Goals

1. To understand how machine learning algorithms differ from other algorithms we have studied
2. To understand what supervised learning is
3. To understand linear regression and the types of problems it can be used for
4. To understand what we mean by pattern recognition and look at three types of pattern recognition algorithms – Nearest Neighbor, Decision Trees and Neural Networks – and see the type of problems they work best for

Goals for this lecture

1. To understand what Machine Learning (ML) means and how the algorithms differ from Brute Force, Divide and Conquer, etc algorithms
2. To see the types of problems ML might be useful for
3. To look at a simplified example of predicting housing prices using a ML approach
4. To understand the concept of a training set.
What is Machine Learning?

Machine Learning (ML) encompasses a lot of things. The field is vast and is expanding rapidly. It is a branch of Artificial Intelligence (AI).

Loosely speaking, ML is the field of study that gives computer algorithms the ability to learn without being explicitly programmed.

The outcome we want from our computer algorithm is PREDICTION. This is different from our previous problems where we wanted the algorithm to solve a specific problem such as finding the best web page for our search, sorting a list of items, or generating a secure means to computing a shared secret in cryptography.

What are we going to use to predict?
Suppose we ask ourselves the following question.

*Will my friend be willing to play tennis today because the weather forecast is lightly cloudy but hot with no wind?*

How can we predict this outcome? Your friend may find it too hot to play or may never play on that day of the week or may always play on cloudy, windless days.

To predict an accurate outcome we must know something about the previous behavior of your friend vis-a-vis playing tennis. This is where the concept of a **training set** is
important.

What does it mean to not be explicitly programmed?

- First, we develop a **generic** algorithm i.e., one that is not a custom code for the problem.
- In the most common type of ML we **train** the algorithm with a set of known data.
- Then we give it some new data and ask the algorithm to **predict** a reasonable result.

So instead of writing many custom programs, we write a generic ML program which can work on a variety of problems.

ML can be used to solve problems where other standard methods don’t work.

For example, suppose you want to write a computer program to predict traffic patterns at a busy intersection like the one at Monroe and Tennessee at various hours of the day. How can this be accomplished?
The City of Tallahassee had an active traffic camera (\#093) at this location for several months; images were uploaded to the web every 2 minutes. This information could be put in a format that is usable by an algorithm. The algorithm is then trained with past hourly traffic patterns and if it has successfully learned, it will be able to predict future hourly traffic patterns.
Two Distinct Types of Machine Learning Algorithms

1. **Supervised machine learning** - The algorithm is trained on a predefined set of examples (called training examples) which allow the algorithm to obtain a prediction when given a new set of data.

   Often the prediction is a classification such as a zipcode or identifying an email as spam or not spam.

2. **Unsupervised machine learning** - The algorithm is given a bunch of unlabeled data and it must find pattern relationships without being trained and try to label the data.

   The most common type of unsupervised ML algorithm is clustering which we will consider later.
For now we will concentrate on Supervised ML. Supervised ML teaches computer algorithms to do what comes naturally to humans and animals – to learn from experience.

Classification techniques in pattern recognition predict discrete outcomes - Is the email spam? Is the object a triangle or a square? Is the image of a tumor cancerous or benign?

Regression techniques predict continuous outcomes - What is the listing price for a
house? What will the temperature be in one hour?

Clustering is used to find patterns or groupings in data that might not be obvious.

Remember that Pattern Recognition is something that humans can do easily but it is difficult for a computer algorithm to capture our thought processes.
Watson played against two top Jeopardy winners and beat them resoundingly.

Watson: $77,147  Ken Jennings: $24,000  Brad Rutter: $21,600

Watson is a supercomputer that combines Artificial Intelligence and sophisticated software to answer questions posed to it.

What type of questions do you think Watson had the most trouble with?
Watson Oncology

Oncology research grows at a rate of approximately 8000 academic papers a day. Too many for even the best of physicians to stay abreast of.

Watson has been trained with over 15 million pages of medical content, including more than 200 medical textbooks and is continually being fed information as it becomes available.

Youtube video How IBM Watson Works
In What Other Real-World Problems Might ML be Useful?

Images of Handwritten Numbers → Machine Learning Algorithm → "1" → "2" → "3" → "4" → etc.

Pattern Recognition: Training an algorithm to read a zip code
Pattern Recognition: Classifying email messages
Self-driving cars

ML in The Google Self-Driving Car
Classroom of the Future where each student will be assessed over the course of their education, helping students master the skills critical to meeting their goals. A system fueled by sophisticated analytics over the cloud will help teachers predict students who are most at risk, their roadblocks, and then suggest measures to help students overcome their challenges.
Classifying a painting by artist and style

A Rutgers University team of computer scientists have used machine learning to train algorithms to recognize the artist and style of fine art painting. They can also reveal connections between artists and between painting styles.

Video: njtvonline.org - The Art & Artificial Intelligence Lab at Rutgers
Linear Regression

Linear regression can be viewed as a type of supervised ML.

We use some data to train the algorithm and then make a prediction.

In linear regression we predict a **continuous quantity**. For example, we could predict the cost of an item, the temperature, the force acting on an object, etc. This is in contrast to classification which predicts a **discrete quantity**. For example, is the picture a lion or a zebra? Is the number a 7 or a 9?

In **linear** regression we are assuming the underlying behavior is **linear**. That means if we have only one variable, then we get a straight line.
What does “depend linearly” mean?

We use linearity all the time without realizing it.

- If we are baking cookies and the recipe calls for 2 cups of flour and we want to double the recipe, then we know that we need 4 cups of flour. If we want to triple the recipe, then we need 6 cups of flour. The amount of flour needed depends linearly on the number of recipes we are making.

- If we buy one Starbucks Grande coffee for $2.10 we know that if we buy our friend one too, it will cost us $4.20. The cost depends linearly on the number of cups we purchase.

What does “depend linearly” mean graphically? We know that when we plot a linear function we get a straight line.
Linear functions have a **constant rate of change** which can be seen graphically.

When $x$ changes by 1, $y$ changes by 0.5
Example. Below are some plots of functions; determine in which cases $y$ depends linearly on $x$. 
A line is uniquely determined by two points and we write its equation as

\[ y(x) = mx + b \]

where \( m \) is the slope of the line and \( b \) is the \( y \)-intercept, i.e., where the line crosses the \( y \)-axis. We say that \( y \) depends linearly on \( x \). Note that the power of \( x \) is 1.

If we have an equation like \( y(x) = 4x^2 \) then \( y \) does NOT depend linearly on \( x \) because the power of \( x \) in the equation is 2. In fact, this is the equation of a parabola.

As a practical example of a linear function, assume you are going shopping and there is a 25\% off the original price sale, then the amount of savings for any item is

\[ \text{savings} = \frac{1}{4} \times \text{original price} \]

so that your savings depends linearly on the original price. If \( x \) represents the original price and \( y \) the savings then

\[ y = \frac{x}{4}. \]

Note that this is an equation of a line where the \( y \)-intercept is zero. For this reason, if the original price is $100 then your savings is $25.00 and if the original price is doubled to $200 then your savings are doubled to $50.00; i.e., the savings depends
linearly on the original price. If the original price is tripled to $300, then we know our savings are tripled to $75.

The amount you pay also depends linearly on the original price because you must pay \( \frac{3}{4} \times \text{original price} \)

So on an item which originally costs $100, then you pay $75 and if the original price is doubled you pay \( 2 \times 75 = 150 \) and if it is halved, you pay half the amount $37.50

Let’s look at an equation of a line where the intercept is not zero such as

\[ y(x) = 1 + 4x \]

If we change \( x \) by an amount \( \Delta x \) then how much does \( y \) change?

\[
y(x + \Delta x) - y(x) = [1 + 4(x + \Delta x)] - [1 + 4x] = [1 + 4x + 4\Delta x] - [1 + 4x] = 4\Delta x
\]

So \( y \) changes by four times the change in \( x \).

If we have the line \( y = 1 + 2x \) then if we change \( x \) by an amount \( \Delta x \) then how much does \( y \) change?
\[ y(x + \Delta x) - y(x) = [1 + 2(x + \Delta x)] - [1 + 2x] = [1 + 2x + 2\Delta x] - [1 + 2x] = 2\Delta x \]

So \( y \) changes by two times the change in \( x \). We see that the slope determines the factor in front of the change.

Now this doesn’t say that the new \( y \) value is twice the old value but rather it says that it changes by an amount 2 times the change in \( x \). Only if the \( y \)-intercept is 0 does the new \( y \) value double as the table below illustrates. In the table below we fix the \( x \) value to be \( x = 1 \) and then add a change to see the change in \( y \) and its new value for different lines. Note that as we evaluate \( y(1 + \Delta x) \) the value doubles as \( 1 + \Delta x \) doubles for the line \( y = 4x \) but not for the line \( y = 4x + 1 \). However, the change in \( y \) is always the same because the slope is 4 in both lines.
Our main example using linear regression will be to predict the listing price of a house in Tallahassee. Because our line will typically not have a zero $y$-intercept we will see that a house for $200K$ will not be twice that of a house for $100K$ but rather the cost goes up the same amount for each $100K$. 

\[
\begin{array}{|c|c|c|c|}
\hline
\Delta x & 1 + \Delta x & y(1) & y(1 + \Delta x) \\
\hline
1 & 2 & 5 & 9 \\
3 & 4 & 5 & 17 \\
7 & 8 & 5 & 29 \\
\hline
\end{array}
\] 

\[
\begin{array}{|c|c|c|}
\hline
\Delta x & 1 + \Delta x & \Delta y \\
\hline
4 & 4 \Delta x \\
12 & 4 \Delta x \\
28 & 4 \Delta x \\
\hline
\end{array}
\]
What is an example of something that doesn’t depend linearly?

If you tell someone you have been in an earthquake, the first question they ask is “What was the magnitude?” This is measured by the Richter scale which is NOT linear. For example, an earthquake that measures 5.0 has a shaking amplitude 10 times that of an earthquake of magnitude 4.0 and corresponds to an energy release of 31.6 times greater.

6.0 earthquake damage 7.0 earthquake damage
Exercises.

1. In the plot, which curve(s) represents a linear function?

2. Assume a store is having a 40% off the original price sale. If you purchase 2 items, one for $100 and another for $200, then the savings on the $200 item is

   (a) the same as for the $100 item
   (b) twice as much as for the $100 item
   (c) three times as much as for the $100 item
   (d) four times as much as for the $100 item
3. For the line $y = 3x - 1$ we know that $y(1) = 2$. If we change $x$ by two (i.e., $x = 3$), then which of the following statement(s) is/are true?

(a) $y$ changes by $3 \times 1 = 3$
(b) $y$ changes by $3 \times 2 = 6$
(c) the new value of $y$ is changed by 2
(d) (b) and (c)
(e) none of the above
Using Linear Regression for a Science Experiment

To see how linear regression works we first look at a simple example from a science lab. Assume we have an object with a given weight of 2 kilograms. We know from Newton’s Second Law of Motion that the force $F$ acting on an object is given by the mass times the acceleration $a$ in meters per second squared of the object, i.e., $F = ma$. Because the weight is fixed then the force depends linearly on the acceleration. This means that if you plot the acceleration on the $x$-axis and the force on the $y$-axis you get a straight line. Also, the $y$-intercept is zero here because if we have no acceleration, we have no force.

In our case $F = 2a$. Suppose you are in a lab taking measurements of the force on the object for 2 different values of the acceleration. Due to experimental errors you probably won’t get the force exactly equal to $2a$. Below is a table of the given acceleration and the measured force compared with the force given by $F = 2a$. 
<table>
<thead>
<tr>
<th>acceleration</th>
<th>measured force</th>
<th>$F = 2a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 m/sec$^2$</td>
<td>2.1</td>
<td>2</td>
</tr>
<tr>
<td>3 m/sec$^2$</td>
<td>4.9</td>
<td>6</td>
</tr>
</tbody>
</table>

I have plotted the two points in a scatter plot which can be done using PLOTLY in a manner analogous to plotting a straight line; you simply select the “scatter plot” option.

Because there are only two points, we can write the equation of the line that passes through them. To write the equation of the line The points (1,2.1) and (3,4.9) must satisfy

$$y = mx + b$$

so when $x = 1$ and $y = 2.1$ we have

$$2.1 = m(1) + b$$

and when $x = 3$, $y = 4.9$ we have

$$4.9 = m(3) + b.$$ If we subtract these two equations then $b$ disappears and we have an equation for the
Once we know \( m \), we can substitute into either equation to get the \( y \)-intercept \( b \). We have

\[
2.1 = 1.4 + b \Rightarrow b = 2.1 - 1.4 = 0.7
\]

So our final equation is \( y = 1.4x + 0.7 \). Note that this is quite different from Newton’s law which says \( y = 2x \). When we plot our line it passes through our two experimental data points.
Experimental data for Newton’s Second Law using 2 points

We can now use this line to predict the force for different values of the acceleration. For example, with an acceleration of 2 we get a force of

\[ 1.4(2) + .7 = 2.8 + 0.7 = 3.5 \]
and with an acceleration of 5 we get a force of 7.7:

\[ y(5) = 1.4(5) + 0.7 = 7 + 0.7 = 7.7 \]

Now suppose you have to measure the force for 4 points, instead of 2. Assume you have the measurements given below.

<table>
<thead>
<tr>
<th>acceleration</th>
<th>measured force</th>
<th>force predicted by ( F = 2a )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 m/sec(^2)</td>
<td>2.1</td>
<td>2</td>
</tr>
<tr>
<td>3 m/sec(^2)</td>
<td>4.9</td>
<td>6</td>
</tr>
<tr>
<td>5 m/sec(^2)</td>
<td>9.1</td>
<td>10</td>
</tr>
<tr>
<td>8 m/sec(^2)</td>
<td>18.2</td>
<td>16</td>
</tr>
</tbody>
</table>

If it was a perfect world then all your points would lie on a straight line but we all know measurements are susceptible to errors. Before our line through the 2 points predicted a force of 7.7 when the acceleration is 5 but our measurement gets 9.1 so clearly these additional points don’t lie on our line. In fact we can’t find any line which passes through all 4 points!
Our goal is to find a line which best represents the data.
Which line is the “best” linear approximation to the data?

It depends on what you mean by “best”!

We choose the line which minimizes the following errors.
Make the sum of these $y$-distances (or errors) as small as possible.

This is called linear regression or a least squares fit to the data. In practice we actually use the sum of the squares of the distances but we won’t worry about that here.

What are these errors for our example? The first $y$-distance (at acceleration $= 1$) is
the distance between the experimental force (2.1) and the predicted force (i.e., the "best" line $b + ax$ evaluated at $x = 1$).

$$\text{first error} = 2.1 - (b + 1 \cdot a)$$

The second $y$-distance (at acceleration = 2) is the distance between the experimental force (4.9) and the predicted force (i.e., the line $b + ax$ evaluated at $x = 2$).

$$\text{second error} = 4.9 - (b + 2 \cdot a)$$

So we want to find $a, b$ which makes the sum

$$[2.1 - (b + a \cdot 1)]^2 + [4.9 - (b + a \cdot 2)]^2 + [9.1 - (b + a \cdot 5)]^2 + [18.2 - (b + a \cdot 5)]^2$$

as small as possible. Mathematically, there is a straightforward way to do this which we won’t go into here. PLOTLY or EXCEL can actually compute this for us.

It turns out that for our problem, the line which best fits our data in this sense is

$$y = 2.32x - 1.29$$
which we plot below. Note that it does not pass through any of our data points.

Just as before with two points, we can use this line to predict the force at other values of the acceleration.
Exercise. How do you think the line would change if we only used the first 3 points?

Exercise. How do you think the line would change if we added the point (10,25)?
1. Machine Learning is a type of Artificial Intelligence.
2. An example of a Machine Learning algorithm is public key encryption.
3. An example of a Machine Learning algorithm is pattern recognition.
4. In unsupervised Machine Learning one uses a training set to “train” the algorithm.
5. Machine learning has only been used to advance the sciences, not the arts.
6. Any linear function has a constant rate of change.
7. If $y$ depends linearly on $x$ then its plot will be a parabola.
8. The slope of a line dictates how much $y$ will change when $x$ is changed.
9. If we have 5 data points to use for linear regression, then the goal is to find a line which passes through these 5 points.
10. Linear regression will make different predictions if different data sets are used.
Suppose Karen is a real estate broker in Tallahassee and she feels that she can walk through a home in Leon county and have a pretty good idea what a fair listing price is.
Now suppose that she hires a trainee who doesn’t have her experience and she wants to devise a strategy to help him learn to accurately price single family homes. What can she do?

Karen decides to write down information that she feels is important for each house that is sold in Tallahassee in the last year. For example, she might have a table like the following.
The idea is that her trainee could look at this data and predict a listing value for a 3 bedroom, 2.5 bath 2850 sq ft house in Betton Hills with a pool and a 2-car garage.

In ML we would use Karen’s table of information about houses as a training data set for a generic algorithm which would then be able to predict the listing price of a house that is just coming on the market.

Of course the prediction relies heavily on the

1. quality of the training set

2. choice of criteria used in training set
For example, the training set might include homes having 1500-4500 sq. ft of living space and ranging in price from $189,000 to $750,000 but we want to know a listing price for a home with 8000 sq. ft. of living space. Then the prediction is probably not very good because we are in the “extrapolation” regime.

Training Set

Prediction
Beware: Potential Problem with using ML to Predict

MY HOBBY: EXTRAPOLATING

As you can see, by late next month you’ll have over four dozen husbands. Better get a bulk rate on wedding cake.
Exercise. Assume that we use the training set below to predict the listing price of a house.

<table>
<thead>
<tr>
<th>Sq. ft.</th>
<th>No. Bedrooms</th>
<th>No. Baths</th>
<th>Zip Code</th>
<th>Year Built</th>
<th>Listing Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>2222</td>
<td>3</td>
<td>3.5</td>
<td>32312</td>
<td>1981</td>
<td>$250,000</td>
</tr>
<tr>
<td>1628</td>
<td>3</td>
<td>2</td>
<td>32308</td>
<td>2009</td>
<td>$185,000</td>
</tr>
<tr>
<td>2893</td>
<td>4</td>
<td>3</td>
<td>32312</td>
<td>1994</td>
<td>$699,000</td>
</tr>
<tr>
<td>1997</td>
<td>3</td>
<td>3</td>
<td>32311</td>
<td>2006</td>
<td>$295,000</td>
</tr>
<tr>
<td>2097</td>
<td>4</td>
<td>3</td>
<td>32311</td>
<td>2016</td>
<td>$290,000</td>
</tr>
<tr>
<td>3200</td>
<td>5</td>
<td>4</td>
<td>32312</td>
<td>1964</td>
<td>$465,000</td>
</tr>
<tr>
<td>2533</td>
<td>3</td>
<td>2</td>
<td>32310</td>
<td>1991</td>
<td>$365,000</td>
</tr>
<tr>
<td>2497</td>
<td>4</td>
<td>4</td>
<td>32309</td>
<td>1990</td>
<td>$289,000</td>
</tr>
</tbody>
</table>

Which of the following 4 houses do you think the algorithm trained with the above data will predict the listing price least accurately? Why?

<table>
<thead>
<tr>
<th>Sq. ft.</th>
<th>No. Bedrooms</th>
<th>No. Baths</th>
<th>Zip Code</th>
<th>Year Built</th>
</tr>
</thead>
<tbody>
<tr>
<td>1137</td>
<td>3</td>
<td>2</td>
<td>32309</td>
<td>1983</td>
</tr>
<tr>
<td>2483</td>
<td>4</td>
<td>3</td>
<td>32312</td>
<td>2016</td>
</tr>
<tr>
<td>2400</td>
<td>4</td>
<td>4</td>
<td>32312</td>
<td>2002</td>
</tr>
<tr>
<td>3160</td>
<td>6</td>
<td>4</td>
<td>32309</td>
<td>1973</td>
</tr>
</tbody>
</table>
What if we use only one criteria to predict listing price?

If we only use one criteria in the training set, then we might expect that the result is not too good. Why?

Suppose we are using the square feet of living space as our criteria in the training set. In this case we expect that if we have two homes with 2800 sq ft of living space then the algorithm will predict the same listing price. However if

- the first home is located in a prestigious neighborhood, is new construction, has a pool, 4 bedrooms, 4 baths, and a 2-car garage
- and the second home was built in 1930, located on a dirt road, and has 3 bedrooms, 2 baths, and no garage

then clearly both homes should not have the same listing price. The reason the prediction failed is that our training set did not include enough training criteria.
We first need to get some training data for the algorithm. How can we get this data?

There are various programs (such as import.io) which “scrape” data from the web and put into a spreadsheet for use.

We will use data from Zillow for Tallahassee which gives the listing price and information about the property. Our training set will consist of all or part of the following 20 pieces of information.
<table>
<thead>
<tr>
<th>Sq. ft.</th>
<th>No. Bedrooms</th>
<th>No. Baths</th>
<th>Zip Code</th>
<th>Year Built</th>
<th>Listing Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>2222</td>
<td>3</td>
<td>3.5</td>
<td>32312</td>
<td>1981</td>
<td>$250,000</td>
</tr>
<tr>
<td>1628</td>
<td>3</td>
<td>2</td>
<td>32308</td>
<td>2009</td>
<td>$185,000</td>
</tr>
<tr>
<td>3824</td>
<td>5</td>
<td>4</td>
<td>32312</td>
<td>1954</td>
<td>$399,000</td>
</tr>
<tr>
<td>1137</td>
<td>3</td>
<td>2</td>
<td>32309</td>
<td>1983</td>
<td>$150,000</td>
</tr>
<tr>
<td>3560</td>
<td>6</td>
<td>4</td>
<td>32309</td>
<td>1973</td>
<td>$315,000</td>
</tr>
<tr>
<td>2893</td>
<td>4</td>
<td>3</td>
<td>32312</td>
<td>1994</td>
<td>$699,000</td>
</tr>
<tr>
<td>3631</td>
<td>4</td>
<td>3</td>
<td>32309</td>
<td>1996</td>
<td>$649,000</td>
</tr>
<tr>
<td>2483</td>
<td>4</td>
<td>3</td>
<td>32312</td>
<td>2016</td>
<td>$399,000</td>
</tr>
<tr>
<td>2893</td>
<td>4</td>
<td>4</td>
<td>32312</td>
<td>2002</td>
<td>$613,000</td>
</tr>
<tr>
<td>1997</td>
<td>3</td>
<td>3</td>
<td>32311</td>
<td>2006</td>
<td>$295,000</td>
</tr>
<tr>
<td>2097</td>
<td>3</td>
<td>4</td>
<td>32311</td>
<td>2016</td>
<td>$290,000</td>
</tr>
<tr>
<td>3200</td>
<td>5</td>
<td>4</td>
<td>32312</td>
<td>1964</td>
<td>$465,000</td>
</tr>
<tr>
<td>4892</td>
<td>5</td>
<td>6</td>
<td>32311</td>
<td>2005</td>
<td>$799,900</td>
</tr>
<tr>
<td>1128</td>
<td>2</td>
<td>1</td>
<td>32303</td>
<td>1955</td>
<td>$89,000</td>
</tr>
<tr>
<td>1381</td>
<td>3</td>
<td>2</td>
<td>32301</td>
<td>2006</td>
<td>$143,000</td>
</tr>
<tr>
<td>4242</td>
<td>4</td>
<td>5</td>
<td>32303</td>
<td>2007</td>
<td>$569,000</td>
</tr>
<tr>
<td>2533</td>
<td>3</td>
<td>2</td>
<td>32310</td>
<td>1991</td>
<td>$365,000</td>
</tr>
<tr>
<td>1158</td>
<td>3</td>
<td>2</td>
<td>32303</td>
<td>1993</td>
<td>$155,000</td>
</tr>
<tr>
<td>2497</td>
<td>4</td>
<td>4</td>
<td>32309</td>
<td>1990</td>
<td>$289,000</td>
</tr>
<tr>
<td>4010</td>
<td>5</td>
<td>3</td>
<td>32309</td>
<td>2002</td>
<td>$549,900</td>
</tr>
</tbody>
</table>

To get an idea of how the algorithm might work we first look at a simplified case where we just list the total living area in square feet and the listing price. Of course this will NOT be a very good predictor because it doesn’t include the number of
bedrooms, baths, neighborhood, whether there is a pool, etc. but it’s a good way to start.

Using the information in the table on the previous slide we do a scatter plot of all the data where the square feet of living space is on the $x$-axis and the listing price is on the $y$-axis. This can be done with PLOTLY in the same manner that you create a line plot except, of course, you request a scatter plot.
Goal: After we have trained the algorithm, we will input the living area of a new house (not in the training set) in square feet and then the output of the algorithm will be a predicted listing price.

Remember though that this is a very simplified problem because we only have one input feature (the size of living space).

The first step in writing the algorithm is to decide how we want to describe our hypothesis (i.e., a good listing price) on a computer. To make things easy, let’s say it depends linearly on the single input variable which is the square feet of living space.

We have seen what “depends linearly” means in general, so we formulate what it means in terms of our pricing houses example.

Assumption: Listing price depends linearly on the total square feet of the house.

- Let $P$ be the listing price of the house
- Let $x$ be our single input parameter which is the square footage of the home.

If $P$ depends on $x$ linearly then we know that
where $a, b$ are unknown; here $b$ is the $y$—intercept and $a$ is the slope of the line. In order to predict the listing price of a new house we need to know $a, b$.

To understand how we determine $a, b$ we first consider the simplified case where we only use two houses in our data set. Recall that the information for the first two houses is:

<table>
<thead>
<tr>
<th>Sq. ft.</th>
<th>Listing Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>2222</td>
<td>$250,000</td>
</tr>
<tr>
<td>1628</td>
<td>$185,000</td>
</tr>
</tbody>
</table>

Using these two houses we can find $a, b$ because this is equivalent to saying it takes two distinct points to determine a line.
Here our first value of $x$ is 2222 with a value of $P$ as 250,000 so

$$250,000 = b + a(2222)$$

For the second house $x = 1628$ and $P = 185,000$ so

$$185,000 = b + a(1628)$$

So we solve these two equations simultaneously by eliminating $b$ first to find $a$. To do this, we simply subtract the two equations to eliminate $b$ so we have a single equation for $a$

$$250000 - 185000 = (2222 - 1628)a \implies a = 109.428$$

To find $b$ we use either of the two equations and substitute this value in for $a$; i.e.,

$$250000 = b + (2222)(109.428) \implies b = 6851.85$$

Then the straight line has slope 109.428 and crosses the $y$-axes at $(0, 6851.85)$. We have

$$P = 6851.85 + 109.428x$$

Note that the two houses we used in our data set lie on the line.
Listing Price

2 points in training set

sq. ft.

$100,000

$200,000

$300,000

$400,000

$500,000

$600,000

$700,000

$800,000
We can now predict the listing price for any house by substituting the number of square feet of living space for $x$ in our linear equation $P = 6851.85 + 109.428x$.

<table>
<thead>
<tr>
<th>Sq. Ft</th>
<th>Predicted Listing Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>$116,280</td>
</tr>
<tr>
<td>2000</td>
<td>$225,708</td>
</tr>
<tr>
<td>2350</td>
<td>$264,008</td>
</tr>
<tr>
<td>3000</td>
<td>$335,136</td>
</tr>
<tr>
<td>4000</td>
<td>$444,564</td>
</tr>
</tbody>
</table>

Note that the price for a 2000 sq ft house is not twice that of a 1000 sq ft house because the $y$-intercept in our equation is not zero.

What is true, is that the change in the price from 1000 to 2000 sq ft is the same as the change from 2000 to 3000 sq ft and from 3000 to 4000 sq ft.

- difference in price from 2000 sq ft to 1000 sq ft: $225,708 - 116,280 = 109,428$
- difference in price from 3000 sq ft to 2000 sq ft: $335,136 - 225,708 = 109,428$
- difference in price from 4000 sq ft to 3000 sq ft: $444,564 - 335,136 = 109,428$
How good are these predictions?

Let’s predict the listing price of some houses from our data set and see how well the algorithm does.

<table>
<thead>
<tr>
<th>Sq. Ft</th>
<th>Predicted Listing Price</th>
<th>Actual Listing Price</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1628</td>
<td>$ 185,000</td>
<td>$ 185,000</td>
<td>$ 0</td>
</tr>
<tr>
<td>3824</td>
<td>$ 425,305</td>
<td>$ 399,000</td>
<td>+$ 26,305</td>
</tr>
<tr>
<td>1137</td>
<td>$ 131,271</td>
<td>$ 150,000</td>
<td>- $ 18,729</td>
</tr>
<tr>
<td>3560</td>
<td>$ 396,416</td>
<td>$ 315,000</td>
<td>+$ 81,416</td>
</tr>
<tr>
<td>2893</td>
<td>$ 323,427</td>
<td>$ 699,000</td>
<td>-$ 375,573</td>
</tr>
<tr>
<td>3631</td>
<td>$ 404,185</td>
<td>$ 649,000</td>
<td>-$ 244,815</td>
</tr>
</tbody>
</table>

Why did the algorithm predict the listing price of the 1628 sq ft house exactly?

From this table we can conclude that we definitely need to improve either the quality or quantity of items in our in the training set.

First, let’s try two different houses for the training set.
Example. Before we had houses with 2222 and 1628 sq ft. in our training set. We see from the table that our predictions were especially bad for homes with larger square feet so this time let’s choose the following 2 homes for our training set.

<table>
<thead>
<tr>
<th>Sq. ft.</th>
<th>Listing Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>3631</td>
<td>$649,000</td>
</tr>
<tr>
<td>1628</td>
<td>$185,000</td>
</tr>
</tbody>
</table>

Determine the line that fits these two houses and use it to predict the listing price of the houses in the previous table.
Explanation of example.

Let $P(x)$ denote the listing price of a house with sq ft $x$. We have

$$P(x) = ax + b$$

and we want to find $a, b$. Using the two points in the training set we have the two equations

$$649000 = a(3631) + b$$
$$185000 = a(1628) + b$$

Subtracting the two equations gives

$$649000 - 185000 = (3631 - 1628)a \implies 464,000 = 2003a$$

or $a = 231.65$ and then

$$b = 185000 - 1628(231.65) = 185,000 - 377,130.3 = -192,130.3$$

Our equation is

$$P(x) = 231.65x - 192130.3$$
Predicting the listing price of the homes gives

\[ P(1628) = 231.65(1628) - 192130.3 = 184,995.9 \]
\[ P(3824) = 231.65(3824) - 192130.3 = 693,699.3 \]
\[ P(1137) = 231.65(1137) - 192130.3 = 71,255.75 \]
\[ P(3560) = 231.65(3560) - 192130.3 = 632,543.7 \]
\[ P(2893) = 231.65(2893) - 192130.3 = 478,033.15 \]
\[ P(3631) = 231.65(3631) - 192130.3 = 648,990.85 \]

The reason the two houses in the training set are not quite exact is because we rounded the values for \( a, b \). As you can see, the predictions are still not very good.

<table>
<thead>
<tr>
<th>Sq. Ft</th>
<th>Predicted Listing Price</th>
<th>Actual Listing Price</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1628</td>
<td>$185,000</td>
<td>$184,996</td>
<td>-</td>
</tr>
<tr>
<td>3824</td>
<td>$693,699</td>
<td>$399,000</td>
<td>+$294,699</td>
</tr>
<tr>
<td>1137</td>
<td>$71,256</td>
<td>$150,000</td>
<td>-$78,744</td>
</tr>
<tr>
<td>3560</td>
<td>$632,544</td>
<td>$315,000</td>
<td>+$317,544</td>
</tr>
<tr>
<td>2893</td>
<td>$478,033</td>
<td>$699,000</td>
<td>-$220,967</td>
</tr>
<tr>
<td>3631</td>
<td>$404,185</td>
<td>$648,991</td>
<td>-</td>
</tr>
</tbody>
</table>
To improve the results we add a house to our training set to use 3 houses.

<table>
<thead>
<tr>
<th>Sq. ft.</th>
<th>Listing Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>2222</td>
<td>$250,000</td>
</tr>
<tr>
<td>1628</td>
<td>$185,000</td>
</tr>
<tr>
<td>3824</td>
<td>$399,000</td>
</tr>
</tbody>
</table>

Here is a scatter plot of the data.
Since the points do not lie on a line, we can’t find values for $a$ and $b$ so that the line $ax + b$ passes through all 3 points! What can we do?

Just like our example using Newton’s second law of motion we must find the line which best fits in the data in the sense of making the $y$ distance from the line to the
data point as small as possible.

Our data points are

\[(2222, \$250,000) \quad (1628, \$185,000) \quad (3824, \$399,000)\]

and if our line is \(y = ax + b\) we see that the sum of the squares of the errors which we want to minimize is

\[\left(250000 - (b + a \cdot 2222)\right)^2 + \left(185000 - (b + a \cdot 1628)\right)^2 + \left(399000 - (b + a \cdot 3824)\right)^2\]

Minimizing the sum of the squares of the errors gives the line

\[P = 31092.8 + 96.5235x\]

when 3 data points are used.

In the plot below, we have drawn the straight line which is the best approximation to the first three houses in our training set and compare it with the line we got using 2 houses in the training set.
If we add the next house which has 1137 sq ft with a listing price of $150,000 to our training set, then we get a different line because the line changes to incorporate this information. Note that both the $y$-intercept and the slope change each time.
2 points  \[ P = 6851.85 + 109.428x \]
3 points  \[ P = 31092.8 + 96.5235x \]
4 points  \[ P = 38624 + 94.1396x \]
As we include additional houses from our training set, the line is modified to account for the new information.

<table>
<thead>
<tr>
<th>Sq. ft</th>
<th>Listing Price</th>
<th>Price per sq. ft.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2222</td>
<td>$250,000</td>
<td>$113</td>
</tr>
<tr>
<td>1628</td>
<td>$185,000</td>
<td>$114</td>
</tr>
<tr>
<td>3824</td>
<td>$399,000</td>
<td>$104</td>
</tr>
<tr>
<td>1137</td>
<td>$150,000</td>
<td>$132</td>
</tr>
<tr>
<td>3560</td>
<td>$315,000</td>
<td>$88</td>
</tr>
<tr>
<td>2893</td>
<td>$699,000</td>
<td>$241</td>
</tr>
</tbody>
</table>

Using 5 houses the slope of the line is reduced because we see that the price per square foot has been greatly reduced (to $88). However, when we add the 6th house it has a very high per square foot price and so the slope must be dramatically increased.

2 points \( P = 6851.85 + 109.428x \)

3 points \( P = 31092.8 + 96.5235x \)

4 points \( P = 38624 + 94.1396x \)

5 points \( P = 55232.5 + 82.6802x \)

6 points \( P = 59807.1 + 107.387x \)
Finally, including all 20 houses in our training set, we have the predictive line $-18625.1 + 151.972x$ given in the plot below.
Now let’s use the algorithm to predict a fair listing price for a house with 2150 square feet and another house with 4110 square feet. Now we will get a different answer depending on how many houses we used in the training set. Below is a table.
<table>
<thead>
<tr>
<th>No. houses in training set</th>
<th>line</th>
<th>Predicted value for 2150 sq ft house</th>
<th>Predicted value for 4110 sq ft house</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6851.9 + 109.4(x)</td>
<td>$242,122</td>
<td>$456,601</td>
</tr>
<tr>
<td>3</td>
<td>31092.8 + 96.5(x)</td>
<td>$238,618</td>
<td>$427,804</td>
</tr>
<tr>
<td>4</td>
<td>38634 + 94.1(x)</td>
<td>$241,034</td>
<td>$425,548</td>
</tr>
<tr>
<td>5</td>
<td>55232.5 + 82.7(x)</td>
<td>$232,995</td>
<td>$395,048</td>
</tr>
<tr>
<td>6</td>
<td>59807.1 + 107.4(x)</td>
<td>$290,689</td>
<td>$501,168</td>
</tr>
<tr>
<td>7</td>
<td>67225.4 + 122.9(x)</td>
<td>$331,355</td>
<td>$572,143</td>
</tr>
<tr>
<td>8</td>
<td>34343.9 + 141.2(x)</td>
<td>$337,883</td>
<td>$614,598</td>
</tr>
<tr>
<td>10</td>
<td>69057.2 + 126.612(x)</td>
<td>$341,273</td>
<td>$589,433</td>
</tr>
<tr>
<td>20</td>
<td>−18625.1 + 151.972(x)</td>
<td>$308,115</td>
<td>$605,980</td>
</tr>
</tbody>
</table>
In our example there is clearly more going on in pricing a house than just the square feet of living space.

We can add another input feature and see what happens. It’s not easy to plot this (we need a 3D plot) but we can still get a result.

Assume that we want to also use the combined number of bedrooms and bathrooms as an input parameter and that the listing price depends linearly on the square footage and this combination. The data we use is given in the table below.
<table>
<thead>
<tr>
<th>Sq. ft.</th>
<th>Sum of No. Bedrooms &amp; Baths</th>
<th>Listing Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>2222</td>
<td>6.5</td>
<td>$250,000</td>
</tr>
<tr>
<td>1628</td>
<td>5</td>
<td>$185,000</td>
</tr>
<tr>
<td>3824</td>
<td>9</td>
<td>$399,000</td>
</tr>
<tr>
<td>1137</td>
<td>5</td>
<td>$150,000</td>
</tr>
<tr>
<td>3560</td>
<td>10</td>
<td>$315,000</td>
</tr>
<tr>
<td>2893</td>
<td>7</td>
<td>$699,000</td>
</tr>
<tr>
<td>3631</td>
<td>7</td>
<td>$649,000</td>
</tr>
<tr>
<td>2483</td>
<td>7</td>
<td>$399,000</td>
</tr>
<tr>
<td>2400</td>
<td>8</td>
<td>$613,000</td>
</tr>
<tr>
<td>1997</td>
<td>6</td>
<td>$295,000</td>
</tr>
<tr>
<td>2097</td>
<td>7</td>
<td>$290,000</td>
</tr>
<tr>
<td>3200</td>
<td>9</td>
<td>$465,000</td>
</tr>
<tr>
<td>4892</td>
<td>11</td>
<td>$799,900</td>
</tr>
<tr>
<td>1128</td>
<td>3</td>
<td>$89,000</td>
</tr>
<tr>
<td>1381</td>
<td>5</td>
<td>$143,000</td>
</tr>
<tr>
<td>4242</td>
<td>9</td>
<td>$569,000</td>
</tr>
<tr>
<td>2533</td>
<td>5</td>
<td>$365,000</td>
</tr>
<tr>
<td>1158</td>
<td>5</td>
<td>$155,000</td>
</tr>
<tr>
<td>2497</td>
<td>8</td>
<td>$289,000</td>
</tr>
<tr>
<td>4010</td>
<td>8</td>
<td>$549,900</td>
</tr>
</tbody>
</table>
We now make the assumption that the listing price depends linearly on the square feet of living space AND linearly on the total number of bedrooms and baths. We have

\[ P = a + bx + cy \]

where \( x \) is the square feet of living space and \( y \) is the combined number of bedrooms and baths. In this case we have 3 unknowns \( a, b, c \). If we have 3 houses in the training set then we can determine them exactly. If we have more than 3 we have to find the coefficients that fits the data in the same way as before.

If we train our algorithm on the data set of 20 houses then

\[ P = 1915.75 + 163.371x - 7216.75y \]

The negative sign in front of the \( y \) term is a bit scary! This means that as the total number of bedrooms and baths increase, then the listing price goes DOWN which is
counterintuitive. For example, if we have a house with 2150 square feet then different combinations of the total number of bedrooms and baths will give different listing prices.

3 bedrooms, 2.5 baths  \[ \Rightarrow 1915.75 + 163.371(2150) - 7216.75(5.5) = $313,471 \]

4 bedrooms, 3 baths  \[ \Rightarrow 1915.75 + 163.371(2150) - 7216.75(7) = $302,646 \]

What is happening here? The reason is that we have a very small training set with extremely variable dependence on the number of bedrooms/baths. For example, the house with 3560 square feet (listing price $315,000) has the second highest combination of bedrooms/baths but in listing price it ranks 9th highest. We need more data in the training set to get a reasonable result. For example, using all houses from the past year in Tallahassee.

This example points out that

- We need enough data in our training set.
- The quality of information (such as number of criteria used) is important.
This ML example is not a Classification algorithm because the output (the listing price) can be any numerical value. This is in contrast to, for example, identifying a zip code which can only contain the numbers from 0 to 9.
Exercise. In this exercise we want to use linear regression to predict the satisfaction an employee has based on his/her/their salary. In the figures, the $x$-axis represents the employee salary in units of $1,000 and the $y$-axis represents the employee satisfaction rating from 0 to 100 where 100 is completed satisfied.

1. In Figure 1 we have plotted the following 2 data points:

<table>
<thead>
<tr>
<th>Employee No.</th>
<th>Salary</th>
<th>Satisfaction Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55K</td>
<td>61</td>
</tr>
<tr>
<td>2</td>
<td>80K</td>
<td>79</td>
</tr>
</tbody>
</table>

Draw the line which best fits the data using linear regression.

2. In Figure 2 we have plotted the line which fits best the data

<table>
<thead>
<tr>
<th>Employee No.</th>
<th>Salary</th>
<th>Satisfaction Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55K</td>
<td>61</td>
</tr>
<tr>
<td>2</td>
<td>80K</td>
<td>95</td>
</tr>
<tr>
<td>3</td>
<td>65K</td>
<td>48</td>
</tr>
<tr>
<td>4</td>
<td>90K</td>
<td>85</td>
</tr>
</tbody>
</table>
using linear regression. This line minimizes the sum of certain distances/errors. In Figure 2 indicate graphically these 4 distances/errors.

3. In Figure 3, we have plotted the line which best fits the 4 data points from the previous question. If we add the additional point (indicated on the plot in blue) sketch what you believe will be the new line. Does the slope increase or decrease? Why?

<table>
<thead>
<tr>
<th>Employee No.</th>
<th>Salary</th>
<th>Satisfaction Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>72K</td>
<td>33</td>
</tr>
</tbody>
</table>

4. In Figure 4, we have plotted the best line which fits 10 data points using linear regression. Use this approximation to predict the satisfaction of the following employees based on their salaries. Which do you think is a better approximation and why?

<table>
<thead>
<tr>
<th>Employee No.</th>
<th>Salary</th>
<th>Predicted Satisfaction Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>58 K</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>105 K</td>
<td></td>
</tr>
</tbody>
</table>
Figure 1
Figure 2
Figure 3
Figure 4
Reading Assignment

Algorithm # 6

9 Algorithms That Changed the Future
The Ingenious Ideas That Drive Today's Computers
John MacCormick
Suppose we have an object of mass 1 kilogram so that $F = 1 \times a$ and we have collected the following experimental data for the force and acceleration:

<table>
<thead>
<tr>
<th>acceleration</th>
<th>measured force</th>
<th>force predicted by $F = a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 m/sec$^2$</td>
<td>1.2</td>
<td>1</td>
</tr>
<tr>
<td>3 m/sec$^2$</td>
<td>2.9</td>
<td>3</td>
</tr>
<tr>
<td>5 m/sec$^2$</td>
<td>5.2</td>
<td>5</td>
</tr>
<tr>
<td>8 m/sec$^2$</td>
<td>7.7</td>
<td>8</td>
</tr>
</tbody>
</table>

1. If we use linear regression to find the line which best fits all the data, then the line will pass through all four points.
2. If we use linear regression to find the line which best fits the first two data points, then the line will pass through these two points.
3. If we use linear regression to find the line which best fits the first two data points, then the line will pass through all four points.
4. If we use linear regression to find the line which best fits the first two data points, then the error we make is 0.

5. If we use linear regression to find the line which best fits all the data, then the error we make is 
   \[(1.2 - 1)^2 + (3 - 2.9)^2 + (5.2 - 5)^2 + (8 - 7.7)^2\]

6. If we use linear regression with these four points to predict the force with an acceleration of 10, then we will get the same prediction if we only used two points.

7. If we add the point with acceleration = 10 and force = 9.1, then the slope of the line from linear regression fitting these 5 points will be greater than the slope of the line using just the given 4 points.

8. If we add the point with acceleration = 10 and force = 12.1, then the slope of the line from linear regression fitting these 5 points will be greater than the slope of the line using just the given 4 points.

9. The line we get from using linear regression with all four data points will probably predict the force when the acceleration is 7 better than when the acceleration is 20.

10. Use the plot to determine which line is the best linear approximation to the given data.
(a) red
(b) green
(c) blue
(d) cyan
Goals for this lecture:

1. To begin to understand why pattern recognition is easy for us but difficult to “teach” a computer algorithm to do
2. To see an example of pattern recognition in action - Google’s photo sorting app
3. To look at our first pattern recognition algorithm called Nearest Neighbor and apply it to different problems.
Computer algorithms such as Page Match and Page Rank for searching and ranking web pages, codes for encrypting/decrypting files, etc. far exceed what a human can do. However this is not the case when it comes to pattern recognition. Humans have a natural advantage here.
In **Linear Regression** we predicted an outcome which was a continuous variable such as the listing price of a house or the force on an object.

In pattern recognition, the outcome is typically *discrete* – does the fingerprint match one in the data base, is the picture a palm tree or a cypress tree, etc.

However our procedure is similar in that we *train* the algorithm with a set of data and then use the trained algorithm to *predict* an outcome for data it has never seen before.
Pattern recognition is a type of Machine Learning algorithm which is considered Artificial Intelligence. It encompasses applications such as

- fingerprint identification
- face recognition
- object recognition
- speech recognition
- handwriting recognition
Object Recognition

Handwriting Recognition
The state of the art in facial recognition (approximately first 2 minutes of video) youtube video “How does facial recognition work? - Brit Lab”
Here is an example of how the Google app for sorting your photos work.

Google for Education - Findable photos video (edu.google.com)
We can easily identify the objects in the picture and read someone’s handwriting (if legible). You can argue that we have been “trained” to do this through school and life experiences. Somehow we have to train an algorithm to identify objects, read handwriting, etc.

What is the strategy for doing this?

Do we take a different approach for facial recognition than from handwriting recognition or do we look at a unifying approach which all these pattern recognition problems have in common.

If we think about it, pattern recognition problems are Classification problems unlike our house listing price example. For example, for fingerprint recognition we compare the given fingerprints with those in a data file; the algorithm should either classify the given fingerprints as matching one in the data file or classify it as “no match”.

For object recognition, we might have only pictures of mammals and the algorithm is used to predict what type of mammal a picture represents. Thus there are a finite number of choices to classify the images.
Our goal is to look at three different ML Classifier Algorithms and see some examples of pattern recognition problems they are best suited for.

1. Nearest-Neighbor Classifier and its Variants
2. Decision Trees
3. Neural Nets
Nearest-Neighbor Classifier & Variants

• This is probably the simplest classification algorithm. In fact it is so simple that the algorithm doesn’t learn anything!

• It is based on classifying an object by the classification of its nearest neighbor (or neighbors)

• For example, if you move to a new city you might want to determine the Post Office that is closest to your house. So you look at the addresses of several Post Offices in your area on a map and determine the one that is nearest to your new home. You are using the Nearest Neighbor approach.

• We have to define what we mean by “nearest neighbor”. If we are talking about the person/persons who live closest to you, we understand this meaning but if we are talking about identifying a number in a zip code by its “nearest neighbor” then it’s not as clear what this means.

• We will look at a couple of examples before looking at identifying a zip code.
Example of Nearest-Neighbor Classifier using Visual Analysis

Suppose you want to predict whether a person living in the Midtown area of Tallahassee is a Democrat or a Republican based on the political persuasion of neighbors. Suppose we have mapped out the area and identified those homes where at least one of the occupants is registered as a Democrat or a Republican. A simplified map is given below.
We want to predict whether each of the two residences (represented by a question mark) is a Democrat or Republican. Let’s start with the residence marked in cyan in the upper left hand corner. Clearly, if **WE** use a Nearest Neighbor strategy to guess the political affiliation of the home then we would say they are Republicans because the surrounding neighbors are all Republicans.
Visually, to determine the nearest neighbor to a residence we would draw circles around the residence until we touch a neighbor. This would be its “nearest neighbor”. Clearly for this residence the nearest neighbor is a Republican.
However, when we look at the bottom residence (green) we see that its nearest neighbor is Republican but the others surrounding it are Democrats. So if we use an algorithm to determine the political persuasion based on the single closest neighbor, the prediction would be Republican for both. However, our intuition tells us that the bottom residence (green) is probably Democrat.
What can we do? Instead of taking a single closest neighbor, we could take its nearest
2 neighbors.

Top - 2 Republican neighbors $\implies$ Republican
Bottom - 1 Democratic neighbor, 1 Republican $\implies$ ?
Top - 3 Republican neighbors $\Rightarrow$ Republican

Bottom - 2 Democratic neighbors, 1 Republican $\Rightarrow$ Democrat
Top - 4 Republican neighbors $\Rightarrow$ Republican

Bottom - 3 Democratic neighbors, 1 Republican $\Rightarrow$ Democrat
This variant of the Nearest Neighbor (NN) algorithm is called k Nearest Neighbor or kNN for short. So when $k=1$ it is the usual NN algorithm but if $k$ is some other integer then it means to use $k$ nearest neighbors to classify the object.

Using NN or 2 nearest neighbors the green dot is classified as a triangle but using 5 nearest neighbors (kNN when $k=5$) it is classified as a square.
Exercise. Use the schematic to answer the following questions.
1. Using a single nearest neighbor classifier, how would the residence in question (marked in green) be classified?

2. Using a 2-nearest neighbor classifier, how would the residence in question (marked in green) be classified?

3. Using a 3-nearest neighbor classifier, how would the residence in question (marked in green) be classified?
Suppose in our previous example, we had data that recorded the amount of donations to the respective party that each residence has made in the last year.

If your closest neighbor is a Republican but has never donated to the RNP and your second closest neighbor is a Democrat who has donated $1000 to the DNP you might believe that your Democratic neighbor has much stronger feelings about his/her/their political persuasion and thus might be a greater influence on you. Weighted kNN classifiers can take this type of information into account. The word “weighted” is used in the sense of IMPORTANCE.
For example, when we used 2 nearest neighbors we were unable to determine the political persuasion of the green house because one of the nearest neighbors was Republican and the other Democrat. However, if we give the Democratic neighbor a weight of 5 and the Republican neighbor a weight of 2, then we would classify the political persuasion of the green house as Democratic because the Democratic neighbor has 2.5 times as much importance (i.e., weight) as the Republican neighbor.
Exercises. Use the schematic below to answer the following questions.

1. Using the NN approach, how would the residence in question (marked in green) be classified?
2. If we use NN, is the result always unique?
3. If we use kNN with k=2, how would the residence in question (marked in green) be classified?
4. If we use kNN, is the result always unique?
5. If we use a weighted 2-nearest neighbor classifier, how would the residence in question (marked in green) be classified if Democratic residences have a larger weight than Republican residences?
Scaling an image using Nearest Neighbor

Many times we have a photo that we want to rescale. A nearest neighbor approach can be used here, although there are now much better algorithms.

To see how we can do this, let’s first understand how an image is stored on your phone or computer.

Youtube video: How do computers store images?
As we have seen, each pixel is stored on the computer. Each grayscale color is assigned an integer between 0 and 256 and each color pixel is assigned three values for RGB (red, green and blue).

For simplicity, assume that we have a grayscale image that is stored as 3 by 3 pixels, i.e., only 9 with the values below.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>127</td>
<td>131</td>
</tr>
<tr>
<td>122</td>
<td>124</td>
<td>125</td>
</tr>
<tr>
<td>120</td>
<td>121</td>
<td>124</td>
</tr>
</tbody>
</table>

We want to rescale this image so that it is 9 by 9 pixels, or 81 total. An obvious approach is to assign 9 pixels corresponding to their location in the smaller image.
How do we assign the remaining pixels?

We can use a nearest neighbor approach to do this. Let’s call the bottom left cell in the (1,1) position meaning the first row and second column. The pixel labeled 121 is in the (1,5) position because it is in the first row and 5th column.

The pixel in the (1,2) position has the pixel labeled 120 as its nearest neighbor but the pixel in the (1,4) position has 121 so we assign each pixel the value of its nearest
neighbor. The pixel in position (1,3) is equally close to the pixel labeled 120 and 121 so we can assign it either value. We continue this way assigning all the pixels that need values.

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>125</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>131</td>
<td>131</td>
</tr>
<tr>
<td>125</td>
<td>125</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>131</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>122</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td>122</td>
<td>122</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td>122</td>
<td>122</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>120</td>
<td>121</td>
<td>121</td>
<td>121</td>
<td>124</td>
<td>124</td>
</tr>
<tr>
<td>120</td>
<td>120</td>
<td>121</td>
<td>121</td>
<td>121</td>
<td>124</td>
<td>124</td>
</tr>
</tbody>
</table>

The pixels which haven’t been assigned possess two neighbors which are equally distance apart so either value can be taken. The images enlarged in this manner appear “pixelated” as the following two images illustrate.
1. In pattern recognition the outcome is typically continuous, not discrete.
2. Facial recognition is a more difficult problem than fingerprint recognition.
3. If we are using the Nearest Neighbor approach (with $k=1$) to classify an item, then there is always a unique solution.
4. If we are using the $k$ Nearest Neighbor approach with $k=2$ to classify an item, then there is always a unique solution.
5. Suppose we want to classify an object as either a circle, triangle or square. If we are using the $k$ Nearest Neighbor approach with $k=2$ and its nearest neighbor is a triangle and the second and third nearest neighbors are circles then how is the object is classified?
6. Suppose we want to classify an object as either a circle, triangle or square. If we are using the $k$ Nearest Neighbor approach with $k=3$ and its nearest neighbor is
a triangle and the second and third nearest neighbors are circles then how is the object is classified?

7. Suppose we want to classify an object as either a circle, triangle or square. If we are using the k Nearest Neighbor approach with \( k = 3 \) and its nearest neighbor is a triangle, the second nearest is a circle and the third nearest neighbor is a square then how is the object is classified?

8. Classifying an object using kNN with different values of \( k \) always gives the same result.

9. To store an image on a computer each pixel’s value is converted to binary and stored.

10. When you enlarge an image using a Nearest Neighbor approach then the scaled image is identical to the original image.
Goals for this lecture:

1. See how the Nearest Neighbor algorithm can be used to identify a zip code.
2. To see how we can quantify our visual approach to Nearest Neighbor so that it can be implemented on a computer.
3. To do a class exercise on kNN.
Application of Nearest Neighbor Classifiers to Reading Postal Codes

In the previous example we just used our usual definition of distance to visually decide which is the nearest neighbor. What can we do for handwritten numbers?

Suppose we have a set of handwritten numbers which we have classified as 0,1,2,3,4,5,6, 7,8 or 9. Assume further that we have scaled all of these numbers so they are the same sizes.
Now we have a handwritten number which we want to identify.

What do we do?

First we scale it as we did with our known set of numbers. We start with the first number in our training set and compare the two

\[ 9 - 4 = 5 \]

We somehow “subtract” the two scaled images and see what remains. We look at the portion of the number we are trying to identify which matches with the first number in our training set. This is the amount of similarity that they have.
We proceed through the data base and see which handwritten number has the smallest remainder and claim that it is our letter’s nearest neighbor and classify it the same way.

\[ 9 - 9 = 9 \]
Exercise. Using the training set below, classify the image using the nearest neighbor classifier.
Suppose we have a group of objects consisting of circles and squares and we have the \((x, y)\) coordinates of each object as indicated in the schematic.

To classify a new object visually we drew circles around the unknown object until we found its nearest neighbor or \(k\) nearest neighbors. We can’t write a computer algorithm to do this so we want to actually calculate the distance between the unknown object and its neighbors to find its nearest neighbor.

Typically when you find the distance between two points you use the standard Euclidean distance.

How do we find the Euclidean distance between two points?

If we are in one dimension (i.e., on the \(x\)-axis) then we just take the absolute value of the difference in their locations. We take the absolute value because distance is always \(\geq 0\).
What do we do if the points are in the $(x, y)$ plane?

Recall that to find the distance between two points we add the square of the difference in their $x$-components and their $y$-components and then take the square root.

\[
\text{The Euclidean distance between two points } (x_1, y_1) \text{ and } (x_2, y_2) \text{ is}
\]
\[
\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}
\]

Note that this formula also works in one dimension when $y = 0$. In this case we have
\[
\sqrt{(x_1 - x_2)^2} = |x_1 - x_2|
\]
Example. Find the Euclidean distance between the points (4,9) and (1,5).

\[
\text{distance} = \sqrt{(4 - 1)^2 + (9 - 5)^2} = \sqrt{(3)^2 + (4)^2} = \sqrt{9 + 16} = \sqrt{25} = 5
\]

Because distance is always positive we take the positive square root of 25 as the answer.
Example. Find the Euclidean distance between the points \((-1, -2)\) and \((3, 5)\).

\[
\text{distance} = \sqrt{(3 - (-1))^2 + (5 - (-2))^2} = \sqrt{(3 + 1)^2 + (5 + 2)^2} = \sqrt{16 + 49} = \sqrt{65} \approx 8.06
\]
Example. Find the Euclidean distance between the points (-1,-2) and (-3,-4).

$$
\text{distance} = \sqrt{\left( -3 - (-1) \right)^2 + \left( -4 - (-2) \right)^2} = \sqrt{(-3 + 1)^2 + (-4 + 2)^2}
$$
$$
= \sqrt{(-2)^2 + (-2)^2} = \sqrt{4 + 4} = \sqrt{8} \approx 2.83
$$
Let’s return to our problem of classifying an object as either a circle or a square where we have the coordinates of each object. We can envision a Brute Force approach to finding the nearest neighbor of the given object in order to classify it. We find the Euclidean distance from the unknown object to every other object and determine which is closest. Then we classify the object the same as its nearest neighbor or k nearest neighbors.

Example. Classify the unknown object located at (4,1) by using a Brute Force approach and the Nearest Neighbor method. Below we tabulate the location, classification and an identifying ID for each known object.

<table>
<thead>
<tr>
<th>Shape</th>
<th>ID &amp; Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circles:</td>
<td>C1: (2,3)</td>
</tr>
<tr>
<td></td>
<td>C2: (-2,-1)</td>
</tr>
<tr>
<td>Squares:</td>
<td>S1: (5,2)</td>
</tr>
<tr>
<td></td>
<td>S2: (3,5)</td>
</tr>
<tr>
<td></td>
<td>S3: (1,3)</td>
</tr>
</tbody>
</table>

We first compute the Euclidean distance from each known object to the one we want to classify.

Distance from (4,1) to C1:

\[
\sqrt{(2 - 4)^2 + (3 - 1)^2} = \sqrt{(-2)^2 + (2)^2} = \sqrt{4 + 4} = \sqrt{8}
\]
Distance from (4,1) to C2:
\[
\sqrt{(-2 - 4)^2 + (-1 - 1)^2} = \sqrt{(-6)^2 + (-2)^2} = \sqrt{36 + 4} = \sqrt{40}
\]

Distance from (4,1) to S1:
\[
\sqrt{(5 - 4)^2 + (2 - 1)^2} = \sqrt{(1)^2 + (1)^2} = \sqrt{1 + 1} = \sqrt{2}
\]

Distance from (4,1) to S2:
\[
\sqrt{(3 - 4)^2 + (5 - 1)^2} = \sqrt{(-1)^2 + (4)^2} = \sqrt{1 + 16} = \sqrt{17}
\]

Distance from (4,1) to C3:
\[
\sqrt{(1 - 4)^2 + (3 - 1)^2} = \sqrt{(-3)^2 + (2)^2} = \sqrt{9 + 4} = \sqrt{13}
\]

We tabulate the results below.
Using the Nearest Neighbor approach and Euclidean distance we see that its nearest neighbor is the square S1 located at (5,2) so we classify the object as a square.

Exercise. Using the results from the previous example,

1. classify the object, if possible, using kNN with k=2.
2. classify the object, if possible, using kNN with k=3.
3. classify the object, if possible, using kNN with k=4.

Let’s write out the Brute Force algorithm for a general problem. Assume \((x, y)\) are the coordinates of the object that we want to classify and the locations of the other objects are \((x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots (x_N, y_N)\). Assume further that we have a
list which identifies each of these $N$ objects; e.g., as a square, circle, triangle, etc. Basically all we have to do is calculate the distance from $(x, y)$ to each of the other $N$ objects and pick the object where the distance is shortest. Instead of calculating all the distances and then picking out the shortest, we can do this “on the fly”. Here we save the number of the point closest using the variable `point_no`.

**Brute Force Algorithm for Nearest Neighbor**

Set `min` = a big number

Loop: For $i = 1, 2, \ldots, N$

Compute $d = \sqrt{(x - x_i)^2 + (y - y_i)^2}$

If $d < \text{min}$ then set $\text{min} = d$ & set `point_no`= $i$

end loop

After we find the location of the nearest neighbor we check our list with the identity of the known objects and classify the unknown object with its classification.
Example. Use the Brute Force Algorithm above to determine the point from the list below that is closest to the point (2, 5). Then classify the object at (2,5) from the list.

| Pt. #1 | (2.5,4.8) | Pt. #2 | (2,4.4) | Pt. #3 | (1.5,5) | Pt. #4 | (2.5,4.9) |

ID list: triangle, circle, square, square

- Set $\text{min} = 10$
- For $i = 1$
  
  $d = \sqrt{(2 - 2.5)^2 + (5 - 4.8)^2} = 0.538516$
  
  $d < \text{min}$ so set $\text{min}=d= 0.538516$ & $\text{point_no}=1$

- For $i = 2$
  
  $d = \sqrt{(2 - 2)^2 + (5 - 4.4)^2} = 0.6$
  
  $d > \text{min}=0.538516$ so do nothing

- For $i = 3$
  
  $d = \sqrt{(2 - 1.5)^2 + (5 - 5)^2} = 0.5$
\[ d < \min = 0.538516 \text{ so set } \min = d = 0.5 \text{ & point}\_\text{no}=3 \]

- For \( i = 4 \)
  \[ d = \sqrt{(2 - 2.5)^2 + (5 - 4.9)^2} = 0.5099 \]
  \[ d > \min = 0.5 \text{ so do nothing} \]

Closest point: Point \# 3 with a distance of 0.5

From our list we see that the object at point \# 3 is a square so we classify our object as a square.
How can we use this approach to implement kNN with $k=2$?

The naive approach would be to go through all the points using the algorithm above and find the nearest neighbor. Then eliminate that point from our list and go through all the points again (except the one we found as the nearest neighbor) to find the second nearest neighbor. However, this is very wasteful because we can go through the points once and keep track of the two nearest neighbors.

How do we modify our algorithm to do kNN with $k=2$?

In this case we want to keep the TWO smallest distances.
Brute Force Algorithm for k Nearest Neighbor with k=2

Set \( \text{min1} = \) a big number; set \( \text{min2} = \text{min1} + 1 \); set \( \text{point\_no1} = 0 \)

Loop: For \( i = 1, 2, \ldots, N \)

Compute \( d = \sqrt{(x - x_i)^2 + (y - y_i)^2} \)

If \( d < \text{min1} \)

then set \( \text{min2} = \text{min1} \), \( \text{min1} = d \) & set \( \text{point\_no2} = \text{point\_no1} \); set \( \text{point\_no1} = i \)

else if \( d < \text{min2} \) set \( \text{min2} = d \) and set \( \text{point\_no2} = i \)

end loop

Example. Use the Brute Force Algorithm for kNN with k=2 above to determine the two points from the list below that are closest to the point \((2, 5)\). Then classify the object at \((2,5)\) from the list.

<table>
<thead>
<tr>
<th>Pt. #1</th>
<th>(2.5,4.8)</th>
<th>Pt. #2</th>
<th>(2,4.4)</th>
<th>Pt. #3</th>
<th>(1.5,5)</th>
<th>Pt. #4</th>
<th>(2.5,4.9)</th>
</tr>
</thead>
</table>

ID list: triangle, circle, square, square

- Set \( \text{min1} = 10 \), \( \text{min2} = 11 \)
• For $i = 1$

$$d = \sqrt{(2 - 2.5)^2 + (5 - 4.8)^2} = 0.538516$$

$d < \min_1$ so set

$$\min_2 = \min_1 = 10; \ point\_no2 = \point\_no1 = 0$$

$$\min_1 = d = 0.538516 \ & \ \point\_no1 = i = 1$$

• For $i = 2$

$$d = \sqrt{(2 - 2)^2 + (5 - 4.4)^2} = 0.6$$

$d > \min_1 = 0.538516$

$d < \min_2 = 10$ so set $\min_2 = d = 0.6$ and set $\point\_no2 = i = 2$

• For $i = 3$

$$d = \sqrt{(2 - 1.5)^2 + (5 - 5)^2} = 0.5$$

$d < \min_1 = 0.538516$ so set

$$\min_2 = \min_1 = 0.538516; \ \point\_no2 = \point\_no1 = 1$$

$$\min_1 = d = 0.5 \ & \ \point\_no1 = i = 3$$

• For $i = 4$
\[d = \sqrt{(2 - 2.5)^2 + (5 - 4.9)^2} = 0.5099\]

\[d > \text{min1}=0.5 \text{ so check if it is smaller than min2}\]

\[d < \text{min2}=.5385 \text{ so set } \text{min2}=d =0.5385 \text{ and set point_no2}=i=4\]

Closest point: Point # 3 with a distance of 0.5, second closest point is Point #4 with a distance of 0.5099

From our list we see that the object at point # 3 is a square and point #4 is also a square so we classify our object as a square.

Here is a table of the values after checking each point.

<table>
<thead>
<tr>
<th>Point No.</th>
<th>d</th>
<th>min1</th>
<th>point_no1</th>
<th>min2</th>
<th>point_no1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>10</td>
<td>0</td>
<td>11</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0.5385</td>
<td>0.5385</td>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
<td>0.5385</td>
<td>1</td>
<td>0.6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>0.5</td>
<td>3</td>
<td>0.5385</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0.5099</td>
<td>0.5</td>
<td>3</td>
<td>0.5099</td>
<td>4</td>
</tr>
</tbody>
</table>
The Brute Force approach might work okay if we only have to do it once but in practice we typically have a large set of points and want to probe it many times to find a nearest neighbor.

**What should we do in this case?**

We should sort the points first. A popular technique for this situation is the Bucket/Bin Sort which was explored in the homework.

Suppose we have a rectangular domain which contains all of the points. Then we divide our domain into “buckets/bins” and determine which bin each data point is in. We know the $x$— and $y$-coordinates of the boundaries of each bin.
If we want to find the nearest neighbor to a point, say (2.9,6.8) (marked in red) what do we have to do? What about the point (2.1,7.2) (marked in blue)?

Is it enough to just search the bin that the point is in?
1. A real-world application for the Nearest Neighbor method is identifying a zip code.

2. The Euclidean distance between two distinct points is always positive.

3. The Euclidean distance between the points (0,2) and (0,5) is
   (a) 3
   (b) 4
   (c) 5
   (d) -3

4. The Euclidean distance between the points (-4,0) and (1,0) is
   (a) 3
   (b) 4
   (c) 5
5. The Euclidean distance between the points (1,2) and (4,1) is
   (a) $\sqrt{10}$
   (b) 4
   (c) $\sqrt{34}$
   (d) 2

6. The Euclidean distance between the points (-1,-2) and (-1,-5) is
   (a) 2
   (b) 3
   (c) 4
   (d) 5

7. In the kNN algorithm different classifications can be obtained when different values of k are used.

8. In the Brute Force algorithm for Nearest Neighbor we must go through ALL the points to find the nearest neighbor.
9. In the Brute Force algorithm for kNN with k=2 we must go through ALL the points TWICE to find the two nearest neighbors.

10. Suppose we have a set of 100,000 points and we want to find the nearest neighbor out of these 100,000 points for 200 new points. The most efficient way to do this is to sort the 100,000 points first.
1. Consider the diagram above where we have two classifications: red circles and blue squares. We want to classify the green “X” as either a red circle or a blue rectangle by using KNN.
(a) Use the KNN approach with $k = 1$ to determine the classification of the green point. Justify your answer graphically.

(b) Use the KNN approach with $k = 2$ to determine the classification of the green point, if possible. Justify your answer graphically.

(c) Use the KNN approach with $k = 3$ to determine the classification of the green point, if possible. Justify your answer graphically.

2. In this problem we want to apply the Brute Force algorithm discussed in class to find the point closest to (2,3) out of the set of points given below. Then we want to classify it using the given ID list. Fill in the table below. I have filled in the results for the first loop.

Points: (2,1), (4,2) (1,2) (0,1)

ID: triangle, circle, circle, triangle

Set $\min = 10$. 
<table>
<thead>
<tr>
<th>Point #</th>
<th>distance between (2,3) and point</th>
<th>result</th>
<th>min</th>
<th>point_no</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$d = \sqrt{(2 - 2)^2 + (3 - 1)^2} = 2$</td>
<td>$d &lt; \text{min}$ so set $\text{min}=d$</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Answer: minimum distance is _________ at point number _____

Classification: _________

3. Suppose we have the following line with one red circle and one blue rectangle. Draw a line perpendicular to the given line that divides the interval into two parts so that any point located in one part should be classified as a red circle and any point located in the other part should be classified as a blue rectangle using NN. Give the point where the two lines intersect. Then use your partition to classify the green and brown points. Explain your reasoning.

4. Suppose we have the following box with one red circle and one blue rectangle. Generalize your approach in the previous exercise to draw a line that divides the
box into two parts so that any point located in one part should be classified as a red circle and any point located in the other part should be classified as a blue rectangle. Then use your partition to classify the green and brown points. Explain your reasoning.
Decision Trees Classifiers
Goals for this lecture:

1. Understand what a decision tree is.
2. Use an existing decision tree for classification.
3. See how to use training data to create a decision tree.

A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decision.
The advantage of this method is that it is easy to use and doesn’t require setting parameters as we must do in the next example of pattern recognition algorithms.
Example. Suppose we have a list of homes located in either Tallahassee or Atlanta and we wanted to separate the list by city. If we knew the elevation of each home we could determine the city it is located in (i.e., classifying it) by a single question. The elevation of Atlanta is around 1000 ft and Tallahassee around 200 ft. So we could simply say if the elevation is \( > 500 \) feet the house is in Atlanta, otherwise it is in Tallahassee. A simple decision tree with only one conditional would be as follows

![Decision Tree Diagram]

Example How to predict an outcome given a decision tree. “Should I play tennis today based on the weather forecast?”

This is an example where we sort through the tree to the appropriate leaf node to get the correct classification which in this case is Yes or No for whether we should play tennis.
We start at the root node of the tree and test the attribute specified by this node and then move down the tree branch corresponding to the response of the attribute.

In this example, the attribute we are checking for the root node is the Outlook and the three options are Sunny, Overcast or Rainy.

For example, if the Outlook is Sunny then we proceed to the next leaf of the tree which tests the Humidity level. The choices are either High or Normal. If the Humidity level is Normal then we follow the branch down to get the classification Yes and if it is High we get the classification No.

What is the classification for the following?

Outlook: Rain

Wind: Strong
Decision Tree as a Classifier

Example: 20 Questions Game
Example. Identify a particular vertebrate by asking questions

**Question 1:** Is it warm-blooded? (assume it is either warm-blooded or cold-blooded vertebrate)

What is a good second question? Clearly it is based on the answer to the first question. We know that cold-blooded animals include fish, reptiles, amphibians while warm blooded animals include birds and mammals so we might have the first level of the decision tree look like the following.
Now if the answer to the first question is “yes” then we know it is a mammal or a bird. There are many choices here for the next question. We could ask any of the following questions; however if we recall, a platypus is a mammal which lays eggs so probably asking if it is a bird or mammal is best.

1. Does it give live birth?
2. Does it lay eggs?
3. Is it a bird?
4. Is it a mammal?

If the answer to the first question is “no” then we know that it is a fish, reptile or amphibian. Maybe the best strategy here is just to ask if it is one of these three.

Our decision tree might look like the following.
To get the correct identification we follow the Decision Tree starting at the root (the first question) based on the answer. For example, if the answer to the first question is "yes" and the answer to the second question is "no" we know that the vertebrate is a bird.
Exercises.

1. Use the Decision Tree below to classify the vertebrate if the answers to the first 3 questions are “yes”, “yes”, “no” (in order).
   (a) lives on land
   (b) bat, whale or dolphin
   (c) song bird
   (d) a bird, but not a song bird

2. Use the Decision Tree below to classify the vertebrate if the answers to the first 3 questions are “yes”, “no”, “yes” (in order)
   (a) lives on land
   (b) bat, whale or dolphin
   (c) song bird
   (d) a bird, but not a song bird
Is it warm-blooded?  yes
Is it a mammal?  no
Is it terrestrial?  yes
lives on land  no  bat, whale or dolphin
Is it a song bird?  yes  song bird  no  not a song bird
Is it a fish?  yes

As an application of decision trees and logical thinking, let’s make decision trees to determine the region in which a point is located. We will start with a simple example in one dimension and then increase the difficulty by considering regions in two dimensions. This idea of considering all the cases can help us when we are actually writing a computer program.

**Example.** Construct a decision tree to determine whether a point is in region R1, R2, or R3 as illustrated in the figure.

There is not a unique way to do this but here is one.
Is $x > 0$?

- No
  - Point in R1

- Yes
  - Is $x > 6$?
    - No
      - Point in R2
    - Yes
      - Point in R3
Example. Construct a decision tree to determine whether a point \((x, y)\) is in the interior (not on the boundary) of region R1, R2, or R3 as illustrated in the figure.

There is not a unique way to do this but here is one. Note that we can decide if a point is in R1 simply by checking the \(x\)-coordinate of the point but to determine if a point is in R2 or R3 we must check the \(y\)-coordinate of the point.
Is $x < 4$?

- Yes: point in R3
- No:
  - No: point in R1
  - Yes: Is $y < 3$?
    - Yes: point in R3
    - No: point in R2
Example. Construct a decision tree to determine whether a point \((x, y)\) is in the interior (not on the boundary) of region R1, R2, R3 or R4 as illustrated in the figure.

In this case we could check to see if \(y > 3\); if so then it is in either R3 or R4. If not then we can check the \(x\)-coordinate to see if the point is in R1 or R2.
Is $y > 3$?

- no
  - Is $x < 4$?
    - no
      - point in $R_1$
    - yes
      - point in $R_2$
- yes
  - Is $y > 4$?
    - no
      - point in $R_3$
    - yes
      - point in $R_4$
Example. Construct a decision tree to determine whether a point \((x, y)\) is in the interior (not on the boundary) of region R1, R2, R3, R4 or R5 as illustrated in the figure.

Now we have to modify our previous decision tree because if \(y > 3\) it can now be in R3, R4 or R5
Is $y > 3$?

Is $x < 4$?

Is $x > 5$?

Is $y > 4$?

point in R1

point in R2

point in R3

point in R4

point in R5
Suppose you are asked to develop a Decision Tree which would help you decide where you should post your status.
Most successful Social Networks have a distinct audience. So we want to take this into account in making our Decision Tree. To simplify matters we will only consider 5 popular Social Networks as options plus the “don’t post” option.

Twitter is a Social Network that helps friends, family and coworkers to communicate and stay connected through the exchange of quick, frequent messages.

Foursquare is a Social Network which is available for common smartphones. Its purpose is to help you discover and share information about businesses and attractions around you.

Facebook is a Social Network which makes it easy for you to connect and share with friends and family.
Google+ is a Social Network for discovering and sharing digital content with friends, family and co-workers.

LinkedIn is a Social Network designed for the business community.

Let’s think about what is distinct about each Social Network.

• Clearly LinkedIn is where you post your professional accomplishments or those of others.

• Twitter has the reputation of being a place where people complain and also many well known individuals (singers, politicians, etc.) post messages. Also there are no “likes” as on Facebook.

• Facebook was originally developed for college students and it has the reputation of individuals posting every little thing about their day - what they ate for breakfast, etc. You have the ability to accumulate “likes”.
• **Google+** has not “caught on” yet and so there are limited users.
• **Foursquare** is popular for a quick “on the go” post.
Where should you post your status?

Do you want anyone to actually see it?
- Yes
  - Are you in a bar?
    - Yes
      - Would it be awkward to explain to your boss?
        - Yes
          - Are you addicted to "Likes"?
            - Yes
              - Don't post it!
            - No
              - LinkedIn
              - Facebook
              - Twitter
        - No
          - LinkedIn
          - Facebook
          - Twitter
      - No
        - LinkedIn
        - Facebook
        - Twitter
  - No
    - LinkedIn
    - Facebook
    - Twitter
- No
  - Is it boring?
    - Yes
      - LinkedIn
      - Facebook
      - Twitter
    - No
      - Is it personal?
        - Yes
          - LinkedIn
          - Facebook
          - Twitter
        - No
          - Foursquare
          - Google Plus
          - LinkedIn
          - Facebook
          - Twitter

LinkedIn
Facebook
Twitter
How can Decision Tree Classifiers Learn?

So far the examples we have seen used Decision Trees for inductive reasoning such as in “20 Questions”. Since we are interested in Machine Learning, we want to understand how we can use training data to create a Decision Tree.

The ML goal for a Decision Tree is to

1. achieve perfect classification
2. using the smallest number of decisions (i.e., questions)

Most algorithms for Decision Tree Learning are constructed in a top down fashion. This means that they begin with the question

   **What attribute should be tested at the root node of the tree?**

To do this we have a set of possible attributes to use for the root node. We want to select the attribute that is most useful in classifying examples.
For example, in our vertebrate example we might have the possible attributes

- Is it warm blooded?
- Is it cold blooded?
- Is it a mammal?
- Is it a bird?
- Is it a fish?
- Is it a reptile?
- Is it an amphibian?
- Is it a human?
- etc.

Suppose our training set consists of 100 vertebrates consisting of

53 mammals including 2 humans, 14 fish, 7 amphibians, 26 birds

Now suppose our root or first query is **Is it a fish?** What percentage of the training set is correctly identified with this question? \( \frac{14}{100} = 14\% \)
Now suppose our root or first query is *Is it a mammal?* What percentage of the training set is correctly identified with this question? \( \frac{53}{100} = 53\% \)

So we say that the choice of *Is it a mammal?* for the root question is a better than the choice than *Is it a fish?* The attribute which correctly identifies the most elements of the training set is chosen for the root node.

Of the questions above, which do you think is the worse choice for the root question?

After the root node is chosen we list the possible descendants. In the vertebrate example, the results are either *yes or no*.

Then for each subsequent node we repeat the process that was used in selecting the root node.

To implement the algorithm one computes a number which gives a measure of the worth of an attribute. This is called the *information gain* and is a statistical property which we won’t go into.
Decision Tree Learning for “Play Tennis” Example

Suppose we are trying to decide what attribute to use for the root node in the tennis example and our possible choices are:

Outlook, Temperature, Humidity, Wind

We want to use a training set to decide which of the four attributes gives the best outcome, i.e., which is the most useful for correctly identifying the data in a training set.

Suppose we have the following Training Set with 14 data points.
<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>yes</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>no</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>no</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>no</td>
</tr>
<tr>
<td>D7</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>yes</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>no</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>no</td>
</tr>
</tbody>
</table>
For the 14 data items we have that 9 result in a classification of “Play Tennis” and 5 result in a classification of “Don’t Play Tennis” which we write in the shorthand notation Yes:9, No:5.

Now for each of the four attributes which are candidates for the root node we look at their breakdown and see how good an indicator each is. For example, if we summarize the results for an attribute and get Yes:5, No:5 then it’s not a good indicator because each outcome is just as likely. On the other hand, if we summarize the results for an attribute and get Yes:9, No:1 then it is a good indicator.

For our example, if an attribute is a really good indicator then every time it occurs, then it should always indicate “Play Tennis” OR always indicate “Don’t Play Tennis”. For example if a Weak Wind occurs 6 times and always classifies as “Play Tennis” then it is a good indicator (Yes:6, No:0); equivalently it could always classify as “Don’t Play Tennis”, i.e., Yes:0, No:6 and it would be a good indicator. But if it classifies 3 as “Play Tennis” and 3 as “Don’t Play Tennis”, then it is no better than flipping a coin for the outcome.

In our actual training set for a Strong Wind we have 3 positives and 3 negatives so we describe the set as Yes:3, No:3 whereas for a Weak Wind we have 6 positives and
2 negatives which we describe as Yes:6, No:2. This tells us that the Weak Wind is a better indicator of the correct outcome than a Strong Wind.

We do this for each attribute.

**Outlook**

- Sunny: Yes:4, No:1
- Overcast: Yes:3, No:0
- Rain: Yes:2, No:4

**Temperature**

- Hot: Yes:3, No:1
- Mild: Yes:3, No:3
- Cool: Yes:3, No:1

**Humidity**

- High: Yes:4, No:3
- Normal: Yes:5, No:2

**Wind**
Using statistics, one can compute a “numerical gain” (which is a number between 0 and 1) for each of the four attributes. The larger the number, the more “gain” from using that attribute. This formula involves using logarithms but we will not go into it here.

With our data set Outlook seems to be the best choice because “Sunny” predicts play 4 out of 5 times, “Overcast” always predicts play and “Rain” predicts don’t play 2 out of 3 times. Of course this choice may be different with a different training set. So we choose “Outlook” as our root node. In addition, since in the training set one always plays tennis on overcast days we can complete that leaf.
There are 3 Outlooks: Sunny, Overcast, Rain but we don’t have to select any option for overcast since we always play. For each of the other two we have to decide whether Temperature, Humidity or Wind is a better indicator. We continue in this way to build a Decision Tree.

We look at Sunny first and try to decide if Temperature, Humidity or Wind is a better indicator. So we only look at the 5 entries in our training set where the Outlook is Sunny and see how good an indicator the temperature is. The 5 entries are D1, D2, D8, D9, and D11. We do the same for Humidity and Wind. Summarizing for these 5 entries we have:
<table>
<thead>
<tr>
<th>Day</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hot  Mild  Cool</td>
<td>High  Normal</td>
<td>Strong  Weak</td>
</tr>
<tr>
<td>D1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>D2</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>D8</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>D9</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>D11</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Summary:

**Temperature**  Hot: Yes:1, No:1, Mild: Yes:2, No:0, Cool: Yes:1, No:0

**Humidity**  High: Yes:2, No:1, Normal: Yes:2, No:0

**Wind**  Strong: Yes:1, No:1, Weak: Yes:3, No:0

So Temperature or Wind look equally strong. We choose Wind. Note that if the wind is weak on a sunny day, then we always play tennis. We have extended our decision tree as follows
Now for the two sunny days (D2 and D11) with a strong wind we play one day and not the other. Should we check the humidity or temperature next? Our training set is so small that it really doesn’t matter in this case.
We have to repeat our calculations when the outlook is rain. There are 6 days (D4, D5, D6, D7, D10 and D14) when it rains. We tabulate the results are before and then summarize to see whether temperature, humidity or wind is the best indicator. Note that just because wind was a good indicator on sunny days, it may not be so on rainy days.

<table>
<thead>
<tr>
<th>Day</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hot</td>
<td>Mild</td>
<td>Cool</td>
</tr>
<tr>
<td>D4</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>D5</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>D6</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>D7</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>D10</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>D14</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Summary:

**Temperature**  Hot: none  Mild: Yes:0, No:3, Cool: Yes:2, No:1

**Humidity**  High: Yes:0, No:2, Normal: Yes:2, No:2
Wind Strong: Yes:2, No:2, Weak: Yes:0, No:2

In this case the temperature is probably the best indicator. For a mild temperature on a rainy day 3 out of 3 times they didn’t play tennis and on cool days 2 out of 3 times they did play tennis. For normal humidity it was a toss up as well as for strong wind. So we choose temperature.

Now we can’t do anything else to the leaf where it is rainy and hot because we have no training data for that scenario; our set is just too small. For mild, rainy days we don’t have to test anymore because for those 3 days in the training set we never play tennis. But we would have to test for cool rainy days because two out of three days we play tennis. Again, it is a toss up between testing for humidity and wind because all the days have normal humidity (so 2 out of 3 we play) and all the days have strong wind. Either will work. Our training set is too small. Remember that in practice there is a complicated mathematical formula which computes an information gain for each attribute so it is usually clear which is the best option, plus it is easy to implement and is not a heuristic.

Remember that what may be the best next question after Outlook may differ for Sunny, Overcast or Rainy. We don’t have to choose the same next question.
A pattern recognition problem where a Decision Tree Classifier is typically used is the identification of web spam.

What is web spam?

Artificially created pages are injected into the web in order to influence the results from search engines to drive traffic to certain pages.

Motivation: financial, political, just for fun, etc.

Clearly, when we do a web search we don't want to have to filter out these nonsense pages, we want the browser to do it for us.
This page would be identified as web spam.

Notice the text that doesn’t really make sense and how words like “cash” are repeatedly used so if one searches for “cash” this page will get “hits”.
• We want to train the algorithm with web pages that have been identified manually as spam or non-spam. Recall that pattern recognition is “easy” for humans.

• The hope is that patterns emerge which help to create an accurate Decision Tree.

• What type of content should we look for to determine whether a page is spam or not?

1. **Number of words on the page.**

   A popular practice in creating spam web pages is “keyword” stuffing; that is, they contain words which are irrelevant to the rest of the page. The hope is that the more “keywords” on the page, the more “hits” the page will get.

   One study by scientists at Microsoft showed that over 50% of valid web pages contain 300 words or less and only about 13% contain 1000 words. However, the correlation for the page being spam for a large number of words (some as large as 3500) is not by itself a good heuristic.
2. **Number of words in the title.**

Some search engines give extra weight when a keyword is in the title so when someone is creating a spam web page one strategy is to pack the title with keywords. In the same Microsoft study the authors considered the prevalence of spam relative to the number of words in the title page which ranged from 1-50 words. Titles with a length of $> 24$ words were more likely to be spam than non spam.

3. **Several other more technical criteria were used by Microsoft to identify web spam.**

The algorithm is quite good because we rarely encounter spam pages in our searches.
Use the given decision tree to answer the first 5 questions. The attributes are color, shape, diameter size and whether it contains stones/pits. The outcomes are the following fruits: lemon, banana, grapefruit, cherry, grape, apple, watermelon.
1. If a fruit is green with a diameter greater than 6 inches, what is it?
2. If the fruit is red with a diameter less that 2 inches and contains a stone, what is it?
3. If a round fruit is neither red nor green and has a diameter greater than 4 inches, what is it?
4. If a fruit is red and is greater than 2 inches in diameter, what is it?
5. If a round fruit is neither red nor green and has a diameter less than 4 inches, what is it?
6. The first node of the decision tree is called a leaf.
7. A decision tree should include all possible scenarios in your training set.
8. Suppose you have a training set for whether you should play tennis. If the attribute “a strong wind” predicts you play tennis 2 out of 7 times and the attribute “a sunny date” predicts you play tennis 4 out of 7 times, then
   (a) a strong wind is a better predictor
   (b) a sunny day is a better predictor
   (c) both attributes are equally good/bad
9. Suppose you have a training set for whether you should play tennis. If the attribute “a weak wind” predicts you play tennis 4 out of 8 times and the attribute “a sunny date” predicts you play tennis 3 out of 6 times, then

(a) a weak wind is a better predictor
(b) a sunny day is a better predictor
(c) both attributes are equally good/bad

10. Suppose you have a training set for whether you should play tennis. If the attribute “high humidity” predicts you play tennis 2 out of 5 times, the attribute “rainy” predicts you play tennis 1 out of 4 times and the attribute “cold temperature” predicts you play tennis 3 out of 4 times then

(a) high humidity is the best predictor
(b) rainy is the best predictor
(c) cold temperature is the best predictor
(d) both rainy and cold temperature are equally good/bad
1. Suppose you and your friends in Exploratory Studies want to use the Decision Tree above to help you decide what major to choose.

(a) If someone's responses are yes, no, no (in order), what major(s) should they consider?
(b) If someone’s responses are yes, no, yes (in order), what major(s) should they consider?
(c) If someone’s responses are yes, yes, no (in order), what major(s) should they consider?
(d) If someone’s responses are yes, yes, yes (in order), what major(s) should they consider?

2. Now suppose that we want to create a Decision Tree to predict a loan applicant’s risk of defaulting. The training set consists of individuals who have previously received a loan. Suppose we know three pieces of information about each person in the training set:

(i) if they have been at their job for less than 2 years at time of loan;
(ii) if they missed any payments on this or other loans;
(iii) if they defaulted on the loan.

We want to determine a Decision Tree using the two attributes (i) and (ii) to predict an outcome (default or not default) for a new applicant. We summarize the training data in the table below.
(a) We want to summarize the two attributes (i) & (ii) with the outcome so we can decide which is the best query for the root node.

**Attribute: years on job**
- **< 2 years at current job** (4 total - A2, A6, A9, A10) : 1 Default, 3 No default
- **> 2 years at current job** (6 total - A1, A3, A4, A5, A7, A8) : 2 Default, 4 No default

**Attribute: Did they miss payments?**
Missed payments: (3 total - A5, A7, A8) 2 Default, 1 No default
Didn't miss payments: (7 total - A1, A2, A3, A4, A6, A9, A10) 1 Default, 6 no default

(b) Based upon the results in (a) what do you think the root question should be? Why?

(c) Make a Decision Tree with the root question from (b) and your second attribute.

(d) We now want to use the Decision Tree from (c) to see the fraction of the entries in the training set that are good risks.

   i. If the applicant answered “No” to both questions. (Note: there are only 3 applicants with these responses so your answer will be 1/3, 2/3 or 3/3).

   ii. If the applicant didn’t miss any payments but has been on the job < 2 years

(e) Now add a box under the last leaves of your decision tree, indicating the fraction of applications that are good risks if they answered the questions in that manner. Note that no one answered yes to both questions so there is no data for that case due to the small training set.
Neural Net algorithms are based on how our brain processes information.

In 1943 a neuroscientist and a logician developed the first conceptual model of an artificial neural network.

Neural net algorithms do NOT model how our brain works but they are inspired by how our brain works and designed to solve certain kinds of problems.

The human brain contains approximately 100 billion nerve cells called neurons.

Each neuron is connected to thousands of other neurons and communicates with them through electrochemical signals.

Signals coming into a neuron are received via junctions called synapses which are located at the end of branches of the neuron called dendrites.

The neuron continuously receives signals from these inputs and then performs a little bit of magic. What the neuron does (in a very simplified explanation) is sum up its inputs in some way and then, if the end result is greater than some
threshold value, the neuron “fires”. It generates a voltage and outputs a signal along something called an axon.

- Since the output of the neuron is “fire” or “don’t fire” it is a binary output which can be imitated on a computer easily.

A neural network is a connectionist computational system. The algorithms we have encountered are serial in the sense that the program executes the first line of code, moves to the second, etc in a linear fashion. A true neural network does not follow a linear path but rather information is processed collectively in parallel throughout a network of nodes (neurons).
• **Neural network** algorithms are made up of many artificial neurons; the number needed depends on how difficult the task is. Our first concrete example will have only a single neuron.

• Each neuron can have **multiple inputs** but only a **single output** which is binary.
As before, we want to train the algorithm with a set of training data. How can this be accomplished?

Each input has a weight which we adjust; the weight is just a number typically scaled between -1 and 1.
Initially we guess the value of the weights and then the algorithm adjusts them during the training portion of the algorithm.

- As each input enters the neuron its value is multiplied by its weight.
- These values are summed for all inputs.
- If the summed valued is \( \geq \) threshold (such as 0 or 1) then it “fires”; i.e., it gives a positive output.
- If the summed valued is \( < \) threshold then it does NOT “fire”; i.e., it gives a negative output.
- If the output of the neuron matches the correct output in the training set, then we don’t modify the weights.
- If the output of the neuron does NOT match the correct output in the training set, then we modify the weights.
- The way they are modified will be discussed in an example with one neuron.
Simple explanation of neural nets - Jesus Suarez youtube video “Intro to Neural networks”
Simple Example to Predict Outcome on Exam

As a simple example, suppose you want to write a program to predict how you will do on the final test in a course based on how you have done on previous tests.

Assume that you have recorded the number of hours you spent preparing for the exam and the number of hours you slept the night before the exam.

So there are two inputs (the number of hours studying for exams and number of hours of sleep the night before) and a single output, the predicted score of the exam.

Now if we want the output to be a letter grade, then it is a Classification Problem (since the only options are: A, A-, B+, B, B-, etc.) and if we want it to be a numerical grade then it is basically a Regression Problem like our example of predicting the listing price of a house.

Suppose you have recorded the following information about each test.
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Letter Output</th>
<th>Numerical Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours Prep</td>
<td>Hours Sleep</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>C</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>B-</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>A</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>B+</td>
</tr>
</tbody>
</table>

Notice that the instructor appears to be scaling the grades and a different scale is used for each exam. Realistically you care about the letter grade so it is a Classification Problem.

Suppose your goal for the next exam is to get a “B+” or better.

Assume that it is 7 pm on the night before a 10 a.m. exam and you are about to start studying. Let’s look at some options you have.

1. Study from 7 pm until midnight (with two 30 minute breaks)  \(\Rightarrow\)  4 hours preparation and get 7 hours of sleep
2. Study from 7 pm until 2 am (with three 30 minute breaks)  \(\Rightarrow\)  5.5 hours preparation and get 6 hours of sleep
3. Study from 7 pm until 4 am (with four 30 minute breaks) \[ \implies \] 7 hours of preparation and get 4 hours of sleep

From looking at the recorded data, we recognize the pattern that more hours of preparation yields better test results and the amount of sleep seemed secondary.

However, to write a program to recognize this pattern we would **train** the algorithm with the four test results and then try to use these to **predict** whether the grade on the test is at least a B+.
To understand how a neural network functions, we tackle a simplified problem where we only use a single neuron in the network.

However you can have one or more inputs but a single binary output.

This is called a **Perceptron – A Neural Net with One Neuron**.
If we input data like the number of hours studied, then how does the algorithm learn?

We allow each input to have a weight which indicates the relative importance of each input. For example, we might believe that studying for an extra hour would improve the grade more than sleeping for an extra hour so we would have a larger weight for studying than for sleeping. Weights are usually scaled so that they lie within a certain range such as \([-1, 1]\). So we make an initial guess for the weight for each input and after the first data in the training set, we modify the weight.

Now for each value of the input we have a weight and because we only have one
“neuron” we do a single computation and then give a binary output. Recall that binary is just yes/no or 0/1, +/-, fire/don’t fire, etc.

Let’s look at a geometric example and see how this might work.

Suppose we have the line $y = x + 1$ which we know has slope 1 and passes through the origin (0,1). Our goal is to predict whether a given point lies below the line or above/on the line. If we just do this randomly then the guess will be right approximately half of the time. We want to train the algorithm so that it will accurately predict whether a point is above or below the line.

Is the red point above or below the line?
After we train the algorithm we want to give the algorithm a new point \((x, y)\) and have it accurately predict whether it is above or below the line.

So we actually have two inputs: the \(x\)- and \(y\)-coordinate of the point. Also each input has a weight.

The output is binary; we take positive to mean above/on the line and negative to mean below the line. The question is, *what do we calculate to determine its sign?*

Let \(x, y\) be the coordinates of the point and \(w_x, w_y\) be their weights, respectively. We multiply the first input \(x\) (the \(x\)-coordinate of the point) by its weight \(w_x\) and multiply the second input \(y\) (the \(y\)-coordinate of the point) by its weight \(w_y\). Then we sum the two values to get

\[
(w_x \times x) + (w_y \times y)
\]

to get a number. But this can’t be our output because we said our output is binary. We can simply take the sign of the output which means we are taking a threshold of zero. We take \(+1\) if the sum is \(\geq 0\) and \(-1\) if it is \(< 0\).
However, there is a small problem with this. The point \((0, 0)\) will ALWAYS satisfy \(w_x x + w_y y = 0\) and so we say the point is on or above the line. But this can’t be true for every possible line. For example, consider \(y = x + 1\) where \((0, 0)\) lies below the line but \(w_x x + w_y y = 0\) because \(x = 0, y = 0\) for the point \((0,0)\).

What can we do to fix this? We simply add another input called the bias with its own weight. The bias is typically taken to be fixed at one but its weight is updated during the training set. We won’t go into this here but in the results reported we have used a bias for an input.
How do we get the training set for this problem?

Let’s say we have the line $y = x + 1$ and we pick a random point. For the training set we need to know whether this point lies above or below the line. How do you know this?

So now we can generate a training set by picking a point $(x, y)$ and determine if it lies above or below line.
We put in the first point in our training set and compute the sign of the sum $w_x \times x + w_y \times y$. Let’s say that it is $\geq 0$ which predicts the point is above or on the line when in actuality the point lies below the line, i.e., the sum should be $< 0$. So our algorithm has predicted incorrectly.

What does this mean? It means that our weights are wrong so we have to modify them. How can we do this?

Our intuition says:

- If the sum should be $< 0$ but it is $\geq 0$ this means our weights are too large so we have to decrease them.
- If the sum should be $\geq 0$ but it is $< 0$ this means our weights are too small and we have to increase them.

We need to compute an error but this is a little strange because the output is $\pm 1$.

It would be easy to calculate an error in our numerical grade prediction model using linear regression. If we predicted a grade of 87 and the actual grade on the test was 82 then our error would be $87 - 82 = 5$, i.e., we were 5 points high on the prediction.
If we predicted a grade of 87 and the actual grade on the test was 92 then our error would be $87 - 92 = -5$, i.e., we were 5 points low on the prediction. If we predicted a grade of 87 and the actual grade on the test was 87 then our error would be zero. So in this case our error could be anything between -100 and 100.

When our output is $\pm 1$ the error can only take on the values 0, -2, 2. To see this, look at the following 3 cases for the line $y = x$

<table>
<thead>
<tr>
<th>Point</th>
<th>Actual</th>
<th>Prediction</th>
<th>Error = Actual - Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,-2)</td>
<td>-1(below)</td>
<td>-1(below)</td>
<td>0</td>
</tr>
<tr>
<td>(3,5)</td>
<td>1 (above)</td>
<td>-1(below)</td>
<td>2</td>
</tr>
<tr>
<td>(4,2)</td>
<td>-1(below)</td>
<td>1 (above)</td>
<td>-2</td>
</tr>
</tbody>
</table>

If the error is zero then we do not modify the weights because the prediction was correct.

If the error is not zero then we must modify the weights This means that we calculate
\[ \text{weight}_{\text{new}} = \text{weight}_{\text{old}} + \text{change}. \]

What can we use to modify the weights?

If we predicted the point was below the line (sum < 0) but it was above the line then we have to increase the weights so the change should be positive. This corresponds to the second point in our table where we have a positive error of +2. Likewise if we predicted the point was above/on the line (sum \( \geq 0 \)) but it was below the line then we have to decrease the weights so the change should be negative. This corresponds to the third point in our table where we have a negative error of -2.

Thus the sign of the error is positive when we want to increase the weights and negative when we want to decrease them. So if we use error as a term in the amount to change the weights then we have the correct sign.

We don’t want to just add the error but rather the error times some term. Remember that the calculation the artificial neuron does is multiply each weight times its input and then sum them. So really we want to modify each weight by multiplying the
error times the corresponding input. In practice, we also want to add a scaling factor which is often called the learning rate so that we don’t overcorrect. We have

\[ \text{weight}_{\text{new}} = \text{weight}_{\text{old}} + \text{error} \times \text{input} \times \text{factor} \]

Typically the learning rate is fairly small, e.g., 0.01 which we take in our examples.
Structure of the Perceptron Algorithm for determining if a point is above or below a given line

**Step 0 - Input fixed information:** for given line enter slope and $y$-intercept; enter an initial guess for the weights $w_x, w_y$ which we take to be in $[-1, 1]$; enter learning rate.

**Step 1 - Training Part:** For $i = 1, 2, \ldots, N$

(i) generate random point $(x, y)$

(ii) determine if $(x, y)$ lies below line $\Rightarrow$ Actual $= -1$; otherwise Actual $= +1$

(iii) compute the term $t = x \times w_x + y \times w_y$

(iv) if $t \geq 0$ then Predicted value $= +1$; if $t < 0$ then Predicted value $= -1$

(v) compute error $= \text{Actual} - \text{Predicted}$

(vi) update weights by formulas

\[
\begin{align*}
    w_x &= w_x + \text{error} \times x \times \text{learning rate} \\
    w_y &= w_y + \text{error} \times y \times \text{learning rate}
\end{align*}
\]
Step 2 - Prediction part: Use the algorithm to predict whether the point lies below or above/on the line for \( J \) new points using the final weights \( w_x, w_y \) from the training set. For \( j = 1, 2, \ldots, J \)

(i) input \( j \)th point \((x, y)\)

(ii) calculate term \( t = x \times w_x + y \times w_y \)

(iii) if \( t \geq 0 \) point lies above/on the line; if \( t < 0 \) then point lies below line.
Numerical Results for Training Data

Fixed information:

Random initial weights: \( w_x = -0.9951 \) and \( w_y = 0.1336 \)

Slope of line = 1; \( y \)-intercept = 0

Learning rate = 0.01
<table>
<thead>
<tr>
<th>Point</th>
<th>Term $t$</th>
<th>Actual output</th>
<th>Predicted output</th>
<th>Error</th>
<th>New Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.98, -1.06)</td>
<td>-1.18</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-0.995 0.134</td>
</tr>
<tr>
<td>(-0.15, -3.41)</td>
<td>0.63</td>
<td>-1</td>
<td>+1</td>
<td>-2</td>
<td>-0.992 0.202</td>
</tr>
<tr>
<td>(-3.96, -1.22)</td>
<td>4.59</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
<td>-0.992 0.202</td>
</tr>
<tr>
<td>(-1.26, -2.26)</td>
<td>1.70</td>
<td>-1</td>
<td>+1</td>
<td>-2</td>
<td>-0.967 0.247</td>
</tr>
<tr>
<td>(-2.93, 3.20)</td>
<td>4.52</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
<td>-0.967 0.247</td>
</tr>
</tbody>
</table>
Error $= 0 \implies$ don’t change weights

Actual $= -1$ (below)

$\bullet$ (1.98,-1.06)

X

Guess $= -1$ (below)
Error = 2 \implies \text{increase weights}

X

Guess = +1 (above)

Actual = -1 (below)

(-0.15, -3.41)
How good are our predictions after the training step?

Assume our training set consists of 10 points and then we use these weights to predict 100 new random points. What percent of the points do we predict correctly? 78% for the random points we chose in the program.

Recall that if we just guessed whether the point was above or below the line we would be right about 50% of the time.

What can we do to improve this result?

If possible, increase the number of points in the training set. The following table gives the accuracy of predicting 100 new points correctly as a function of the size of the training set.

<table>
<thead>
<tr>
<th>Training Set Size</th>
<th>Percent Correct for new points</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>78%</td>
</tr>
<tr>
<td>25</td>
<td>83%</td>
</tr>
<tr>
<td>50</td>
<td>83%</td>
</tr>
<tr>
<td>100</td>
<td>88%</td>
</tr>
</tbody>
</table>
As you can see, the accuracy is increasing very slowly. Unfortunately, we do not always have the luxury of making our training set arbitrarily large.

Another approach would be to go through the data set repeatedly to adjust the weights until the algorithm predicts the correct answer 100% of the time for the training set. Note that this doesn’t guarantee that all additional points will be predicted exactly but it should work well. In neural net lingo these iterations through the data set are called epochs.

In the following table we went through the data set repeatedly until the algorithm got 100% correct in the training set and then predicted 100 new random points.

<table>
<thead>
<tr>
<th>Training Set Size</th>
<th>Number of Epochs</th>
<th>Percent Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>90%</td>
</tr>
<tr>
<td>25</td>
<td>15</td>
<td>97%</td>
</tr>
<tr>
<td>50</td>
<td>13</td>
<td>100%</td>
</tr>
</tbody>
</table>

So a combination of a good sized training set and iteration typically works best.
How would we modify our description of the algorithm to incorporate this?

**Step 0 - Input fixed information:** for given line enter slope and $y$-intercept; enter an initial guess for the weights $w_x, w_y$ which we take to be in $[-1, 1]$; enter learning rate

**Step 1 - Training Part:** For $i = 1, 2, \ldots, N$

(i) generate random point $(x, y)$
(ii) determine if $(x, y)$ lies below line $\implies$ Actual $= -1$; otherwise Actual $= +1$
(iii) compute the term $t = x \times w_x + y \times w_y$
(iv) if $t \geq 0$ then Predicted value $= +1$; if $t < 0$ then Predicted value $= -1$
(v) compute error $= \text{Actual} - \text{Predicted}$;
   - for $i = 1$ if error $= 0$ set error_flag $= 0$; if error $\neq 0$ set error_flag $= 1$
   - for $i > 1$ if error $\neq 0$ set error_flag $= 1$
(vi) update weights by formulas

$$w_x = w_x + \text{error} \times x \times \text{learning rate}$$

$$w_y = w_y + \text{error} \times y \times \text{learning rate}$$
If error\_flag = 1 repeat Step 1

**Step 2 - Prediction part:** Use the algorithm to predict whether the point lies below or above/on the line for \( J \) new points using the final weights \( w_x, w_y \) from the training set. For \( j = 1, 2, \ldots, J \)

(i) input \( j \)th point \((x, y)\)

(ii) calculate term \( t = x \times w_x + y \times w_y \)

(iii) if \( t \geq 0 \) point lies above/on the line; if \( t < 0 \) then point lies below line.
Neural Nets as a Linear Classifier
A more practical example of using Neural Nets is as a linear classifier. As in the figure, assume you want to classify a point as red or blue. The goal is to find a line which separates the points so that any point which lies below the line is one color and points lying above the line are the other color. This is called the decision boundary. Of course, it may not be possible to find a line which separates all the training data 100% accurately. This is in contrast to the previous example where we are given the line and just want to determine if a random points lies above or below the line.

To determine a line we know that we need the slope \( m \) and the \( y \)-intercept \( b \)

\[
y(x) = mx + b
\]

So we start with an initial guess for \( m, b \) and adjust them so that we move the separating line in a direction that lowers the classification error.
If our training data is linearly separable, i.e., there exists a line where all blue points lie above it and all red points lie below it, then it was proved that the Perceptron approach will produce a solution to this problem.

However, the convergence is very slow.
Here we look at an extension of the facial recognition example given in your text.

In this example the training set consists of photographs of different individuals; typically there are multiple images of each individual.

The goal is to use the training set to train the Neural Net Algorithm to identify various characteristics in the photographs. In your text an example is described where the goal is to identify whether the person in the photograph is wearing sunglasses. Here we will try to identify a particular person. In both cases the output is binary.

The algorithm used contains many artificial neurons unlike the Perceptron example which used only one and is a much more sophisticated algorithm.

On the following page are some sample images from the training set. Notice how some images of individuals are all face forward and images of other individuals are all profile shots. Also note that the quality of the images is not very good.
• An application of this might be on Facebook where images of yourself are identified on other friends’ pages.
We want to train the algorithm to identify an image of a particular person (Mr. Glickman). There are 4 images of Glickman in the training set.

![Images of Glickman](image1.jpg) ![Images of Glickman](image2.jpg) ![Images of Glickman](image3.jpg) ![Images of Glickman](image4.jpg)

As before, if we train the program by going through the training set only once to modify the weights, then the results are not very good. What we did in the Perceptron example was to repeatedly go through the training set until the model got 100% of the answers correct. Recall that in Neural Net lingo these iterations are called epochs.

In the table below we train the program using a given number of epochs.

Notice that after one epoch it identifies Glickman approximately 94% of the time and it takes 6 epochs for the program to correctly identify each photograph in the training set.

The last column in the table indicates how confident we should be in the answer.
For example, it is possible to get 100% correct without training but of course that would be pure luck and so the confidence level should be low which means we expect a large error. Notice that when it only got 14% of the photographs correct it has a much larger error whereas when it got 100% correct the confidence level is high so the expected error is small.

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Percent Correct.</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.2857</td>
<td>0.109234</td>
</tr>
<tr>
<td>1</td>
<td>94.2857</td>
<td>0.0139558</td>
</tr>
<tr>
<td>2</td>
<td>94.2857</td>
<td>0.0123471</td>
</tr>
<tr>
<td>3</td>
<td>94.2857</td>
<td>0.0109007</td>
</tr>
<tr>
<td>4</td>
<td>94.2857</td>
<td>0.00893908</td>
</tr>
<tr>
<td>5</td>
<td>97.1429</td>
<td>0.00706015</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>0.00571391</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>0.00475462</td>
</tr>
</tbody>
</table>

Now that the algorithm has been trained using 7 epochs and the confidence is high (i.e., predicted error is small), we want to test it on a new set of images. The new
set contains 77 images and the results are that it gets 94% correct with a predicted error of 0.011.

We would like to think that it should identify all the images correctly in the new set so let’s see what is going wrong. Actually it fails to identify the following 4 photographs of Glickman.

Why couldn’t the algorithm identify these photos as Glickman?

Probably because none of the images of Glickman in the training set were profile shots.

Suppose we add two of these images to the training set. We now get the following results which compare to the table above.
Now when we test our algorithm on the same set of images as before it identifies 100% correctly.

This just shows us that the algorithm is only as good as the training set!
1. In a neural network algorithm one tries to exactly mimic how the brain works.
2. In a Perceptron network you can only have one input.
3. In a Perceptron network you can only have one output.
4. The output of a neuron in a neural network is binary.
5. In a neural network you typically only goes through the training set once adjusting the weights.
6. In a neural network algorithm, different training sets may give different results.
7. Using a neural network algorithm for facial recognition, one can always identify a new photo as long as there is at least one photo in the training set of that individual.
8. Suppose we use our Perceptron algorithm to determine if a point lies above/on or below the line \( y = 2x - 3 \). If the point \((4,4)\) is in our training set does it
(a) lie above the given line?
(b) lie below the given line?
(c) lie on the given line?

9. Suppose the point \((1, 3)\) is in the training set and we know that it lies above the line and the Perceptron algorithm predicts that it lies below the line. Then the error is
   (a) 0
   (b) +1
   (c) +2
   (d) -2
   (e) -1

10. Suppose the point \((2, 2)\) is in the training set and we know that it lies above the line but the algorithm predicts that it lies below the line. Assume that if the output of the neuron is +1, then the point is above the line and if it is -1, then the point is below the line. Then
    (a) the weights should stay the same
    (b) the weights should decrease
(c) the weights should increase
Classwork for Neural Nets

Training data: (4,2) (1,-3) (0,-2) (2,2)

Set the initial guesses for the weights to be: weight $w_x = 1/4$ for the $x$–component and weight $w_y = -3/4$ for the $y$-component. Set the learning rate = 0.01

1. Draw the line $y = x/2 - 1$ on the graph. What is the slope of the line?
2. For each training point $(x, y)$ in the set compute the following and fill in the table below.
   (a) Indicate the point on the graph and whether it lies above or below the line. Assign the value +1 if it lies above the line and -1 if it lies below the line. This is the Actual output.
   (b) Compute
       $$t = x \cdot w_x + y \cdot w_y$$
   (c) Determine predicted value for this point. Recall that if $t \geq 0$ then the predicted value = +1 and if $t < 0$ then predicted value = -1.
(d) Compute the error for this point

\[
\text{Error} = \text{Actual value} - \text{Predicted value}
\]

(e) Update the weights from the formulas

\[
w_x = w_x + \text{error} \cdot x \cdot \text{learning rate}
\]

\[
w_y = w_y + \text{error} \cdot y \cdot \text{learning rate}
\]

(f) Put your answers in the table below.

<table>
<thead>
<tr>
<th>Point</th>
<th>Term ( t )</th>
<th>Actual Output</th>
<th>Predicted Output</th>
<th>Error</th>
<th>Weights ( w_x )</th>
<th>( w_y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4,2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,-3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0,-2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2,2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. You have now trained the algorithm and we want to see how well it does on the test data. By training, we mean that you have computed new weights \( w_x, w_y \).
Here is the test data:

$\left(2, \frac{1}{2}\right), \left(-1, -3\right), \left(4, 1\right)$

For each point determine the following using the weights $w_x, w_y$ that you determined from training the algorithm. Put your answers in the table below.

(a) Does the point actually lie above the line or below the line?

(b) Use your algorithm to predict whether the point lies above the line or below the line. Recall that to do this we compute $t = x \cdot w_x + y \cdot w_y$ and if $t \geq 0$ we say it lies above the line and if $t < 0$ then it lies below the line.

(c) Compare the predicted and actual values.

<table>
<thead>
<tr>
<th>Point</th>
<th>Does point lie above or below line</th>
<th>Term $t$</th>
<th>Prediction for point</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\left(2, \frac{1}{2}\right)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\left(-1, -3\right)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\left(4, 1\right)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. What fraction (or percent) of the new data did the algorithm predict correctly? What can be done to improve the algorithm's ability to accurately predict whether a point lies above or below the line?