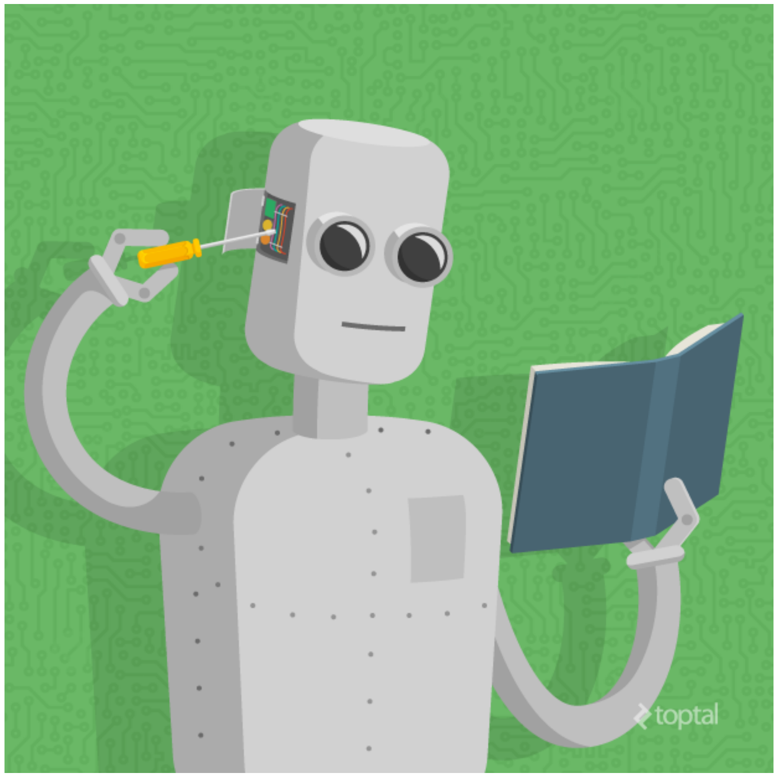
PART ?? - MACHINE LEARNING



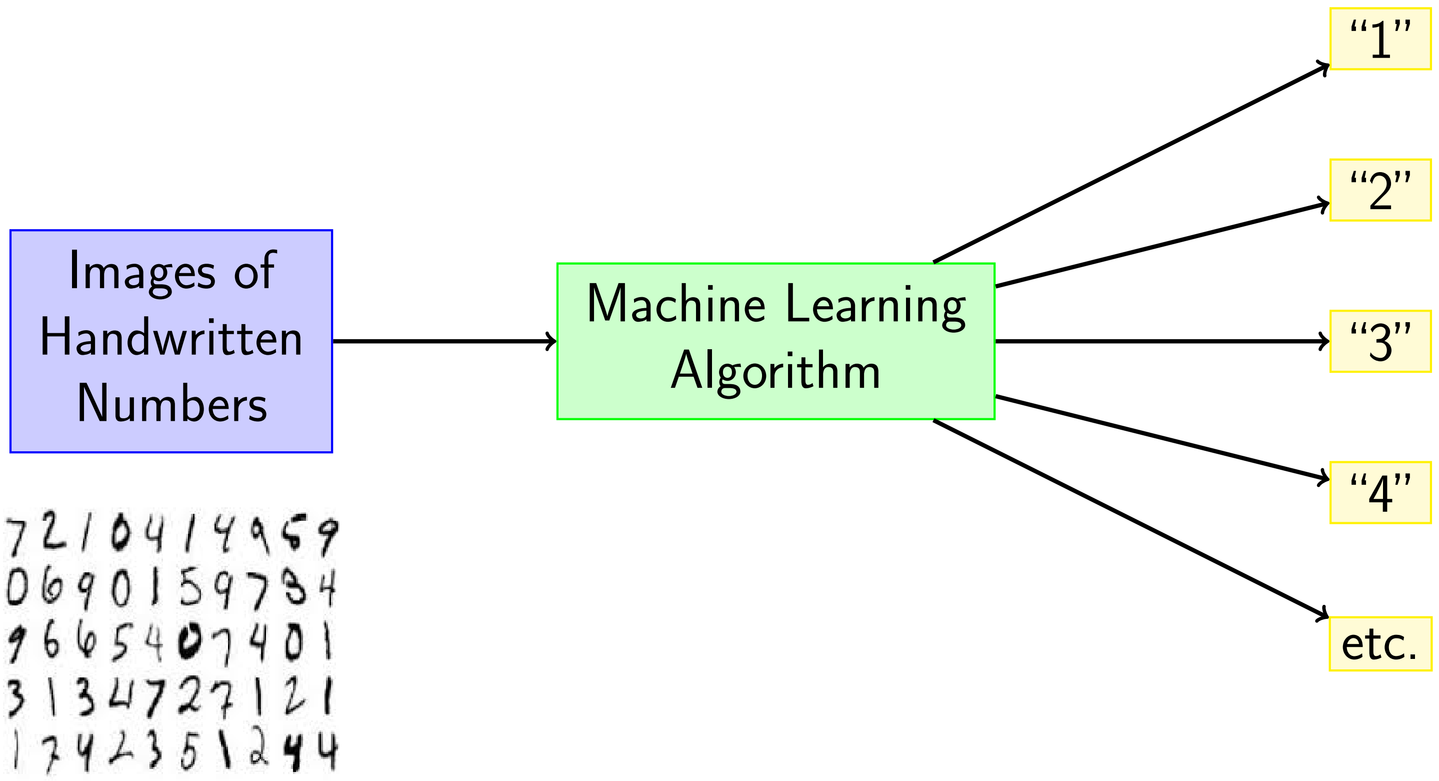
What is Machine Learning?

- 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 a custom code we feed data into the generic algorithm and it builds its own prediction based on the data.

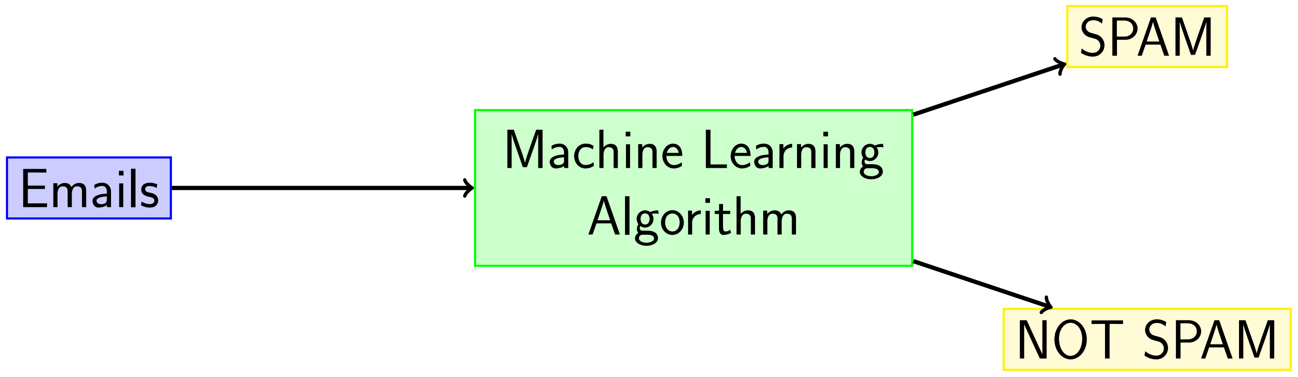
- So instead of writing many custom programs, we write a generic ML program which can work on a variety of problems.
- ML can be used to solve problems where other standard methods don't work.
- For example, suppose you want to write a computer program to predict traffic patterns at a busy intersection 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.

In What Real-World Problems Might ML be Useful?

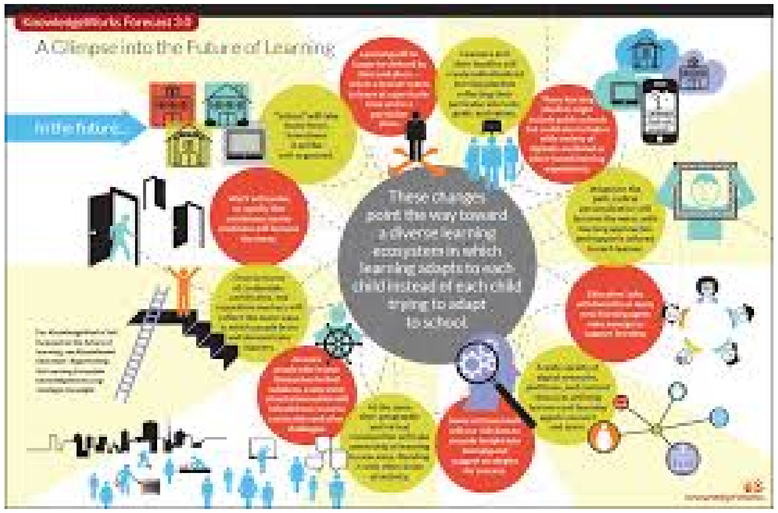


Training an algorithm to read a zip code



Classifying email messages

- Predicting prognosis of cancer patients with specific type, size, and spread of tumors.
- Autonomous Land Vehicles (ALVINN); Robotics Institute, CMU
- Classroom of the Future where each student will be assessed over the course of their education, helping students master the skills critical to meeting their goals. A system fueled by sophisticated analytics over the cloud will help teachers predict students who are most at risk, their roadblocks, and then suggest measures to help students overcome their challenges.


KnowledgeWorks Forecast 3.0
A Glimpse into the Future of Learning
In the future
These changes point the way toward a diverse learning ecosystem in which learning adapts to each child instead of each child trying to adapt to school

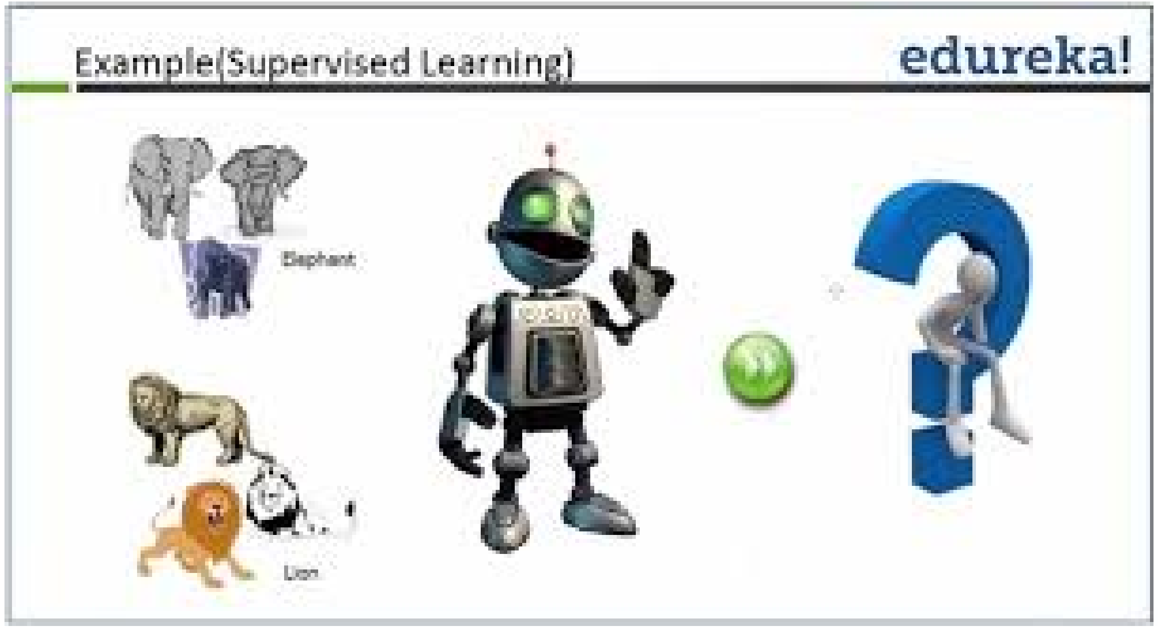
Two Distinct Types of Machine Learning Algorithms

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 classificiation 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.



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

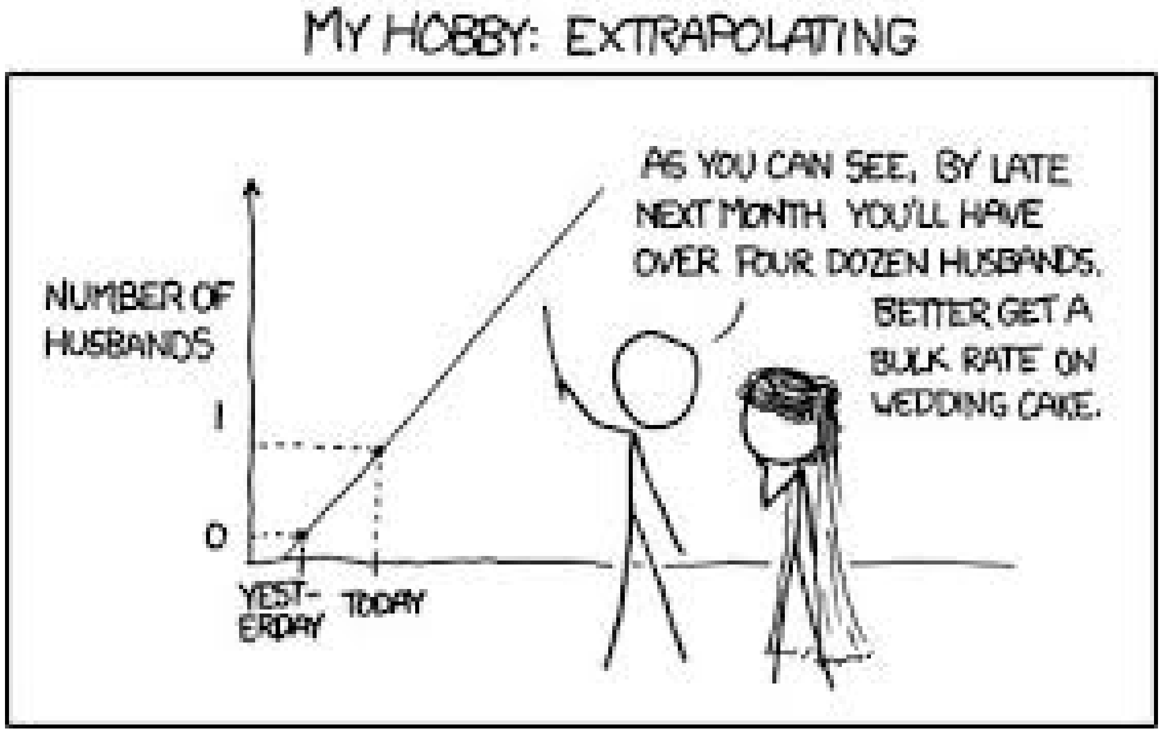
1. quality of the training set
2. choice of criteria used in training set

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
NUMBER OF HUSBANDS
1
0
YEST-ERDAY
TODAY
AS YOU CAN SEE, BY LATE NEXT MONTH YOU'LL HAVE OVER FOUR DOZEN HUSBANDS. BETTER GET A BULK RATE ON WEDDING CAKE.

Socrative Quiz

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

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

If we only use one criteria in the training set, for example the amount of living space, then we expect that the result might not be good.

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. 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?

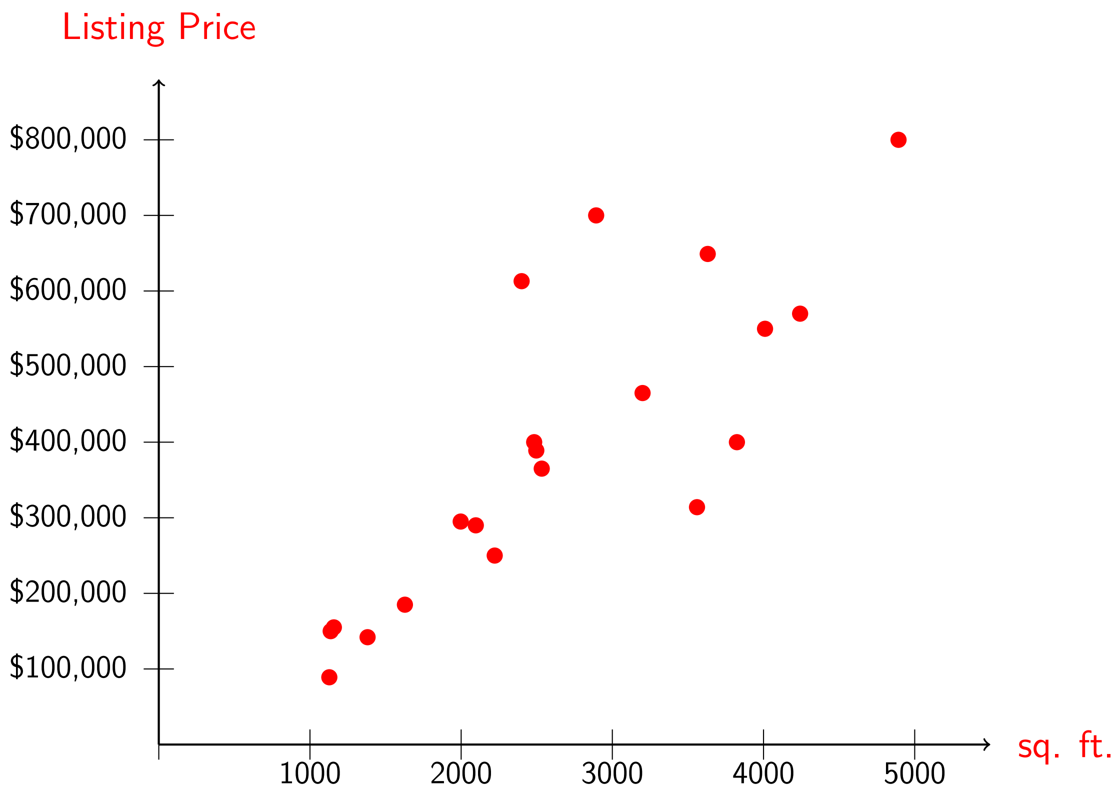
There are various programs (such as `import.io`) which “scrape” data from the web and put into a spreadsheet for use.

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.



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.

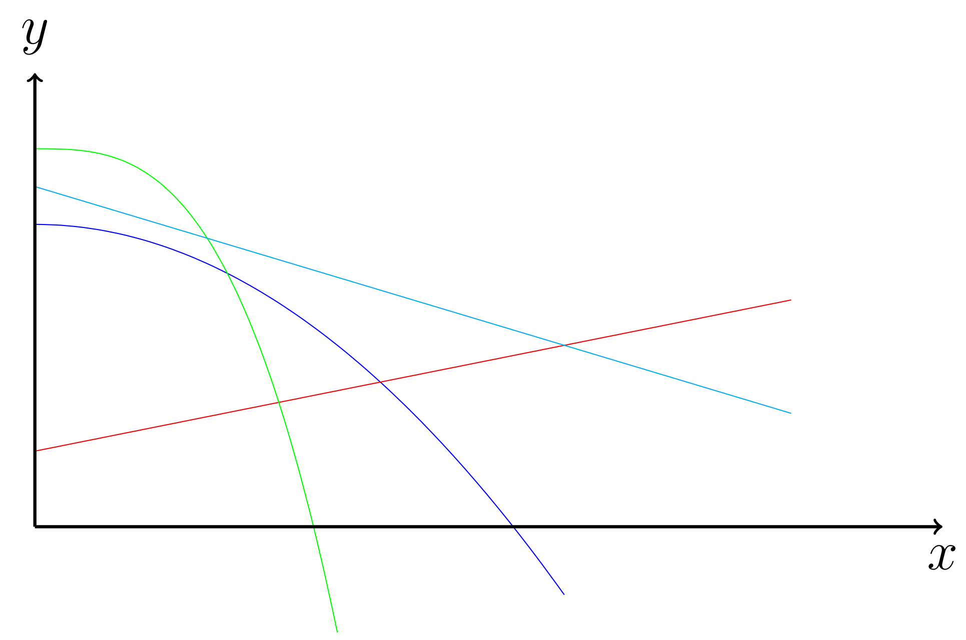
What does "depend linearly" mean?

First, we know that when we plot a linear function we get a straight line. 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.

Below are some plots of functions; determine in which cases y depends linearly on x.



As an example, 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. 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 \$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?

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

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?

$$y(x+\Delta x) - y(x) = \big[1 + 2(x+\Delta x)\big] - \big[1+2x\big] = \big[1 + 2x + 2\Delta x\big] - \big[1+2x\big] = 2\Delta x$$

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

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

Another example of something depending linearly which you have encountered is Newton's Second Law of Motion. This says that the force F acting on an object is given by the mass times the acceleraton a in meters per second squared of the object, i.e., $F = ma$. This says that for an object weighing 5 lb if we plot the acceleration on the x-axis and the force $(=5a)$ on the y-axis we get a straight line with slope 5. Similarly if the acceleration is fixed, then the force depends linearly on the mass of the object. To get twice as much force you need an object twice as large.

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

Socrative Quiz

1. If we have the line $y = 5x - 1$ and we change x by two, then how much does y change by?
   (a) 5
   (b) 6
   (c) 9
   (d) 10
2. 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 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 let's formulate what we mean by listing price depends linearly on the total square feet of 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

$$\boxed{P = ax + b}$$

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 P for any value of x we need to know a, b.

To understand how we determine a, b we first 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

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

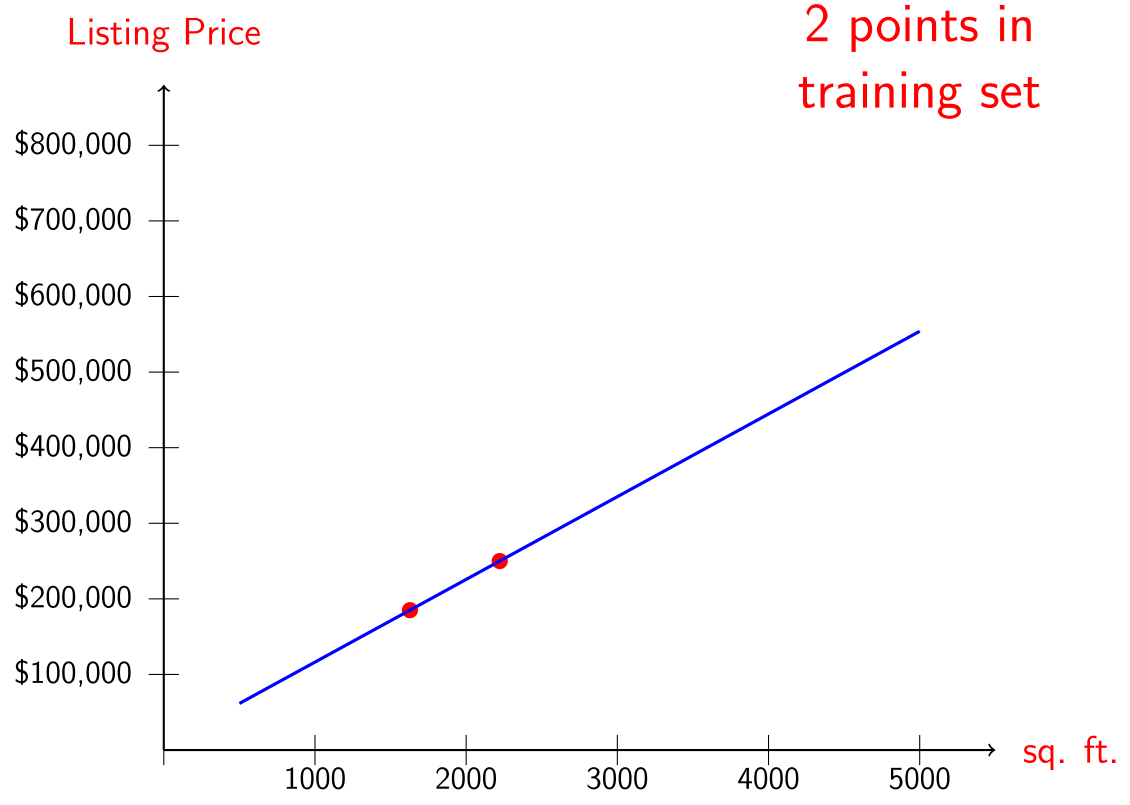
To find b we use either of the two equations and substitute this value in for a; i.e.,

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

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

$$P = 6851.85 + 109.428x$$

.



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 $P = 6851.85 + 109.428x$. For example,

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

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	$ 116,280	$ 116,280	$ 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

So we can conclude that we definitely need more data in the training set.

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

Socrative Quiz - Basic Concepts from Lecture

Answer true (“T”) or false (”F”).

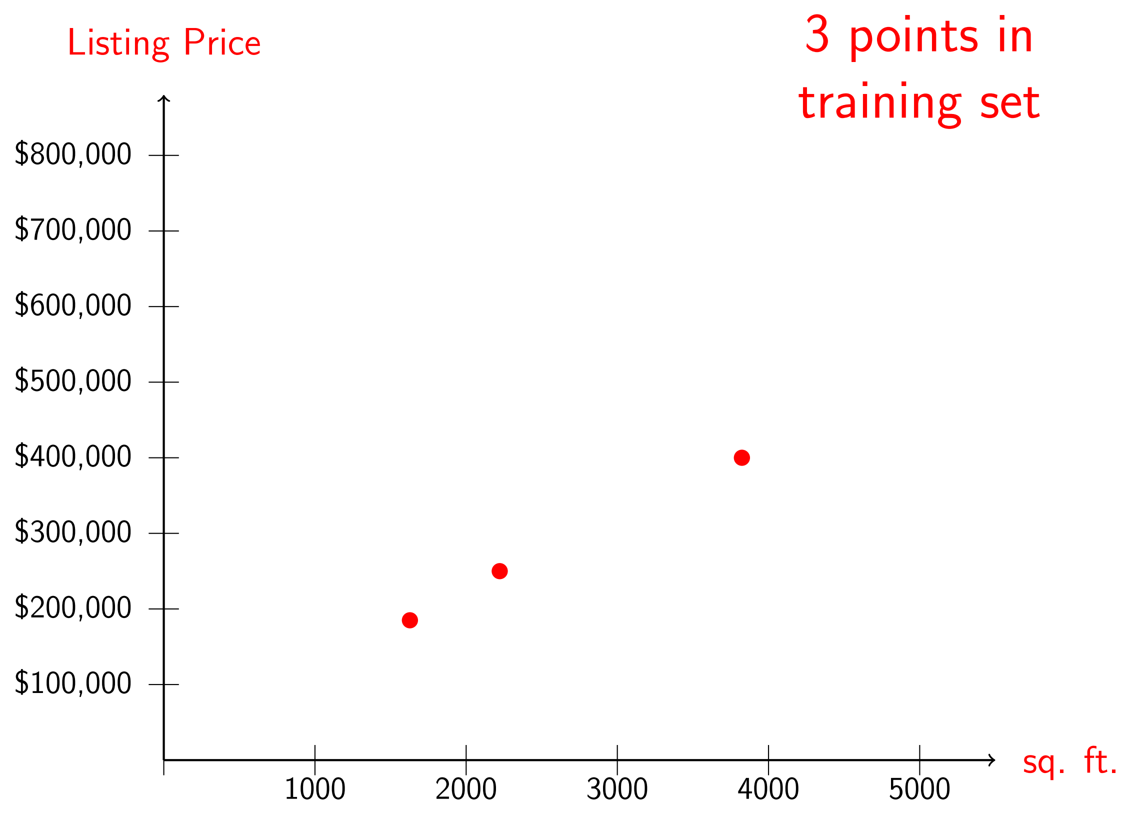
1. Machine Learning is a type of Artificial Intelligence.
2. An example of a Machine Learning algorithm is public key encryption.
3. An example of a Machine Learning algorithm is pattern recognition.
4. To predict accurate values for the listing price of a house between \$100,000 and \$1,000,000 it is enough to have a training set with houses which list between \$100,000 and \$500,000.
5. If y depends linearly on x then its plot will be a parabola.
6. If $y = 8 + 7x$ then when x is changed by -2 then y will change by 8+7(-2)=-6.
7. If $y = 5x$ and $w = 7+5x$ then $w(x)$ and $y(x)$ will change the same amount when x is changed by 2.

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?

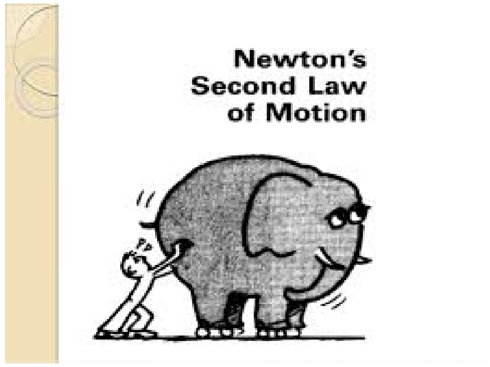
To see this, we begin by using 3 entries in the training set. We know that with 3 points they probably don't lie on a line. Below is a table containing our data and a scatter plot of the data. Clearly, they don't lie on a line.

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



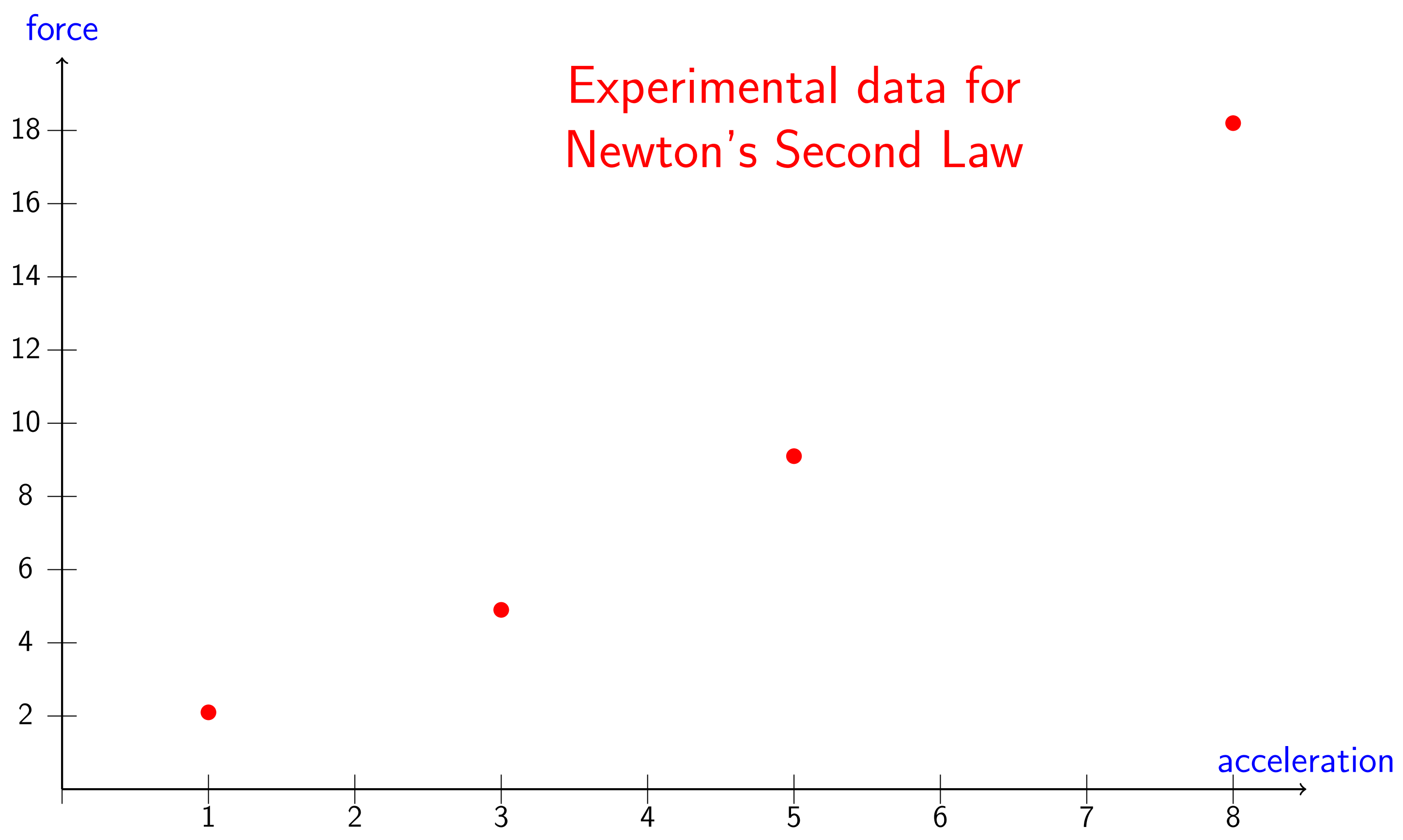
Since the points do not lie on a line, we can't find values for a and b so that the line $ax + b$ passes through all 3 points! What can we do?

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 acceleraton 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.

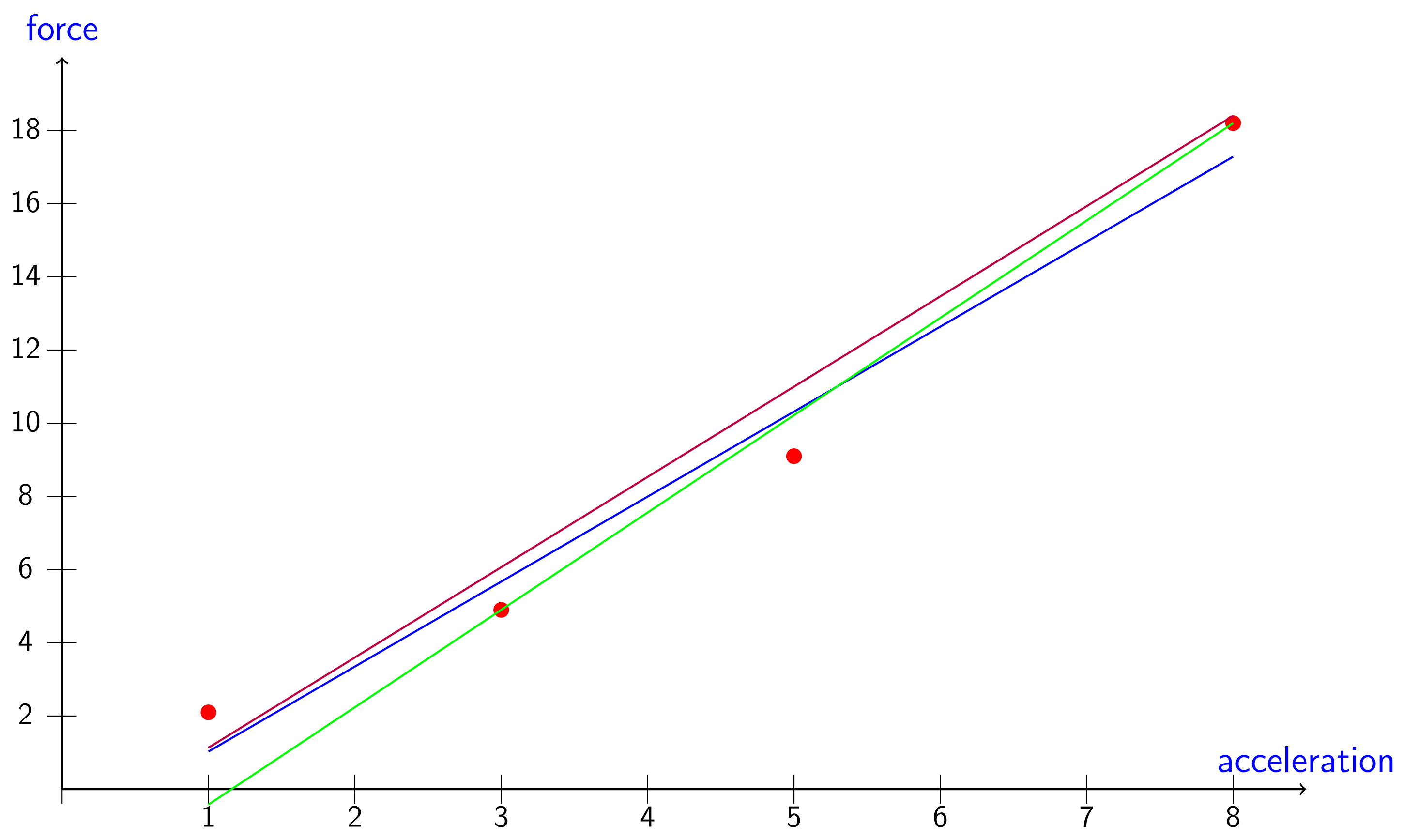


In our case $F = 2a$. Suppose you are in a lab taking measurements of the force on the object for 4 different values of the acceleration. If it was a perfect world then all your points would lie on a straight line but we all know measurements are susceptible to errors. Instead suppose you took the following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



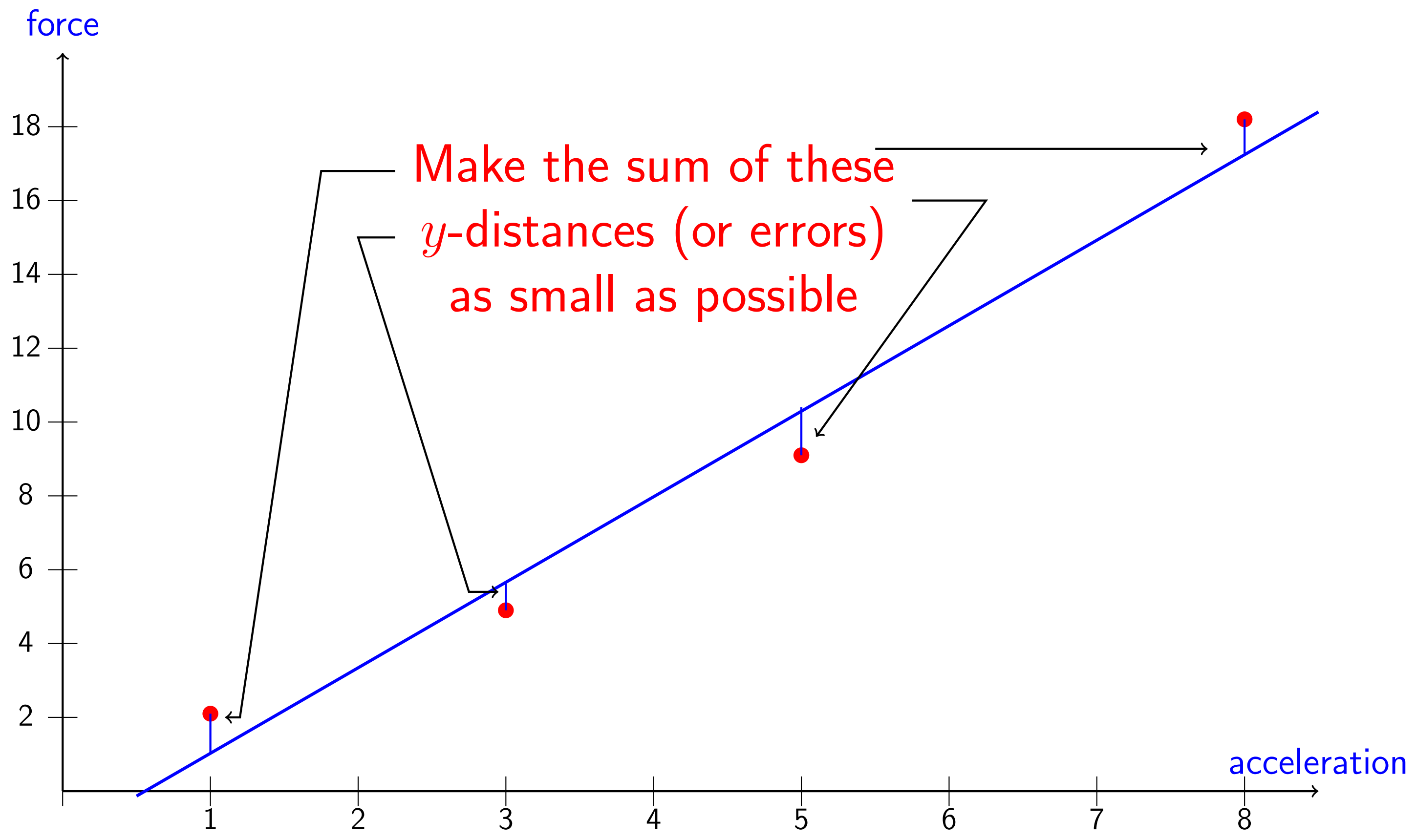
Due to measurement errors, the data does NOT lie in a straight line but we want to find a line which represents the data.



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

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

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

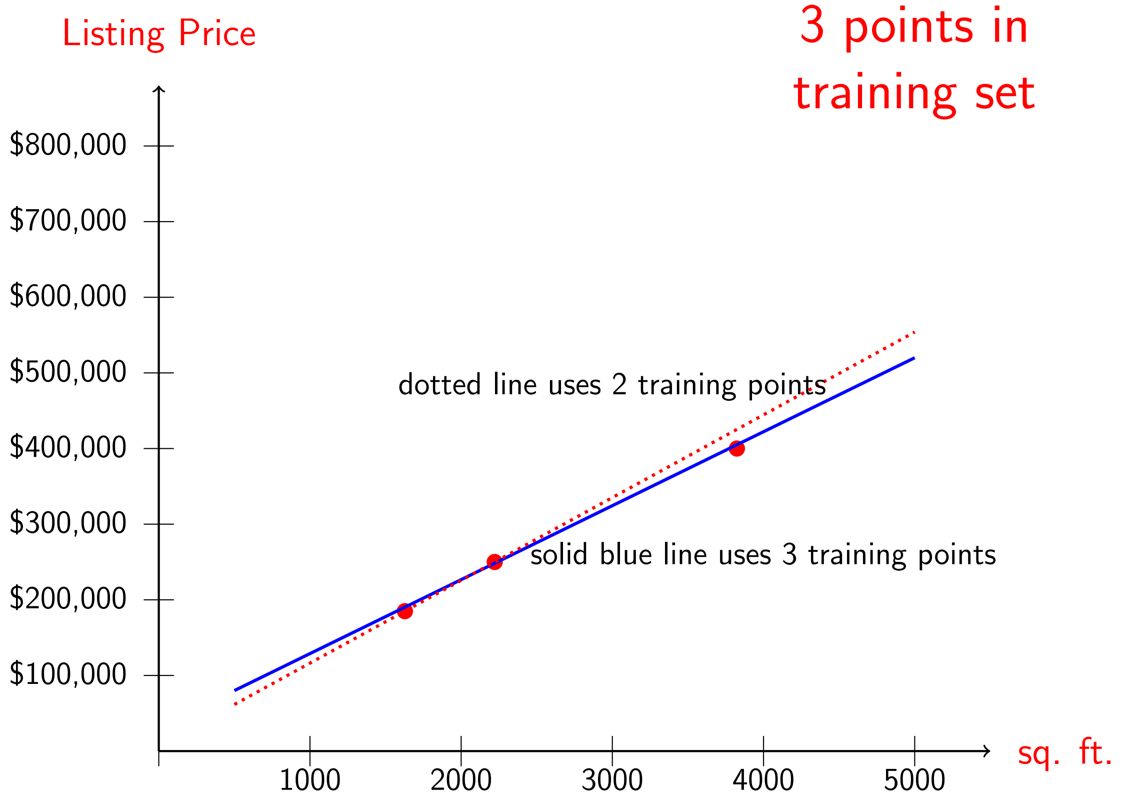
$$\left[250000 - (b + a \cdot 2222)\right]^2 + \left[185000 - (b + a \cdot 1628)\right]^2 + \left[399000 - (b + a \cdot 3824)\right]^2$$

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

$$P = 31092.8 + 96.5235x$$

when 3 data points are used.

In the plot below, we have drawn the straight line which is the best approximation to the first three houses in our training set and compare it with the line we got using 2 houses in the training set.

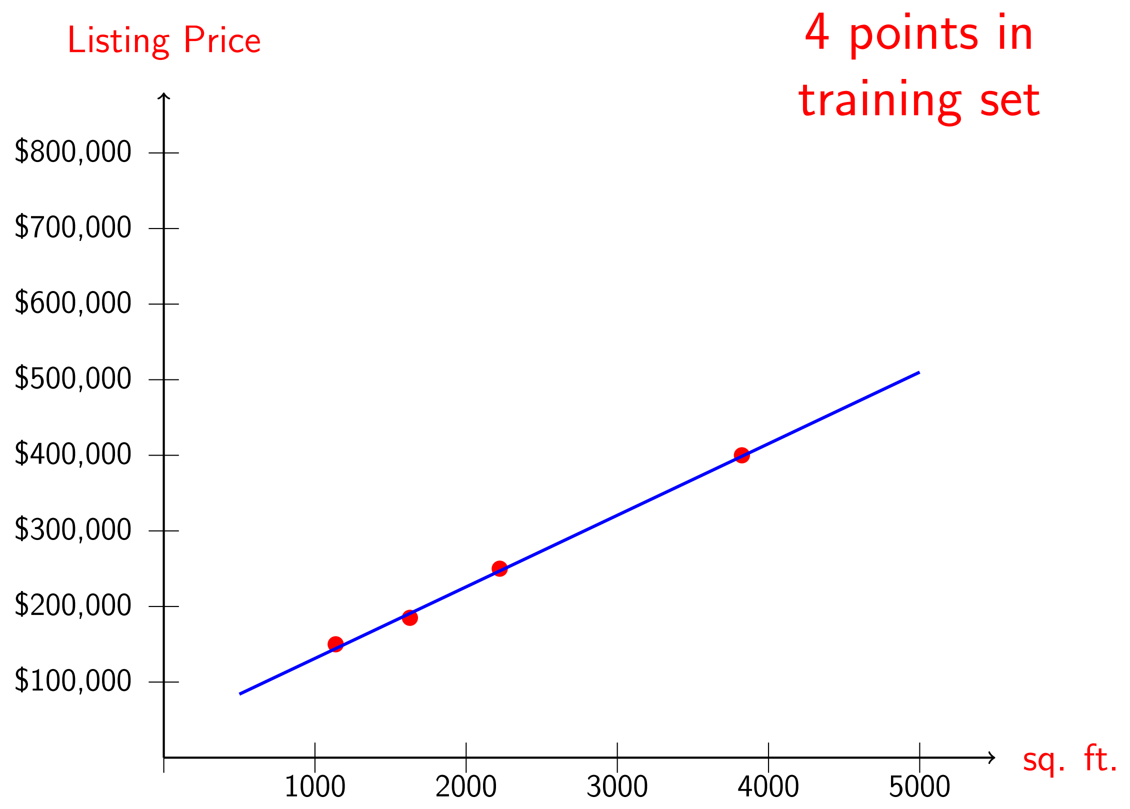


If we add the next house which has 1137 sq ft with a listing price of $150,000 to our training set, then we get a different line because the line changes to incorporate this information. Note that both the y-intercept and the slope change each time.

2 points $P = 6851.85 + 109.428x$

3 points $P = 31092.8 + 96.5235x$

4 points $P = 38624 + 94.1396x$



As we include additional houses from our training set, the line is modified to account for the new information.

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 points $P = 6851.85 + 109.428x$

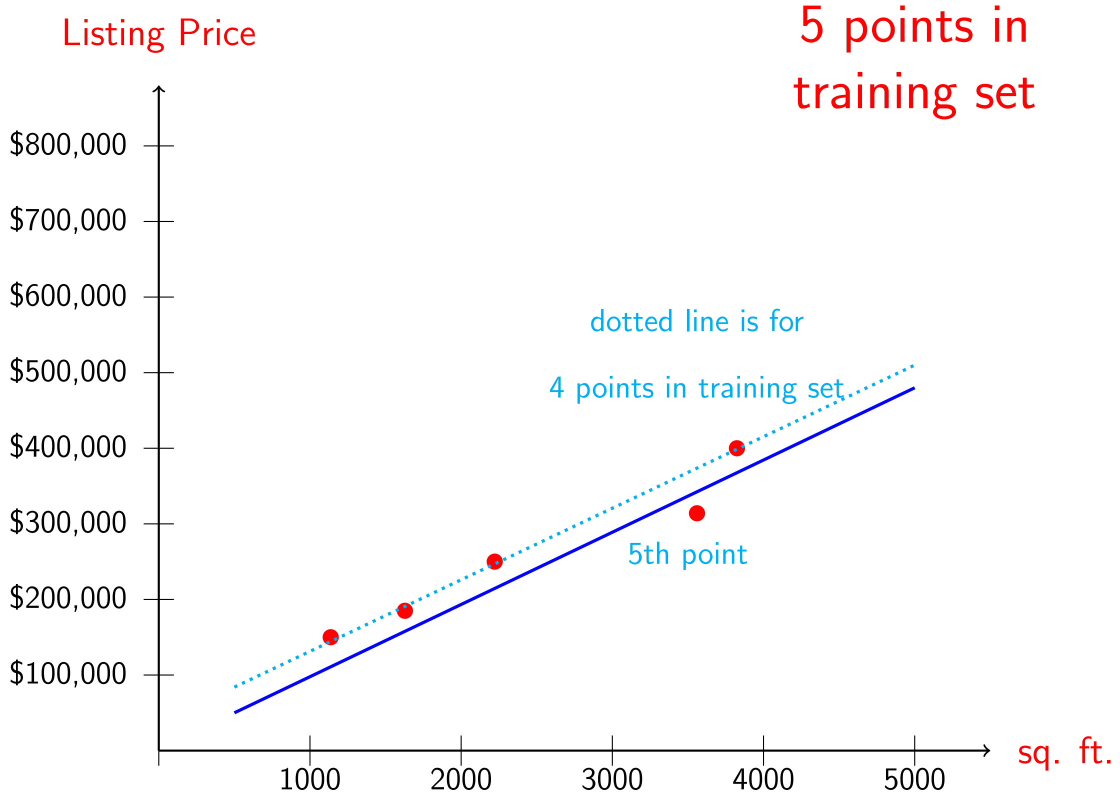
3 points $P = 31092.8 + 96.5235x$

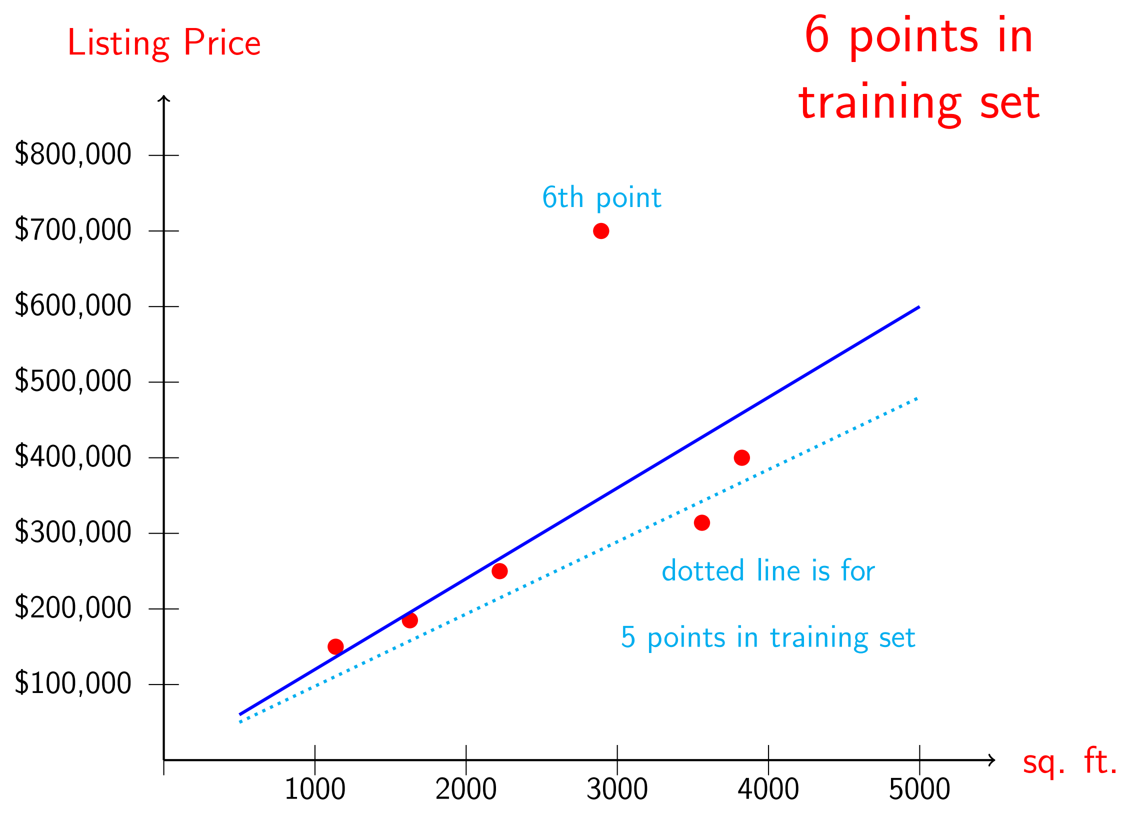
4 points $P = 38624 + 94.1396x$

5 points $P = 55232.5 + 82.6802x$

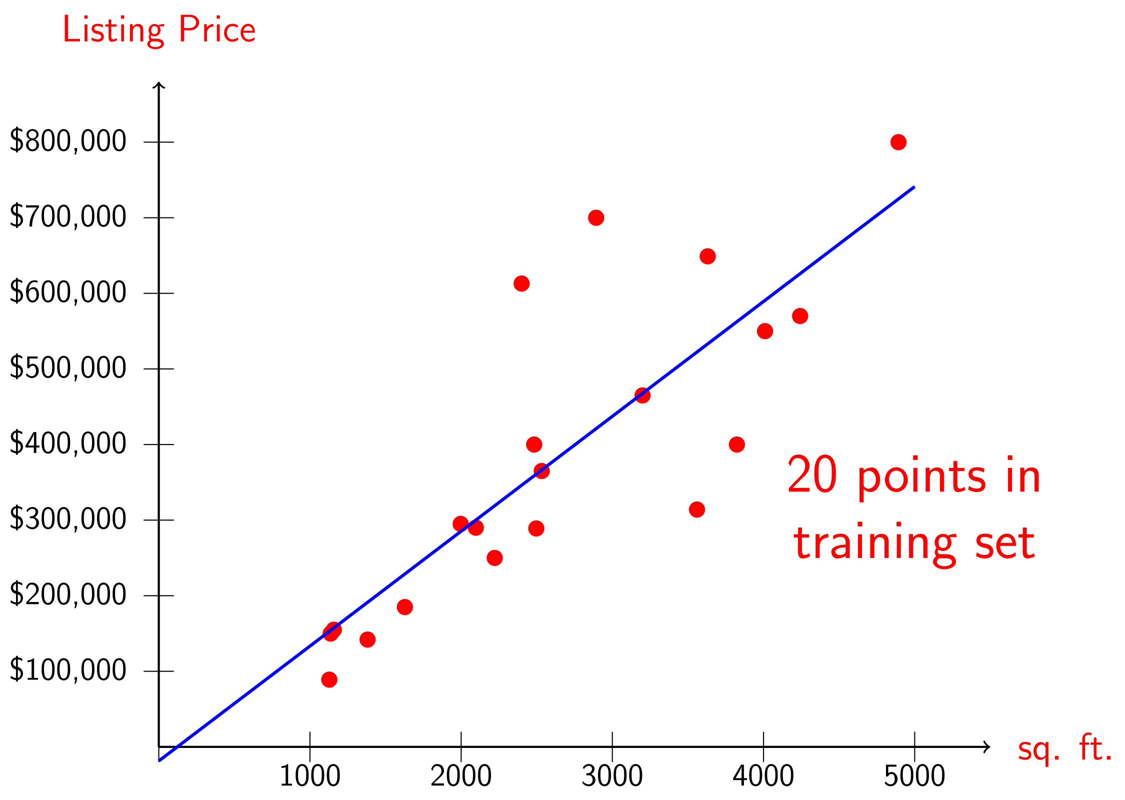
6 points $P = 59807.1 + 107.387x$

5 points in training set





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

Socrative Quiz

Use the plot below to answer the following questions. Assume that we are using linear regression/least squares to find the best line for the given data.

1. Which of the following errors do we minimize?
   (a) xx
2. Which line do you think is the best linear approximation to the given data?
   (a) red
   (b) green
   (c) blue

ADD PLOT

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

$$\boxed{P = a + bx + cy}$$

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 from 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.

$$\text{3 bedrooms, 2.5 baths} \implies 1915.75 + 163.371(2150) - 7216.75(5.5) = \$313,471$$
$$\text{4 bedrooms, 3 baths} \implies 1915.75 + 163.371(2150) - 7216.75(7) = \$302,646$$

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.

Group Exercise on Linear Regression

(I'll hand this out - Turn in at end of class) In this exercise we are trying to use linear regression to predict the satisfaction an employee has based on his/her/their salary. In the figures, the x-axis represents the employee salary in units of$1,000 and the y-axis represents the employee satisfaction rating from 0 to 100 where 100 is completed satisfied.

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

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 these 4 distances/errors.

3. In Figure 3, we have plotted the line which best fits the 4 data points from 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	

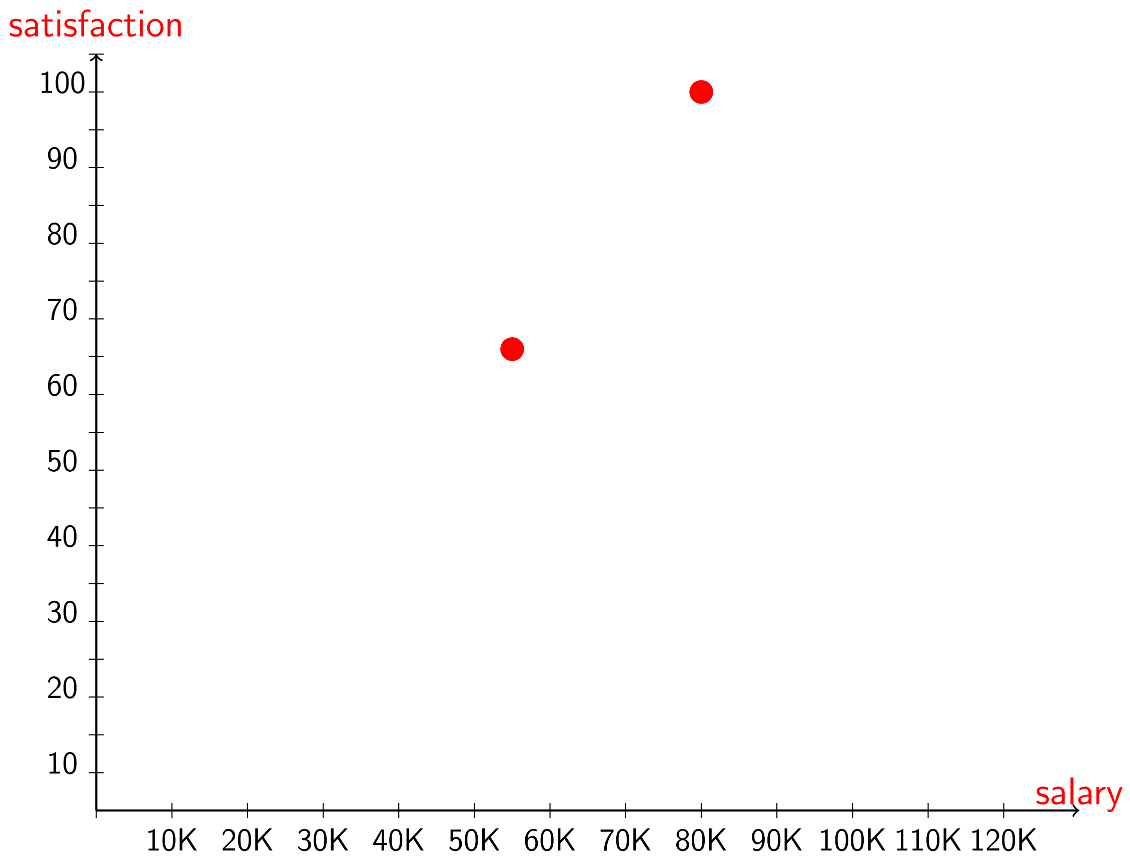

satisfaction
100
90
80
70
60
50
40
30
20
10
10K
20K
30K
40K
50K
60K
70K
80K
90K
100K
110K
120K
salary

Figure 1

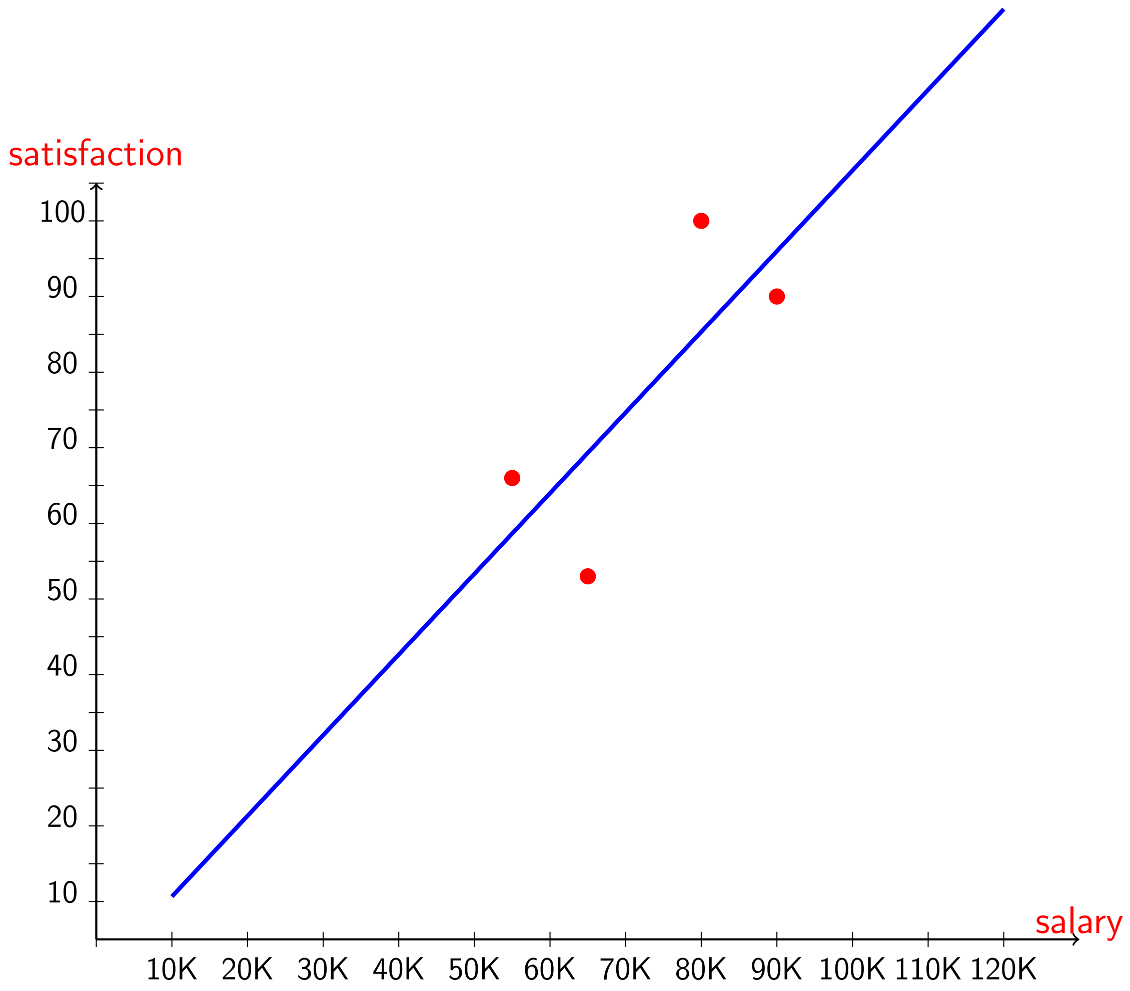

satisfaction
100
90
80
70
60
50
40
30
20
10
salary
10K
20K
30K
40K
50K
60K
70K
80K
90K
100K
110K
120K

Figure 2

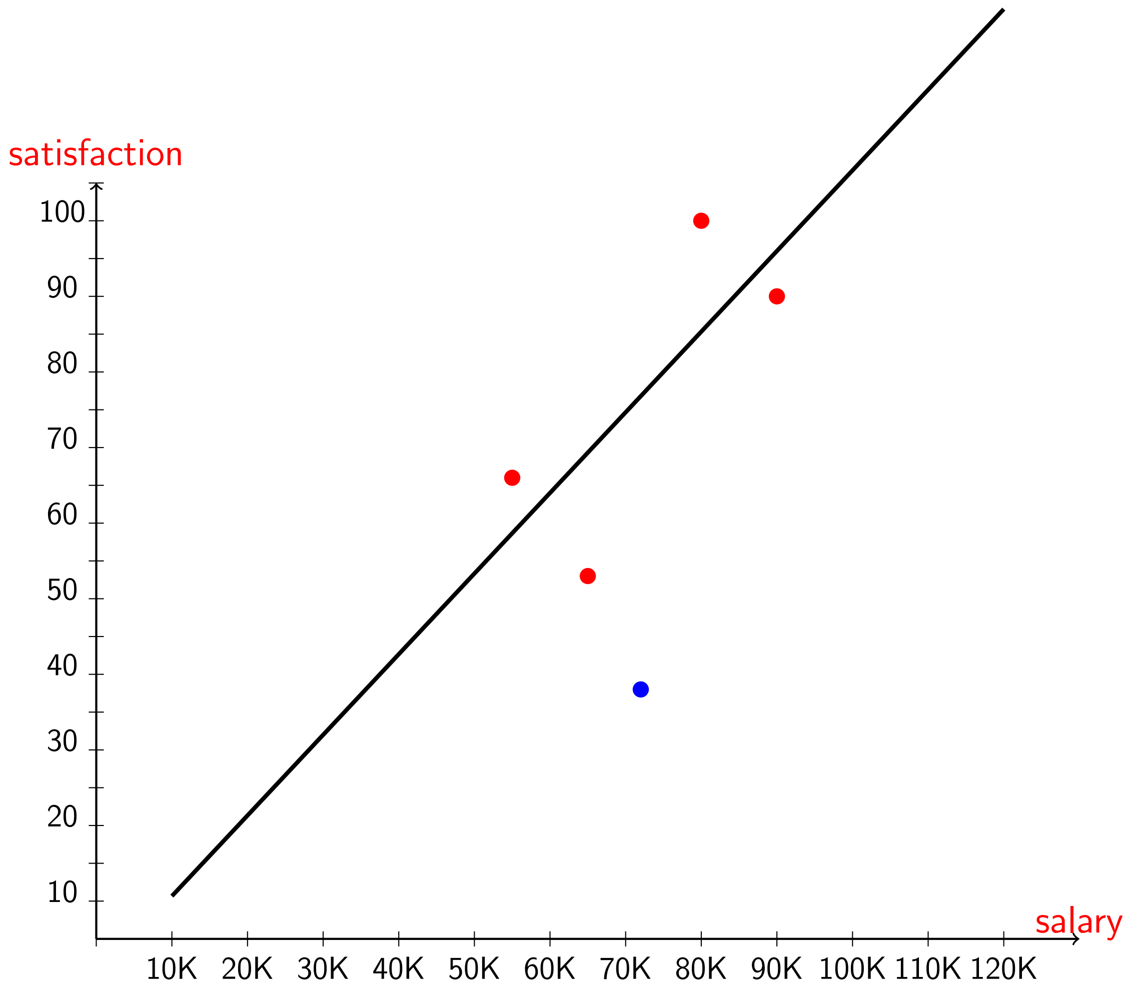

satisfaction
100
90
80
70
60
50
40
30
20
10
salary
10K
20K
30K
40K
50K
60K
70K
80K
90K
100K
110K
120K

Figure 3

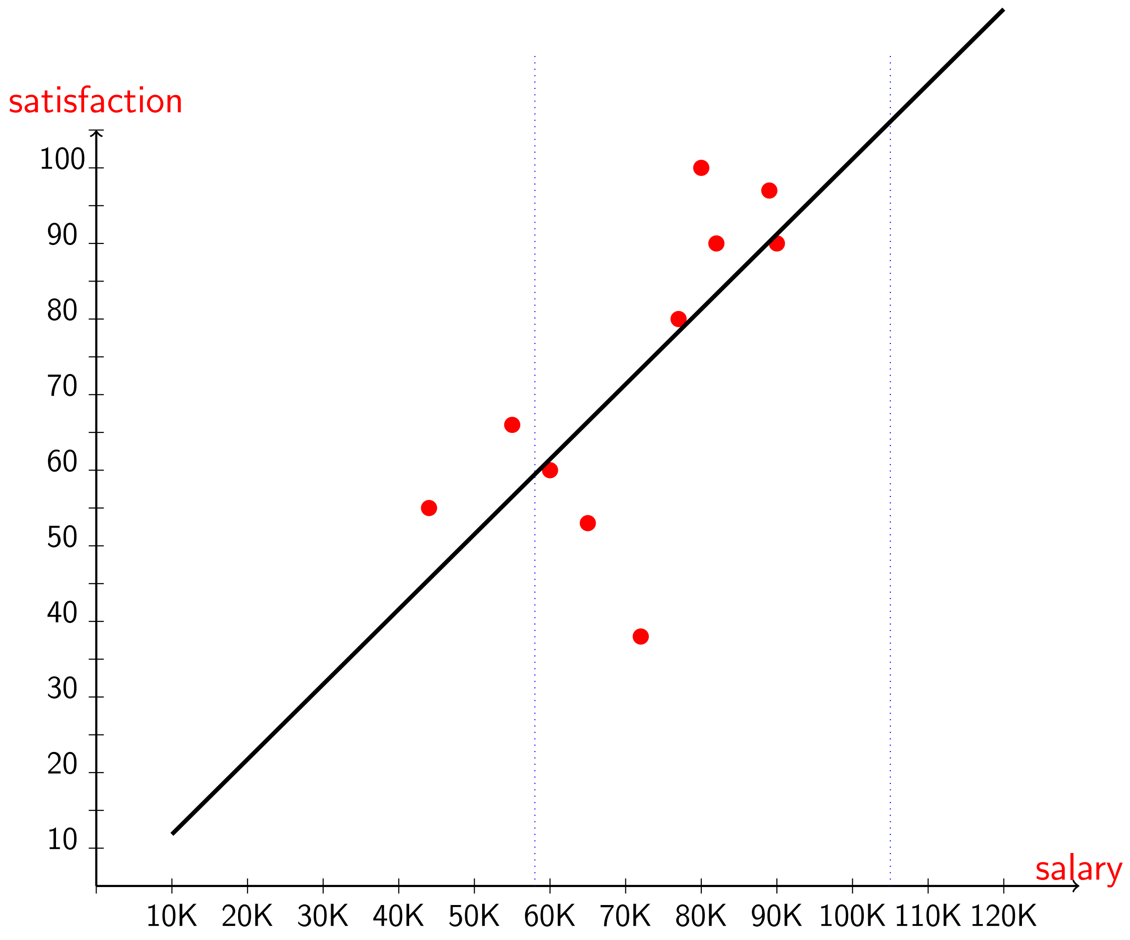

satisfaction
100
90
80
70
60
50
40
30
20
10
salary
10K
20K
30K
40K
50K
60K
70K
80K
90K
100K
110K
120K

Figure 4

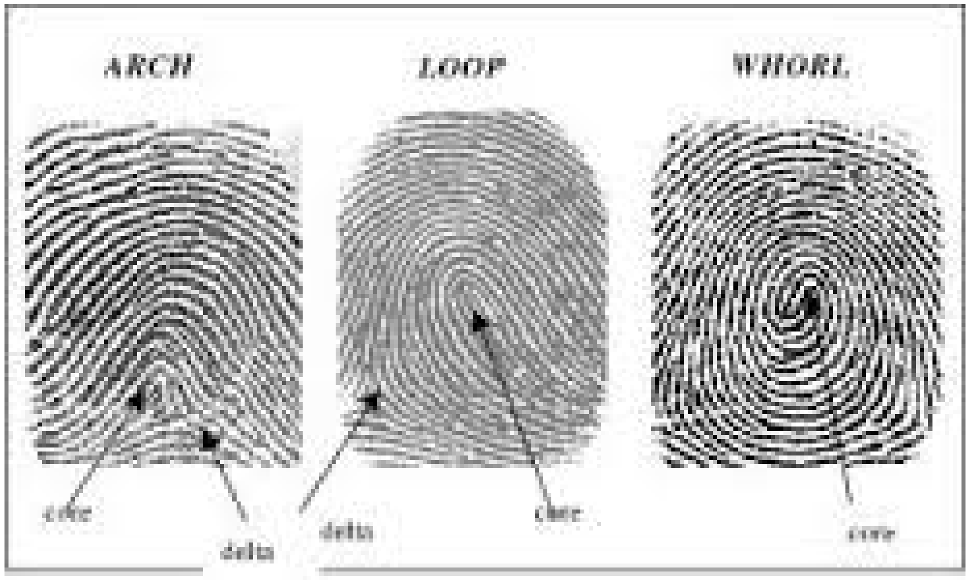
Pattern Recognition - Algorithm # 6

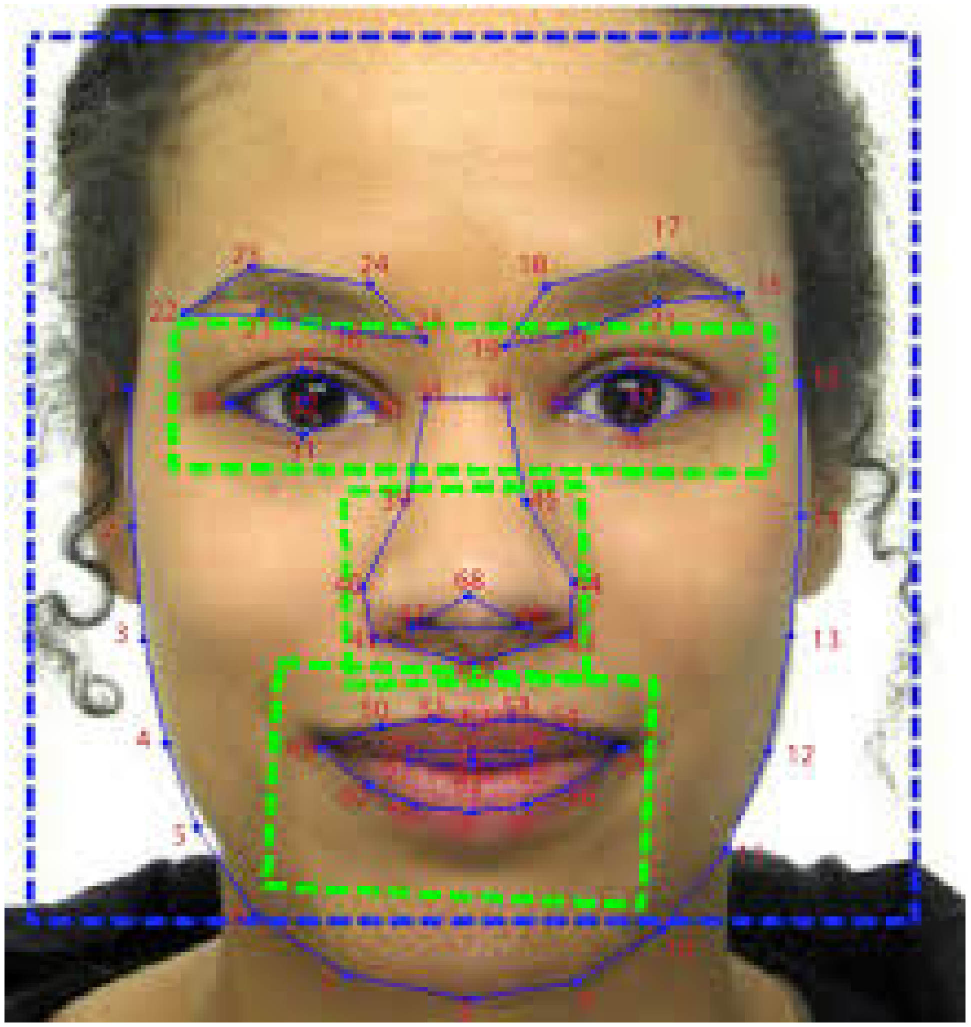
Computer algorithms such as PageRank for searching web pages, codes for encrypting/decrypting files, etc. far exceed what a human can do. However this is not the case when it comes to pattern recognition. Humans have a natural advantage here.

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

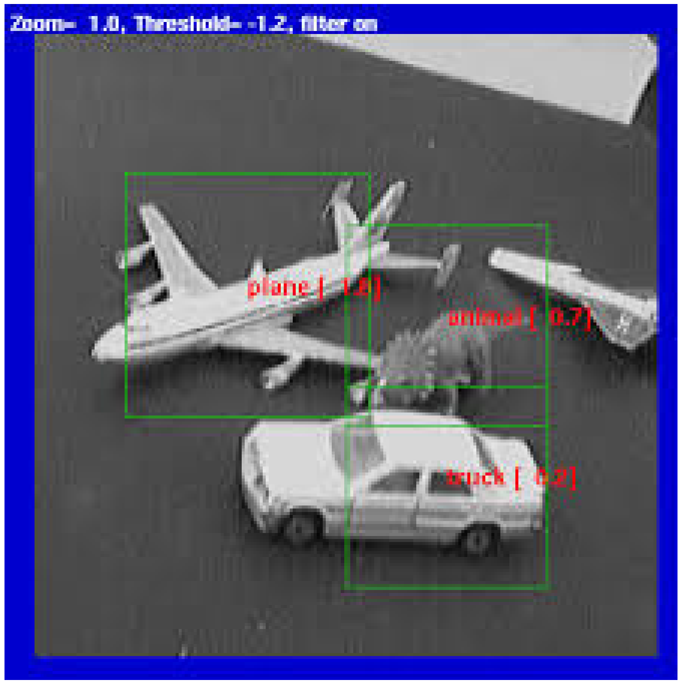
Pattern recognition is a type of Machine Learning algorithm which is considered Artificial Intelligence. It encompasses applications such as

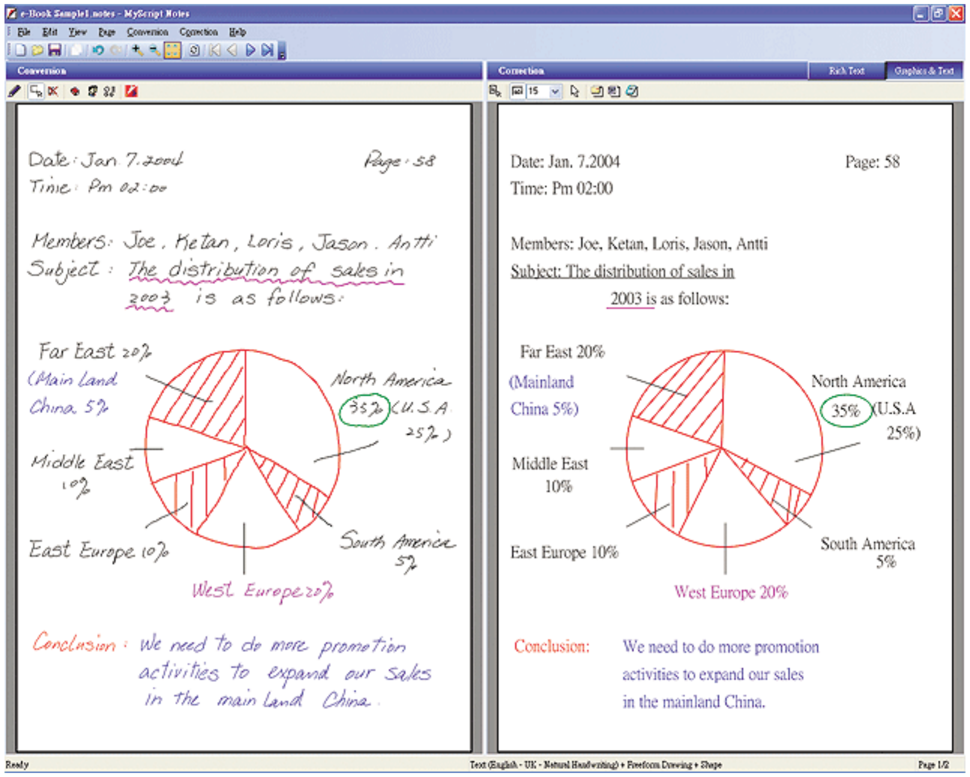
- fingerprint identification
- face recognition
- object recognition
- speech recognition
- handwriting recognition





Face Recognition





e-Book Sample1.notes - MyScript Notes
File Edit View Page Conversion Correction Help
Conversion
Correction
Rich Text
Graphics & Text
Ready
Text (English - UK - Natural Handwriting) + Freeform Drawing + Shape
Page 1/2

Date: Jan 7.2004 Page: 58
Time: Pm 02:00

Members: Joe, Ketan, Loris, Jason, Antti
Subject: The distribution of sales in 2003 is as follows:

Far East 20% (Main Land China 5%)
North America 35% (U.S.A 25%)
Middle East 10%
East Europe 10%
South America 5%
West Europe 20%

Conclusion: We need to do more promotion activities to expand our sales in the main land China.

Date: Jan. 7.2004 Page: 58
Time: Pm 02:00

Members: Joe, Ketan, Loris, Jason, Antti
Subject: The distribution of sales in 2003 is as follows:

Far East 20% (Mainland China 5%)
North America 35% (U.S.A 25%)
Middle East 10%
East Europe 10%
South America 5%
West Europe 20%

Conclusion: We need to do more promotion activities to expand our sales in the mainland China.

Object Recognition Handwriting 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.

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, all the 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.

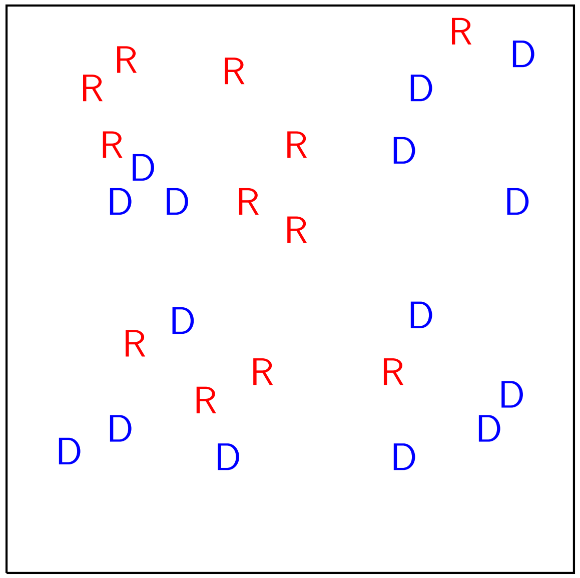
Our goal is to look at three different ML Classifier Algorithms and see some examples of pattern recognition problems they are best suited for.

Nearest-Neighbor Classifier & Variants

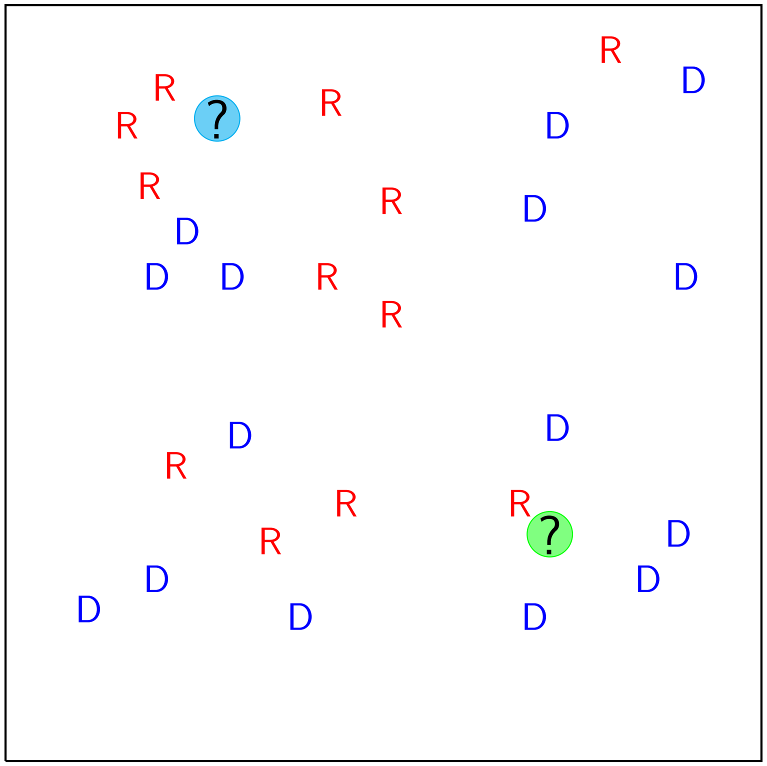
- This is probably the simplest classification algorithm
- It is based on classifying an object by the classification of its nearest neighbor (or neighbors)
- We have to define what we mean by “nearest neighbor”. If we are talking about the person/persons who live closest to you, we understand this meaning 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.

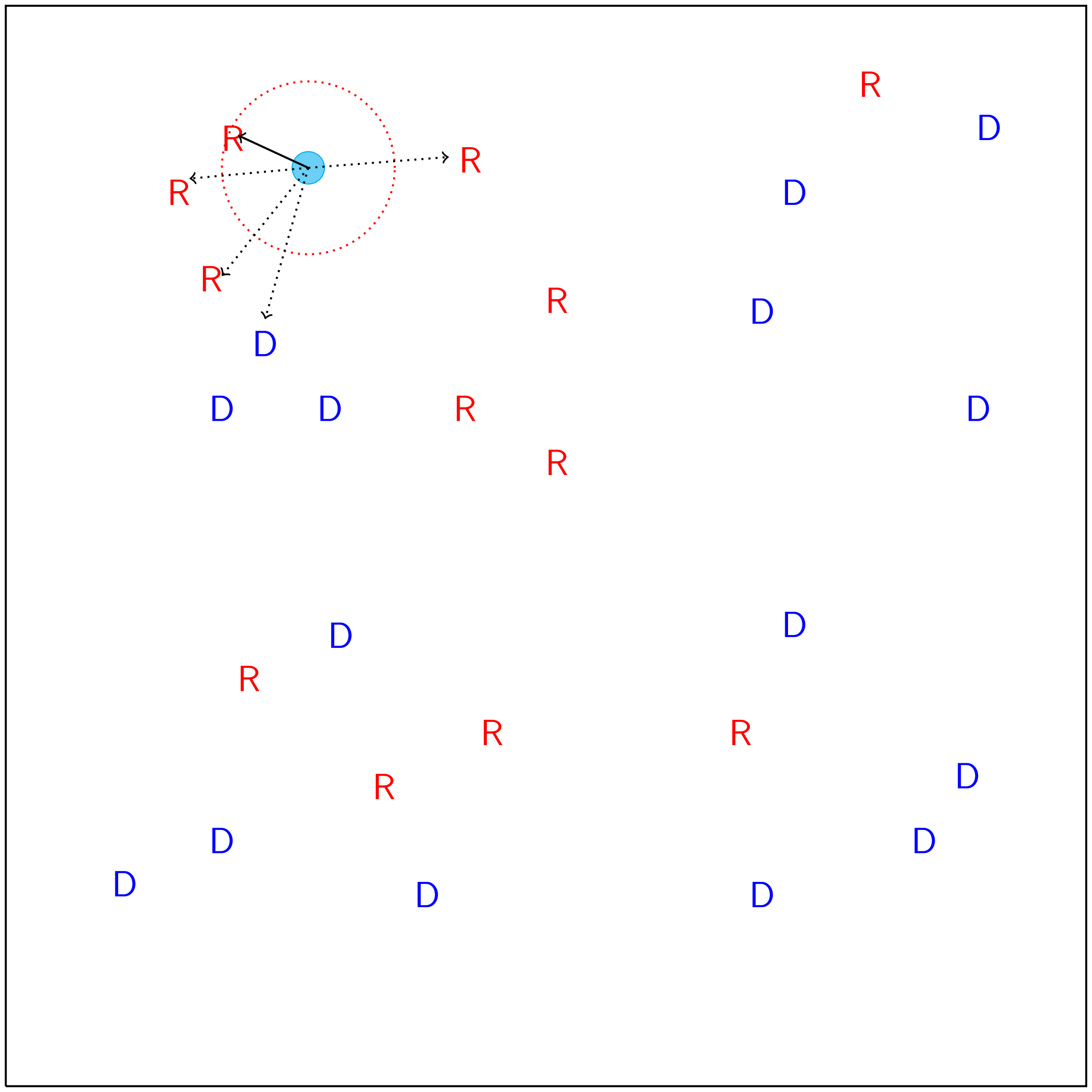


We want to predict whether each of the two residences (represented by a question mark) is a Democrat or Republican.

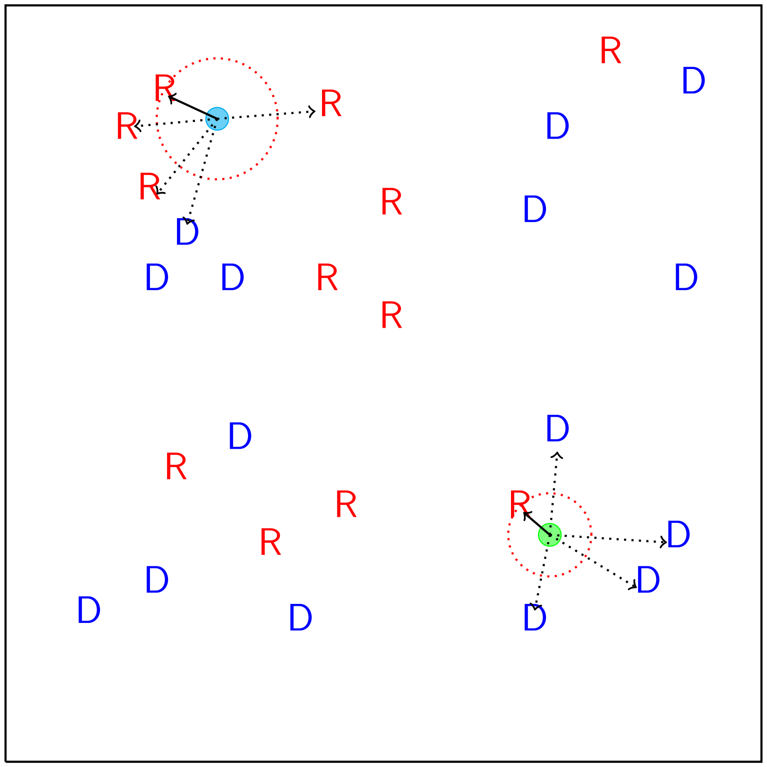


If we used this approach to guess the political affiliation of the home located at the top (marked in cyan) then we would say they are Republicans because the surrounding neighbors are all Republicans.

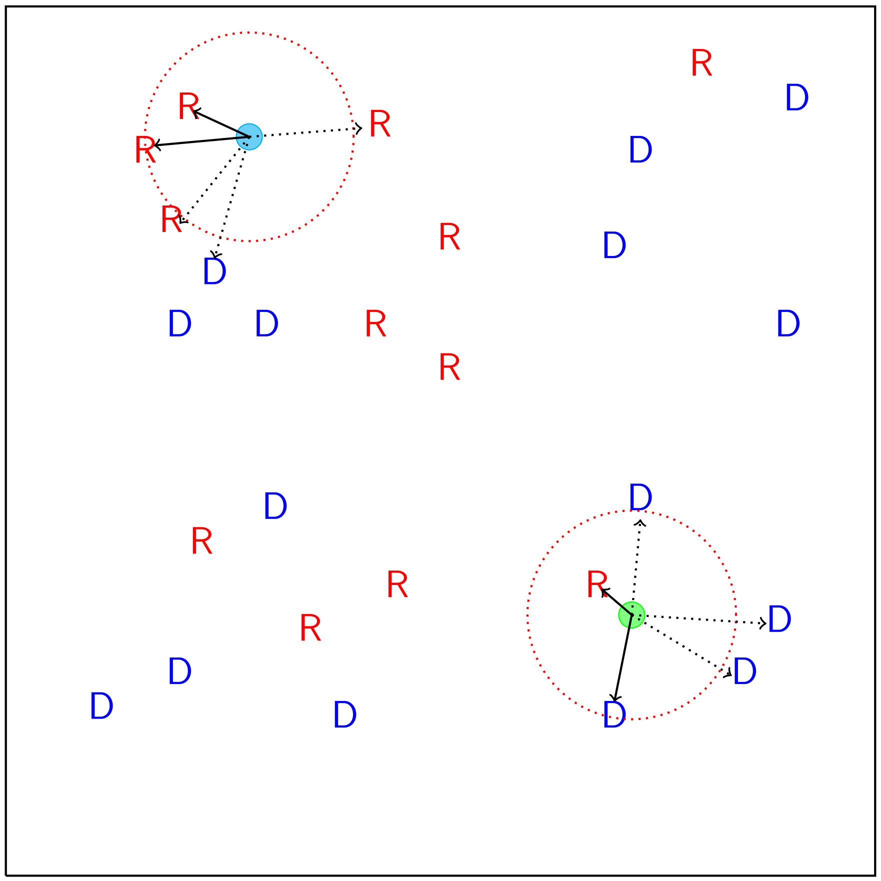
Visually, to determine the nearest neighbor to this residence we would draw circles around the residence until we touch a neighbor. This would be its "nearest neighbor". Clearly for this residence the nearest neighbor is a Republican. When we write an algorithm we would simply calculate the Euclidean distance from each residence to the one in question and determine the classification of the one with the shortest distance.


R
D
R
R
D
R
R
D
D
D
D
R
D
R
D
R
R
R
D
R
D
D
D
D
D

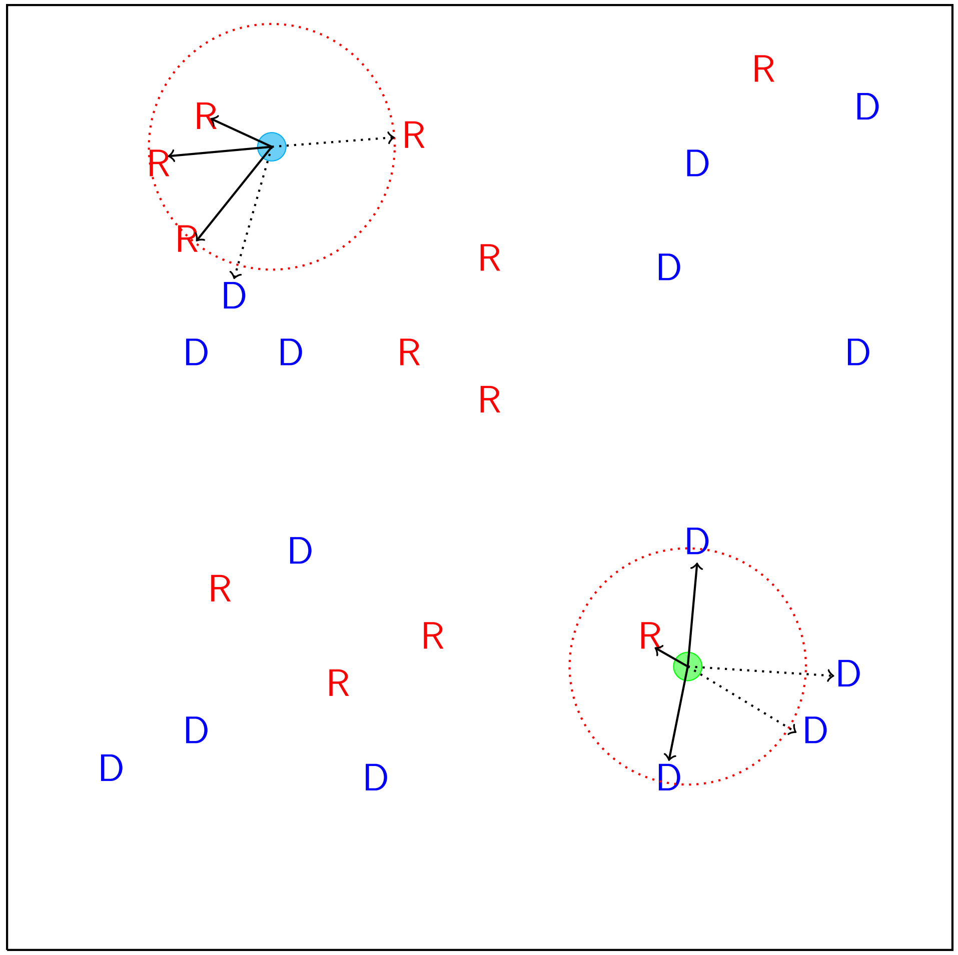
However, when we look at the bottom residence (green) we see that its nearest neighbor is Republican but the others surrounding it are Democrats. So if we use a computer algorithm to determine the political persuasion based on the single closest neighbor, the prediction would be Republican for both.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

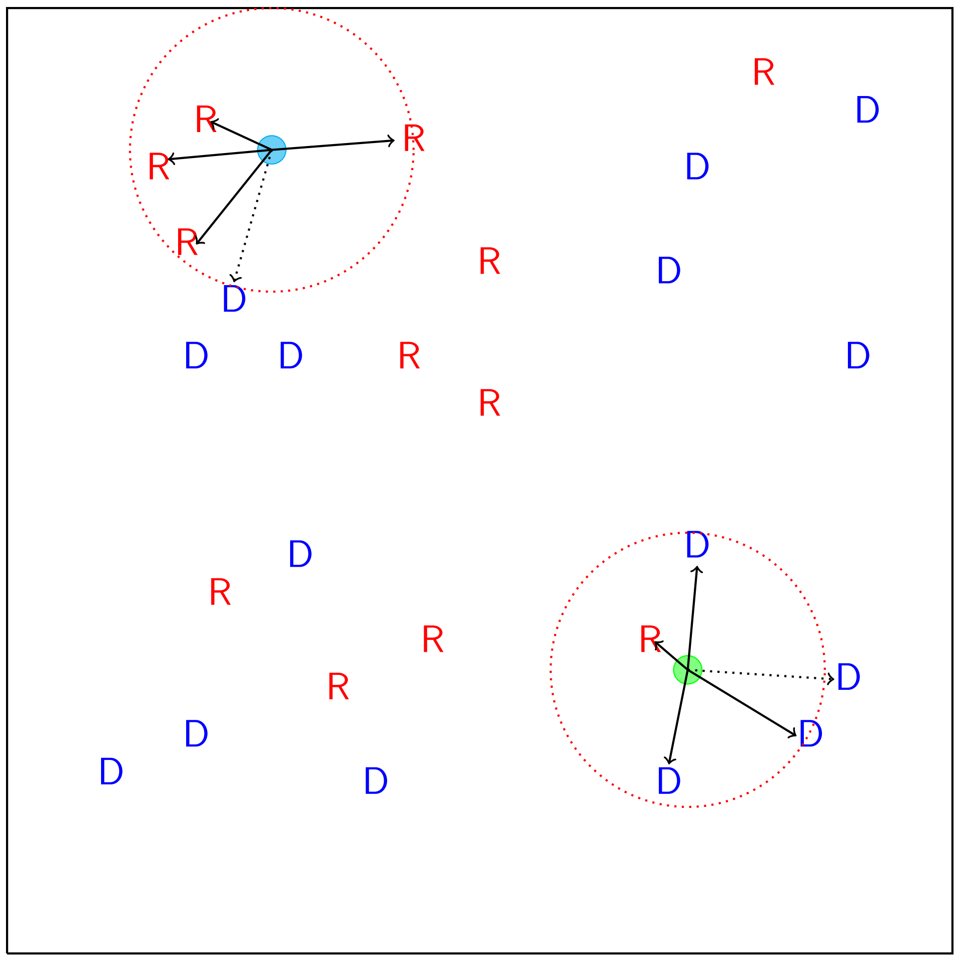


Top - 2 Republican neighbors $\Longrightarrow$ Republican
Bottom - 1 Democratic neighbor, 1 Republican $\Longrightarrow$?



Top - 3 Republican neighbors $\Longrightarrow$ Republican

Bottom - 2 Democratic neighbors, 1 Republican $\Longrightarrow$ Democrat



Top - 4 Republican neighbors $\Longrightarrow$ Republican

Bottom - 3 Democratic neighbors, 1 Republican $\Longrightarrow$ Democrat

Variants of Nearest Neighbor Classifier

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

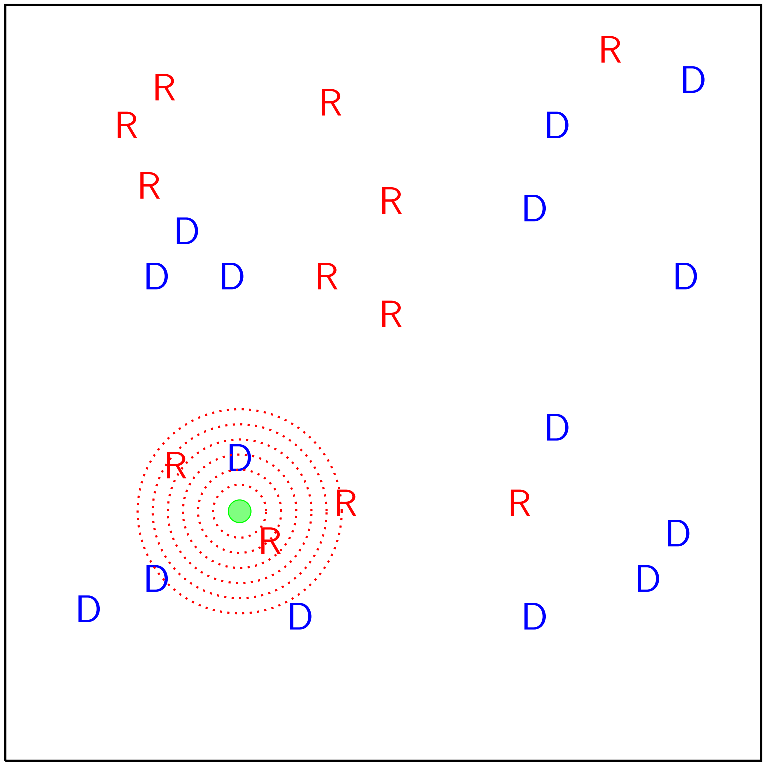
Suppose in our previous example, we had data that recorded the amount of donations to the respective party that each residence has made in the last year.

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.

Socrative Quiz

Use the schematic below to answer the following questions.

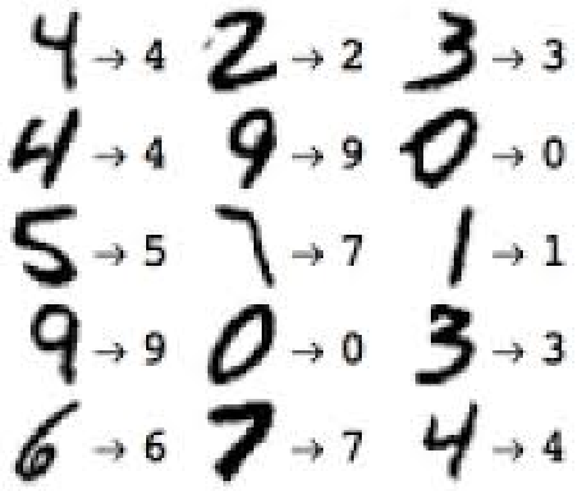
1. Using a single nearest neighbor classifier, would the residence in question (marked in green) be classified as a Democrat (enter “D”) or a Republican (enter “R”).
2. Using a 3-nearest neighbor classifier, would the residence in question (marked in green) be classified as a Democrat (enter “D”) or a Republican (enter “R”).
3. Using a weighted 2-nearest neighbor classifier, would the residence in question (marked in green) be classified as a Democrat (enter “D”) or a Republican (enter “R”) if Democratic residences have a larger weight than Republican residences.


R
D
R
R
D
R
R
D
D
D
D
R
D
R
D
D
R
D
R
R
R
D
D
D
D
D
D

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?

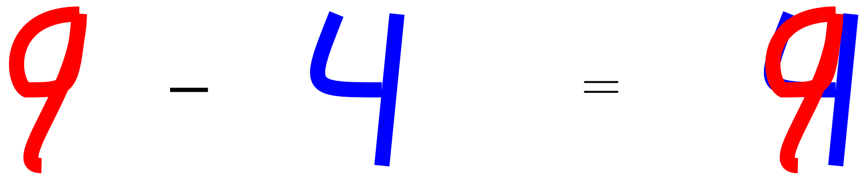
Suppose we have a set of handwritten numbers which we have classified as 0,1,2,3,4,5,6, 7,8 or 9. Assume further that we have scaled all of these numbers so they are the same sizes.



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

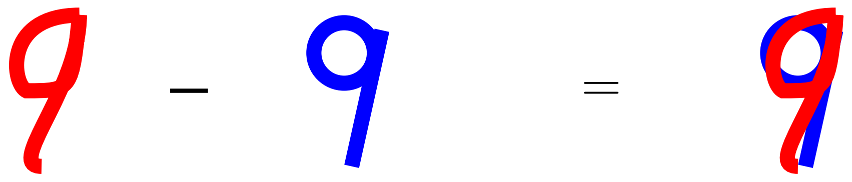
What do we do?

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



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.

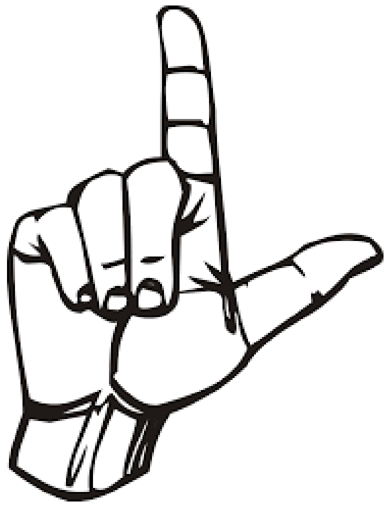
We proceed through the data base and see which handwritten number has the smallest remainder and claim that is our letter's nearest neighbor and classify it the same way.

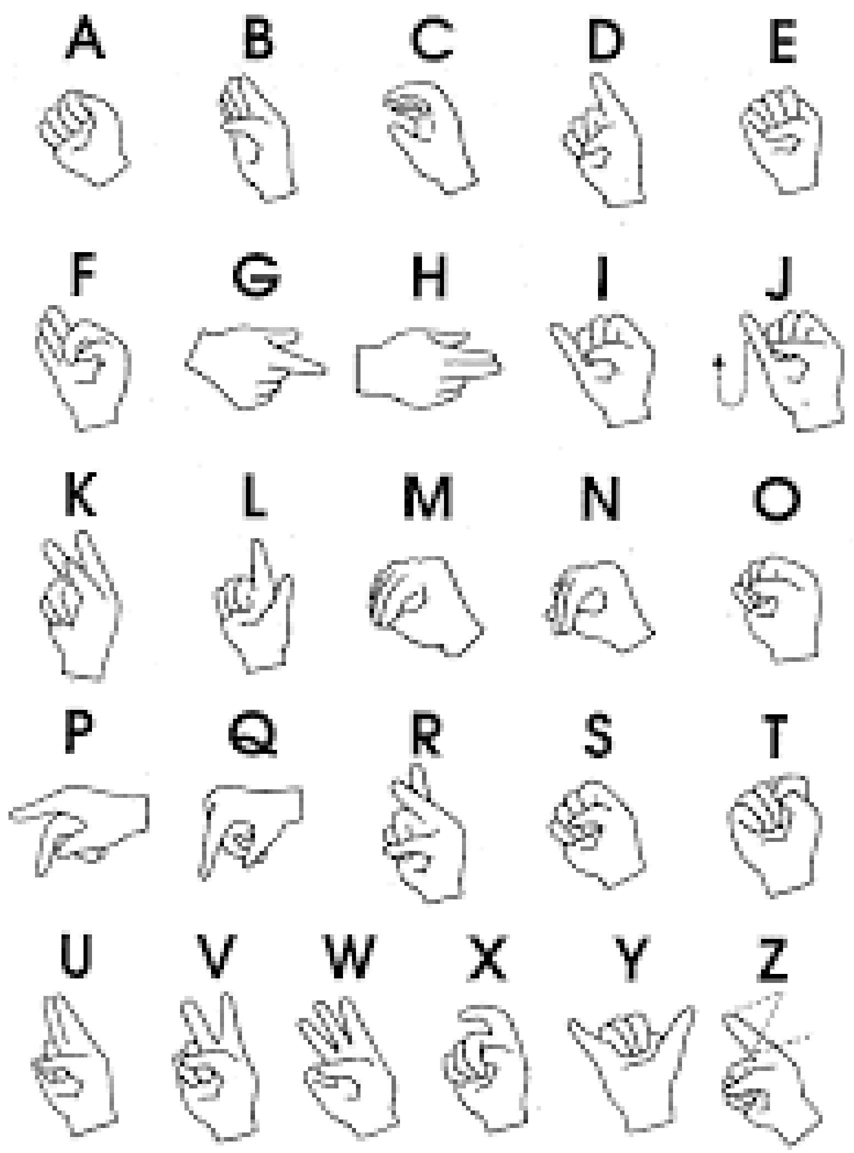


Socrative Quiz

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

Enter the letter of the alphabet for your response.




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

Handout exercise so Pattern Recognition Intro & Nearest Neighbor is 1 lecture

Decision Trees Classifiers

What is a Decision Tree?

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.

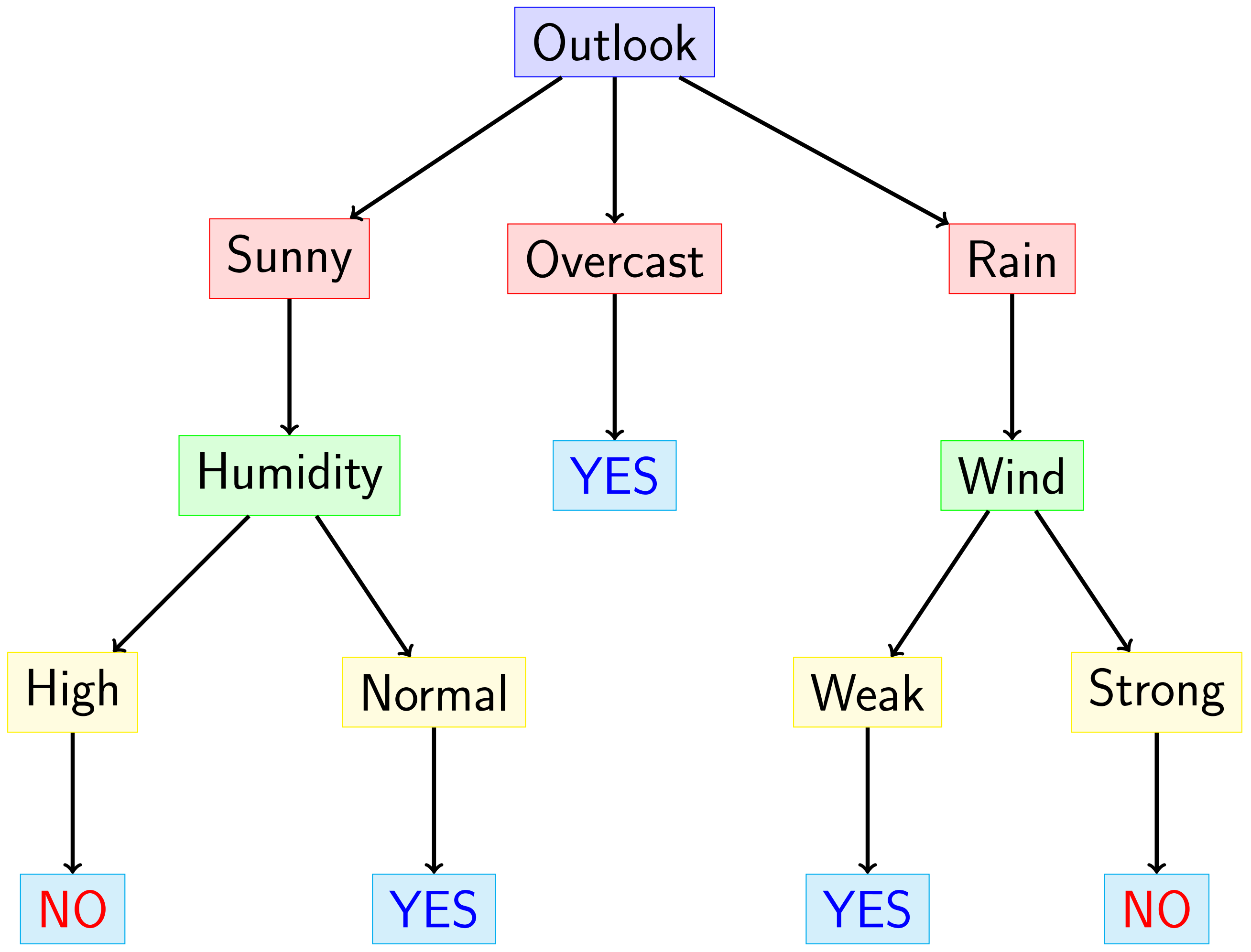
In this example, the attribute we are checking for the root node is the Outlook and the three options are Sunny, Overcast or Rainy.

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?

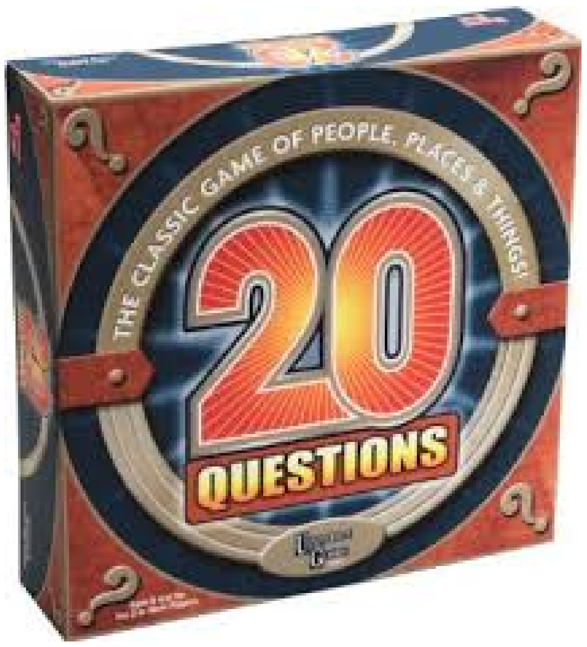
Outlook: Rain

Wind: Strong


Outlook
Sunny
Overcast
Rain
Humidity
YES
Wind
High
Normal
Weak
Strong
NO
YES
YES
NO

Decision Tree as a Classifier

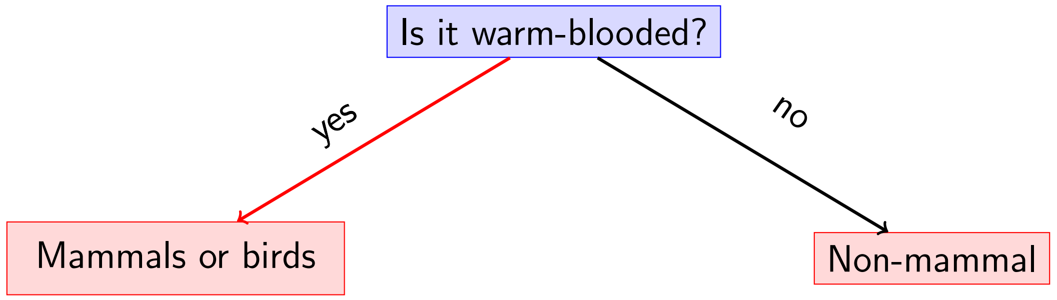
Example: 20 Questions Game



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 .

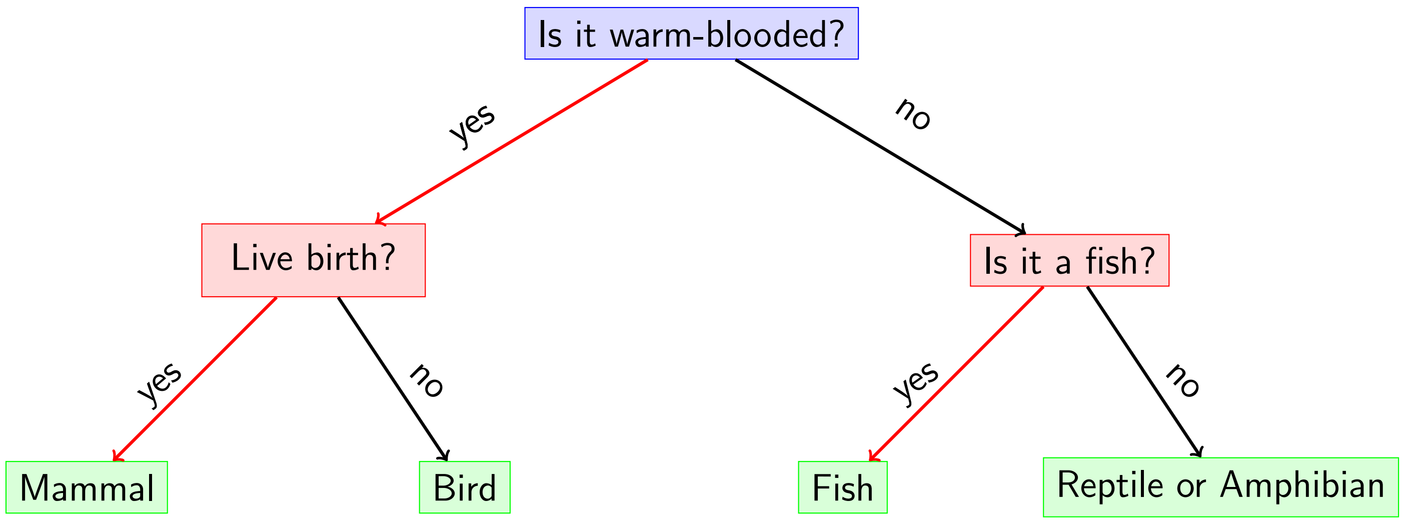


Now if the answer to the first question is “yes” then we know it is a mammal or a bird. There are many choices here for the next question. We could ask any of the following questions which give basically the same information.

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.

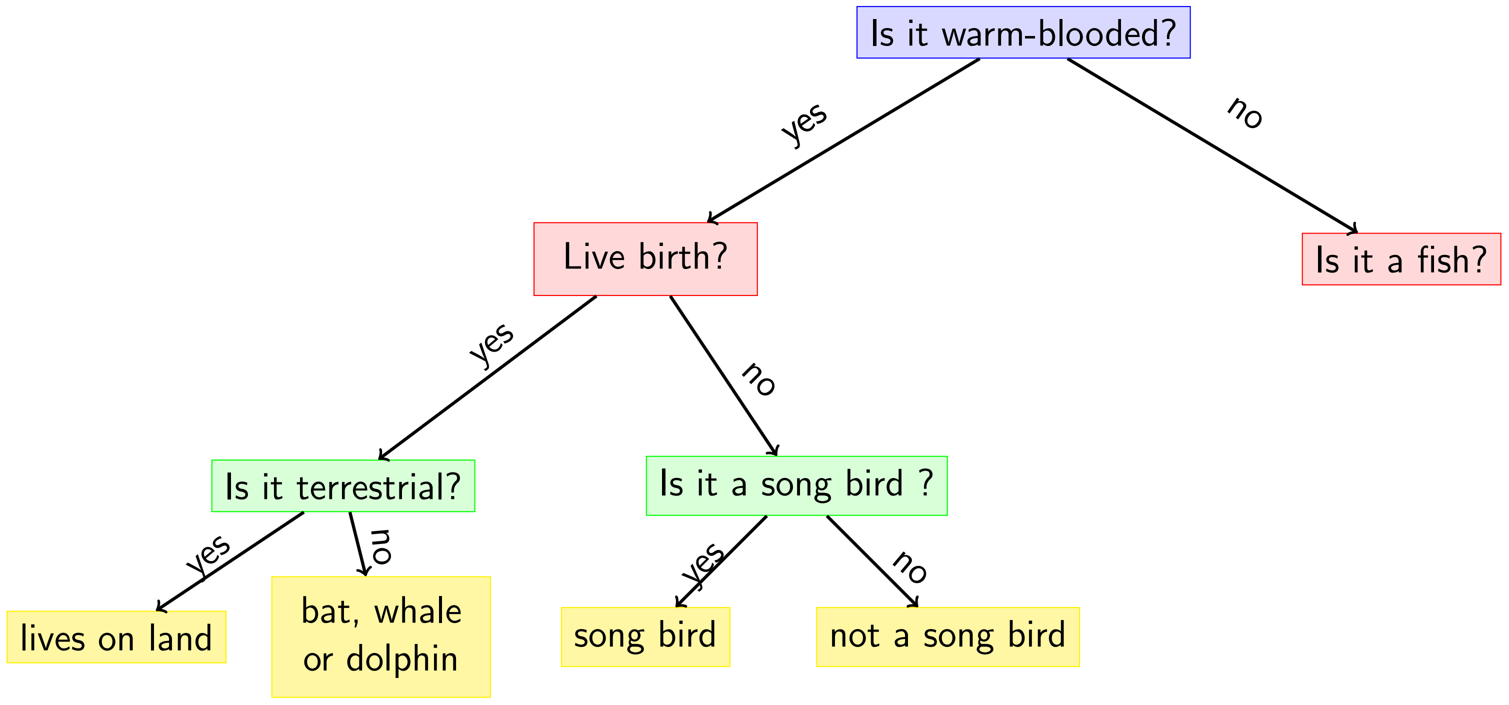


To get the correct identification we follow the Decision Tree starting at the root (the first question) based on the answer. For example, if the answer to the first question is "yes" and the answer to the second question is "no" we know that the vertebrate is a bird.

Socrative Quiz - Decision Trees

Use the Decision Tree below to classify the vertebrate.

1. The answers to the first 3 questions are “yes”, “yes”, “no” (in order)
2. The answers to the first 3 questions are “yes”,“no”, “yes” (in order)



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.

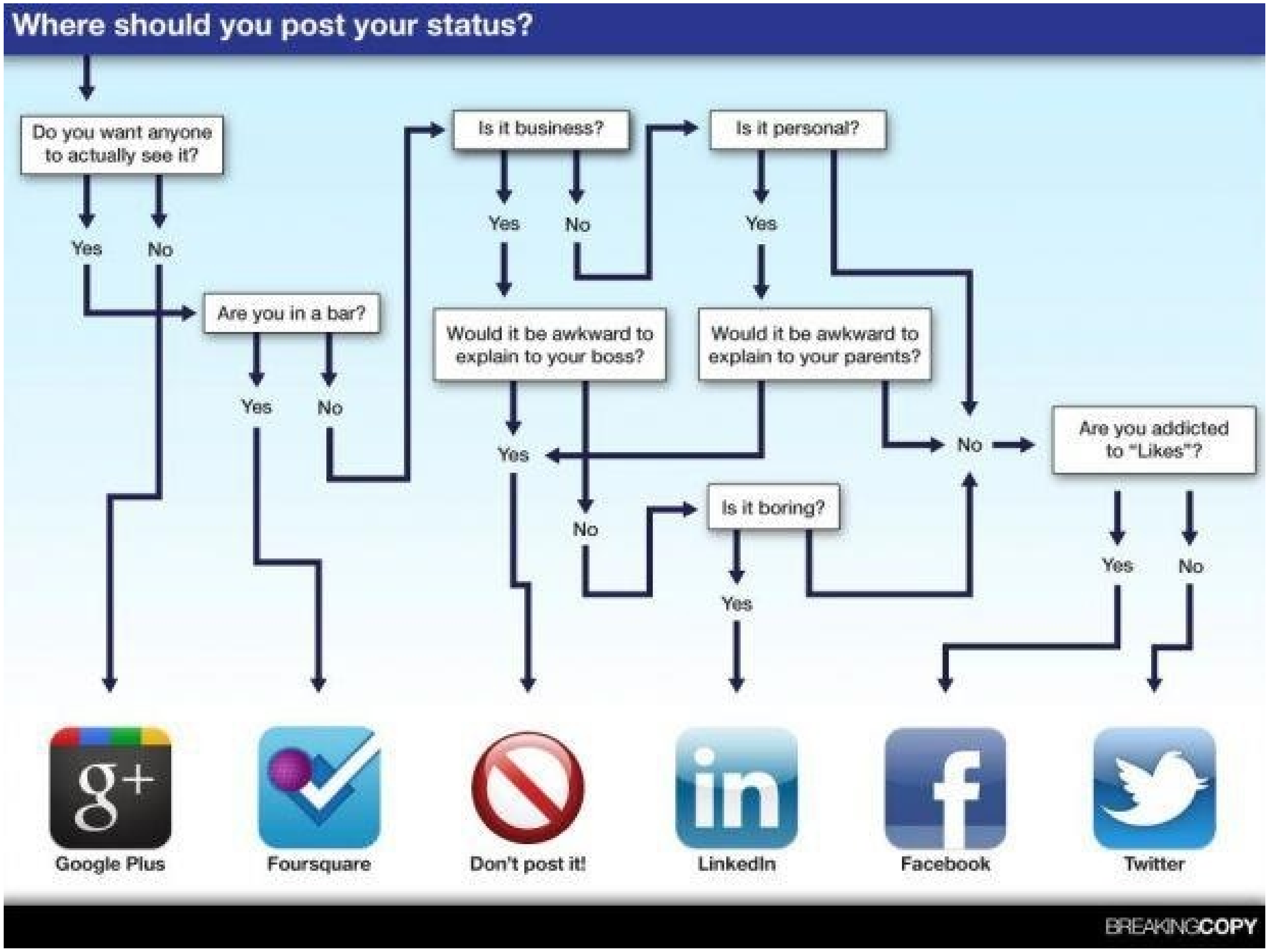
Google+ is a Social Network for discovering and sharing digital content with friends, family and co-workers.

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

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.

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?
- ⋮

We then use the training set and test each element in the set and see if the particular attribute gets the right answer. For example, if the attribute is Is it a fish it will only correctly identify elements of the training set which are fish whereas if the attribute is Is it a mammal? it will correctly identify all mammals. The attribute which correctly identifies the most elements of the training set is chosen for the root node.

After the root node is chosen we list the possible descendants. In the vertebrate example, the results are either yes or no but in the tennis example the descendants were Sunny, Overcast, Rainy.

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.

Decision Tree Leaning for “Play Tennis” Example

Suppose we are trying to decide what attribute to use for the root node in this 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	no
D2	Sunny	Hot	High	Strong	no
D3	Overcast	Hot	High	Weak	yes
D4	Rain	Mild	High	Weak	yes
D5	Rain	Cool	Normal	Weak	yes
D6	Rain	Cool	Normal	Strong	yes
D8	Sunny	Mild	High	Weak	no
D9	Sunny	Cool	Normal	Weak	yes
D10	Rain	Mild	Normal	Weak	yes
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, for Wind the possibilities are Strong, Weak.

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	$\{2^+, 3^-\}$
Overcast	$\{4^+, 0^-\}$
Rain	$\{3^+, 2^-\}$

Temperature

Hot	$\{2^+, 2^-\}$
Mild	$\{3^+, 1^-\}$
Cool	$\{4^+, 2^-\}$

Humidity

High	$\{3^+, 3^-\}$
Normal	$\{6^+, 2^-\}$

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. We have

Outlook	0.246
Temperature	0.029
Humidity	0.151
Wind	0.048

so clearly the best choice for the root node is the attribute Outlook which we could have guessed from the data. Of course this choice may be different with a different training set.

Web Browser Application using Decision Tree Classifier

Now we want to see how we can train an algorithm to form a Decision Tree.

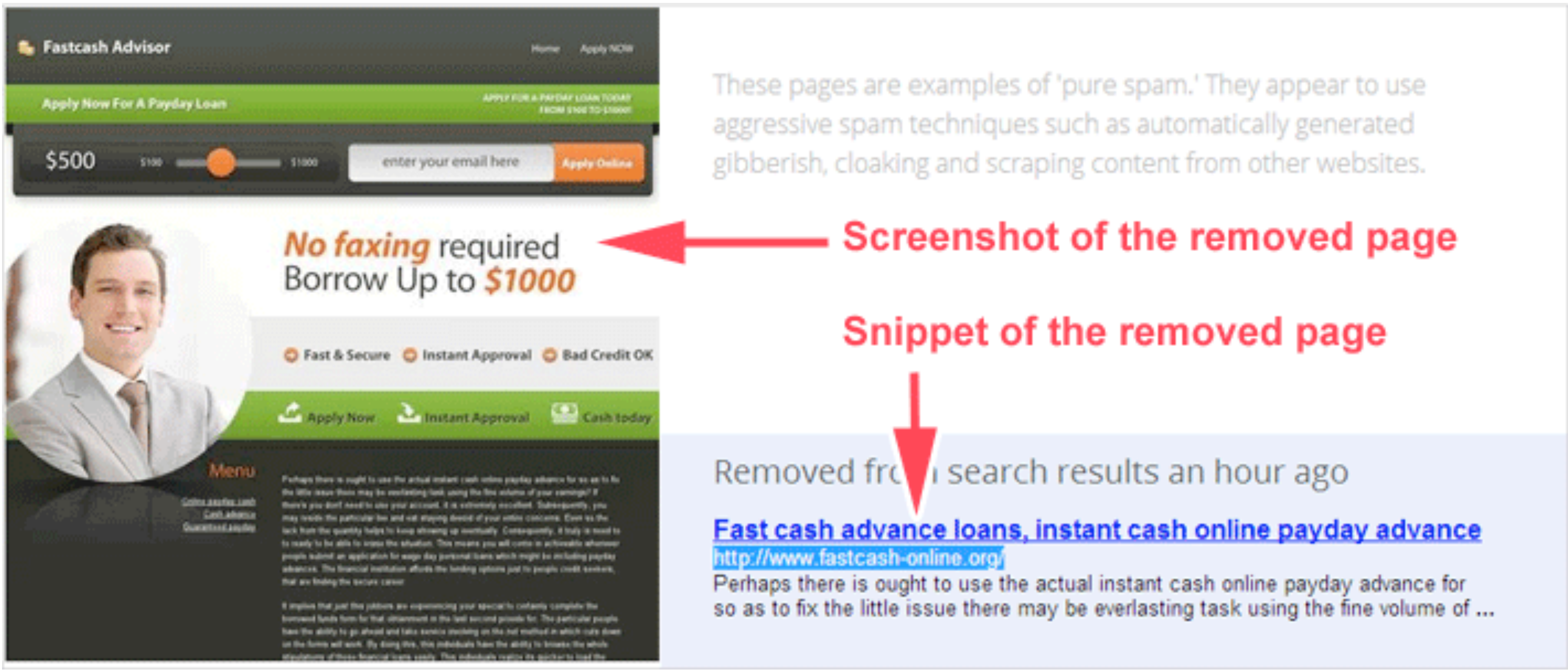
A pattern recognition problem where a Decision Tree Classifier is typically used is the identification of web spam.

What is web spam?

Artifically 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.

Removed from search results an hour ago

Fast cash advance loans, instant cash online payday advance
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.

Socrative Quiz - General Concepts

Answer "T" for True and "F" for False.

1. Nearest Neighbor and Decision Trees are two types of Pattern Recognition algorithms.
2. A commonly used application of Nearest Neighbor algorithms is identifying web spam pages.
3. Most humans are typically better at recognizing patterns than machine learning algorithms.
4. A Decision Tree is a graphical representation of possible solutions to a decision based on certain conditions.
5. The TV show/game *Jeopardy* is a type of Decision Tree.

Do we need handout exercise so Decision trees are 1 lecture ??

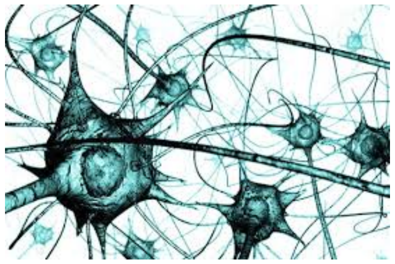
Neural Nets

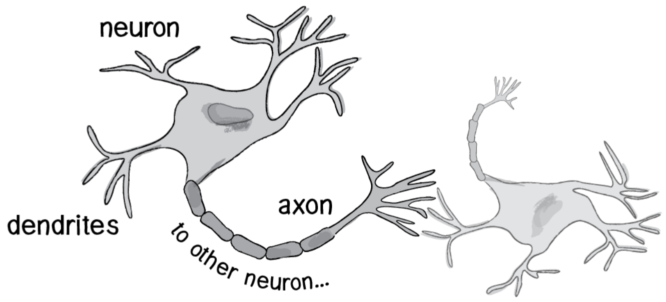
- Neural net algorithms are based on how our brain processes information.
- In 1943 a neuroscientist and a logician developed the first conceptual model of an artificial neural network.
- Neural net algorithms do NOT model how our brain works but they are inspired by how our brain works and designed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

- The human brain contains approximately 100 billion nerve cells called neurons.
- Each neuron is contected 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.

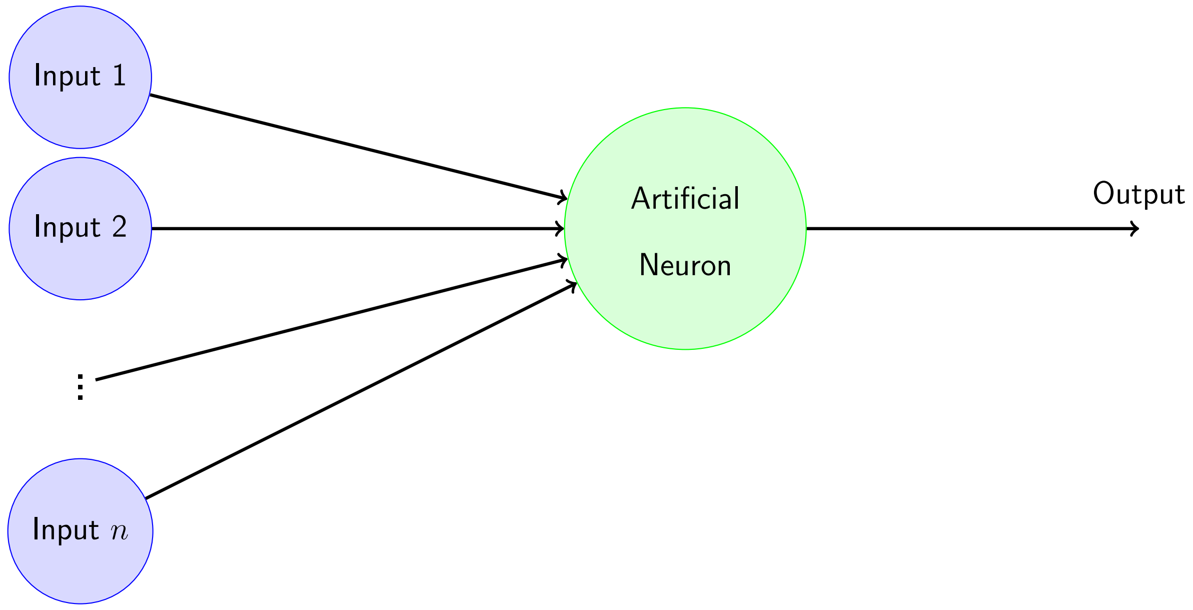
- Since the output of the neuron is "fire" or "don't fire" it is a binary output which can be imitated on a computer easily.





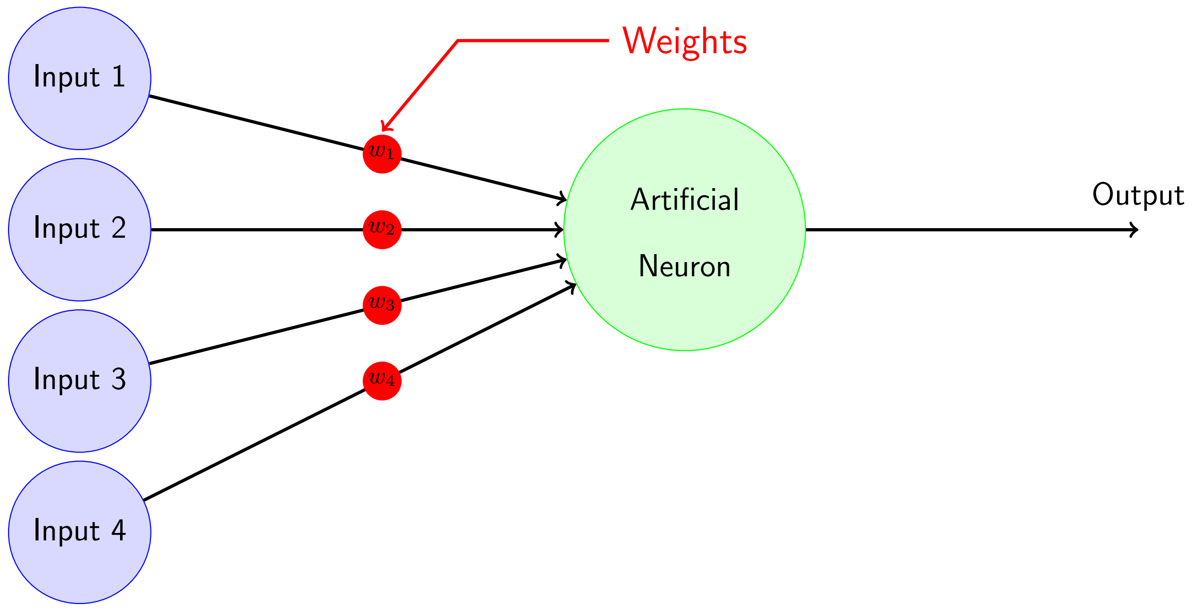
A neural network is a connectionist computational system. The algorithms we have encountered are serial in the sense that the program executes the first line of code, moves to the second, etc in a linear fashion. A true neural network does not follow a linear path but rather information is processed 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 valued is $>=$ threshold (such as 0 or 1) then it “fires”; i.e., it gives a positive output.
- If the summed valued is $<$ threshold then it does NOT “fire ”; i.e., it gives a negative output.
- If the output of the neuron matches the correct output in the training set, then we don’t modify the weights.
- If the output of the neuron does NOT match the correct output in the training set, then we modify the weights.
- The way they are modified will be discussed in an example with one neuron.

Simple 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.

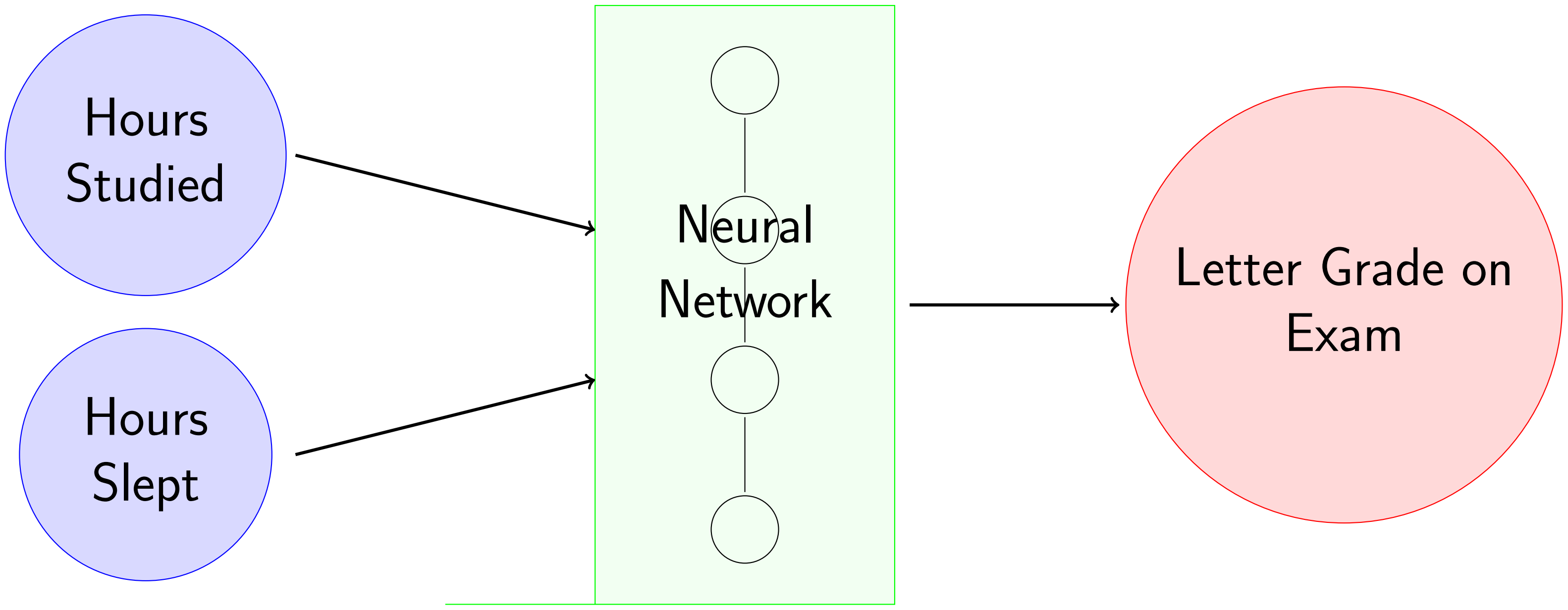
Assume that is is 7 pm on the night before a 10 a.m. exam and you are about to start studying. Let's look at some options you have.

1. Study from 7 pm until midnight (with two 30 minute breaks) $\Longrightarrow$ 4 hours preparation and get 7 hours of sleep
2. Study from 7 pm until 2 am (with three 30 minute breaks) $\Longrightarrow$ 5.5 hours preparation and get 6 hours of sleep

3. Study from 7 pm until 4 am (with four 30 minute breaks) $\Longrightarrow$ 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 the letter grade of the test.

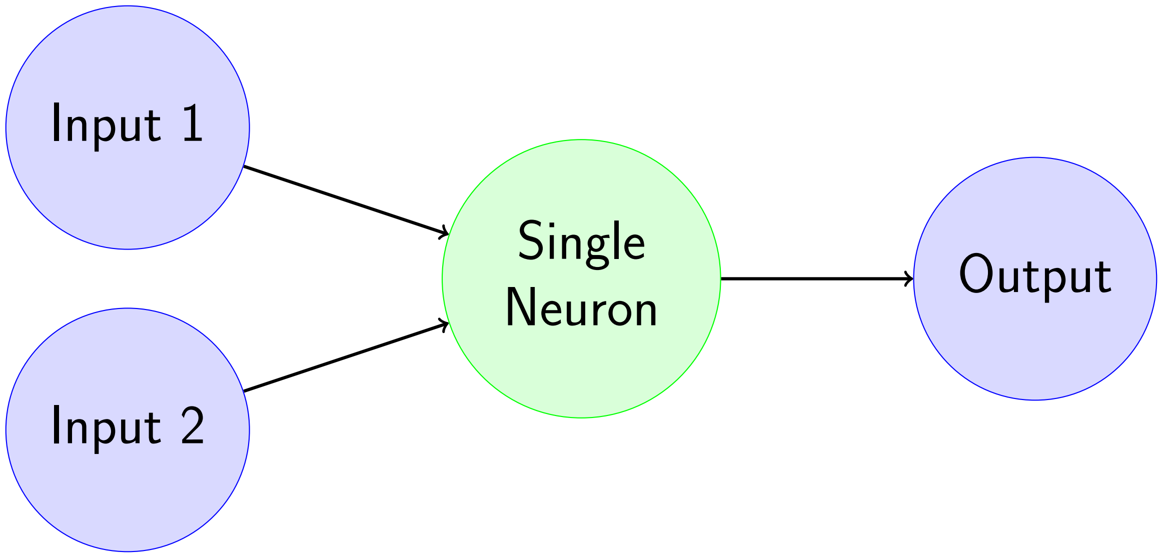


Perceptron

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

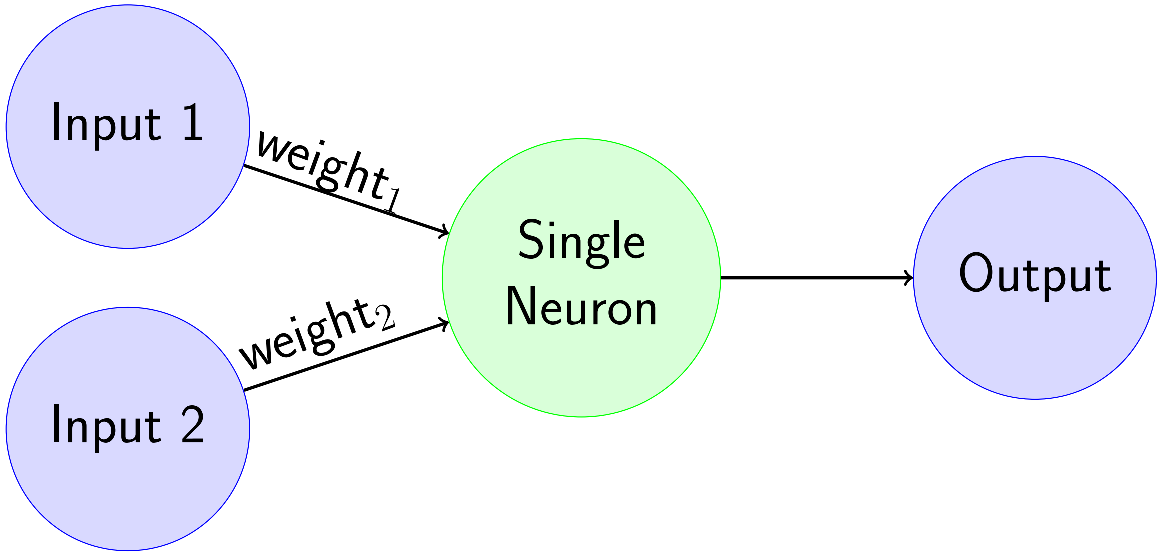
However you can have one or more inputs but a single binary output.

This is called a Perceptron – A Neural Net with One Neuron.



If we input data like the numbers 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.

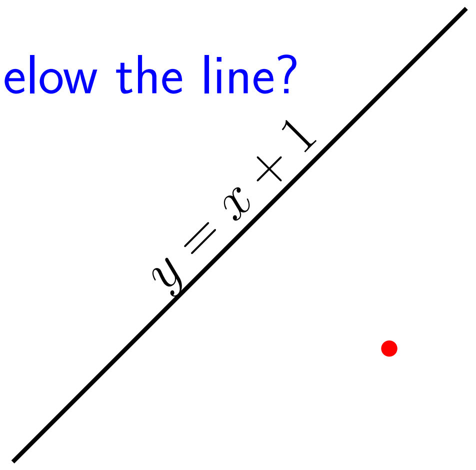


Now for each value of the input we have a weight and because we only have one "neuron" we do a single computation and then give a binary output. Recall that binary is just yes/no or 0/1, +/-, fire/don't fire, etc.

Let's look at a concrete 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 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

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

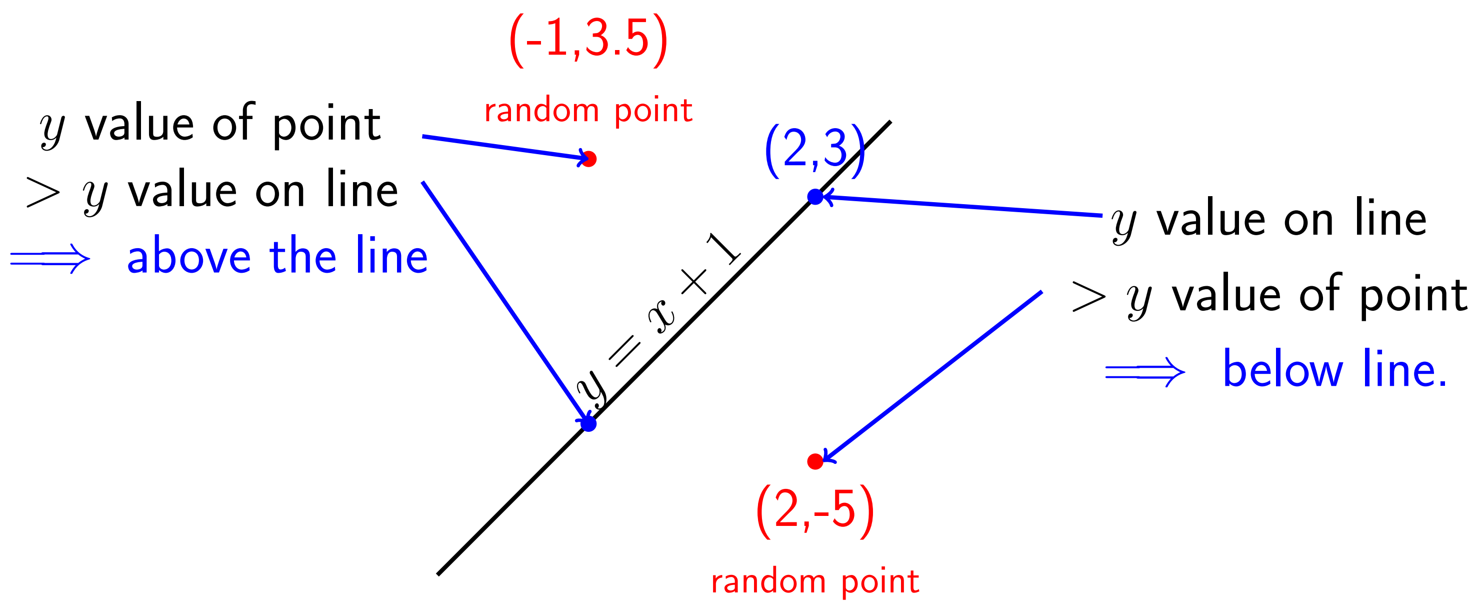
to get a number. But this can't be our output because we said our output is binary. We can simply take the sign of the output which means we are taking a threshold of zero. We take $+1$ if the sum is $>= 0$ and -1 if it is < 0.

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 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 (sum < 0) but it was above the line then we have to increase the weights so the change should be positive. This corresponds to the second point in our table where we have a positive error of $+2$. Likewise if we predicted the point was above/on the line (sum $>= 0$) 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

$$\text{weight}_{\mathrm{new}} = \text{weight}_{\mathrm{old}} + \text{error} \times \text{input} \times \text{factor}$$

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

(i) generate random point (x, y)

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

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

(iv) if $t >= 0$ 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 $j = 1, 2, \ldots, J$

(i) input jth point (x, y)

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

(iii) if $t >= 0$ point lies above/on the line; if $t < 0$ then point lies below line.

Numerical Results for Training Data

Fixed information:

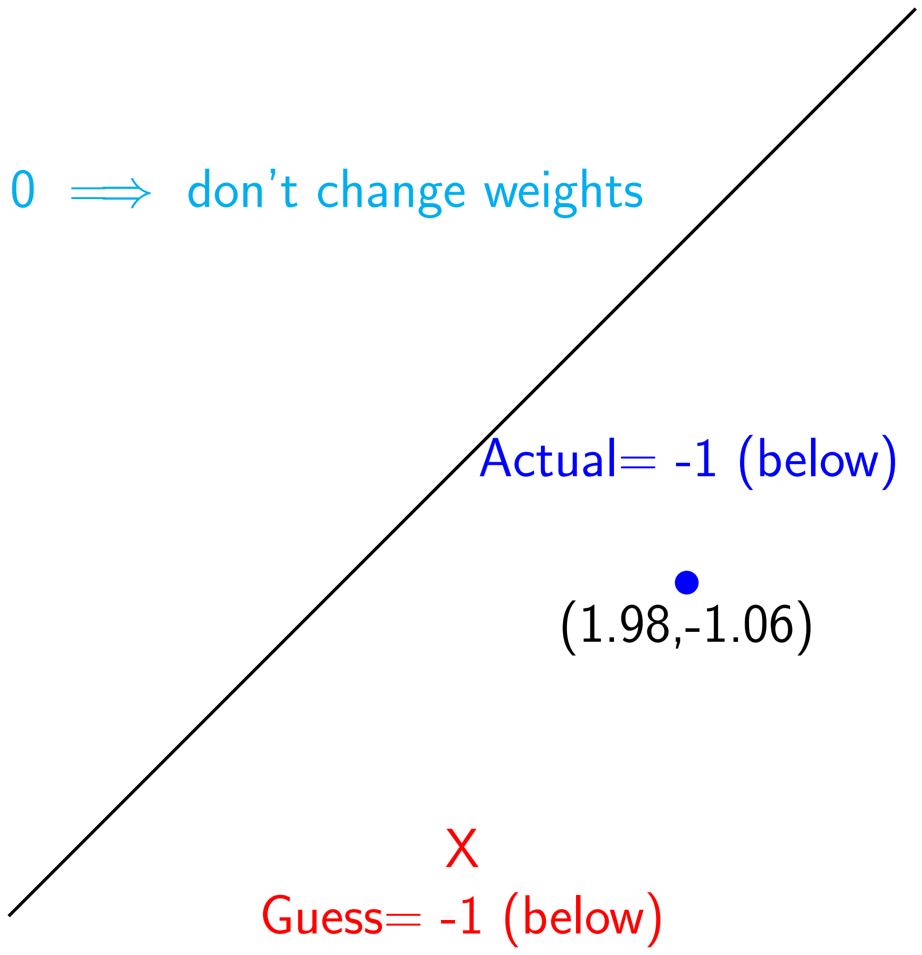
Random initial weights: $w_x = -0.9951$ and $w_y = 0.1336$

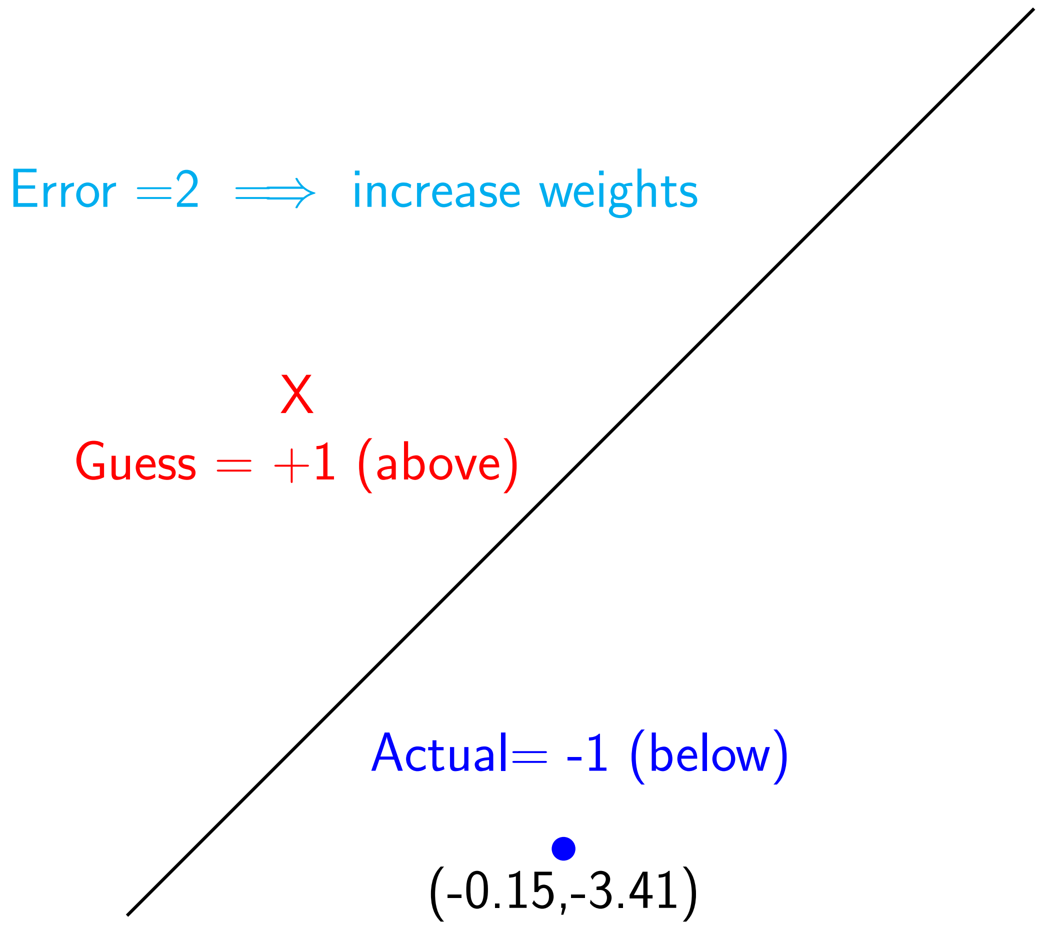
Slope of line $=$ 1; y-intercept $=$ 0

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

$\text{Error} = 0 \implies$ don't change weights




Error =2 $\implies$ increase weights
X
Guess = +1 (above)
Actual= -1 (below)
(-0.15,-3.41)

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

(i) generate random point (x, y)

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

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

(iv) if $t >= 0$ 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 $j = 1, 2, \ldots, J$

(i) input jth point (x, y)

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

(iii) if $t >= 0$ point lies above/on the line; if $t < 0$ then point lies below line.

Socrative Quiz

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) $w_x =$ error $\times\ x \times$ training rate

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

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

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

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

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

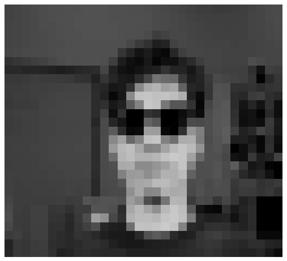
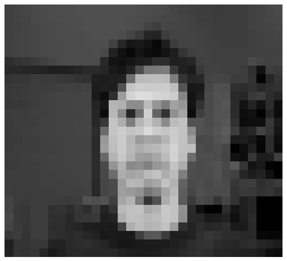
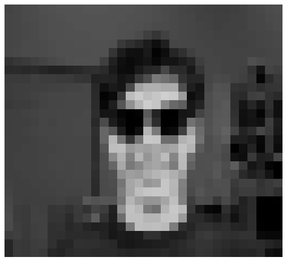
   (g) none of the above

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.

After the program is trained then a new set of images. The testing set includes different images of Glickman than in the training set as well as photographs of other people.

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

Notice that without training, the program identifies Glickman correctly approximately

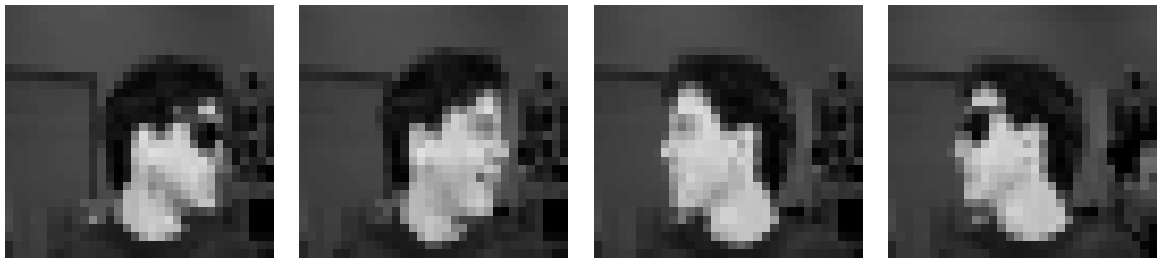
14% of the time in the training set. After one epoch it identifies Glickman approximately 94% of the time and it takes 6 epochs for the program to correctly identify each photograph in the training set.

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	14.2857	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
0	15.493	0.108187
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!

Socrative Quiz - Concepts

Answer “T” for true and “F” for false.

1. A Neural Net Algorithm models how our brain works.
2. The output of a Neural Net algorithm is any number between -1 and 1.
3. In a Perceptron Neural Net algorithm there can be multiple inputs but only a single output.
4. If there are multiple inputs to a neuron then all inputs must have the same weight.
5. In a Perceptron Neural Net algorithm there can be at most 10 artificial neurons.
6. In a Neural Net algorithm each neuron computes a quantity and if it is above a given threshold then the neuron “fires”.
7. etc.