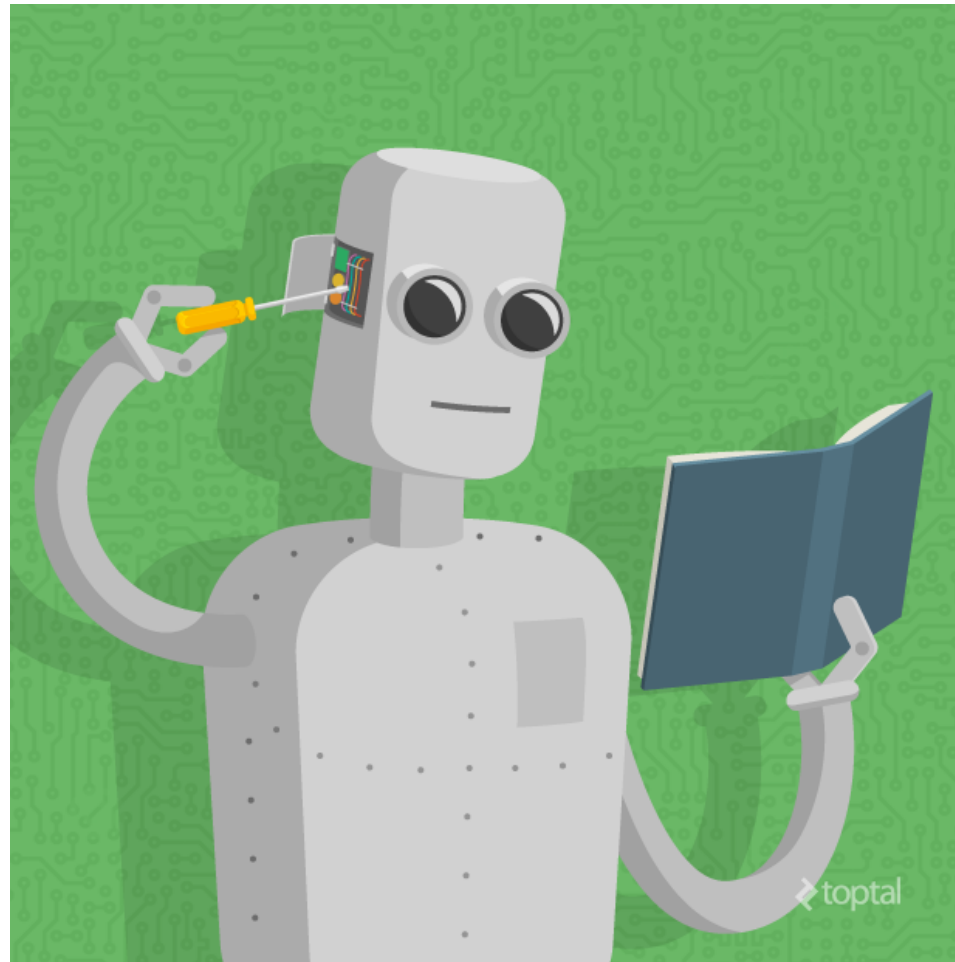

PART V - MACHINE LEARNING



Goals for Part V

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**.
- Loosely speaking, **ML is the field of study that gives computers the ability to learn without being explicitly programmed.**
- 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.

IBM's Watson



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.

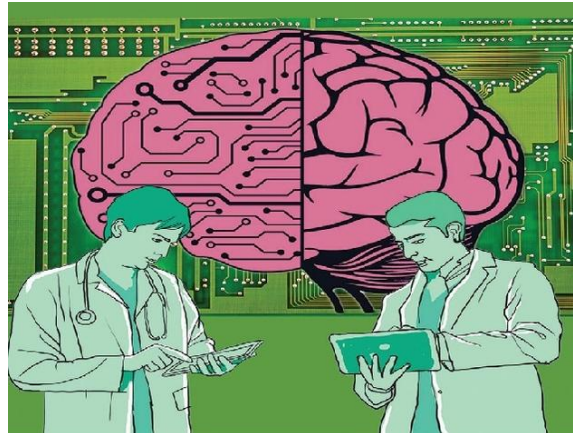
What type of questions do you think Watson had the most trouble with?

Watson Oncology

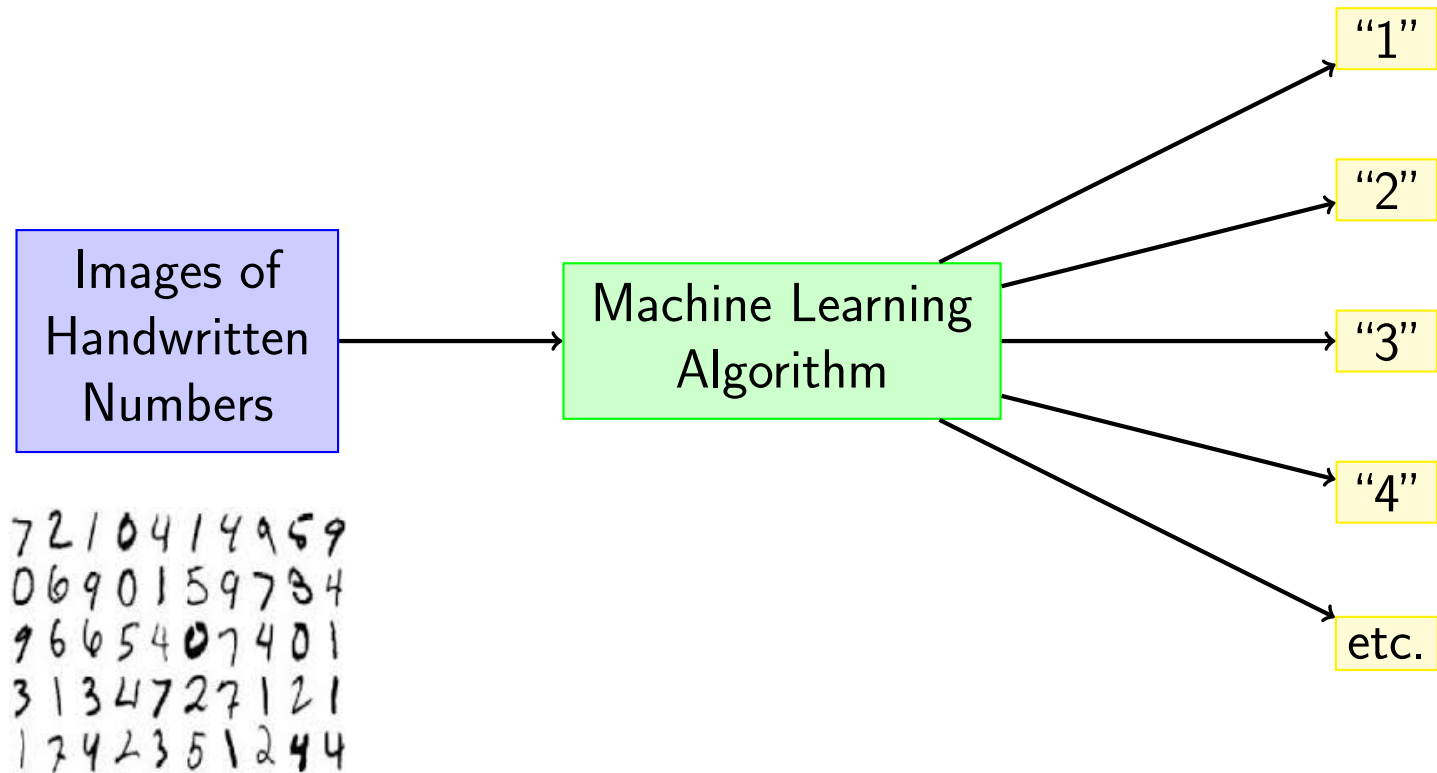
Oncology research grows at a rate of approximately 8000 academic papers a day. Too many for even the best of physicians to stay abreast of.

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.

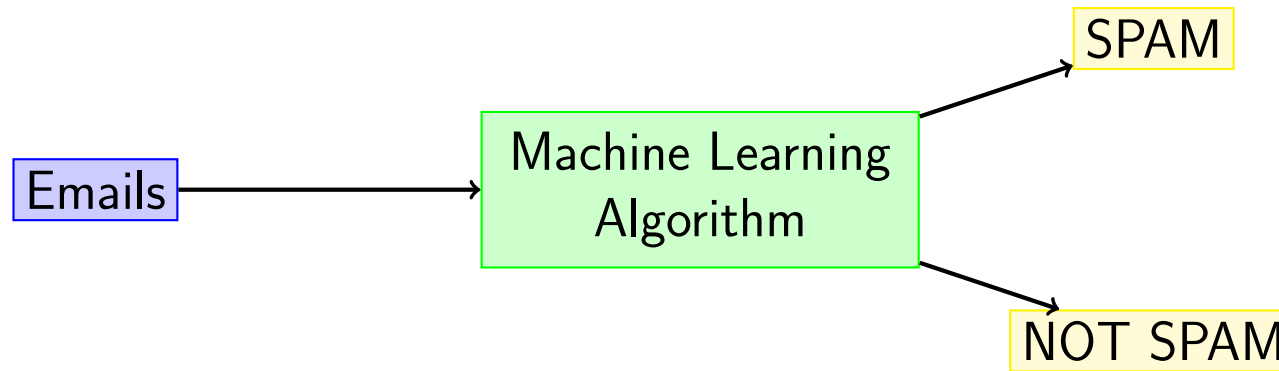
Watson is being used to suggest treatment options for patients with rare forms of cancer. Available for use in U.S. January, 2017.



In What Other Real-World Problems Might ML be Useful?



Training an algorithm to read a zip code



Classifying email messages

- Autonomous Land Vehicles (ALVINN); Robotics Institute, CMU



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.

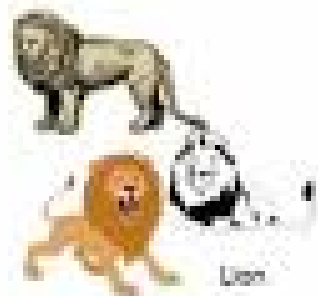
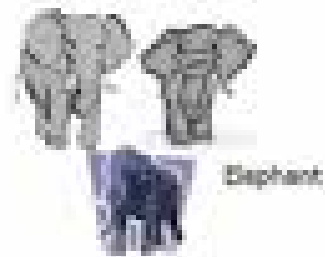
Often the prediction is a **classification** such as a zipcode or identifying an email as spam or not spam.

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.

Example(Supervised Learning)

edureka!



For now we will concentrate on Supervised ML.

Simple Example of Supervised ML



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.

Sq. Ft	Bedrooms	Baths	Pool	Garage	Lot Size	Neighborhood	Listing Price
2900	4	4	no	3 car	1 acre	NE	\$589,000
1895	3	2	no	carport	5 acres	NW	\$248,000
3350	5	4.5	yes	3 car	1 acre	NE	\$644,000
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

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

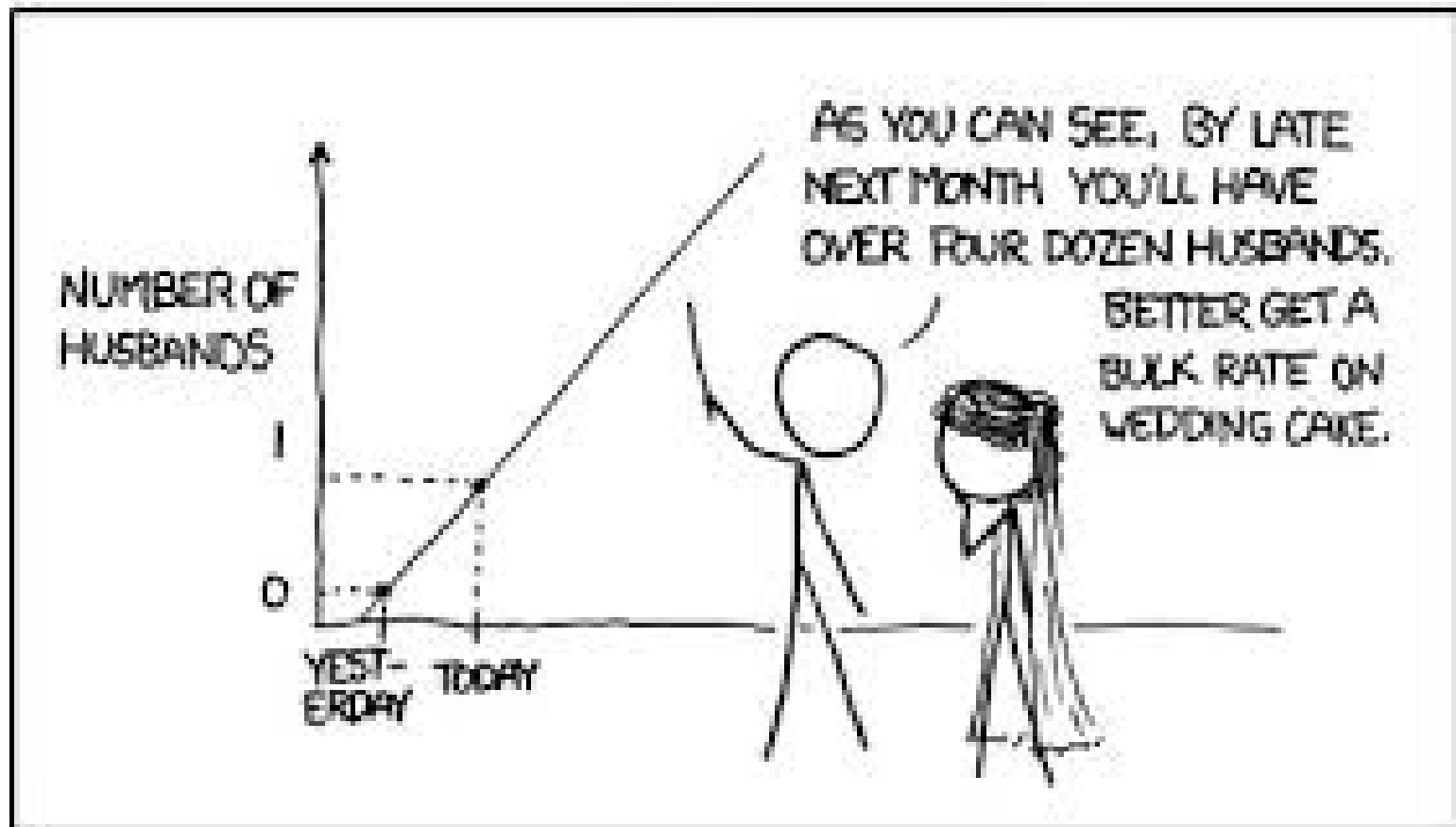


Prediction



Beware: Potential Problem with using ML to Predict

MY HOBBY: EXTRAPOLATING



Example. Assume that we use the training set below to predict the listing price of a house.

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

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

Sq. ft.	No. Bedrooms	No. Baths	Zip Code	Year Built
1137	3	2	32309	1983
2483	4	3	32312	2016
2400	4	4	32312	2002
3560	6	4	32309	1973

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.



ML Applied to House Pricing in Tallahassee

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.

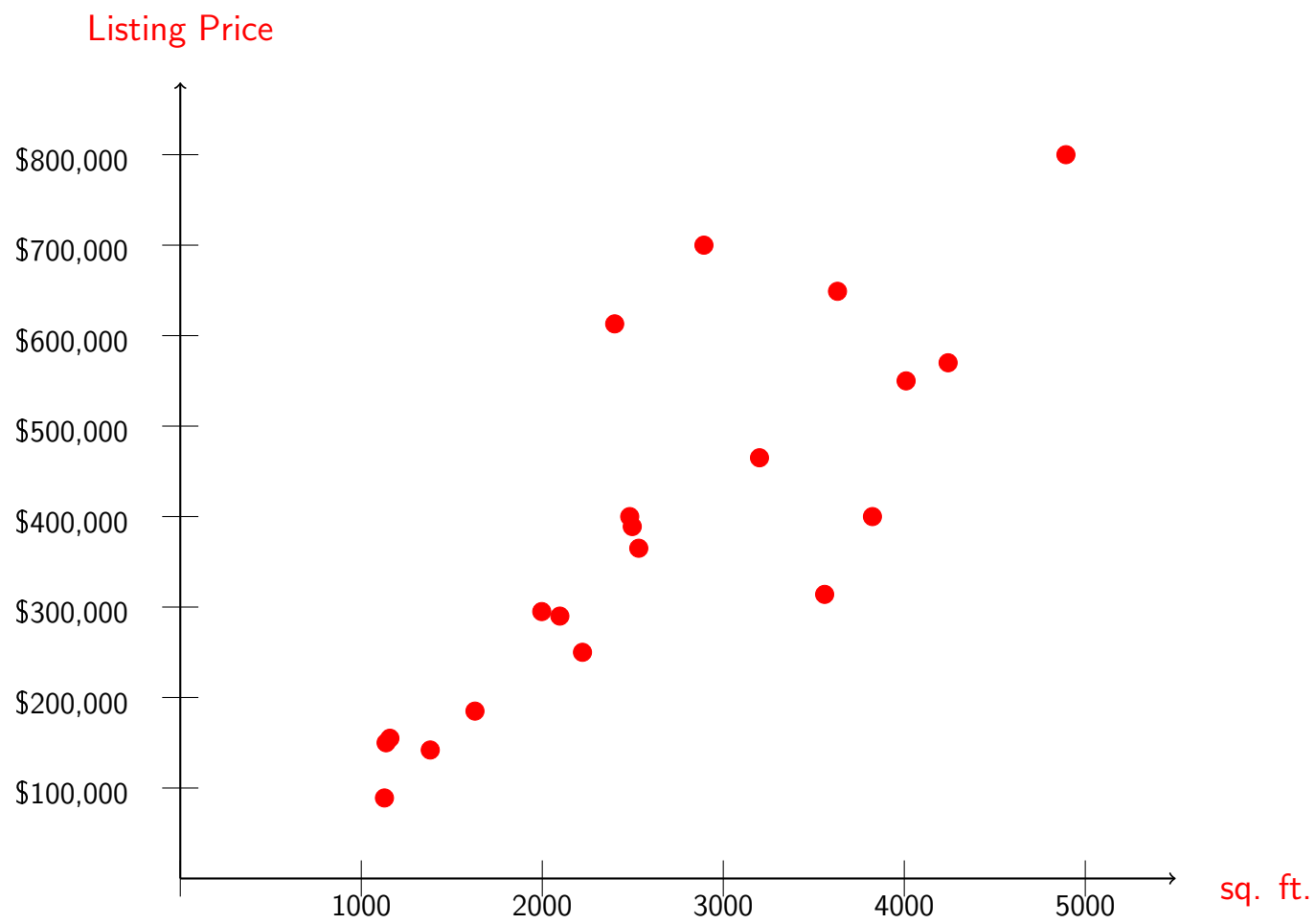


Sq. ft.	No. Bedrooms	No. Baths	Zip Code	Year Built	Listing Price
2222	3	3.5	32312	1981	\$250,000
1628	3	2	32308	2009	\$185,000
3824	5	4	32312	1954	\$399,000
1137	3	2	32309	1983	\$150,000
3560	6	4	32309	1973	\$315,000
2893	4	3	32312	1994	\$699,000
3631	4	3	32309	1996	\$649,000
2483	4	3	32312	2016	\$399,000
2400	4	4	32312	2002	\$613,000
1997	3	3	32311	2006	\$295,000
2097	4	3	32311	2016	\$290,000
3200	5	4	32312	1964	\$465,000
4892	5	6	32311	2005	\$799,900
1128	2	1	32303	1955	\$89,000
1381	3	2	32301	2006	\$143,000
4242	4	5	32303	2007	\$569,000
2533	3	2	32310	1991	\$365,000
1158	3	2	32303	1993	\$155,000
2497	4	4	32309	1990	\$289,000
4010	5	3	32309	2002	\$549,900

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 house 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).

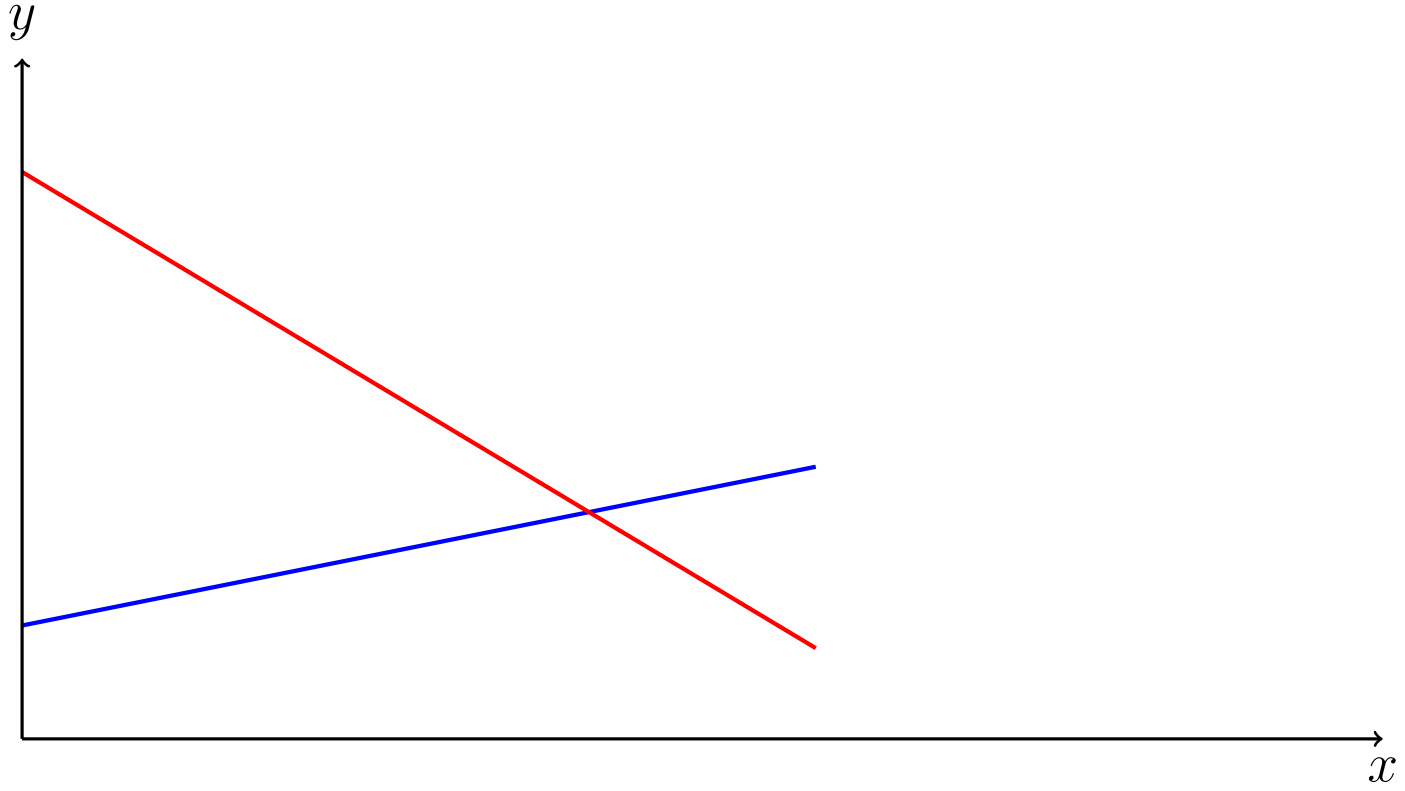
The first step in writing the algorithm is to decide how we want to describe our **hypothesis** (i.e., a good listing price) on a computer. To make things easy, let's say it depends **linearly** on the single input variable which is the square feet of living space.

What does “depend linearly” mean?

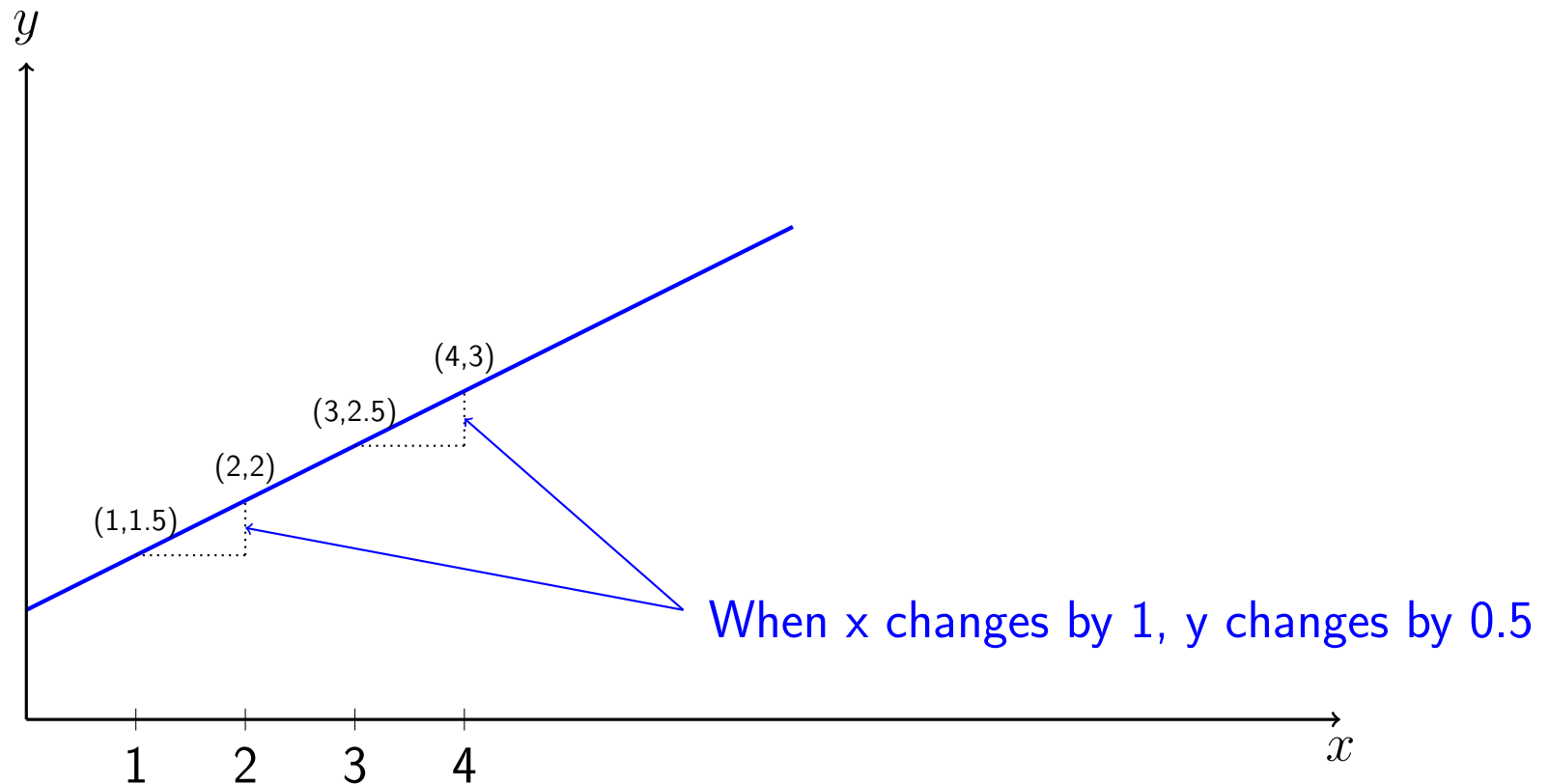
We use linearity all the time without realizing it.

- 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.

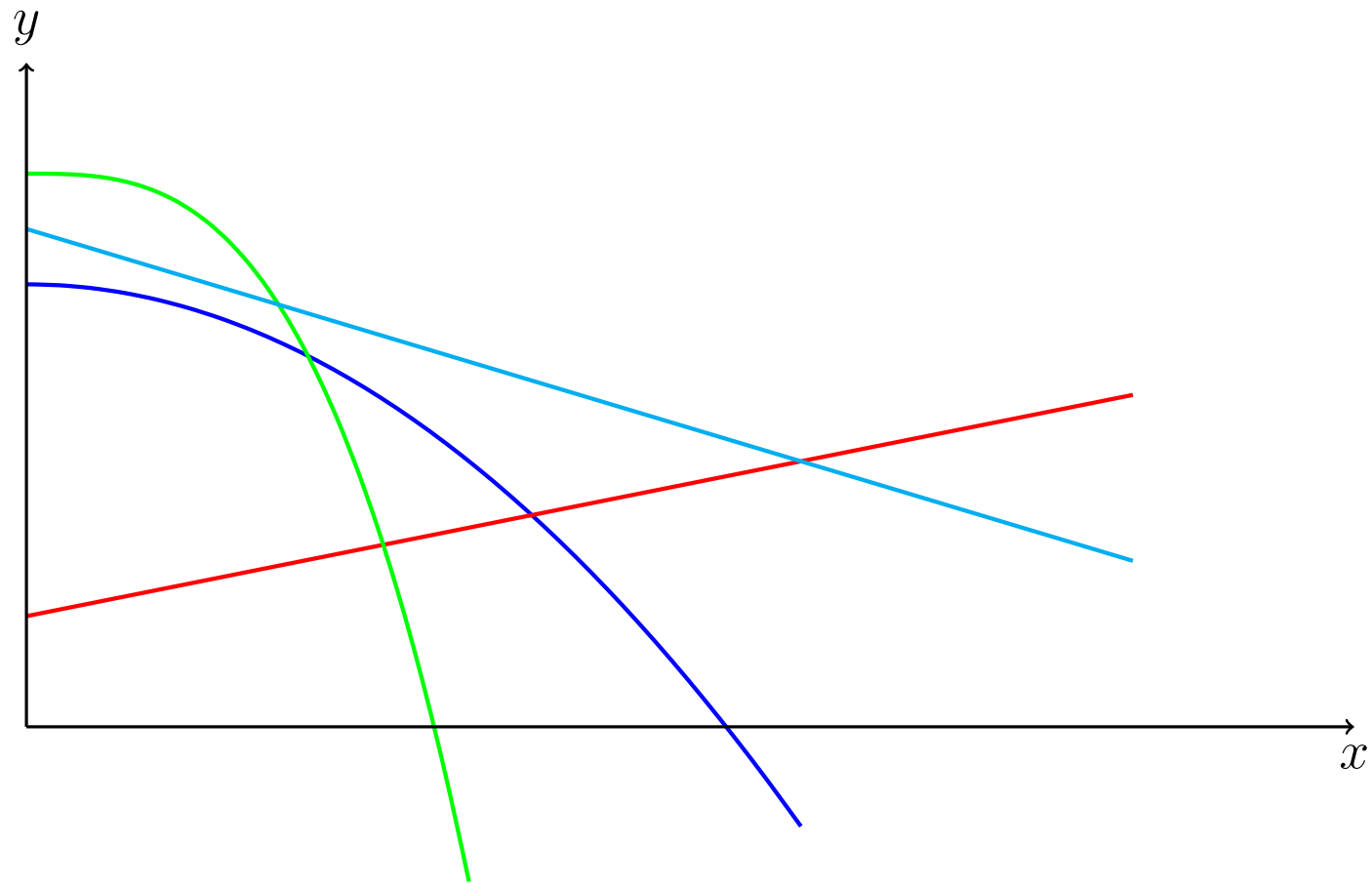
What does “depend linearly” mean graphically? We know that when we plot a linear function we get a **straight line**.



Linear functions have a **constant rate of change** which can be seen graphically.



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 Δx 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 Δx 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 $2\Delta x$. Only if the y -intercept is 0 does the new y value double as the table below illustrates. In the tables 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.

		$y = 4x + 1$			$y = 4x$		
Δx	$1 + \Delta x$	$y(1)$	$y(1 + \Delta x)$	Δy	$y(1)$	$y(1 + \Delta x)$	Δy
1/2	1.5	5	7	$2 = 4\Delta x$	4	6	$2 = 4\Delta x$
1	2	5	9	$4 = 4\Delta x$	4	8	$4 = 4\Delta x$
3	4	5	17	$12 = 4\Delta x$	4	16	$12 = 4\Delta x$
-1	0	5	1	$-4 = 4\Delta x$	4	0	$-4 = 4\Delta x$

		$y = -2x + 3$			$y = -2x$		
Δx	$1 + \Delta x$	$y(1)$	$y(1 + \Delta x)$	Δy	$y(1)$	$y(1 + \Delta x)$	Δy
1/2	1.5	1	0	$-1 = -2\Delta x$	-2	-3	$-1 = -2\Delta x$
1	2	1	-1	$-2 = -2\Delta x$	-2	-6	$-4 = -2\Delta x$
3	4	1	-5	$-6 = -2\Delta x$	-2	-8	$-6 = -2\Delta x$
-1	0	1	3	$2 = -2\Delta x$	-2	0	$2 = -2\Delta x$

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.



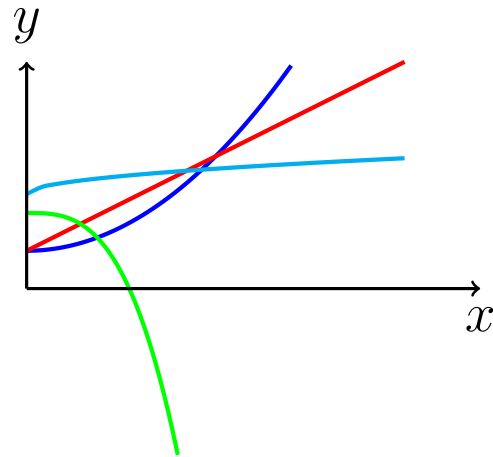
6.0 earthquake damage



7.0 earthquake damage

Socratic Quiz - PartV_Practice_Quiz1

IUZGAZ34E



1. In the plot, which curve represents a linear function?
 - (a) blue
 - (b) red
 - (c) green
 - (d) cyan

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 statements is/are true?
- (a) y changes by 3
 - (b) y changes by 9
 - (c) the new value of y is tripled
 - (d) (b) and (c)
 - (e) none of the above

Now that we know what “depends linearly” means in general, 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

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

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:

Sq. ft.	Listing Price
2222	\$250,000
1628	\$185,000

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.,

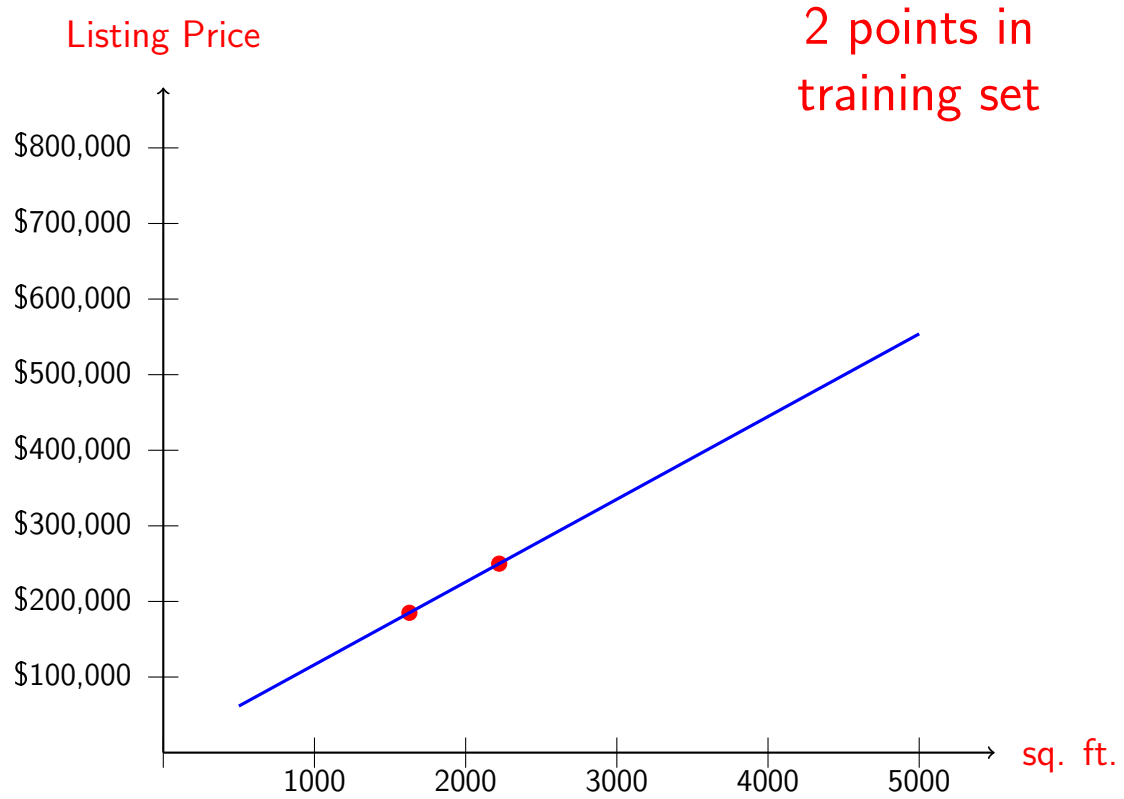
$$250000 = b + (2222)(109.428) \implies b = 6851.85$$

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.



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 $P = 6851.85 + 109.428x$.

Sq. Ft	Predicted Listing Price
1000	\$ 116,280
2000	\$ 225,708
2350	\$ 264,008
3000	\$ 335,136
4000	\$ 444,564

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.

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

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.

Sq. ft.	Listing Price
3631	\$649,000
1628	\$185,000

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

$$P(1628) = 231.65(1628) - 192130.3 = 184,995.9$$

$$P(3824) = 231.65(3824) - 192130.3 = 693,699.3$$

$$P(1137) = 231.65(1137) - 192130.3 = 71,255.75$$

$$P(3560) = 231.65(3560) - 192130.3 = 632,543.7$$

$$P(2893) = 231.65(2893) - 192130.3 = 478,033.15$$

$$P(3631) = 231.65(3631) - 192130.3 = 648,990.85$$

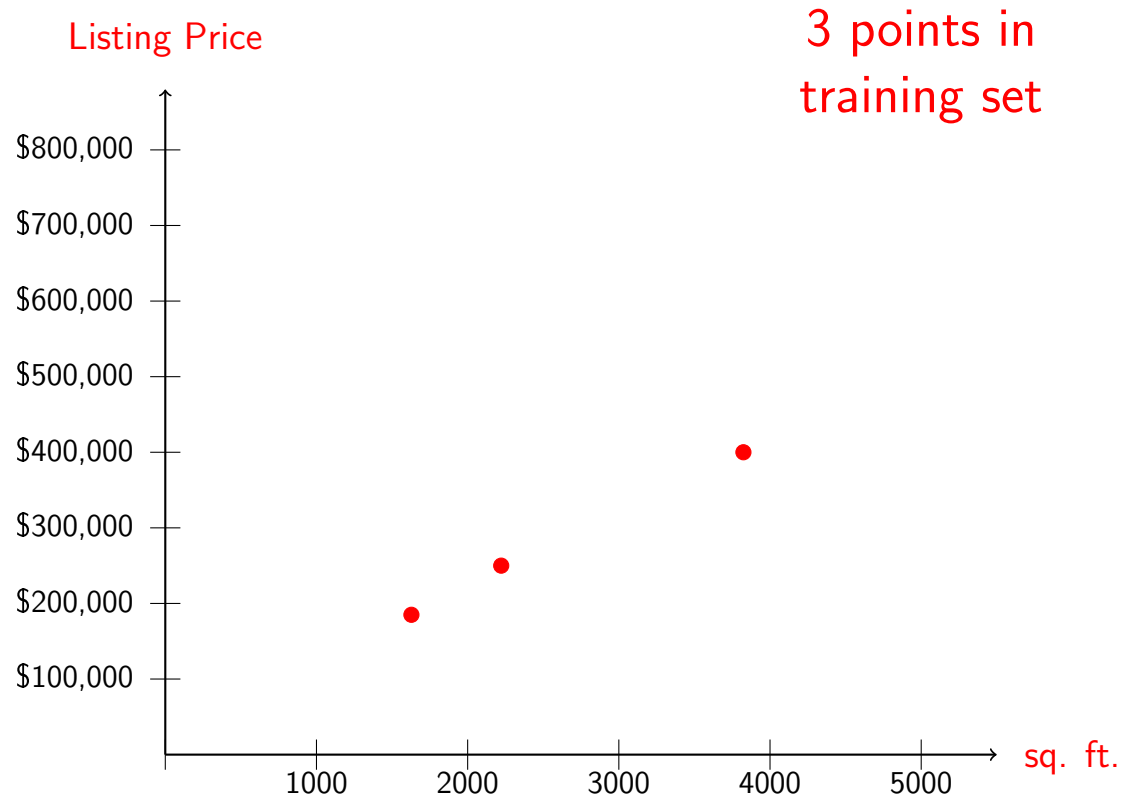
The reason the two houses in the training set are not quite exact is because we rounded the values for a, b . As you can see, the predictions are still not very good.

Sq. Ft	Predicted Listing Price	Actual Listing Price	Difference
1628	\$ 185,000	\$ 184,996	-
3824	\$ 693,699	\$ 399,000	+\$ 294,699
1137	\$71,256	\$ 150,000	- \$ 78,744
3560	\$ 632,544	\$ 315,000	+\$317,544
2893	\$ 478,033	\$ 699,000	-\$ 220,967
3631	\$ 404,185	\$ 648,991	-

To improve the results we add a house to our training set to use 3 houses.

Sq. ft.	Listing Price
2222	\$250,000
1628	\$185,000
3824	\$399,000

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?

Next time we will look at how to do this and to include more criteria in the training set instead of just square feet of living space.

Socratic Quiz - Part V Quiz 1

IUZGAZ34E

Answer true or false.

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. To predict accurate values for the listing price of a house between \$100,000 and \$1,000,000 it is enough to have a training set with houses which list between \$100,000 and \$500,000.
5. Any linear function has a constant rate of change.
6. If y depends linearly on x then its plot will be a parabola.
7. If $y = 8 + 7x$ then when x is changed by -2 then y will always change by $8 + 7(-2) = -6$.

8. If $y = 5x$ and $w = 7 + 5x$ then $w(x)$ and $y(x)$ will change the same amount when x is changed by 2.
9. The slope of a line dictates how much y will change when x is changed.
10. A Machine Learning algorithm will make different predictions if trained on different data sets.

Goals for this lecture:

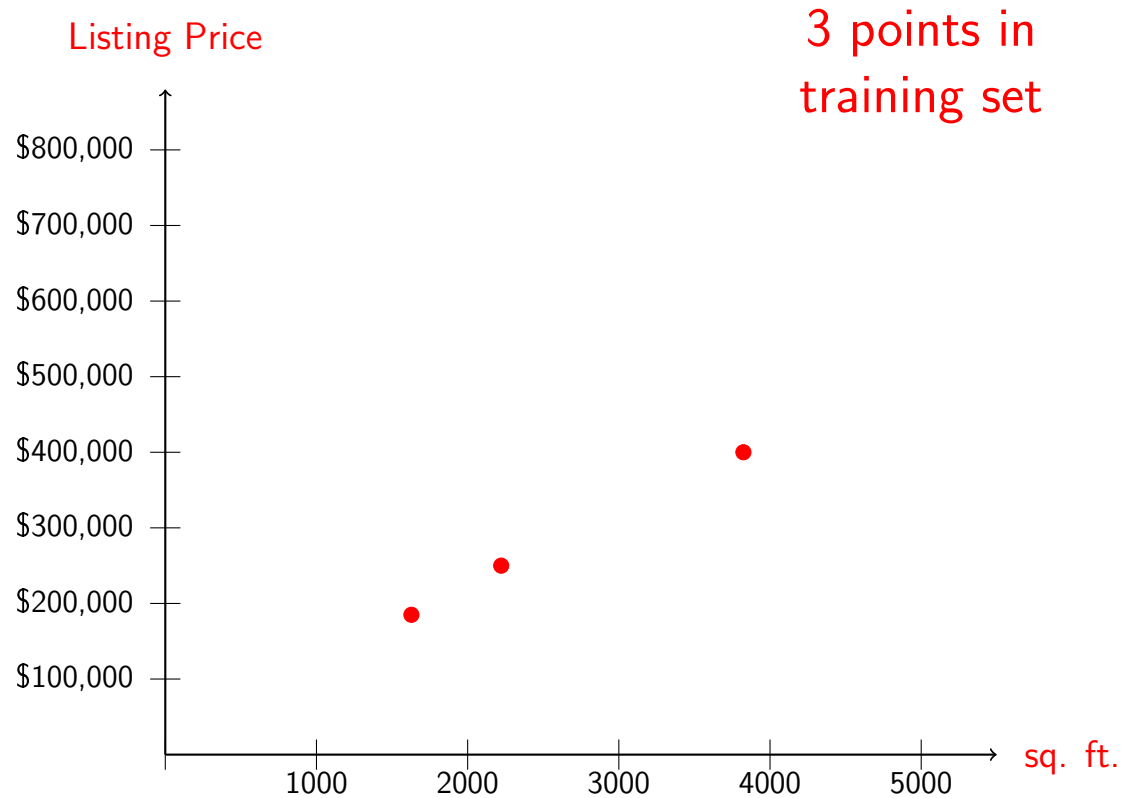
1. To introduce the concept of linear regression.
2. To apply linear regression to our example of predicting the listing price of a house.
3. To do a classroom exercise on linear regression.

Last time we saw that using only two data points didn't give very good answers. Now we want to increase the number of entries in our training set.

However, we can no longer find a line which passes through all the data so what can we do?

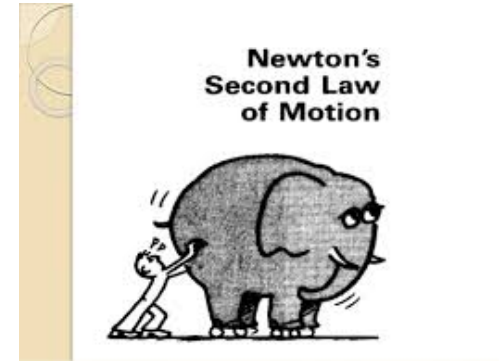
To see this, we begin by using 3 entries in the training set. We know that with 3 points they probably don't lie on a line. Below is a table containing our data and a scatter plot of the data. Clearly, they don't lie on a line.

Sq. ft.	Listing Price
2222	\$250,000
1628	\$185,000
3824	\$399,000



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?

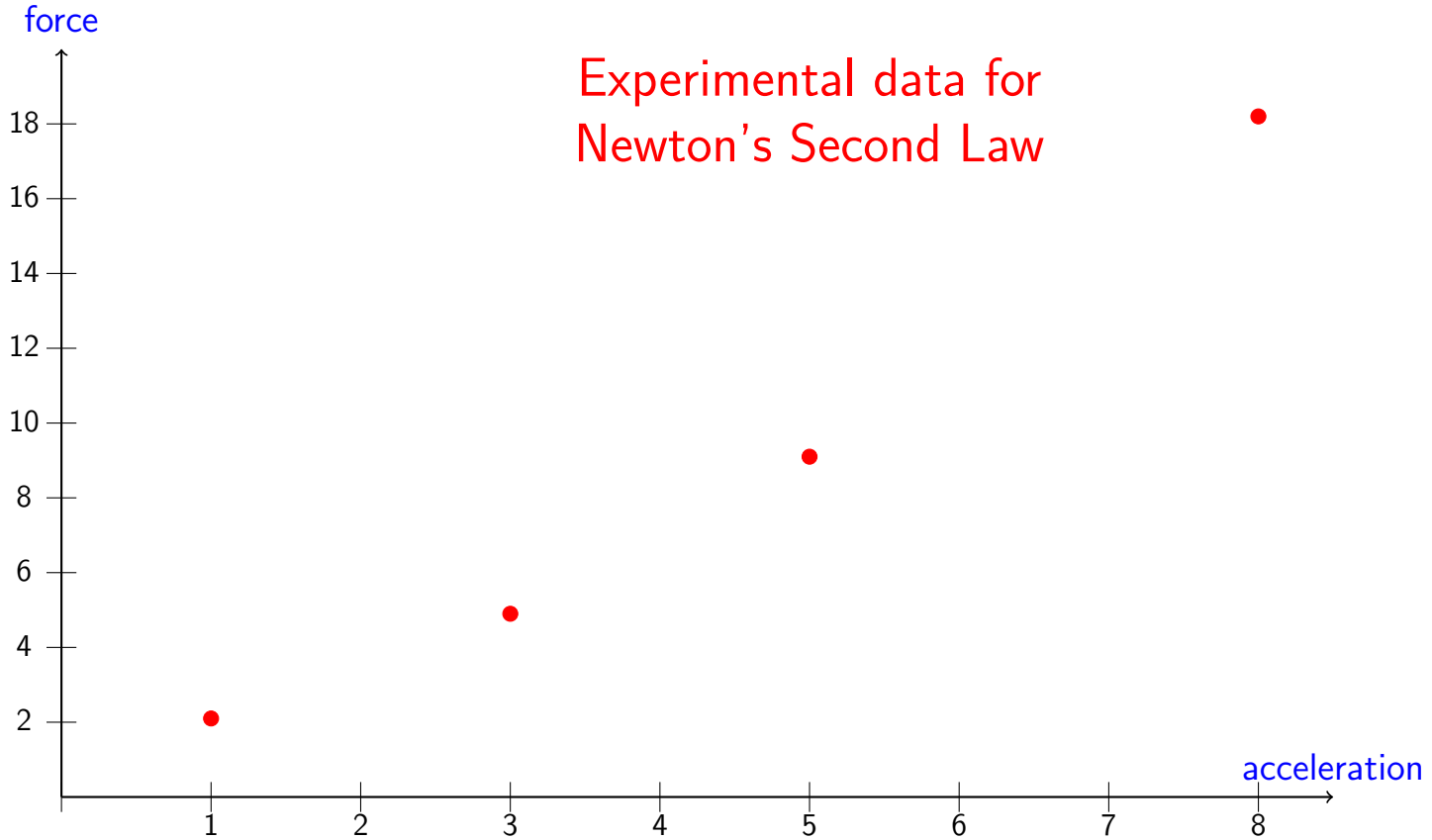
To understand the approach we will use we look at a simple example from a science lab. Assume we have an object with a given weight of 2 kilograms. We know from Newton's Second Law of Motion that the force F acting on an object is given by the mass times the acceleration a in meters per second squared 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.



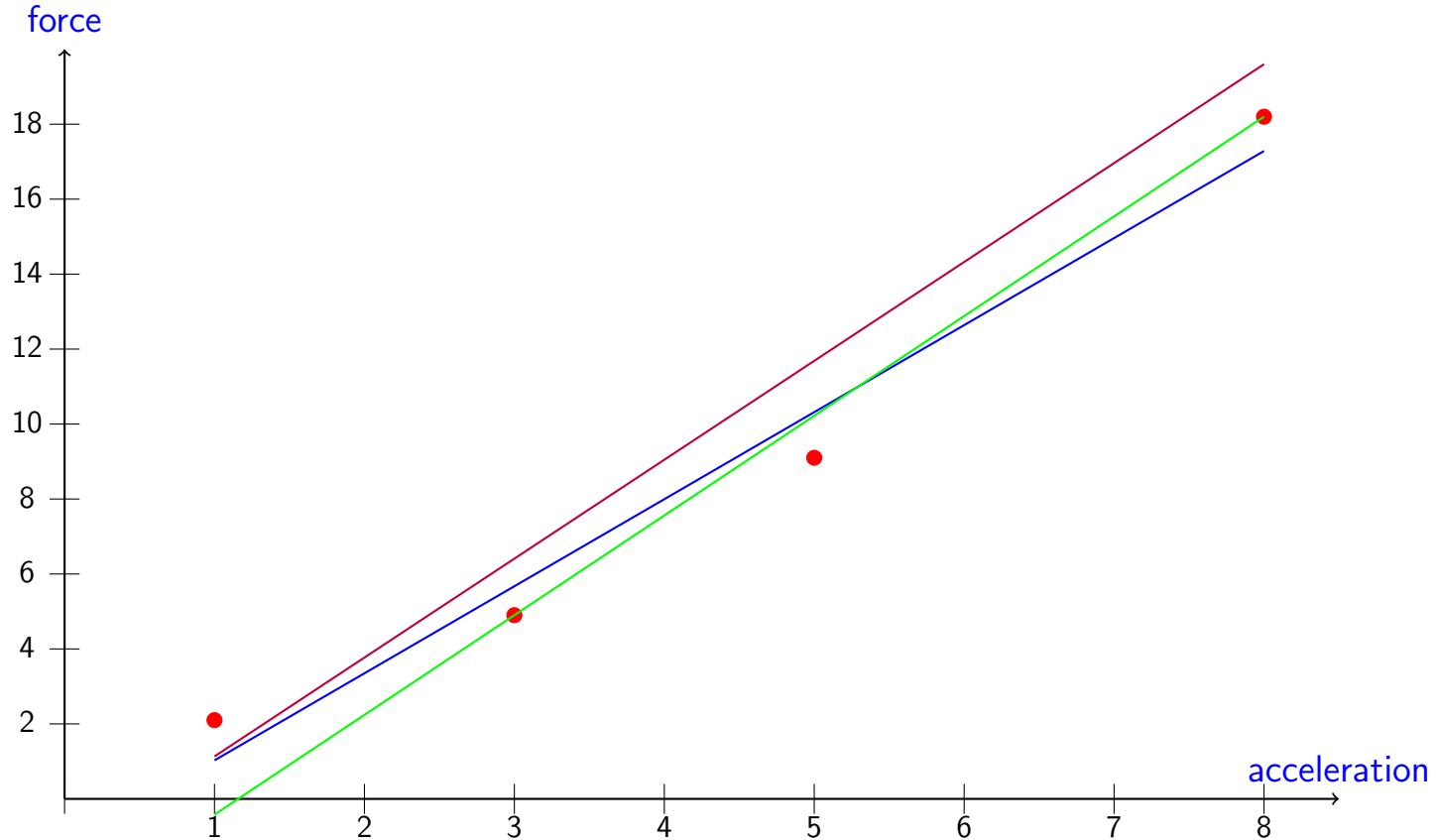
In our case $F = 2a$. Suppose you are in a lab taking measurements of the force on the object for 4 different values of the acceleration. If it was a perfect world then all your points would lie on a straight line but we all know measurements are susceptible to errors. Instead suppose you took the following measurements for the 2 kilogram weight; the scatter plot of the data is included.

acceleration measured force force predicted by $F = 2a$

1 m/sec ²	2.1	2
3 m/sec ²	4.9	6
5 m/sec ²	9.1	10
8 m/sec ²	18.2	16



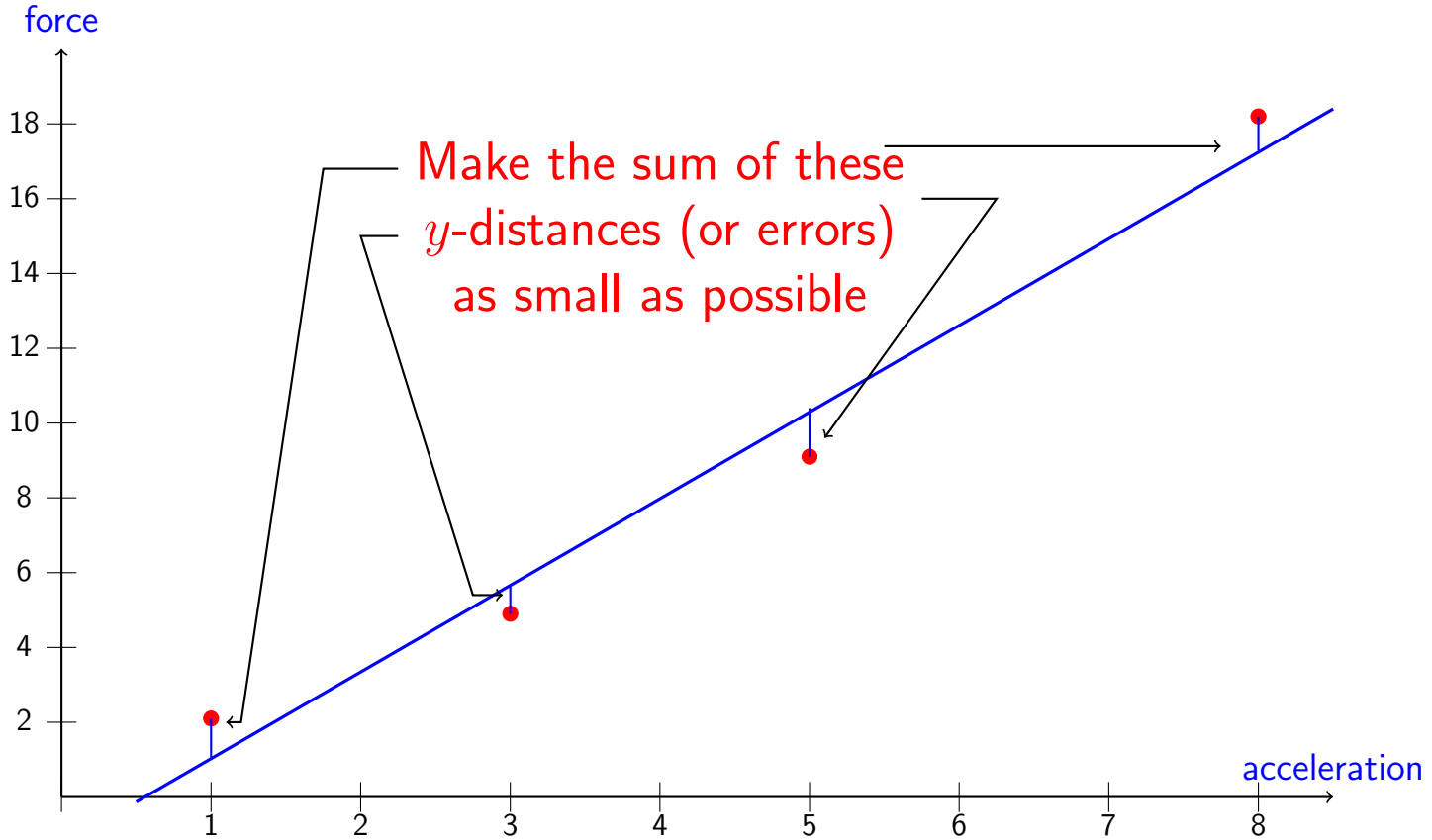
Due to measurement errors, the data does NOT lie on a straight line but we know it should. So our goal is to find a line which best represents the data.



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.



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? 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

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

as small as possible. Mathematically, there is a straightforward way to do this which we won't go into here.

Returning to our example, we want to find the line $P = a + bx$ which makes the sum of the errors as small as possible. Recalling that our three data points are

Sq. ft.	Listing Price
2222	\$250,000
1628	\$185,000
3824	\$399,000

we see that the sum of the squares of the errors which we want to minimize is

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

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.

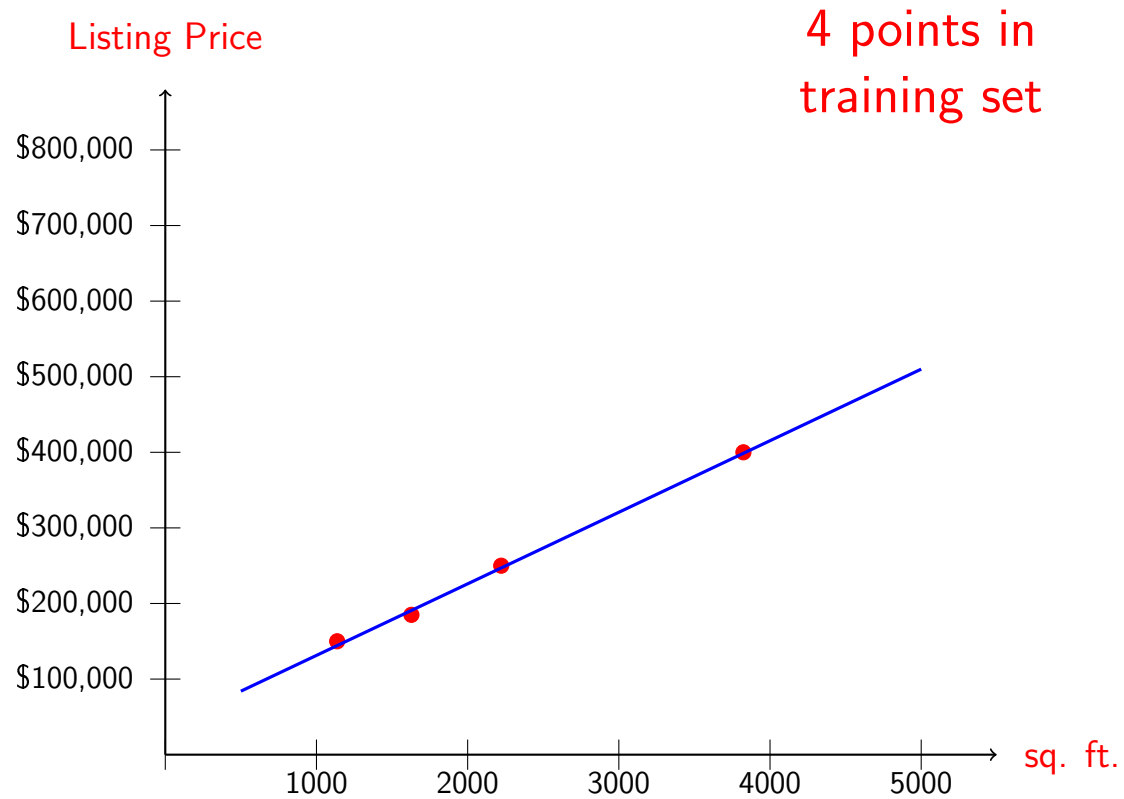


If we add the next house which has 1137 sq ft with a listing price of \$150,000 to our training set, then we get a different line because the line changes to incorporate this information. Note that both the y -intercept and the slope change each time.

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.

Sq. ft.	Listing Price	Price per sq. ft.
2222	\$250,000	\$113
1628	\$185,000	\$114
3824	\$399,000	\$104
1137	\$150,000	\$132
3560	\$315,000	\$88
2893	\$699,000	\$241

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.

$$2 \text{ points } P = 6851.85 + 109.428x$$

$$3 \text{ points } P = 31092.8 + 96.5235x$$

$$4 \text{ points } P = 38624 + 94.1396x$$

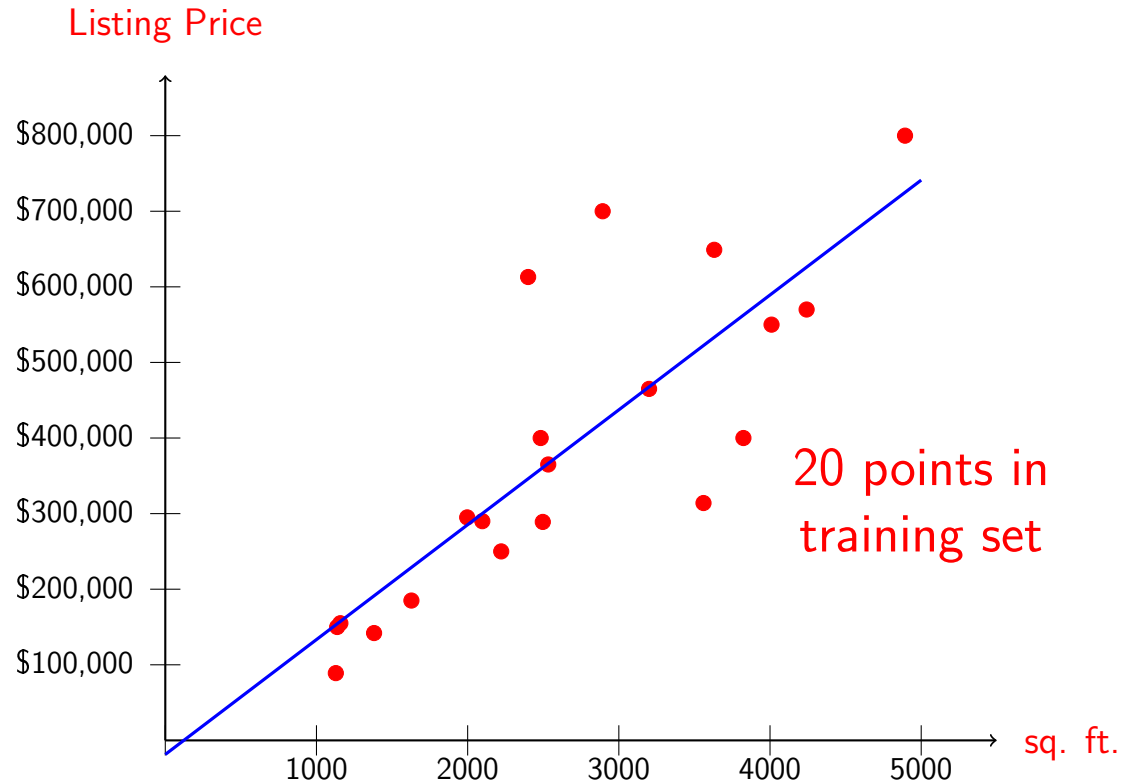
$$5 \text{ points } P = 55232.5 + 82.6802x$$

$$6 \text{ points } P = 59807.1 + 107.387x$$





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.

No. houses in training set	line	Predicted value for 2150 sq ft house	Predicted value for 4110 sq ft house
2	$6851.9 + 109.4x$	\$242,122	\$456,601
3	$31092.8 + 96.5x$	\$238,618	\$427,804
4	$38634 + 94.1x$	\$241,034	\$425,548
5	$55232.5 + 82.7x$	\$232,995	\$395,048
6	$59807.1 + 107.4x$	\$290,689	\$501,168
7	$67225.4 + 122.9x$	\$331,355	\$572,143
8	$34343.9 + 141.2x$	\$337,883	\$ 614,598
10	$69057.2 + 126.612x$	\$341,273	\$589,433
20	$-18625.1 + 151.972x$	\$308,115	\$605,980

Socratic Quiz - Practice Quiz 2

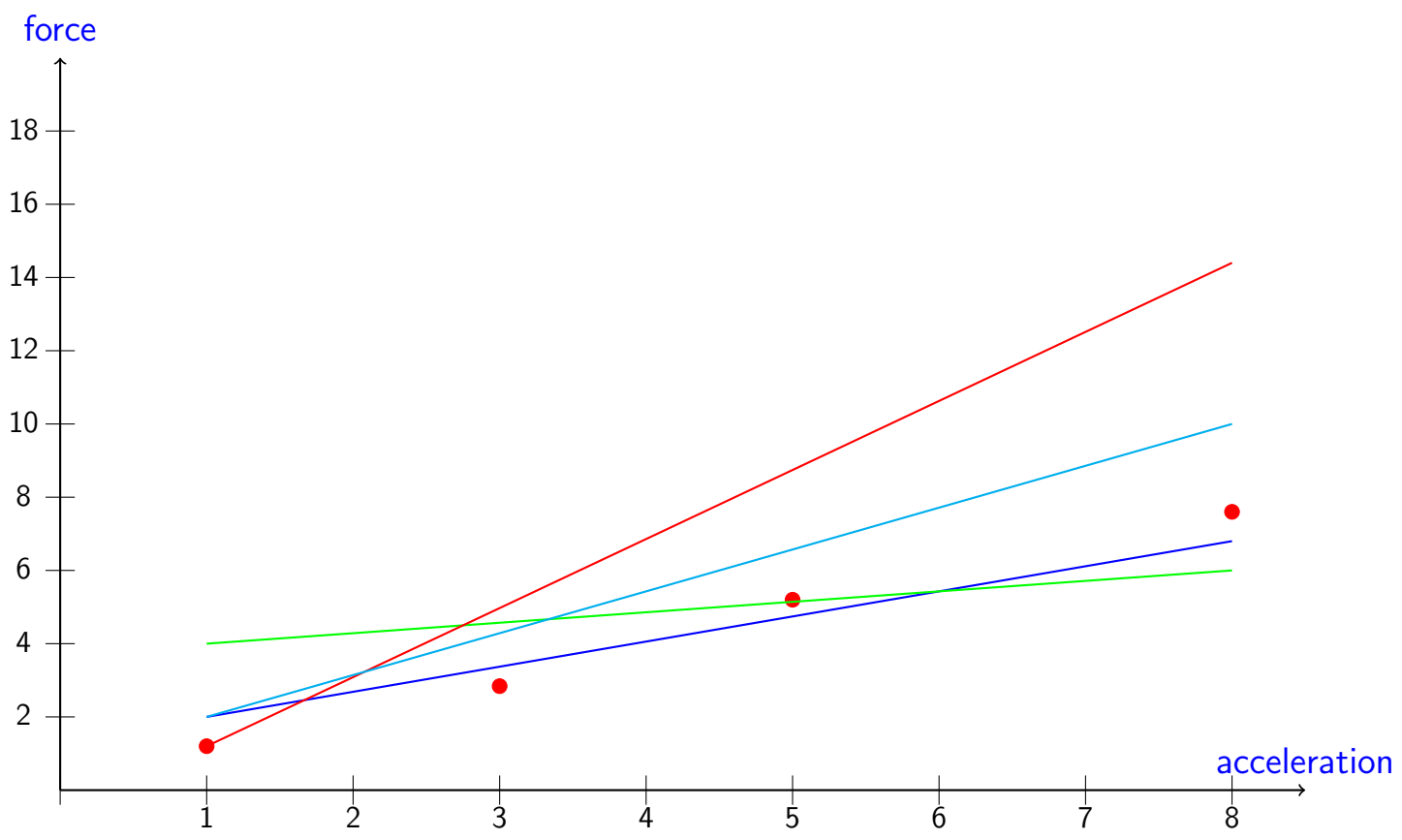
IUZGAZ34E

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

acceleration	measured force	force predicted by $F = a$
1 m/sec ²	1.2	1
3 m/sec ²	2.9	3
5 m/sec ²	5.2	5
8 m/sec ²	7.7	8

1. If we use the first two data points then the line which is the best linear approximation to the data in a linear regression sense passes through the first two points.
2. If we use the first three data points then the line which is the best linear approximation to the data in a linear regression sense passes through the three points.
3. When we use the first three points then we are making the error = $.2^2 + .3^2 + .2^2$ as small as possible.

4. Use the plot to determine which line is the best linear approximation to the given data.
- (a) red
 - (b) green
 - (c) blue
 - (d) cyan
5. If we add a fifth point to the data set where force = 12 for an acceleration of 10 and find the linear approximation to these five points then the slope of the new line is
- (a) greater than the slope of the line for 4 points
 - (b) less than the slope of the line for 4 points
 - (c) the same as the slope of the line for 4 points
 - (d) can't be determined from the given information



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.

Sq. ft.	Sum of No. Bedrooms & Baths	Listing Price
2222	6.5	\$250,000
1628	5	\$185,000
3824	9	\$399,000
1137	5	\$150,000
3560	10	\$315,000
2893	7	\$699,000
3631	7	\$649,000
2483	7	\$399,000
2400	8	\$613,000
1997	6	\$295,000
2097	7	\$290,000
3200	9	\$465,000
4892	11	\$799,900
1128	3	\$89,000
1381	5	\$143,000
4242	9	\$569,000
2533	5	\$365,000
1158	5	\$155,000
2497	8	\$289,000
4010	8	\$549,900

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.

$$3 \text{ bedrooms, } 2.5 \text{ baths} \implies 1915.75 + 163.371(2150) - 7216.75(5.5) = \$313,471$$

$$4 \text{ bedrooms, } 3 \text{ baths} \implies 1915.75 + 163.371(2150) - 7216.75(7) = \$302,646$$

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.

Classwork Exercise on Linear Regression

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 completely satisfied.

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

Employee No.	Salary	Satisfaction Rating
1	55K	61
2	80K	79

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

Employee No.	Salary	Satisfaction Rating
1	55K	61
2	80K	95
3	65K	48
4	90K	85

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)

Employee No.	Salary	Satisfaction Rating
5	72K	33

sketch what you believe will be the new line. Does the slope increase or decrease? Why?

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?

Employee No.	Salary	Predicted Satisfaction Rating
11	58 K	
12	105 K	

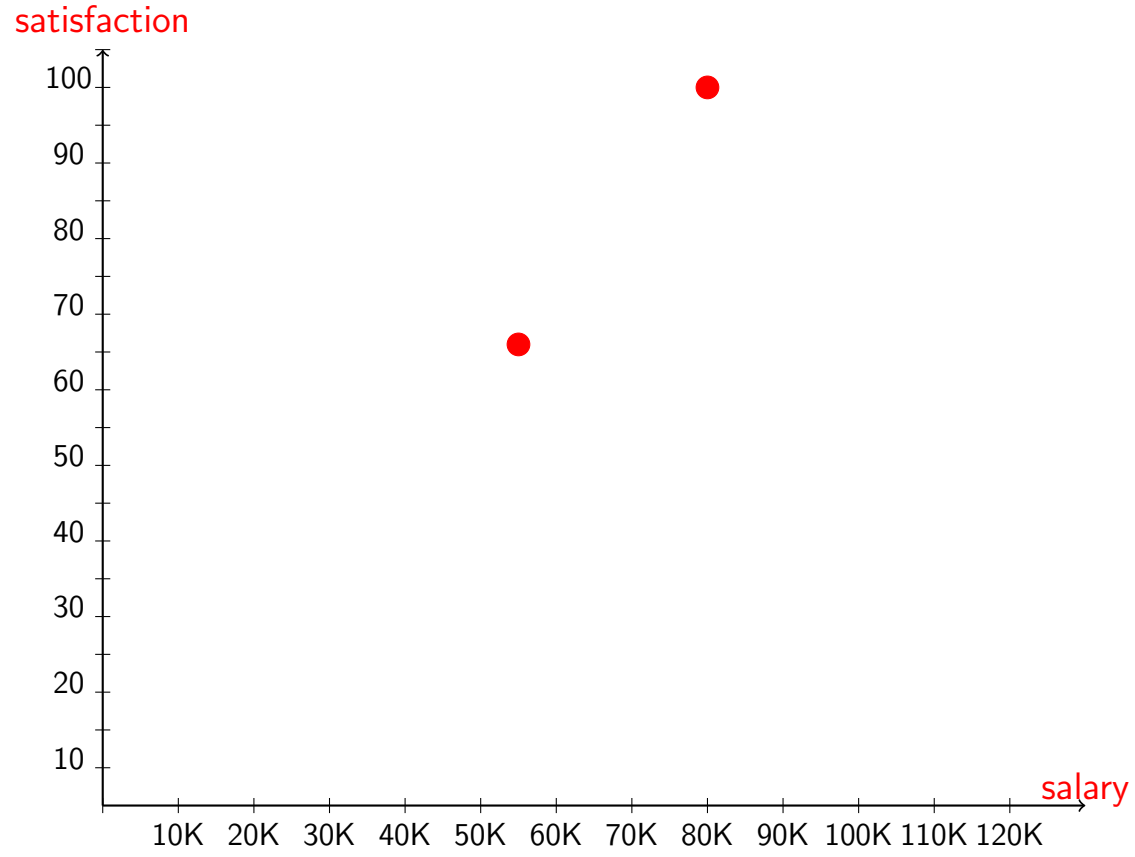


Figure 1

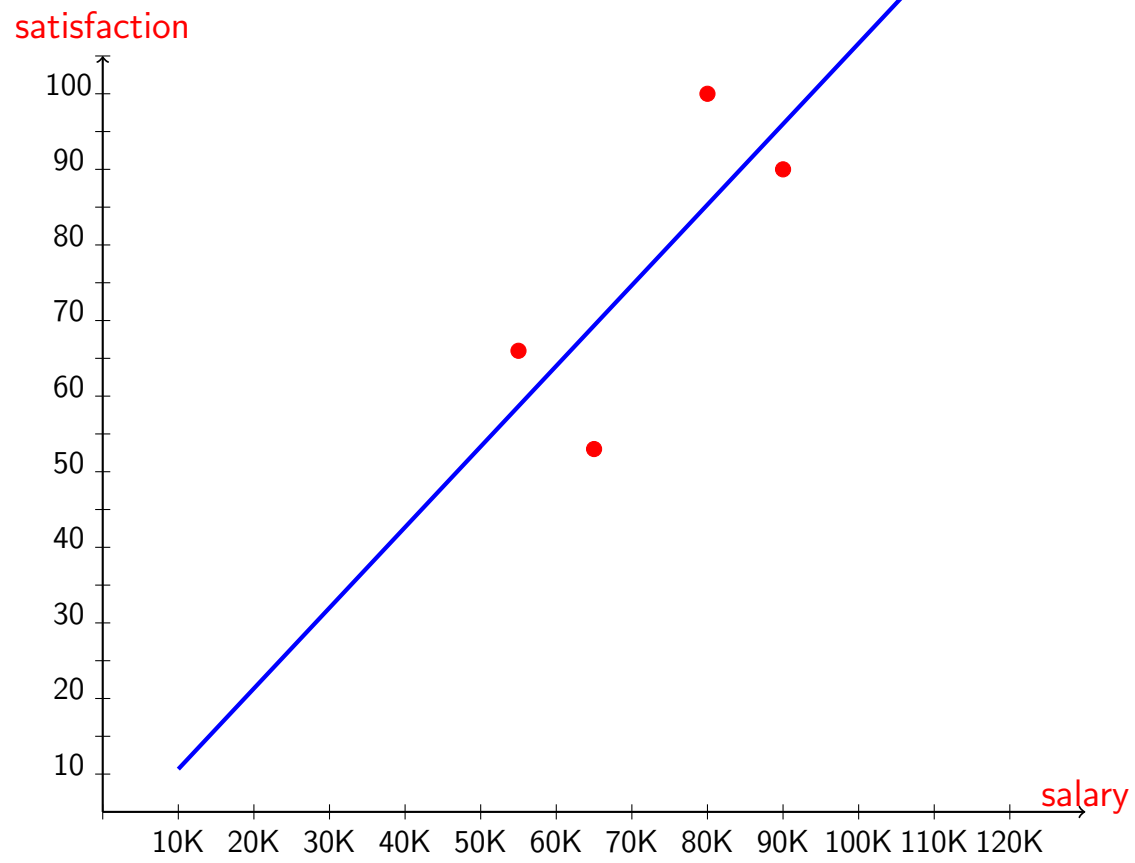


Figure 2

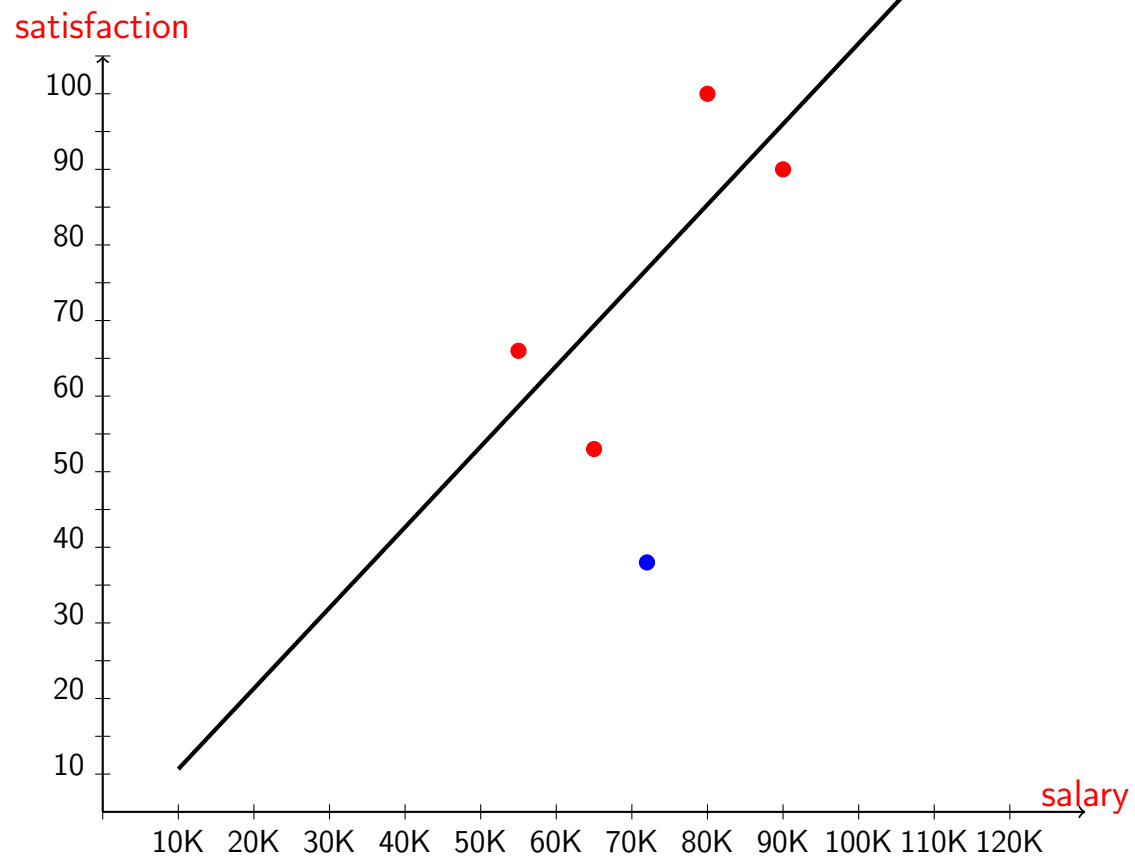


Figure 3

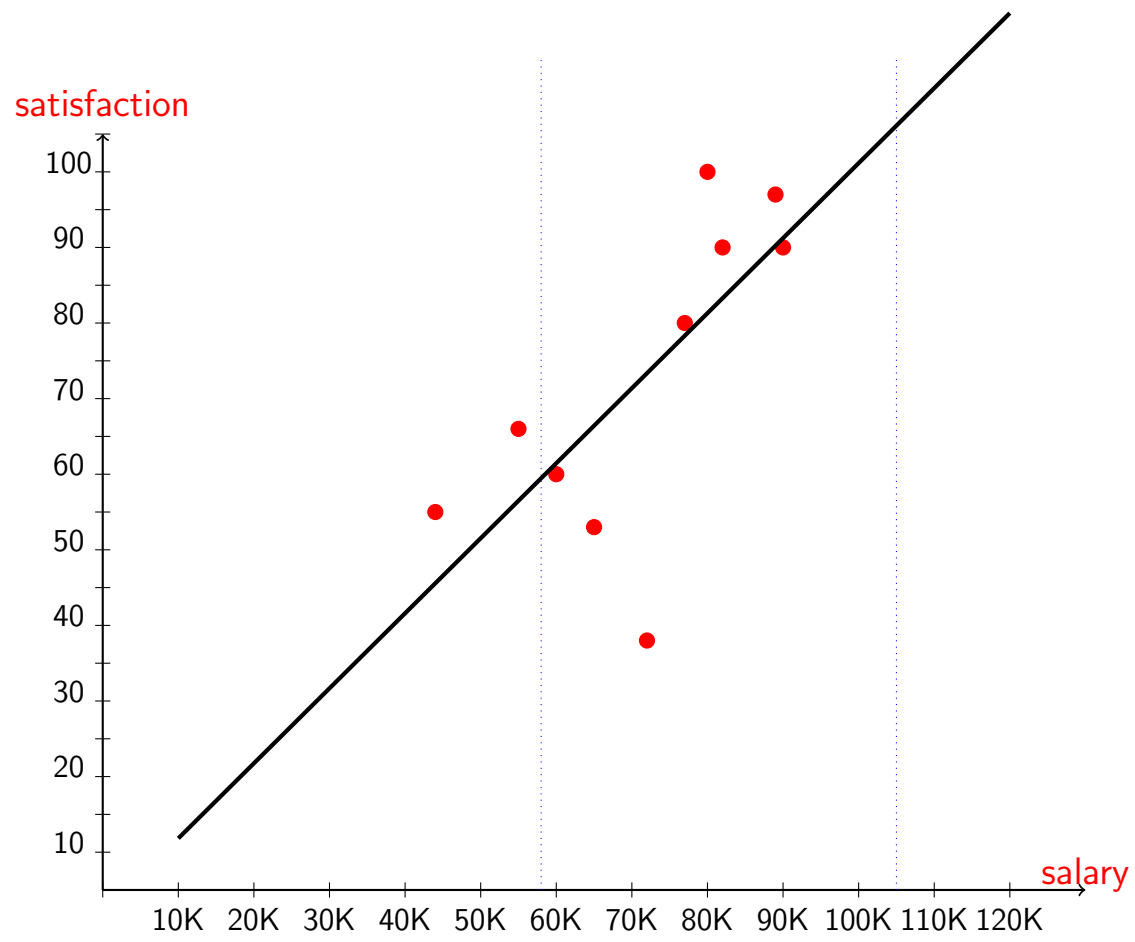
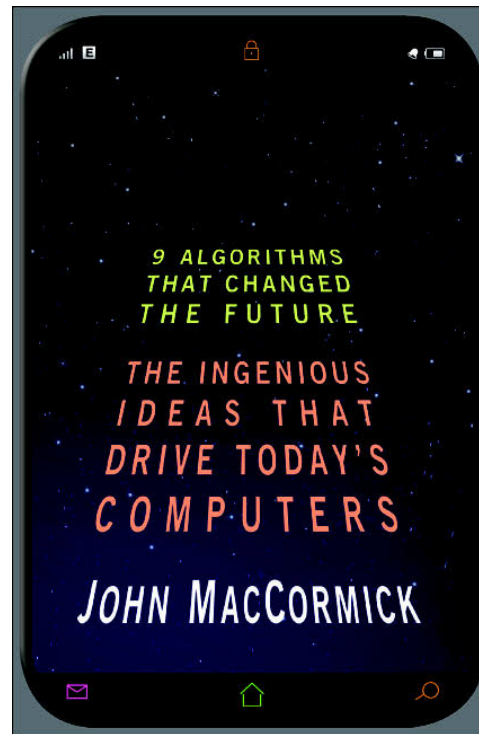


Figure 4

Reading Assignment

Algorithm # 6



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 look at our first pattern recognition algorithm called Nearest Neighbor and apply it to different problems.

Pattern Recognition - Algorithm # 6

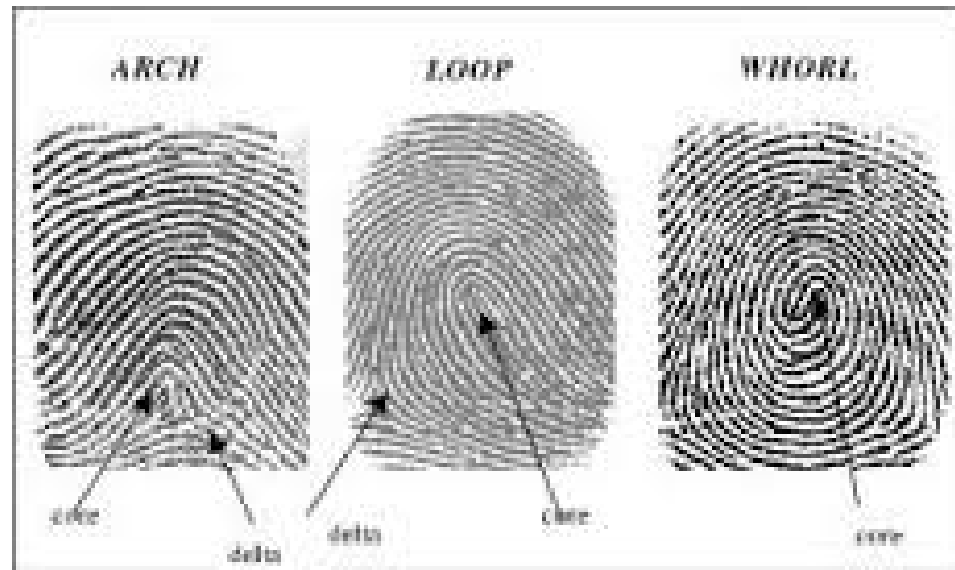
Computer algorithms such as PageRank for searching web pages, codes for encrypting/decrypting files, etc. far exceed what a human can do. However this is not the case when it comes to [pattern recognition](#). Humans have a natural advantage here.

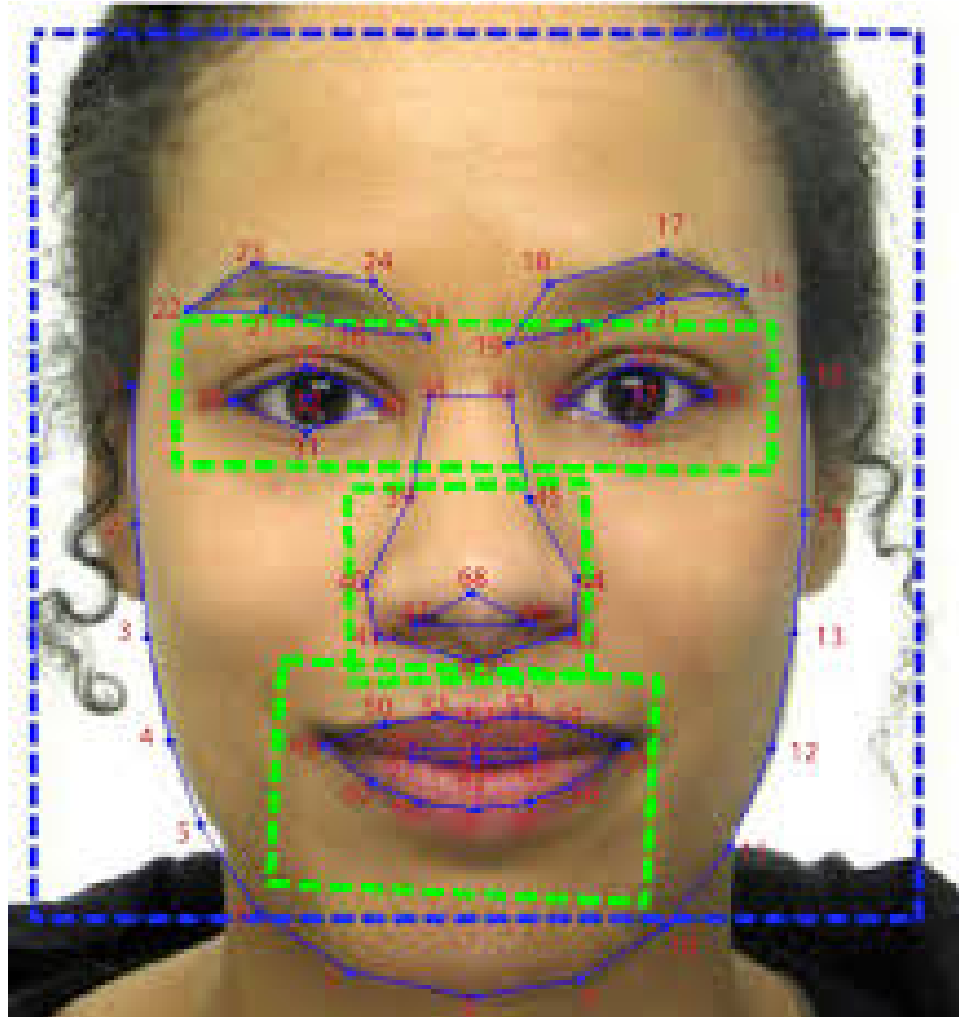


Pattern recognition by a robot - (c) 2012 Scriptol.com

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



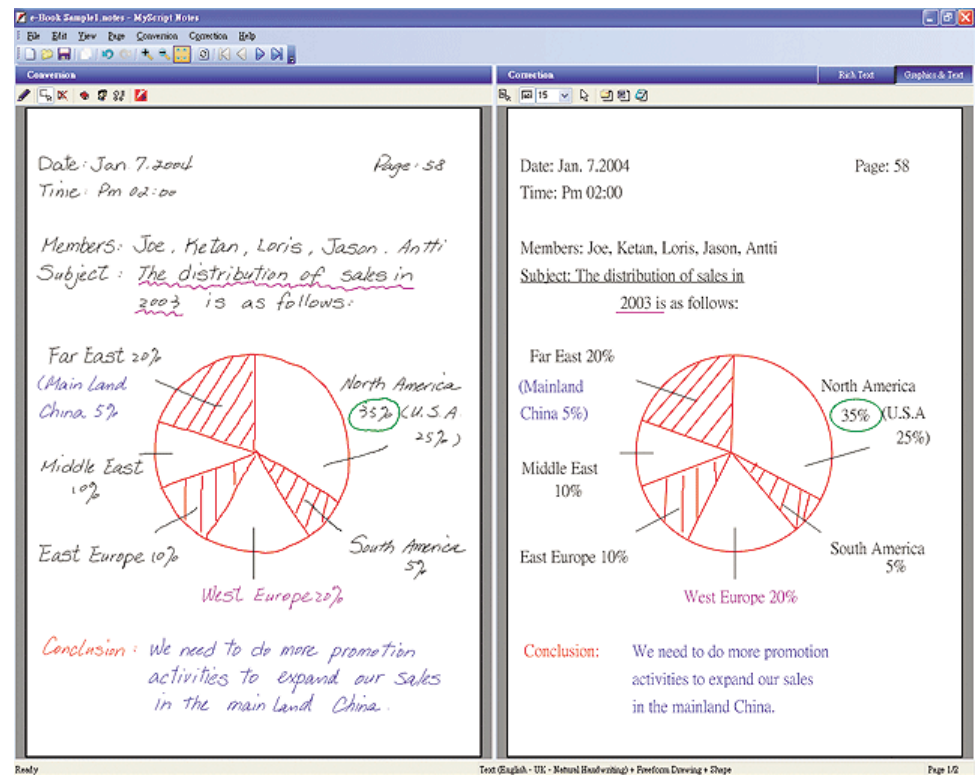


Face Recognition



Object Recognition

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.



Handwriting Recognition

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.

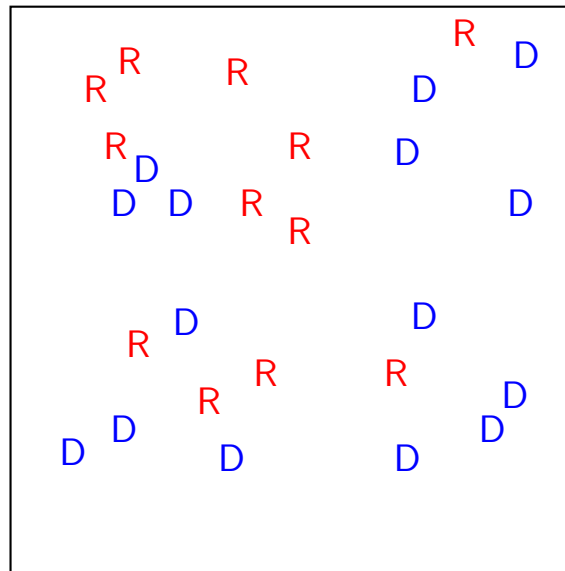
Our goal is to look at three different ML Classifier Algorithms and see some examples of pattern recognition problems they are best suited for.

Nearest-Neighbor Classifier & Variants

- This is probably the simplest classification algorithm
- 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 simplified example before looking at identifying a zip code.

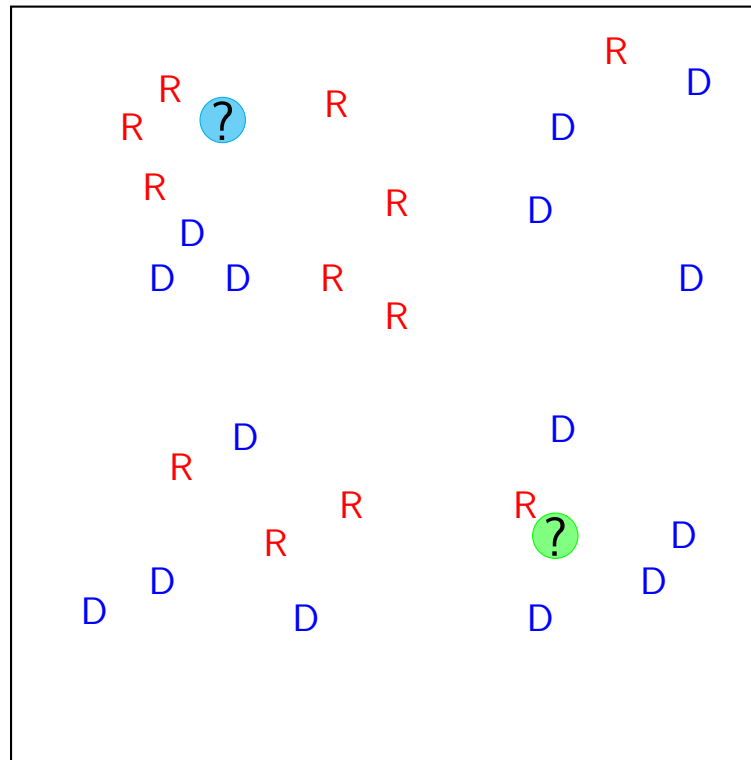
Simple Example of Nearest-Neighbor Classifier

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.

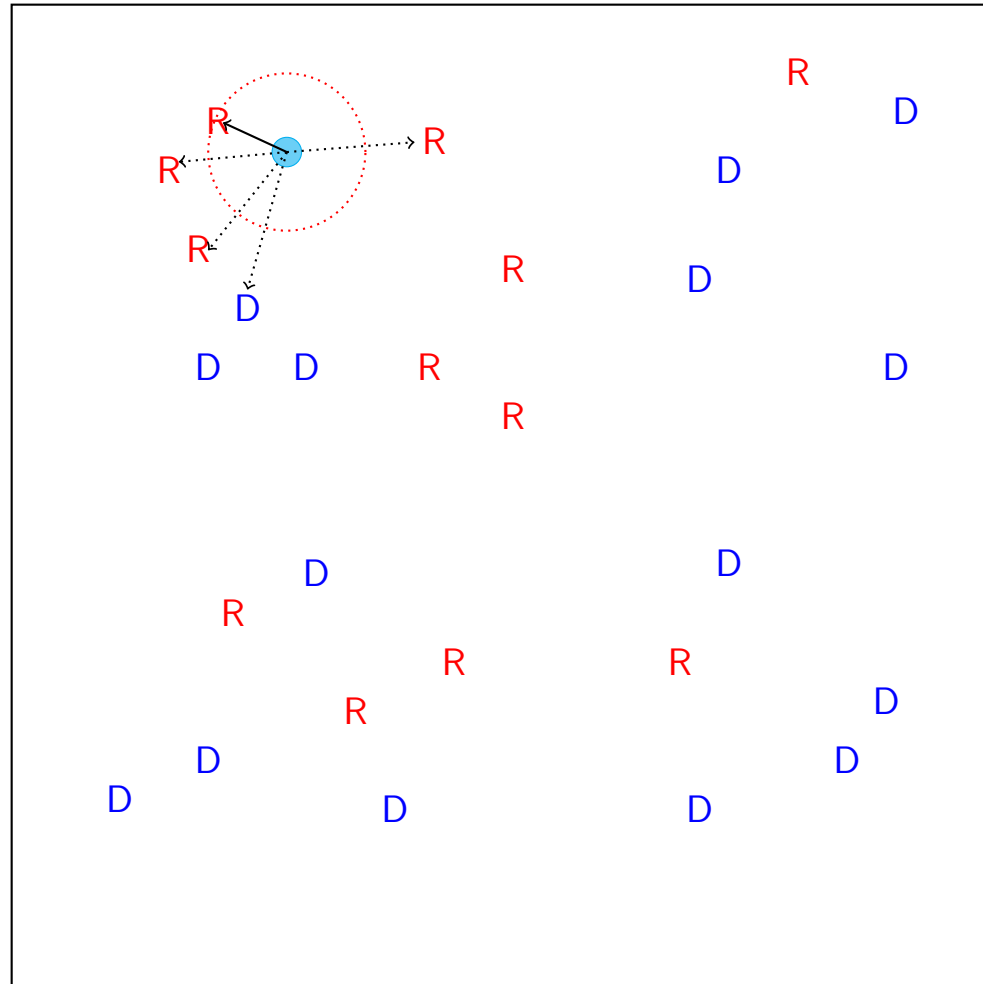


We want to predict whether each of the two residences (represented by a question

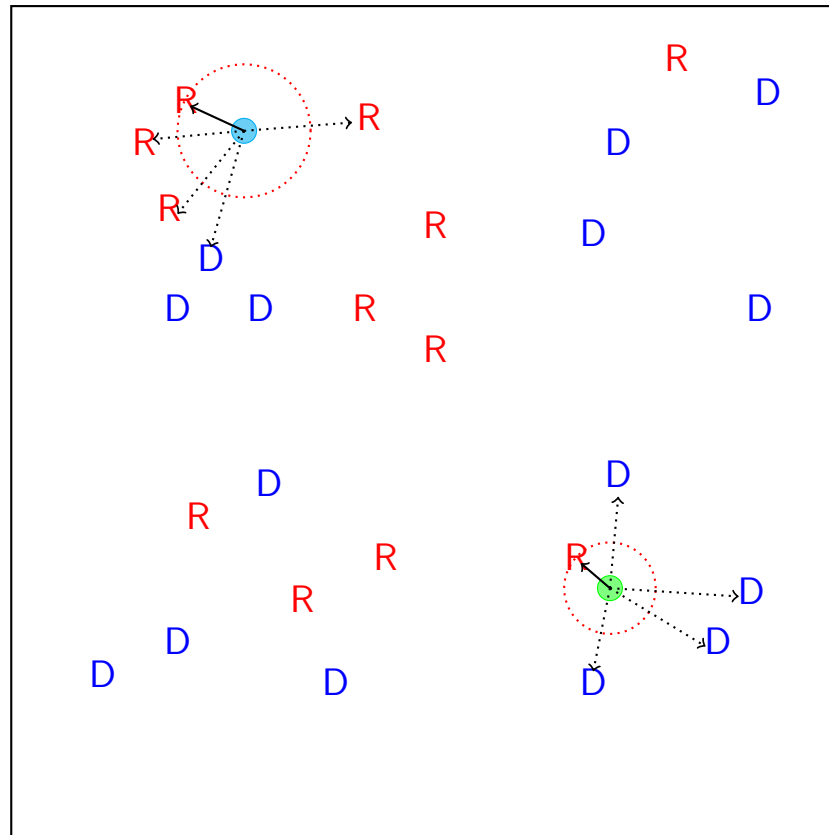
mark) is a Democrat or Republican. Clearly, if **WE** use a Nearest Neighbor strategy to guess the political affiliation of the home located at the top (marked in cyan) then we would say they are Republicans because the surrounding neighbors are all Republicans. So we want to figure out how to tell a computer to do this.



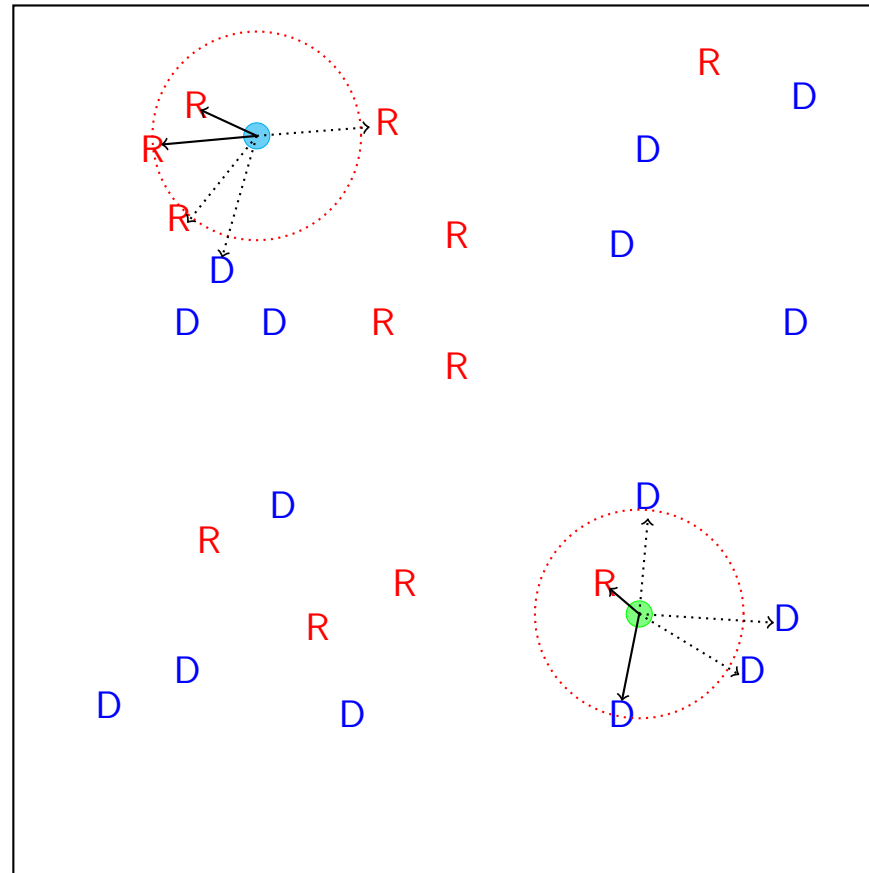
Visually, to determine the nearest neighbor to a residence we would draw circles around the residence until we touch a neighbor. This would be its “nearest neighbor”. Clearly for this residence the nearest neighbor is a Republican.



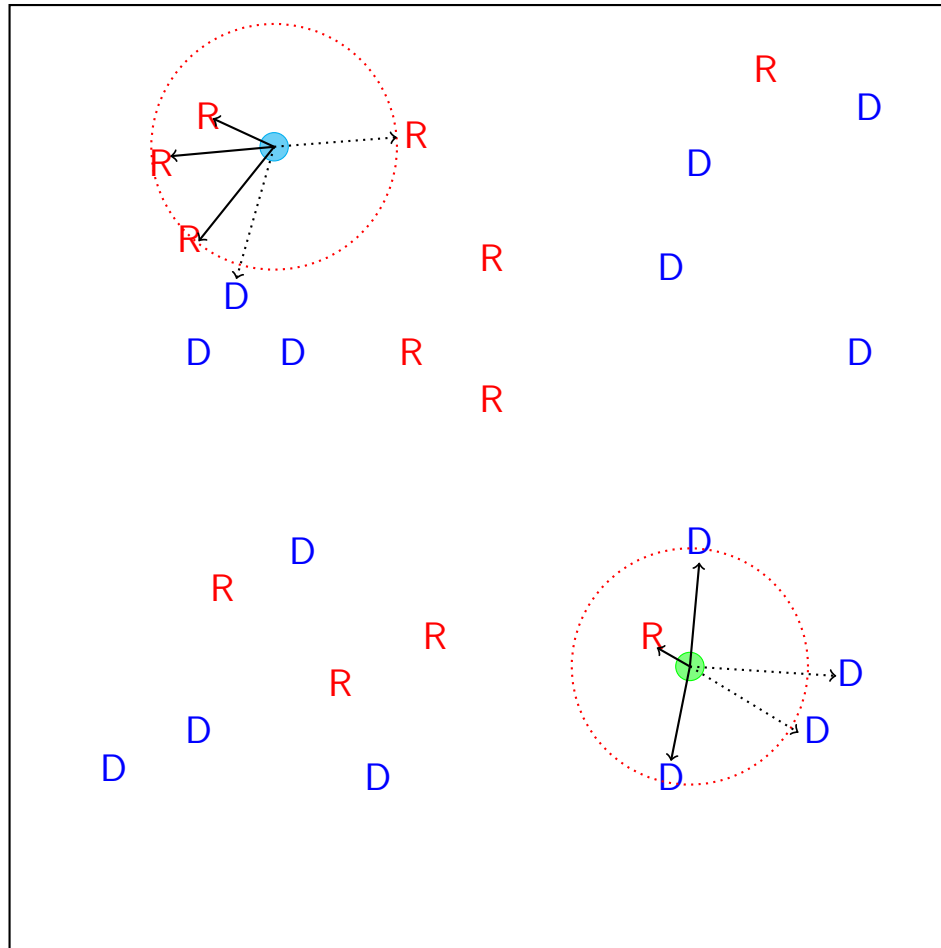
However, when we look at the bottom residence (green) we see that its nearest neighbor is Republican but the others surrounding it are Democrats. So if we use an algorithm to determine the political persuasion based on the single closest neighbor, the prediction would be Republican for both. However, 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.

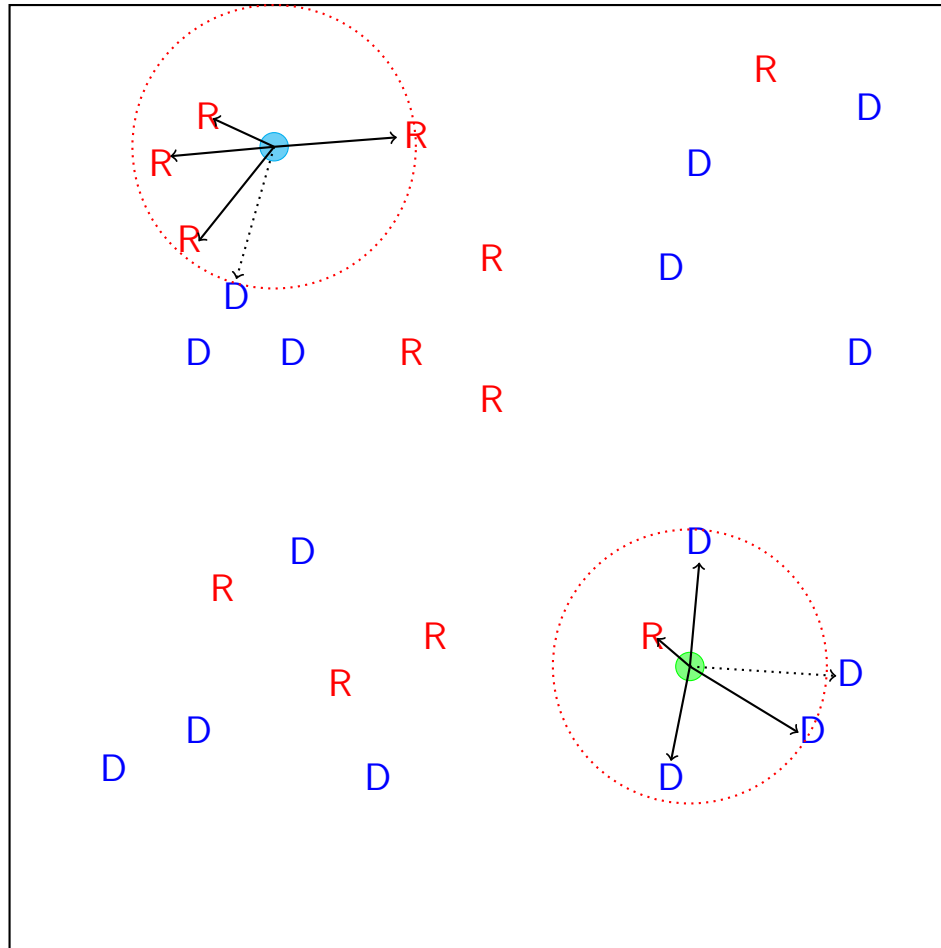


Top - 2 Republican neighbors \implies Republican
Bottom - 1 Democratic neighbor, 1 Republican \implies ?



Top - 3 Republican neighbors \implies Republican

Bottom - 2 Democratic neighbors, 1 Republican \implies Democrat



Top - 4 Republican neighbors \implies Republican

Bottom - 3 Democratic neighbors, 1 Republican \implies Democrat

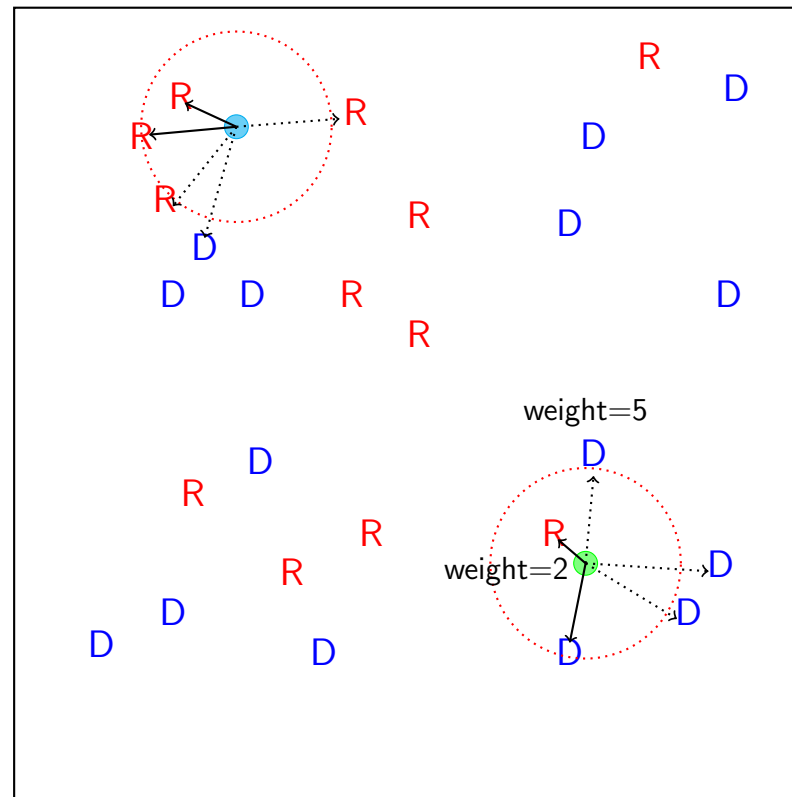
Variants of Nearest Neighbor Classifier

1. Nearest k neighbors called k -Nearest Neighbor Classifier
2. Weighted k -Nearest Neighbor Classifier

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 k -Nearest Neighbor 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.



Socratic Quiz - Practice Quiz 3

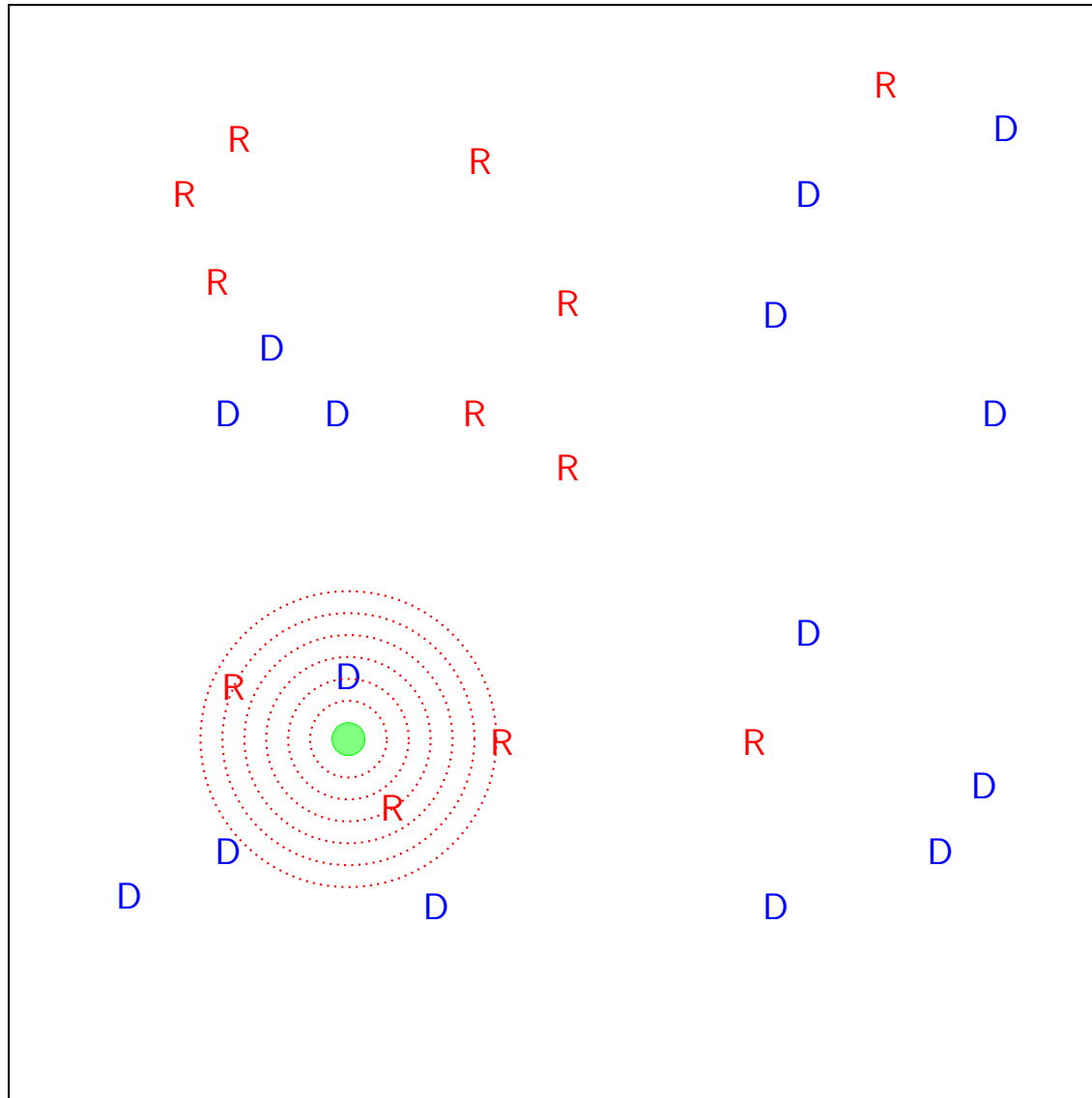
IUZGAZ34E

Use the schematic below to answer the following questions.

1. Using a single nearest neighbor classifier, how would the residence in question (marked in green) be classified?
 - (a) Democrat
 - (b) Republican
 - (c) unable to classify
2. Using a 2-nearest neighbor classifier, how would the residence in question (marked in green) be classified?
 - (a) Democrat
 - (b) Republican
 - (c) unable to classify

3. Using a 3-nearest neighbor classifier, how would the residence in question (marked in green) be classified?
 - (a) Democrat
 - (b) Republican
 - (c) unable to classify

4. Using 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?
 - (a) Democrat
 - (b) Republican
 - (c) unable to classify



How can we write an algorithm to determine the nearest neighbor if we using Euclidean length?

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.

The Euclidean distance between two points (x_1, y_1) and (x_2, y_2) is

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

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.

In our map of the houses in Midtown, we assume we have the coordinates of each house. So we can envision a **Brute Force** approach to finding the nearest neighbor.

Assume (x, y) are the coordinates of the house that we want to determine the political persuasion and the locations of the other houses are $(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots (x_N, y_N)$. Basically all we have to do is calculate the distance from (x, y) to each of the other N houses and pick the house 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

Set **min** = a big number

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

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

 If $d < \mathbf{min}$ then set **min** = d & set **point_no** = i

end loop

Example. Use the Brute Force Algorithm to determine the point from the list below that is closest to the point $x = (2, 5)$.

#1	(2.5,4.8)	#2	(2,4.4)	# 3	(1.5,5)	# 4	(2.5,4.9)
----	-----------	----	---------	-----	---------	-----	-----------

- 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 $i = 3$

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

$d < \text{min}=0.538516$ so set $\text{min} = d=0.5$ & $\text{point_no}=3$

- For $i = 4$

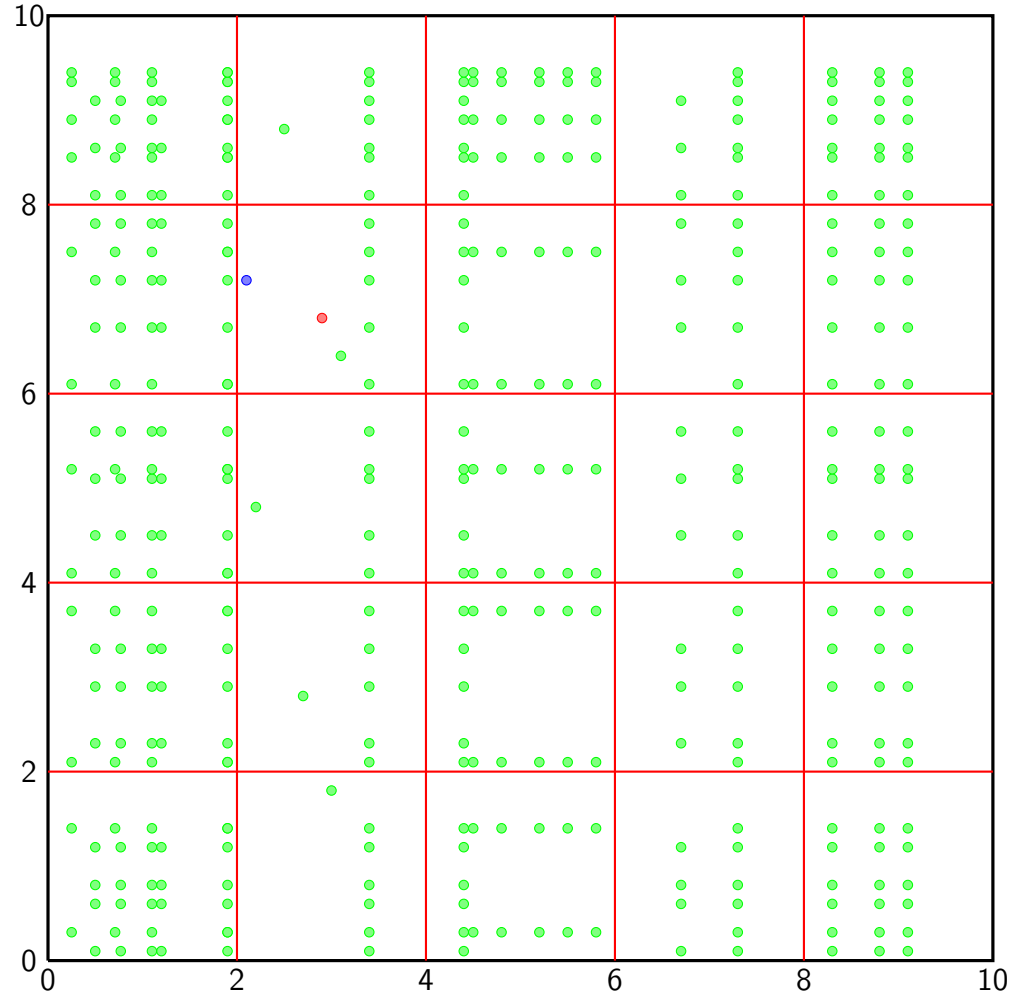
$$d = \sqrt{(2 - 2.5)^2 + (5 - 4.9)^2} = 0.5099$$

$d > \text{min}=0.5$ so do nothing

Answer: Point # 3 with a distance of 0.5

This 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. In this case typically the data is first sorted into points that are closest. It is beyond the scope of this course to go into these algorithms in depth but here is a graphical explanation of one called [binning](#).

Suppose we have a rectangular domain which contains all of the points. Then we divide our domain into “bins” or subdomains and determine which bin each data point is in. We know the x - and y -coordinates 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)?

Application of Nearest Neighbor Classifiers to Reading Postal Codes

In the previous example we just used our usual definition of distance to 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.

4 → 4	2 → 2	3 → 3
4 → 4	9 → 9	0 → 0
5 → 5	7 → 7	1 → 1
9 → 9	0 → 0	3 → 3
6 → 6	7 → 7	4 → 4

Now we have a handwritten number which we want to identify. 

What do we do?

We start with the first number in our training set and compare the two

$$\text{9} - \text{4} = \text{9}$$

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.



Example. Using the training set below, what is the nearest neighbor to the given object?



A



B



C



D



E



F



G



H



I



J



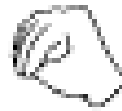
K



L



M



N



O



P



Q



R



S



T



U



V



W



X



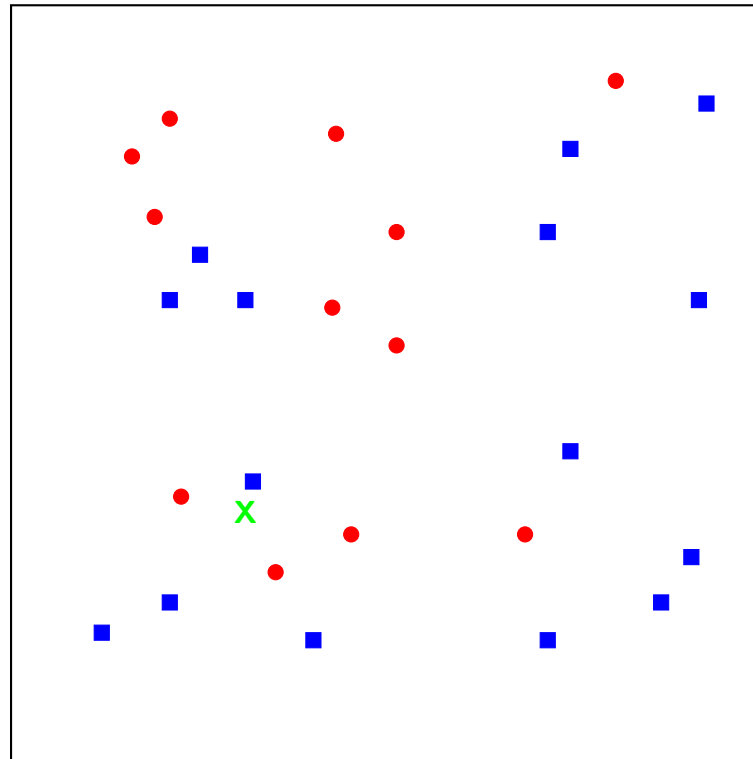
Y



Z



Classwork for k Nearest Neighbor Algorithm (KNN)



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. Fill in the table below. I have filled in the results for the first loop.

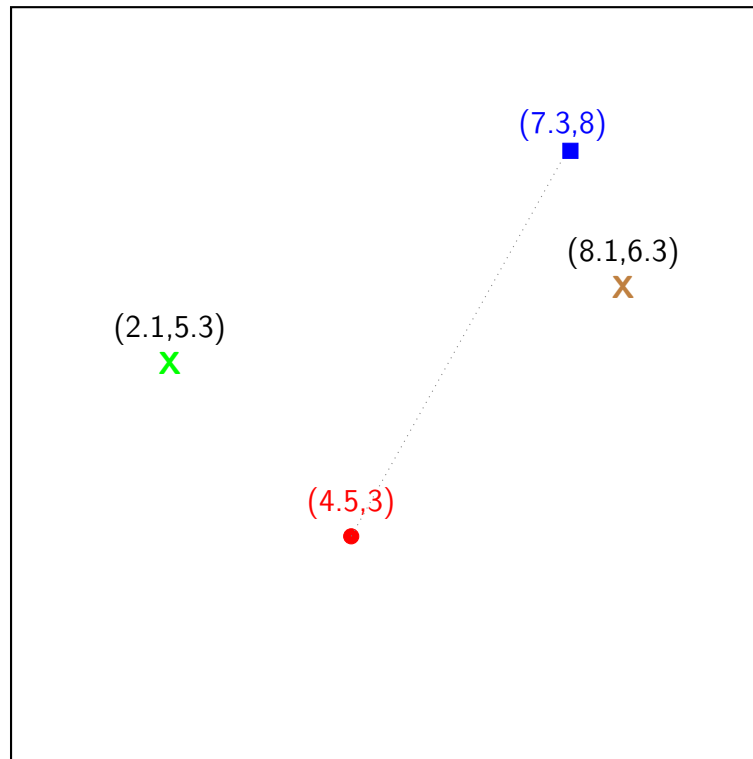
Points: (2,1), (4,2) (1,2) (0,1)

Set $\text{min} = 10$.

Loop #	distance between (2,3) and point	result	min	point_no
1	$d = \sqrt{(2 - 2)^2 + (3 - 1)^2} = 2$	$d < \text{min}$ so set $\text{min}=d$	2	1
2				
3				
4				

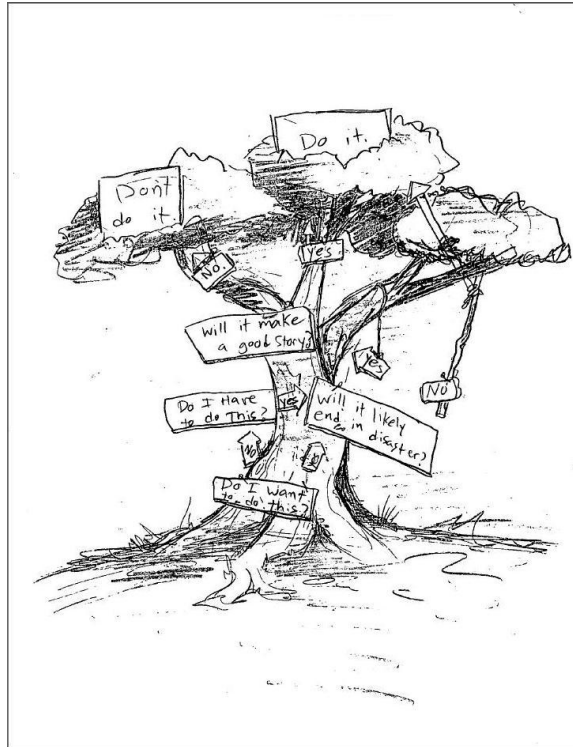
Answer: minimum distance is _____ at point number _____

3. Suppose we have the following box with one red circle and one blue rectangle. 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

What is a Decision Tree?



A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decision.

Example 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.

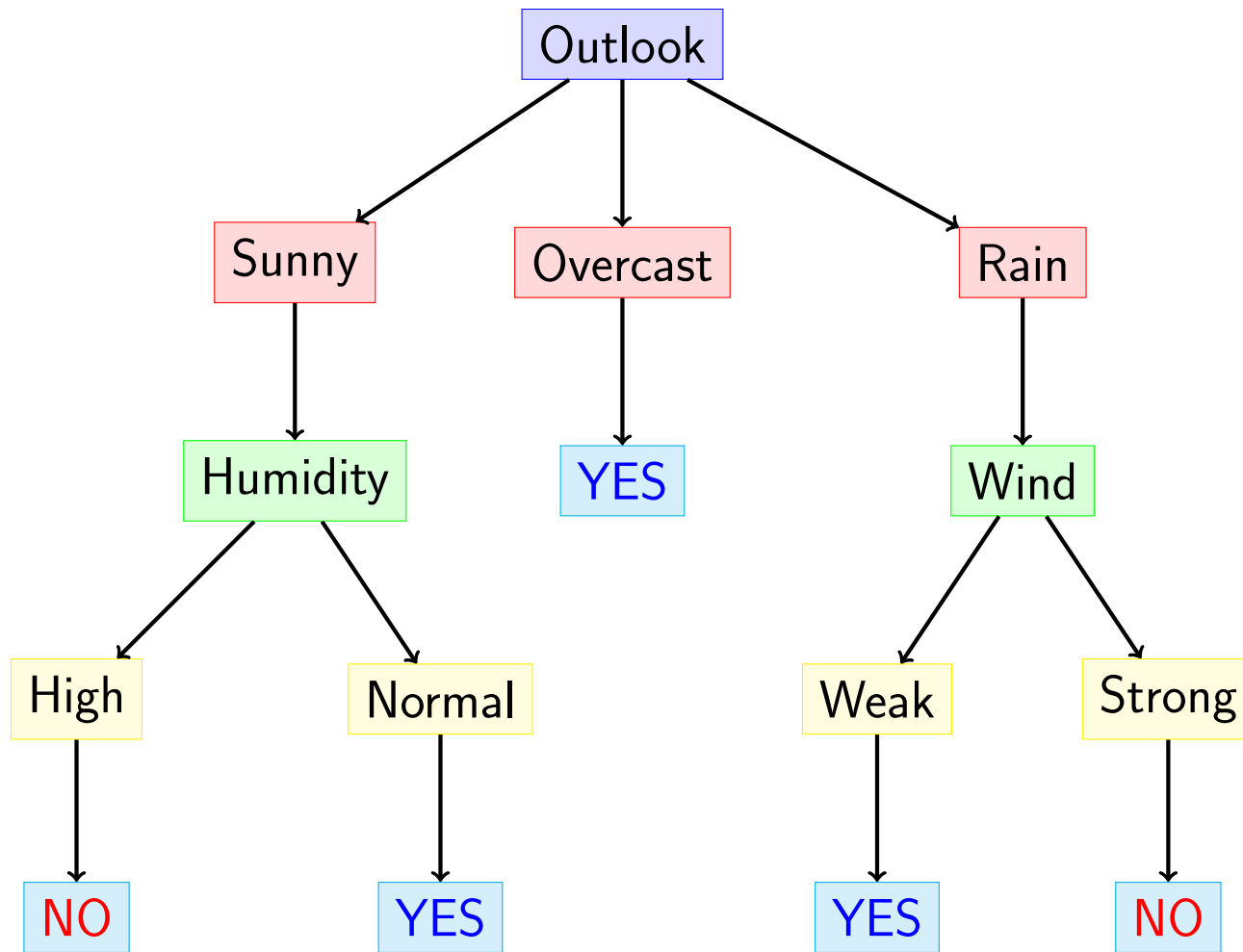
In this example, the attribute we are checking for the root node is the **Outlook** and the three options are **Sunny, Overcast or Rainy**.

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

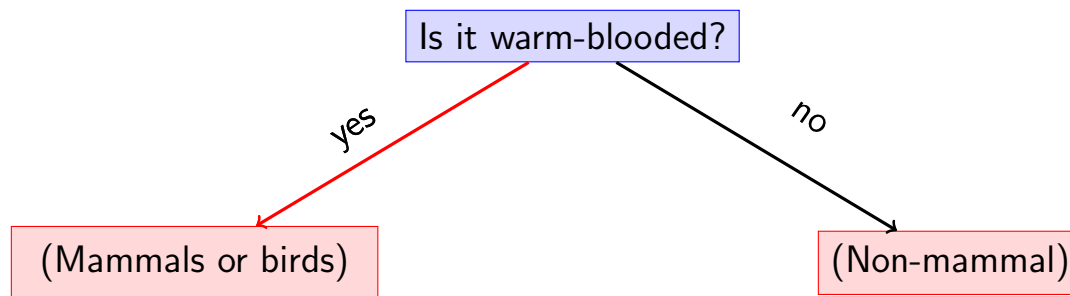
Example: 20 Questions Game



Example. Identify a particular vertebrate by asking 20 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 .

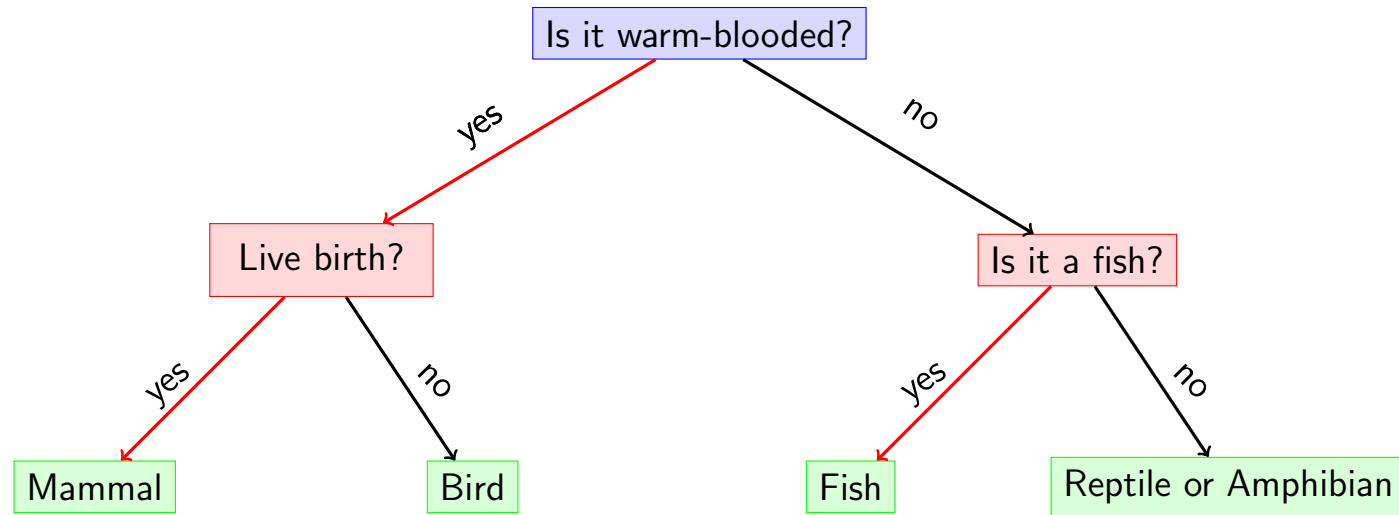


Now if the answer to the first question is “yes” then we know it is a mammal or a bird. There are many choices here for the next question. We could ask any of the following questions which give basically the same information.

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.

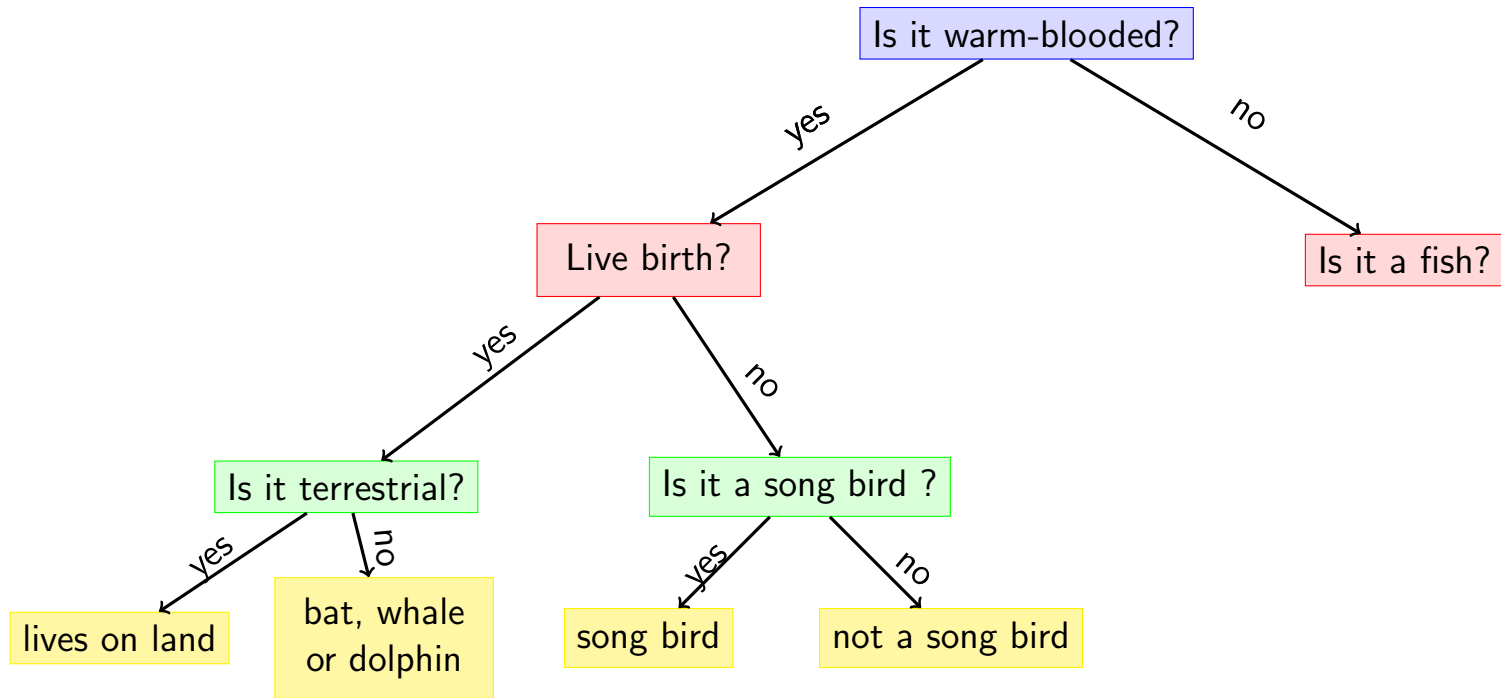


To get the correct identification we follow the Decision Tree starting at the root (the first question) based on the answer. For example, if the answer to the first question is “yes” and the answer to the second question is “no” we know that the vertebrate is a bird.

Socratic Quiz - PartV_Practice_Quiz4

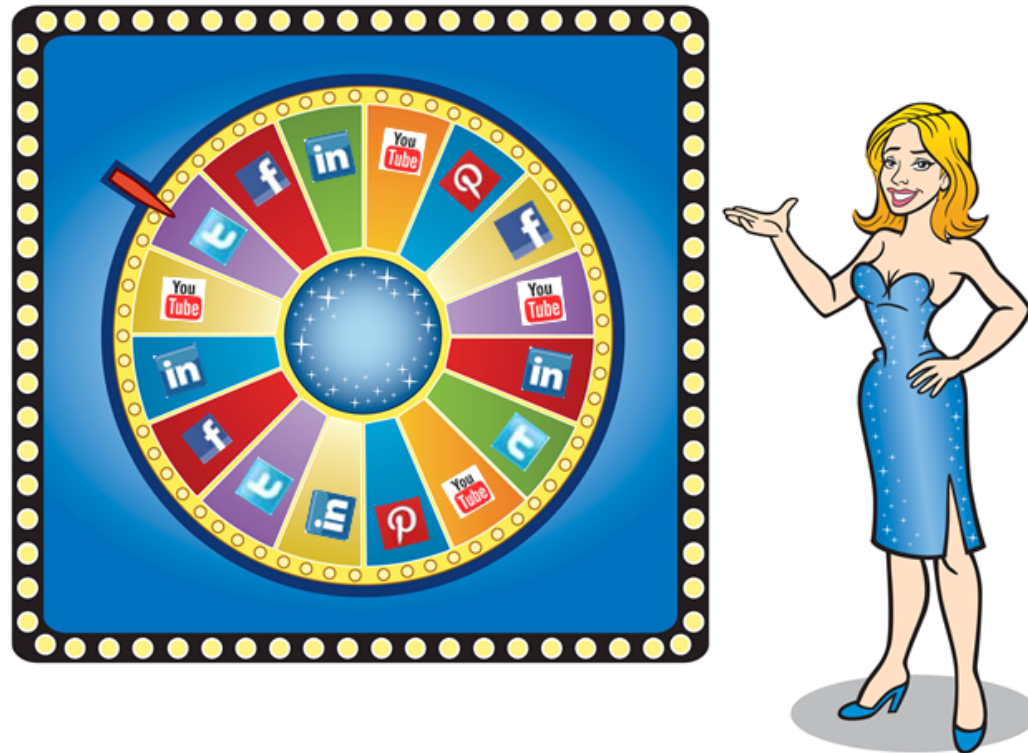
IUZGAZ34E

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



A Social Network Application of Decision Tree Classifier

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.



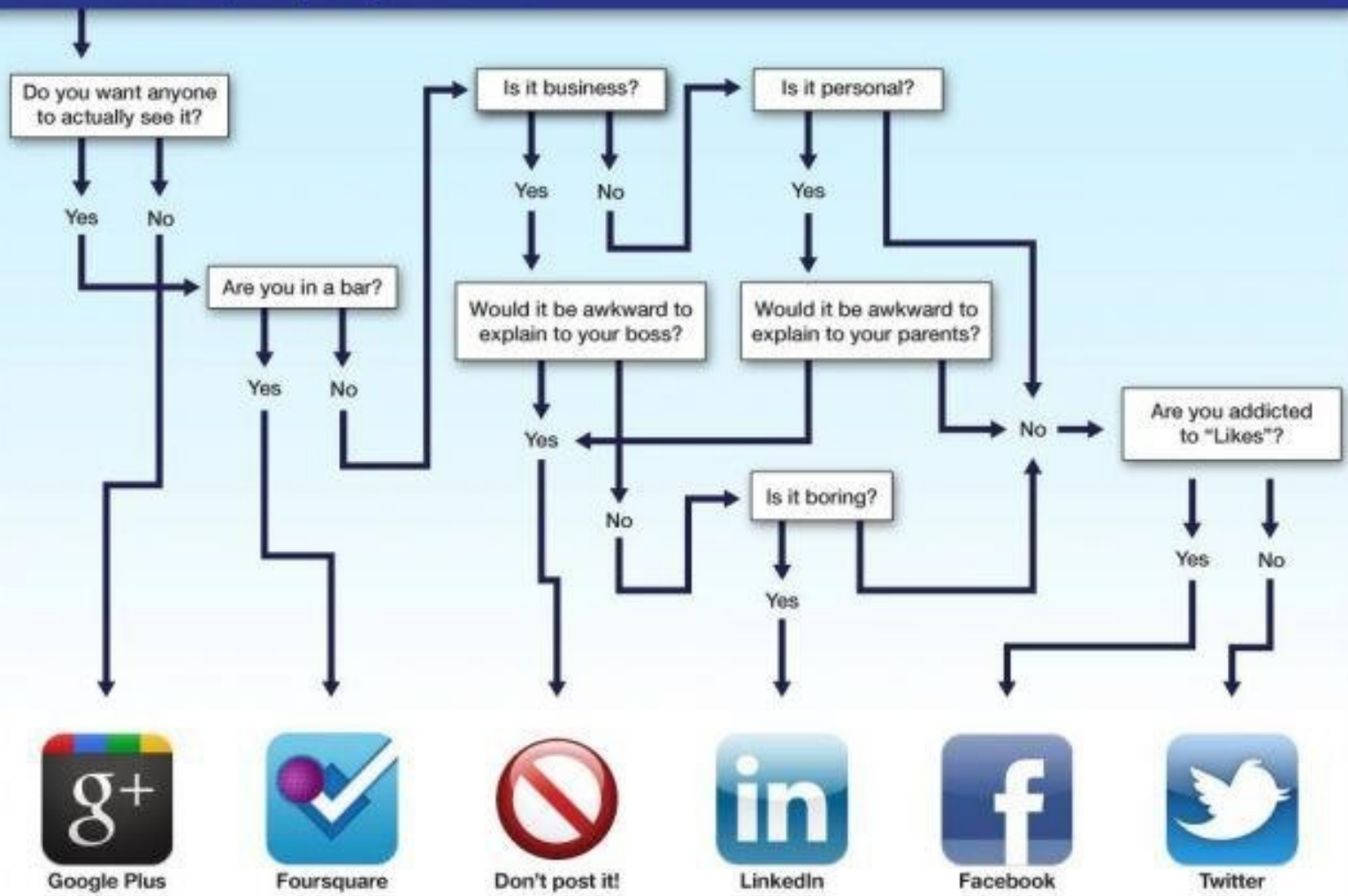
LinkedIn is a Social Network designed for the business community.

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?



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 better Decision Tree.

The ML goal for a Decision Tree is to

1. achieve perfect classification while
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? $14/100 = 14\%$

Now suppose our root or first query is *Is it a mammal?* What percentage of the training set is correctly identified with this question? $53/100 = 53\%$

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 leaf 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.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	yes
D2	Sunny	Hot	High	Strong	no
D3	Overcast	Hot	High	Weak	yes
D4	Rain	Mild	High	Weak	no
D5	Rain	Cool	Normal	Weak	yes
D6	Rain	Cool	Normal	Strong	no
D7	Rain	Cool	Normal	Strong	yes
D8	Sunny	Mild	High	Weak	yes
D9	Sunny	Cool	Normal	Weak	yes
D10	Rain	Mild	Normal	Weak	no
D11	Sunny	Mild	Normal	Strong	yes
D12	Overcast	Mild	High	Strong	yes
D13	Overcast	Hot	Normal	Weak	yes
D14	Rain	Mild	High	Strong	no

To summarize the data we will use the notation “+” to indicate a “Yes” classification (Play Tennis) and “-” to represent a “No” classification (Don’t Play Tennis) . 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 $\{9^+, 5^-\}$.

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 $\{5^+, 5^-\}$ 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 $\{9^+, 1^-\}$ 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 ($\{6^+, 0^-\}$); equivalently it could always classify as “Don’t Play Tennis”, i.e., $\{0^+, 6^-\}$ and it would be a good indicator. But if it classifies 3 as “Play Tennis” and 3 as “Don’t Play Tennis” , i.e., $\{3^+, 3^-\}$ 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 $\{3^+, 3^-\}$ whereas for a **Weak Wind** we have 6 positives and 2 negatives which we describe as $\{6^+, 2^-\}$. 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 $\{4^+, 1^-\}$

Overcast $\{3^+, 0^-\}$

Rain $\{2^+, 4^-\}$

Temperature

Hot $\{3^+, 1^-\}$

Mild $\{3^+, 3^-\}$

Cool $\{3^+, 1^-\}$

Humidity

High $\{4^+, 3^-\}$

Normal $\{5^+, 2^-\}$

Wind

Strong $\{3^+, 3^-\}$

Weak $\{6^+, 2^-\}$

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 out data set Outlook seems to be the best choices. To determine the best one of the two we would compute a numerical gain. Of course this choice may be different with a different training set.

There are 3 Outlooks: Sunny, Overcast, Rain. For each of these we have to decide whether Temperature, Humidity or Wind is a better indicator. For example, if Temperature always predicted correctly then we would use that. We continue in this way to build a Decision Tree.

Web Browser Application using Decision Tree Classifier

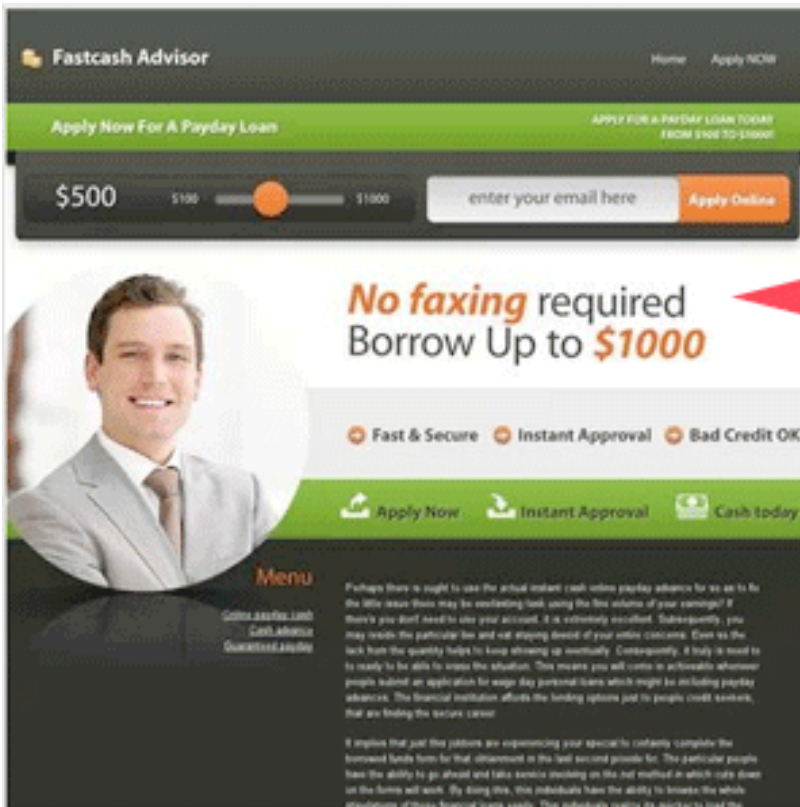
A pattern recognition problem where a Decision Tree Classifier is typically used is the identification of [web spam](#).

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.



These pages are examples of 'pure spam.' They appear to use aggressive spam techniques such as automatically generated gibberish, cloaking and scraping content from other websites.

← Screenshot of the removed page

↓ Snippet of the removed page

Removed from search results an hour ago

[Fast cash advance loans, instant cash online payday advance
http://www.fastcash-online.org/](http://www.fastcash-online.org/)

Perhaps there is ought to use the actual instant cash online payday advance for so as to fix the little issue there may be everlasting task using the fine volume of ...

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.

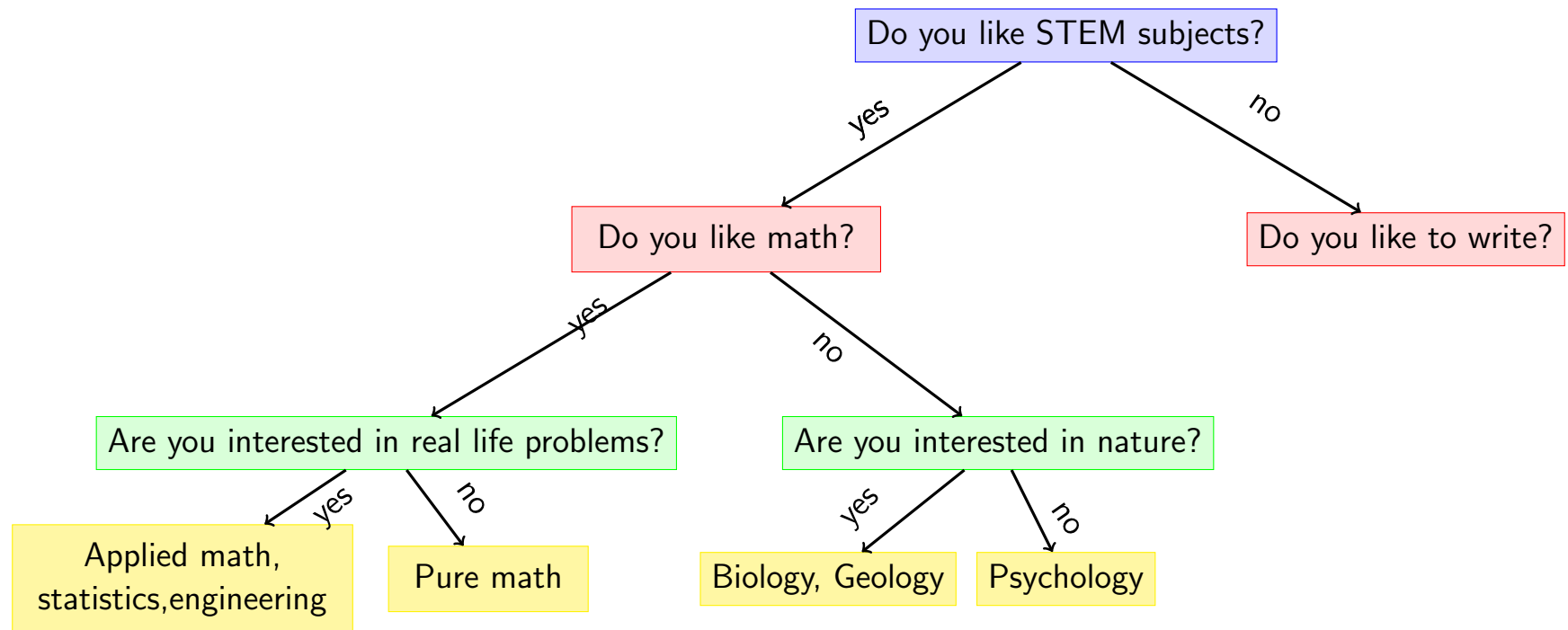
2. Number of words in the title.

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.

Classwork on Decision Trees



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.

< 2 yrs at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

(a) We want to summarize the two attributes (i) & (ii). We will indicate an applicant who did not default as “good” (a good risk) and one who defaulted as “bad” (a bad risk). For each of the two attributes summarize the data by filling in the table below. For example, the first data point in the training set didn’t miss any payments and didn’t default so that applicant is counted in the “good” category under “No” (for attribute Missed payments?) whereas the second didn’t miss any payments but defaulted so that applicant is counted in the “bad” category

under “No” or attribute Missed payments.

	Attribute: < 2 yrs on job?		Attribute: Missed payments?	
	Yes	No	Yes	No
good				
bad				

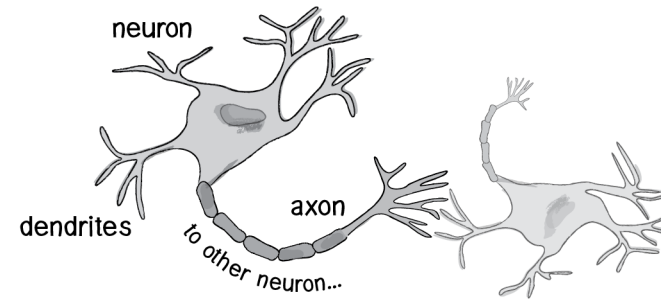
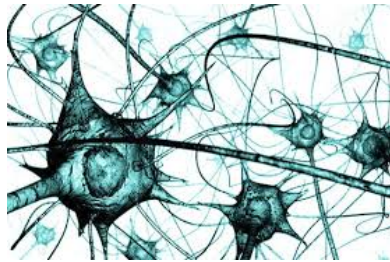
- (b) Based upon the results in (a) what do you think the root question should be? Why?
- (c) Make a Decision Tree with the root question from (b) and your second attribute.
- (d) We now want to use the Decision Tree from (c) to see the fraction of the entries in the training set that are good risks.
- 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 Nets

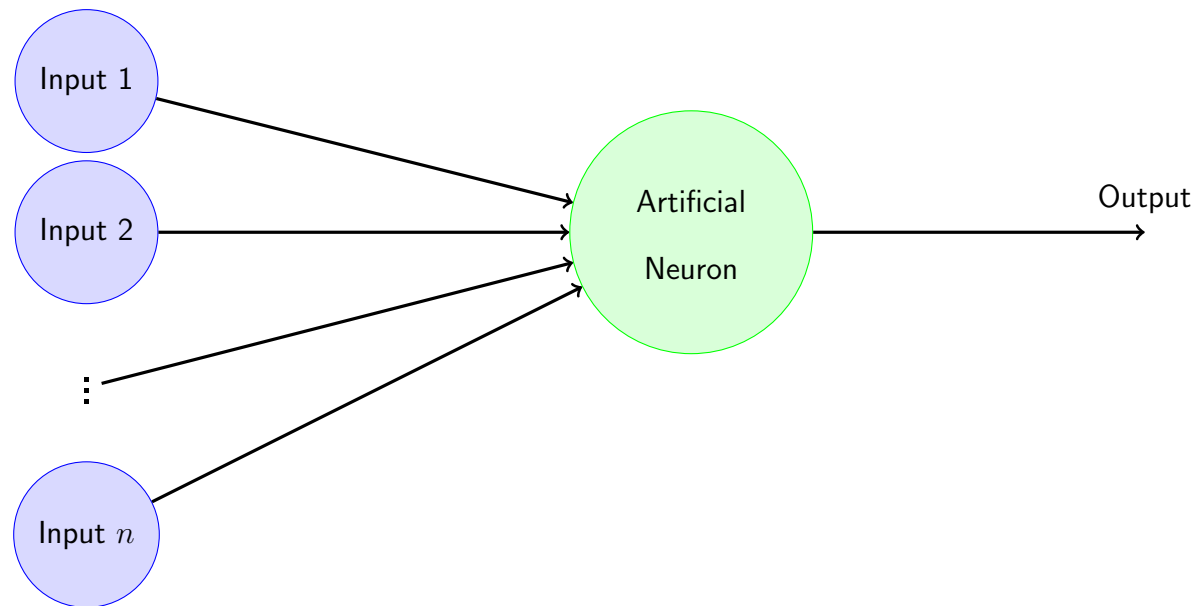
- Neural net algorithms are based on how our brain processes information.
- In 1943 a neuroscientist and a logician developed the first conceptual model of an artificial neural network.
- Neural net algorithms do NOT model how our brain works but they are **inspired** by how our brain works and designed 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.



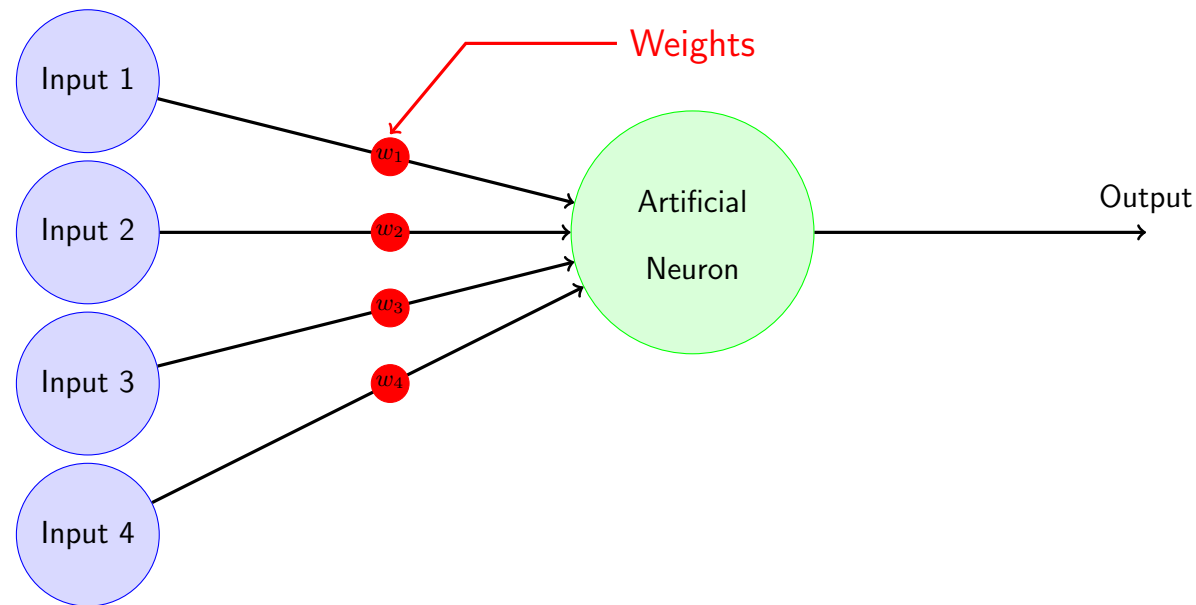
A neural network is a connectionist computational system. The algorithms we have encountered are serial in the sense that the program executes the first line of code, moves to the second, etc in a linear fashion. A true neural network does not follow a linear path but rather information is processed 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.



Artificial neuron with 4 inputs

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 value is \geq threshold (such as 0 or 1) then it “fires”; i.e., it gives a positive output.
- If the summed value 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 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.

Inputs		Letter Output	Numerical Output
Hours Prep	Hours Sleep		
3	5	C	65
5	2	B-	82
10	6	A	91
7	3	B+	84

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.

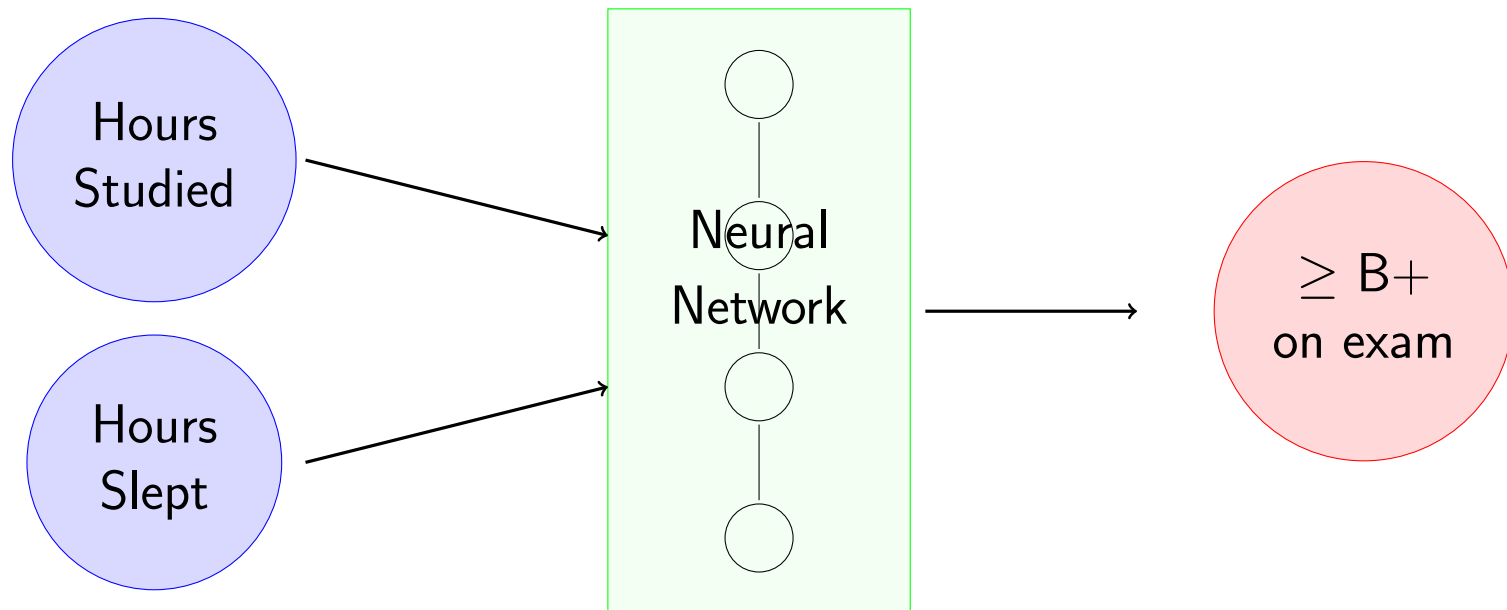
Assume that it is 7 pm on the night before a 10 a.m. exam and you are about to start studying. Let’s look at some options you have.

1. Study from 7 pm until midnight (with two 30 minute breaks) \implies 4 hours preparation and get 7 hours of sleep
2. Study from 7 pm until 2 am (with three 30 minute breaks) \implies 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+.

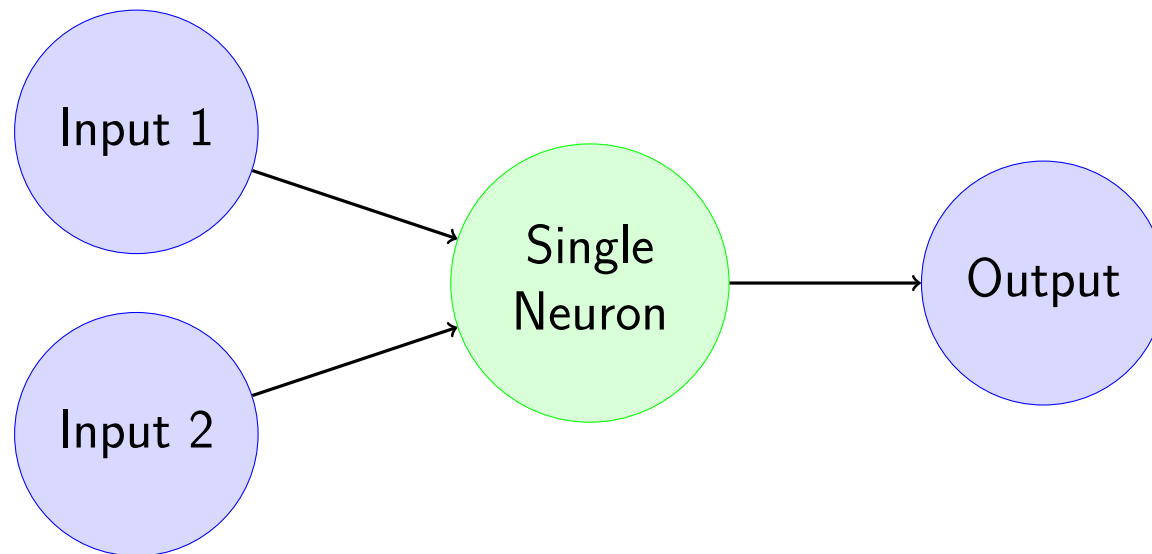


Perceptron

First we tackle a simplified problem where we only use a single neuron in the network.

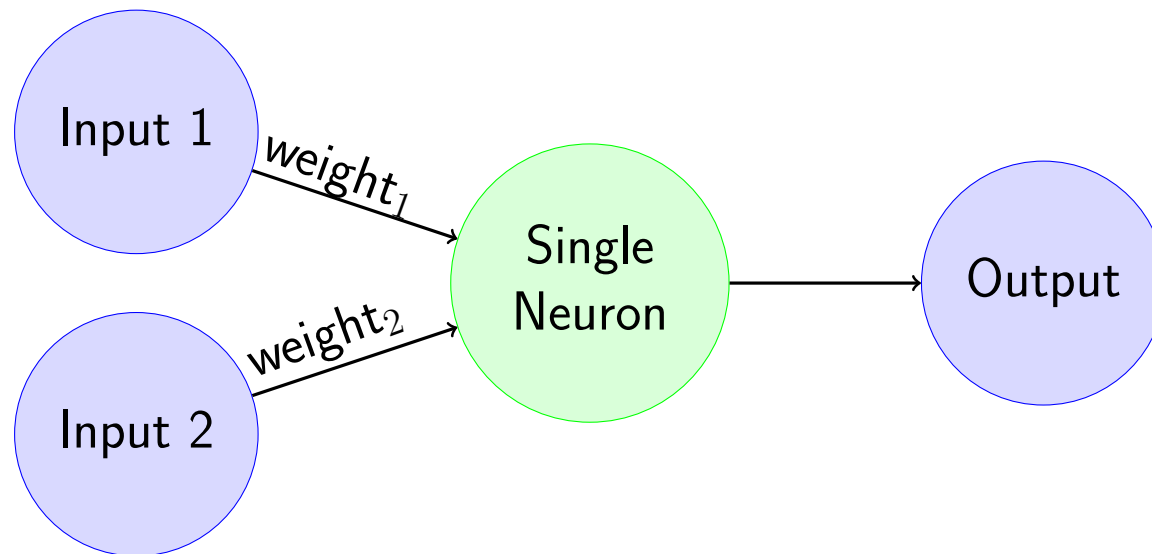
However you can have one or more inputs but a single binary output.

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.



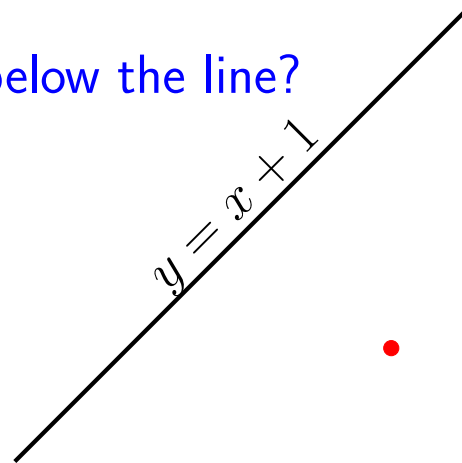
Now for each value of the input we have a weight and because we only have one “neuron” we do a single computation and then give a **binary output**. Recall that

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 the 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

$$(w_x \times x) + (w_y \times y)$$

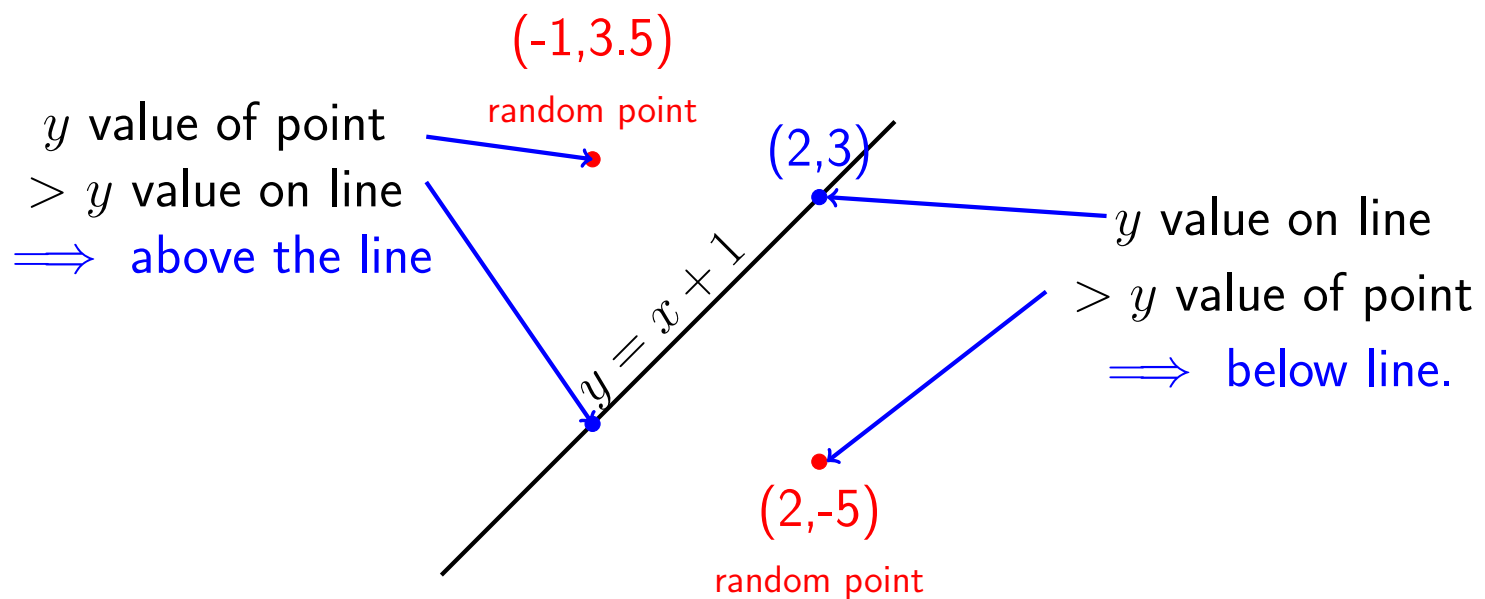
to get a number. But this can't be our output because we said our output is binary. We can simply take the sign of the output which means we are taking a threshold of zero. We take $+1$ if the sum is ≥ 0 and -1 if it is < 0 .

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 $w_x \times x + w_y \times y = 0$ 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 ≥ 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 ≥ 0 this means our weights are too large so we have to decrease them.
- If the sum should be ≥ 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 ± 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 ± 1 the error can only take on the values 0, -2, 2. To see this, look at the following 3 cases for the line $y = x$

Point	Actual	Prediction	Error = Actual - Predicted
(1,-2)	-1 (below)	-1 (below)	0
(3,5)	1 (above)	-1 (below)	2
(4,2)	-1 (below)	1 (above)	-2

We want to use the error to modify the weights so that they agree with our intuition. 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 ($\text{sum} < 0$) 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 ($\text{sum} \geq 0$) 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 scaling factor 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 $i = 1, 2, \dots, N$

(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 = Actual - Predicted

(vi) update weights by formulas

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

$$w_y = w_y + \text{error} \times y \times \text{learning rate}$$

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 $j = 1, 2, \dots, J$

(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

Point	Term t	Actual output	Predicted output	Error	New Weights
(1.98, -1.06)	-1.18	-1	-1	0	-0.995 0.134
(-0.15, -3.41)	0.63	-1	+1	-2	-0.992 0.202
(-3.96, -1.22)	4.59	+1	+1	0	-0.992 0.202
(-1.26, -2.26)	1.70	-1	+1	-2	-0.967 0.247
(-2.93, 3.20)	4.52	+1	+1	0	-0.967 0.247

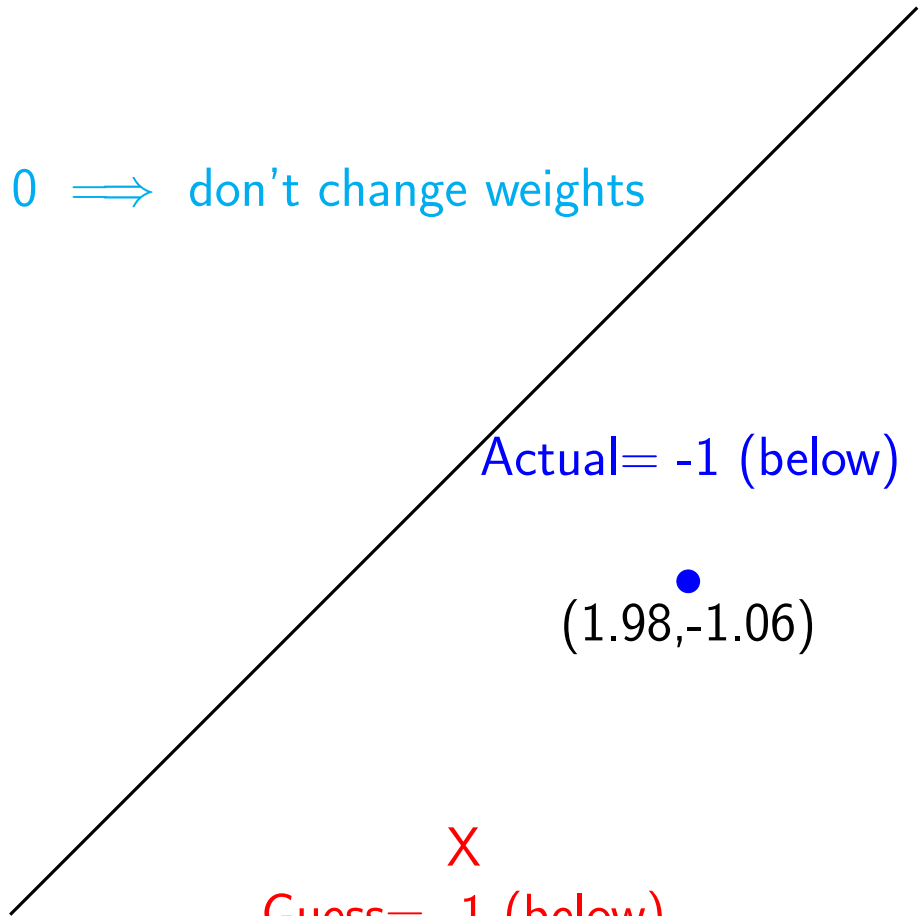
Error = 0 \implies don't change weights

Actual= -1 (below)

(1.98, -1.06)

X


Guess= -1 (below)



Error = 2 \implies 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.

Training Set Size	Percent Correct
10	78%
25	83%
50	83%
100	88%

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. 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.

Training Set Size	Number of Epochs	Percent Correct
10	4	90%
25	15	97%
50	13	100%

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 $i = 1, 2, \dots, N$

(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 = Actual - 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 $j = 1, 2, \dots, J$

(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.

Socratic Quiz - PartV_Practice_Quiz5

IUZGAZ34E

Suppose we use our Perceptron algorithm to determine if a point lies above/on or below the line

$$y = 2x - 3$$

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

(c) +2

(d) -2

(e) -1

3. 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. Then which formula do we use to modify the weight for the x coordinate, w_x ?

(a) $w_x = \text{error} \times x \times \text{training rate}$

(b) $w_x = w_x + \text{error} \times w_x \times \text{training rate}$

(c) $w_x = w_x + \text{error} \times x \times \text{training rate}$

(d) $w_x = w_x - \text{error} \times w_x \times \text{training rate}$

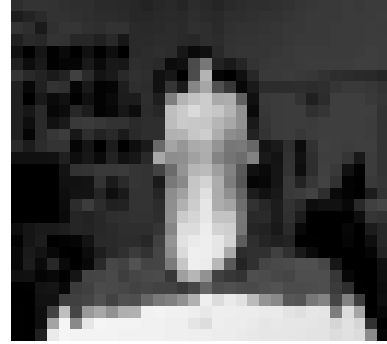
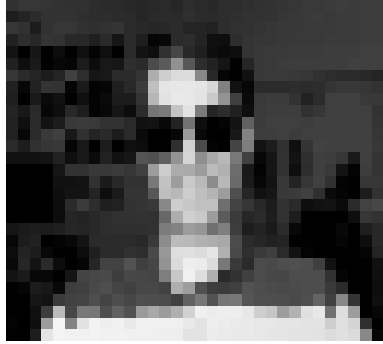
(e) $w_x = w_x - \text{error} \times x \times \text{training rate}$

(f) $w_x = - \text{error} \times x \times \text{training rate}$

(g) none of the above

Neural Net Algorithms for Facial Recognition

- 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.



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.

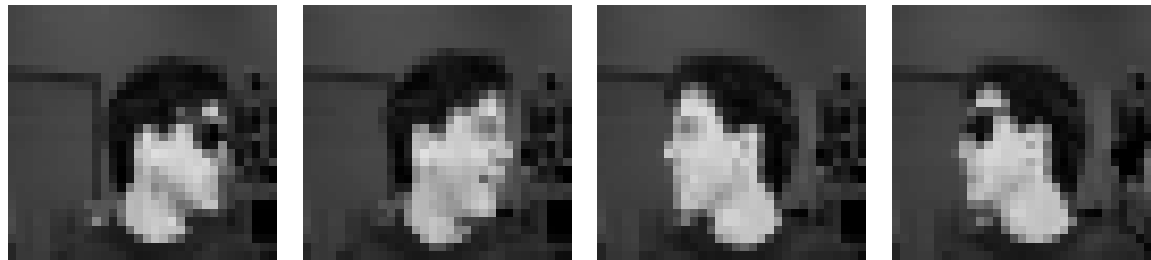
For example, it is possible to get 100% correct without training but of course that would be pure luck and so the confidence level should be low which means we expect a large error. Notice that when it only got 14% of the photographs correct it has a much larger error whereas when it got 100% correct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

Epoch	Percent Correct.	Error
0	4.2857	0.109234
1	94.2857	0.0139558
2	94.2857	0.0123471
3	94.2857	0.0109007
4	94.2857	0.00893908
5	97.1429	0.00706015
6	100	0.00571391
7	100	0.00475462

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.

Epoch	Percent Correct	Error
1	92.9577	0.0168988
2	92.9577	0.0151587
3	92.9577	0.0134871
4	92.9577	0.0113244
5	97.1831	0.00909995
6	98.5915	0.00742321
7	98.5915	0.00615335
8	98.5915	0.00514712
9	98.5915	0.00433465

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!

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.

Point	Term t	Actual	Predicted	Error	Weights	
		Output	Output		w_x	w_y
(4,2)						
(1,0)						
(1,-3)						
(0,-2)						
(2,2)						

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:

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

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.

Point	Does point lie above or below line	Term t	Prediction for point	Error
$\left(2, \frac{1}{2}\right)$				
$(-1, -3)$				
$(4, 1)$				

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?