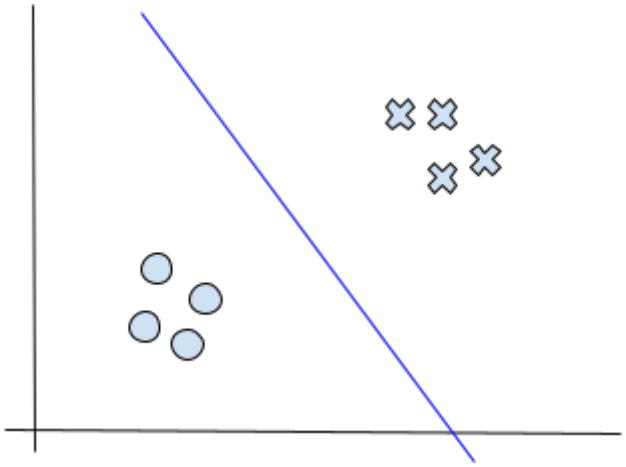
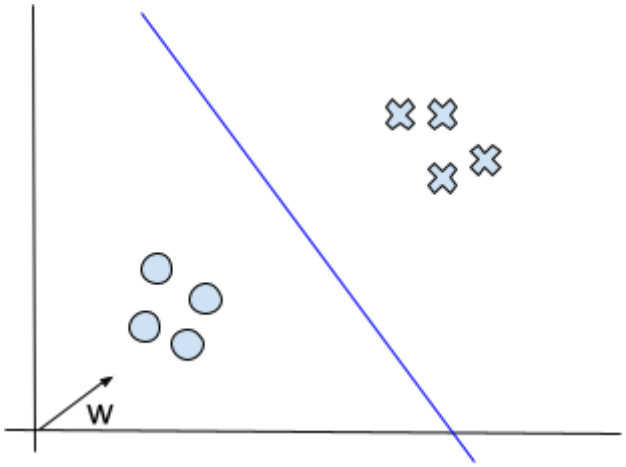
The two algorithms for classification nearest means and naive bayes are based on the mean of the data. We will move away from this and enter more general classification algorithms. Instead of relying upon means we will write classification algorithms that are based upon a linear boundary between two classes.

Geometry of a linear classifier

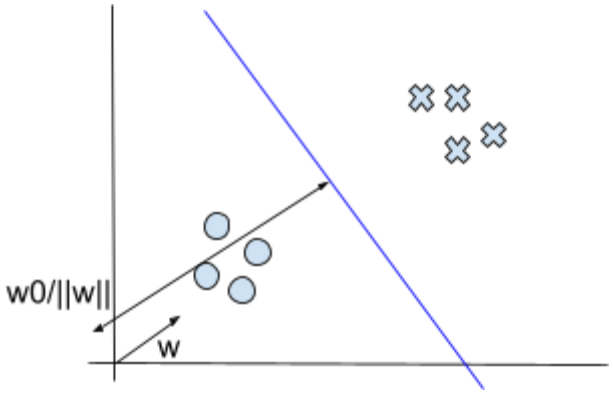
A linear boundary in two dimensional space is just a line. For example in the figure below the blue line is a linear classifier.



How do we represent this line in algebra? The line is determined by a vector that we call w and is perpendicular to the line. The vector w determines the orientation of the line.



We have another quantity called w0 that determines the distance of the line to the origin. The exact distance of the line to the origin is in fact w0/||w||.



For every point x on the blue line (which is our boundary) we have $w^T x + w_0 = 0$. For every point x to the “left” side of the blue line we have $w^T x + w_0 < 0$ and for every point x to the “right” side of the blue line we have $w^T x + w_0 > 0$.

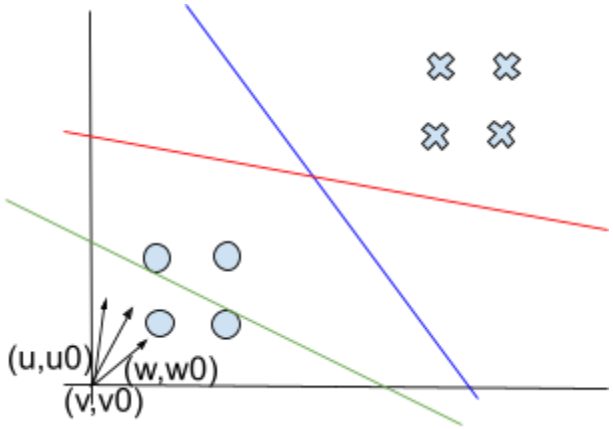
For proof of this see the slides in our google drive called “hyperplanes_as_classifiers.ppt”.

It turns out that nearest means and naive bayes are both also linear classifiers. Again see the above slides for proof.

Least squares objective

We know that the output of a linear classifier is $y' = w^T x + w_0$ where y’ is either negative or positive. The classifier itself is determined by the vector w and the number w0. For nearest means and naive bayes we know what w and w0 are. Are these the *best* values of w and w0? Are there some other values that would give a better classifier?

To better understand the problem consider the hyperplanes in the figure below. We have three lines below and each of them are given by (u,u0) (green), (v,v0) (red), and (w,w0) (blue). If we are given just these three choices how do we evaluate which of them are the best?



Let us write out the data for the above example so that we have specific numbers to work with.

X	Y
(1, 1)	-1
(1, 2)	-1
(2, 1)	-1
(2, 2)	-1
(5, 5)	1
(5, 6)	1
(6, 5)	1
(6, 6)	1

Let us estimate the vectors from the above diagram.

w=(1, 1)
v=(.5, 1)
u=(.2, 1)

After normalizing (which means divide the vector coordinates by the vector Euclidean length) we get

w = $(1/\sqrt{2}, 1/\sqrt{2})$= (.71, .71), w0 = -5
v = $(.5/\sqrt{1.25}, 1/\sqrt{1.25})$= (.45, .9), v0= -2
u = $(.2/\sqrt{1.04}, 1/\sqrt{1.04})$=(.2, .98), u0= -3.5

Now we have the data and each of our hyperplanes fully specified. In order to evaluate which of these is the best we will use the ***least squares*** function. In least squares consider the squared difference between each datapoint's prediction and its true label. Let us determine the prediction of each datapoint in our example above.

HOMEWORK: please finish this table and sum all row values in the bottom. We expect to see the sum in column $(w^Tx+w0-y)^2$ to be lowest, followed by $(u^Tx+u0-y)^2$ and then $(v^Tx+v0-y)^2$. This means that we can use this sum as a way to evaluate which hyperplane to use for classification.

Jake Nhan ->

X	Y	$w^Tx + w_0$	$(w^Tx + w_0 - y)^2$	$v^Tx + v_0$	$(v^Tx + v_0 - y)^2$	$u^Tx + u_0$	$(u^Tx + u_0 - y)^2$
(1, 1)	-1	-3.58	6.6564	-0.65	0.1225	-2.32	1.7424
(1, 2)	-1	-2.87	3.4969	0.25	5.5225	-1.34	0.1156
(2, 1)	-1	-2.87	3.4969	-0.2	0.64	-2.12	1.2544
(2, 2)	-1	-2.16	1.3456	0.7	2.89	-1.14	0.0196
(5, 5)	1	2.1	1.21	4.75	14.0625	2.4	1.96
(5, 6)	1	2.81	3.2761	5.65	21.6225	3.38	5.6644
(6, 5)	1	2.81	3.2761	5.2	17.64	2.6	2.56
(6, 6)	1	3.52	6.3504	6.1	26.01	3.58	6.6564
Sum		-0.24	29.1084	21.8	88.51	5.04	19.9728

Wai Lun ->

X	Y	w^Tx+w0	$(w^Tx+w0-y)^2$	v^Tx+v0	$(v^Tx+v0-y)^2$	u^Tx+u0	$(u^Tx+u0-y)^2$
(1, 1)	-1	-3.6	6.76	-.65	.12	-2.32	1.74
(1, 2)	-1	-2.9	3.61	.25	1.56	-1.34	.12
(2, 1)	-1	-2.9	3.61	-.2	.64	-2.12	1.25
(2, 2)	-1	-2.2	1.44	.7	2.89	-1.14	.02
(5, 5)	1	2.1	1.21	4.75	14.06	2.4	1.96
(5, 6)	1	2.8	3.24	5.65	21.62	3.38	5.66
(6, 5)	1	2.8	3.24	5.2	17.64	2.6	2.56
(6, 6)	1	3.5	6.25	6.1	26.01	3.58	6.66
Sums:		**-0.4**	**29.36**	**21.8**	**84.54**	**5.04**	**19.96**

In our training data we have a label y for each x. Our goal is that the prediction matches the true label. How do we achieve this? We want the output $w^Tx + w_0$ to be close in value to the true label y. We can express that with the objective $argmin_{w,w_0} (w^Tx + w_0 - y)^2$. Why do we have a square in this objective? We want the minimum value to be 0 and so we square the objective.

We can achieve a zero minimum by taking the absolute value. We will see a little later that by taking the square we will be able to solve our problem much more easily than if we were to take the absolute value.

If we are given training data (x_i, y_i), $x_i \in R^m$, $y_i \in \{+1, -1\}$, $i = 0 \ldots n-1$ then we can write the above objective as:

$$argmin_{w,w_0} \sum_{i=0}^{n-1} (w^T x_i + w_0 - y_i)^2$$

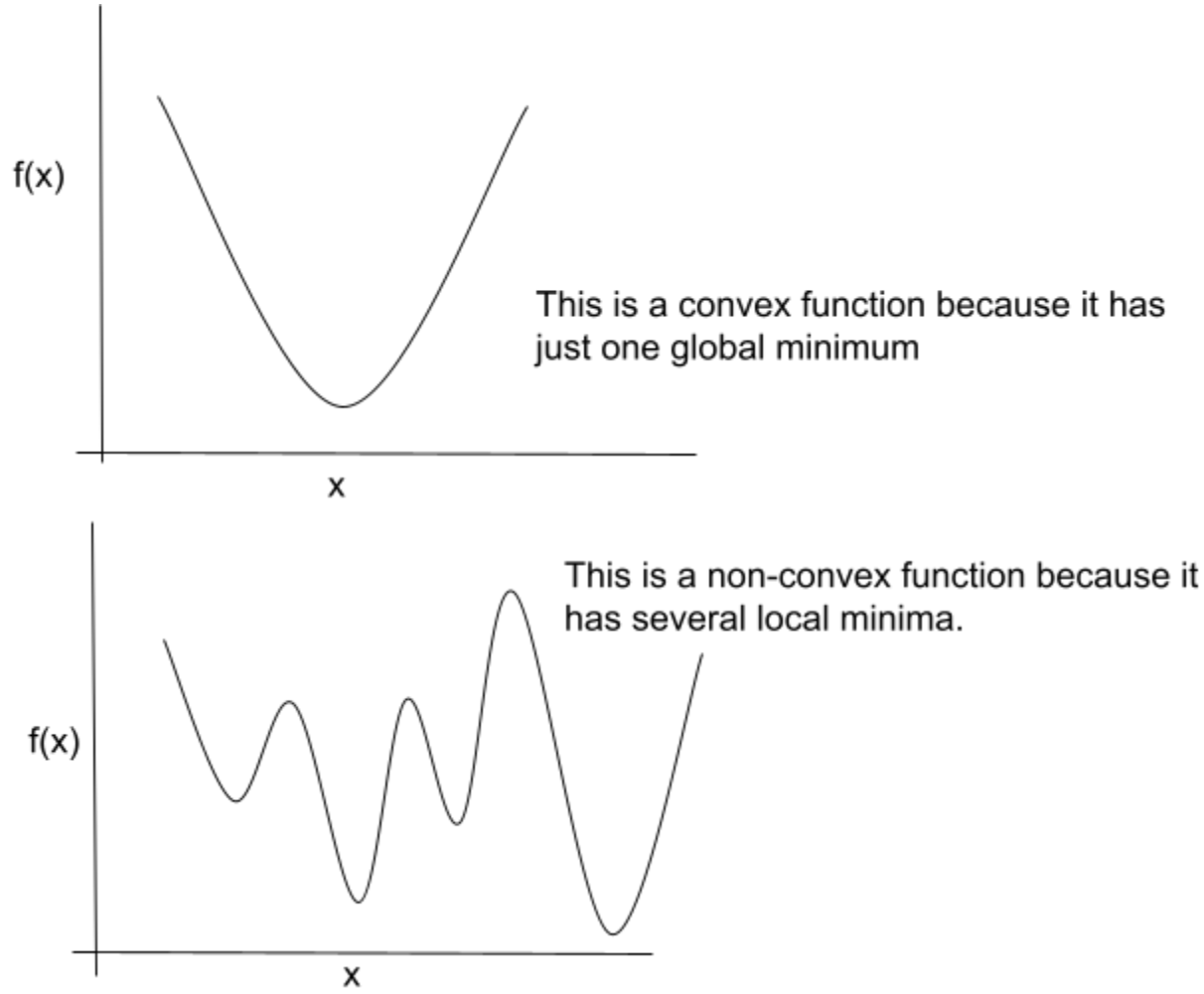
More formally the expression $(w^T x_i + w_0 - y_i)^2$ for a single datapoint x_i is called the ***loss*** of x_i. And the total sum loss of all datapoints $\sum_{i=0}^{n-1} (w^T x_i + w_0 - y_i)^2$ is called the ***empirical risk***. Most of machine learning and AI is entirely to find the model parameters that minimize the empirical risk.

How do we solve the above optimization problem? We use basic calculus.

Gradient descent

Consider a function f(w). How do we find w such that f(w) is minimum? Suppose your function is $f(w) = (w^T x + w_0 - y)^2$. Before determining the minimum value of f(w) we want to know how many such values.

Convexity How many values of w exist where f(w) is minimum? Or how many minimum points does f(w) have? This is known as the convexity of a function. A convex function is one where f(w) has just one unique minimum value. See our examples below. A function f(w) is convex if and only if (iff) the second derivative of f(w) w.r.t. is always non-negative. For our function $f(w) = (w^T x + w_0 - y)^2$ the first derivative $df/dw = 2(w^T x + w_0 - y)x$ and the second derivative $d^2 f/dw^2 = 2x^2 >= 0$.Therefore our least squares loss is convex.



Vector derivative The above example was for one dimensional data where both all variables are just numbers. If we have two dimensional then w is also two dimensional. In that case what is df/dw? The derivative of a vector is just the derivative of each component of the vector $df/dw = (df/dw_1, df/dw_2)$.

For two dimensional data we can write our loss as $f(w, w_0, x, y) = (w^T x + w_0 - y)^2 = (w_1 x_1 + w_2 x_2 + w_0 - y)^2$. Now we can clearly define the derivative.

We can use the derivative df/dw to find the minimum value of f(w) with the gradient descent algorithm. The gradient of a function $\nabla f_w(w, w_0, x, y)$is defined as a vector of first derivatives. For our two dimensional example $f(w, w_0, x, y) = (w^T x + w_0 - y)^2 = (w_1 x_1 + w_2 x_2 + w_0 - y)^2$

the gradient vector is defined as $\nabla f_w(w, w_0, x, y) = (df/dw_1, df/dw_2)$. According to Calculus the gradient of a function gives us a vector of maximum increase of the function. In other words if we move our variables in the direction of the gradient then our function will increase. So for example suppose we did $w' = w + \nabla f_w(w, w_0, x, y)$. Then $f(w', w_0, x, y) > f(w, w_0, x, y)$.

Let us first sketch the algorithm below:

Input: (x_i, y_i), $x_i \in R^m$, $y_i \in \{+1, -1\}$, $i = 0 \ldots n-1, n > 2, m > 1$

Output: $w \in R^m$ and $w_0 \in R$ that minimize the empirical risk $\sum_{i=0}^{n-1} (w^T x_i + w_0 - y_i)^2$

Algorithm:

1. Initialize w and w0 to random values between [-.01, .01]
2. Set learning rate $\eta = .01$
3. Calculate obj = $\sum_{i=0}^{n-1} (w^T x_i + w_0 - y_i)^2$
4. Set prev_obj = obj + 10
5. while(prev_obj - obj > .001):
    a. Set prev_obj = obj
    b. Perform the update $w = w - \eta \nabla f_w$
    c. Perform the update $w_0 = w_0 - \eta \nabla f_{w_0}$
    d. Recalculate objective obj = $\sum_{i=0}^{n-1} (w^T x_i + w_0 - y_i)^2$

The key things to work out for this problem are the gradients. Can you calculate and say what the gradient should be?

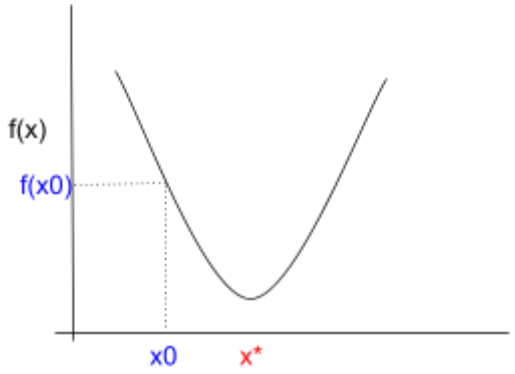
$$df/dw_1 = 2 \sum_{i=0}^{n-1} (w^T x_i + w_0 - y_i) x_{i1}$$

In general for the jth coordinate we have

$$df/dw_j = 2 \sum_{i=0}^{n-1} (w^T x_i + w_0 - y_i) x_{ij}$$

Remember that $\nabla f(w) = (df/dw_1, df/dw_2, \ldots, df/dw_m)$

$$df/dw_0 = 2 \sum_{i=0}^{n-1} (w^T x_i + w_0 - y_i)$$



So far it looks like least squares is a reasonable good strategy for linearly separable.