ADVANCED CLASSIFICATION TECHNIQUES

David Kauchak
CS 159 – Fall 2014

Admin

Quiz #3
- mean: 25.25 (87%)
- median: 26 (90%)

Assignment 5 graded

ML lab next Tue (there will be candy to be won ☺)

Admin

Project proposal: tonight at 11:59pm

Assignment 7: Friday at 5pm
- See my e-mail (Wednesday)
- Both p(*|positive) and p(*|negative) should use exactly the same set of features
  - specifically, all the words that were seen during training (with **either** label)
  - this is one of the main reasons we need smoothing!)

$$p(positive)\prod_{j=1}^{m} p(w_j \mid positive)^{x_j} \longleftrightarrow p(negative)\prod_{j=1}^{m} p(w_j \mid negative)^{x_j}$$

Machine Learning: A Geometric View

Apples vs. Bananas

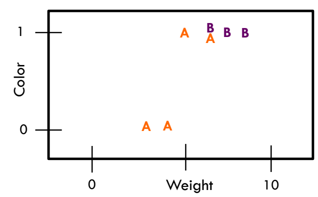
Weight	Color	Label
4	Red	Apple
5	Yellow	Apple
6	Yellow	Banana
3	Red	Apple
7	Yellow	Banana
8	Yellow	Banana
6	Yellow	Apple

Can we visualize this data?

Apples vs. Bananas

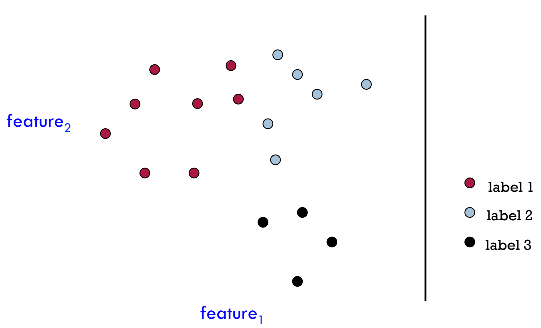
Turn features into numerical values

Weight	Color	Label
4	0	Apple
5	1	Apple
6	1	Banana
3	0	Apple
7	1	Banana
8	1	Banana
6	1	Apple

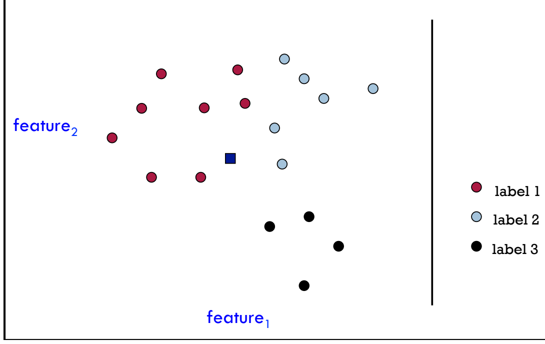


We can view examples as points in an *n*-dimensional space where *n* is the number of features called the feature space

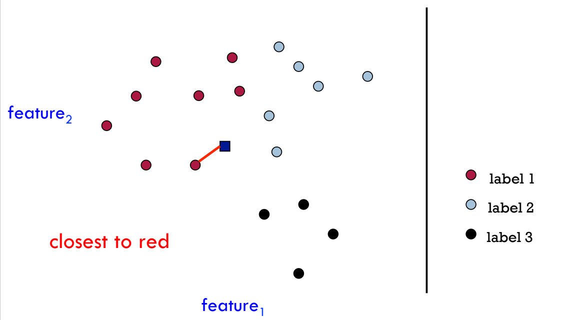
Examples in a feature space



Test example: what class?



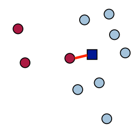
Test example: what class?



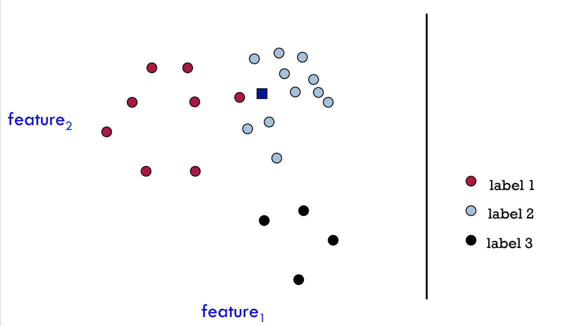
Another classification algorithm?

To classify an example ***d***:

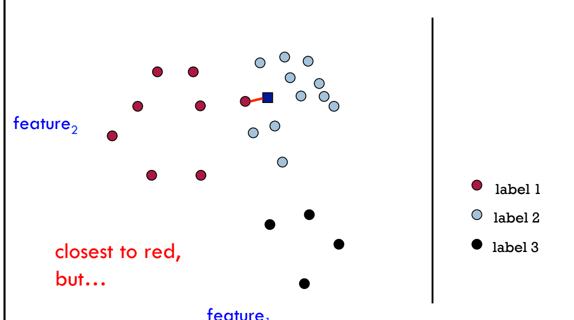
Label ***d*** with the label of the closest example to ***d*** in the training set



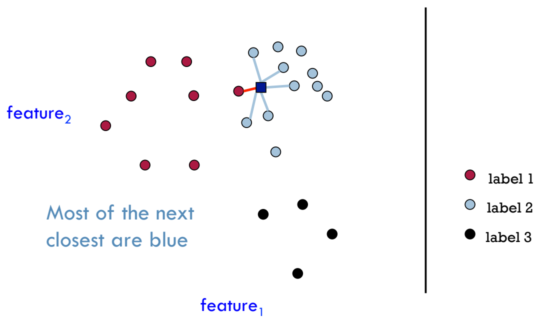
What about this example?



What about this example?



What about this example?



k-Nearest Neighbor (k-NN)

To classify an example ***d***:

- Find ***k*** nearest neighbors of ***d***
- Choose as the label the majority label within the ***k*** nearest neighbors

k-Nearest Neighbor (k-NN)

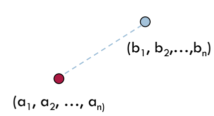
To classify an example ***d***:

- Find ***k*** *nearest* neighbors of ***d***
- Choose as the label the majority label within the ***k*** nearest neighbors

How do we measure "nearest"?

Euclidean distance

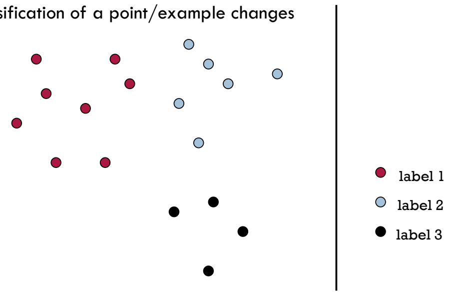
Euclidean distance! (or L1 or …)



$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

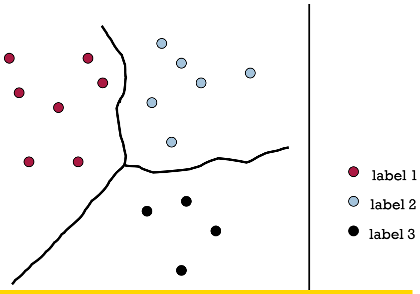
Decision boundaries

The **decision boundaries** are places in the features space where the classification of a point/example changes



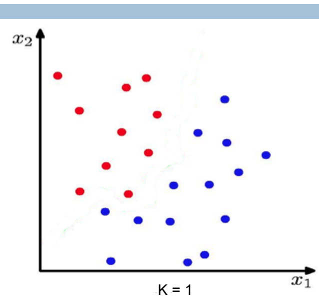
Where are the decision boundaries for k-NN?

k-NN decision boundaries



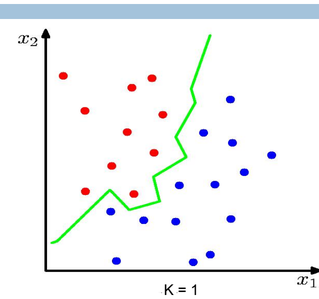
k-NN gives locally defined decision boundaries between classes

K Nearest Neighbour (*k*NN) Classifier



What is the decision boundary for k-NN for this one?

K Nearest Neighbour (*k*NN) Classifier



Machine learning models

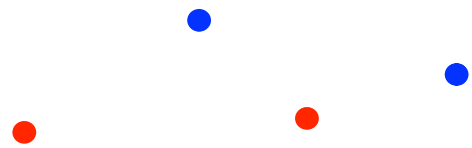
Some machine learning approaches make strong assumptions about the data

- If the assumptions are true this can often lead to better performance
- If the assumptions aren't true, they can fail miserably

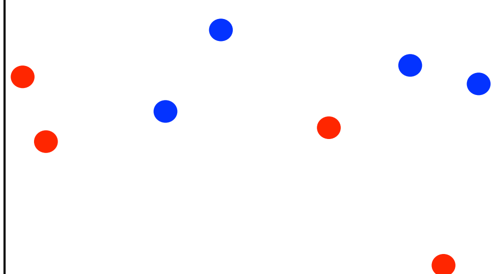
Other approaches don't make many assumptions about the data

- This can allow us to learn from more varied data
- But, they are more prone to overfitting
- and generally require more training data

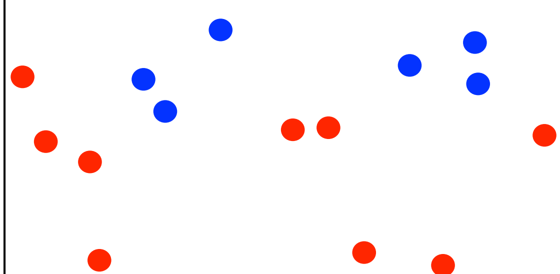
What is the data generating distribution?



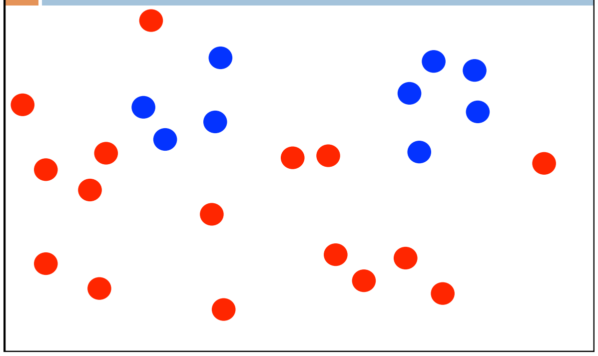
What is the data generating distribution?



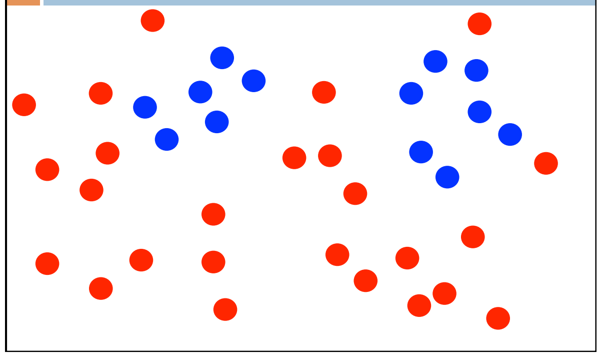
What is the data generating distribution?



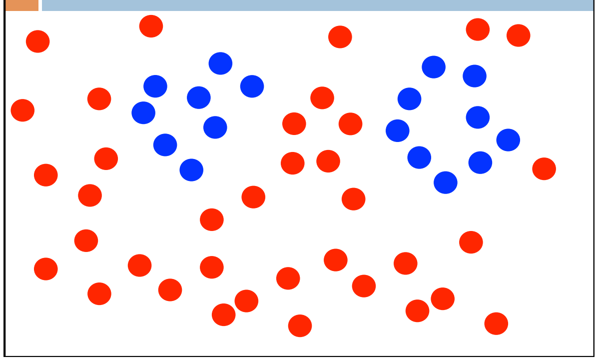
What is the data generating distribution?



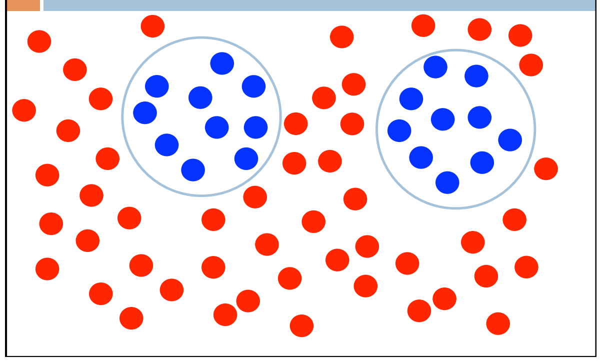
What is the data generating distribution?



What is the data generating distribution?



Actual model

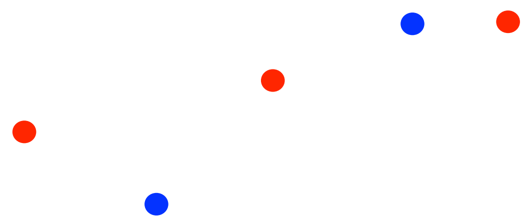


Model assumptions

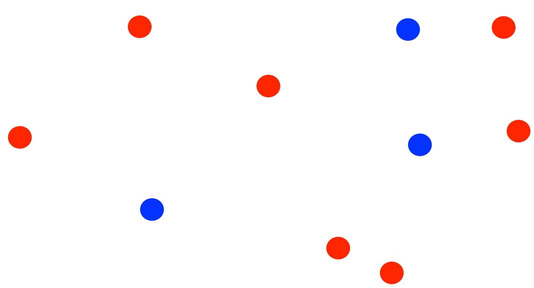
If you don't have strong assumptions about the model, it can take you a longer to learn

Assume now that our model of the blue class is two circles

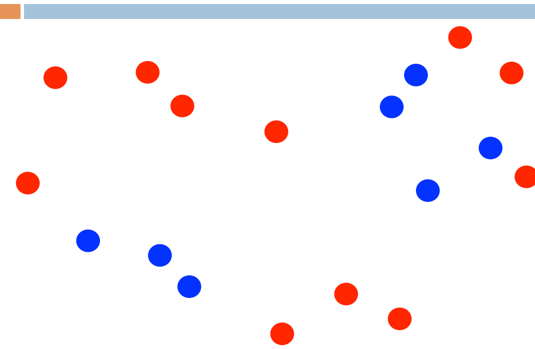
What is the data generating distribution?



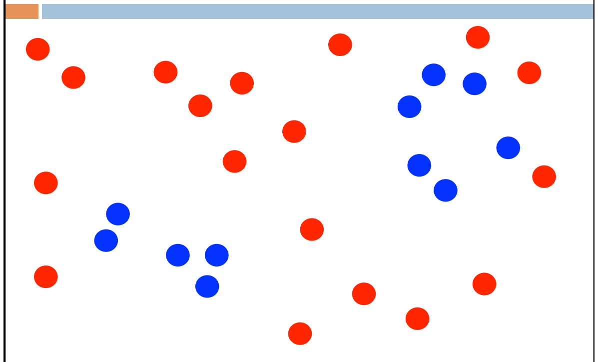
What is the data generating distribution?



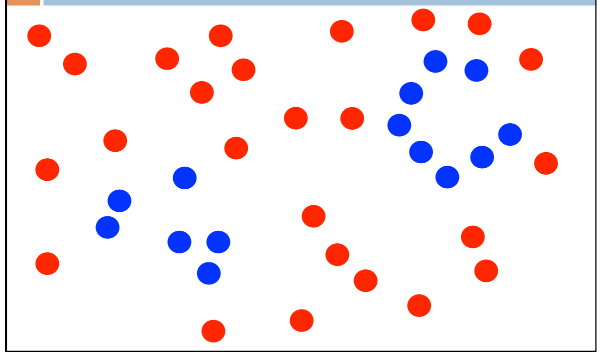
What is the data generating distribution?



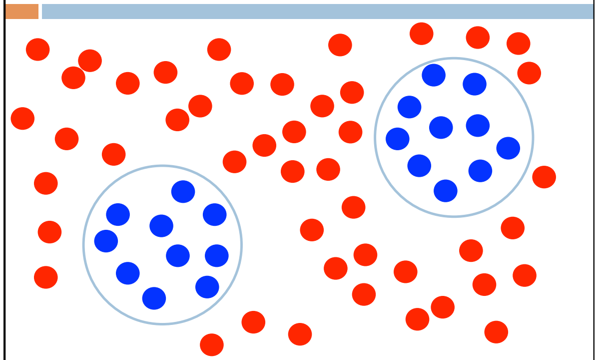
What is the data generating distribution?



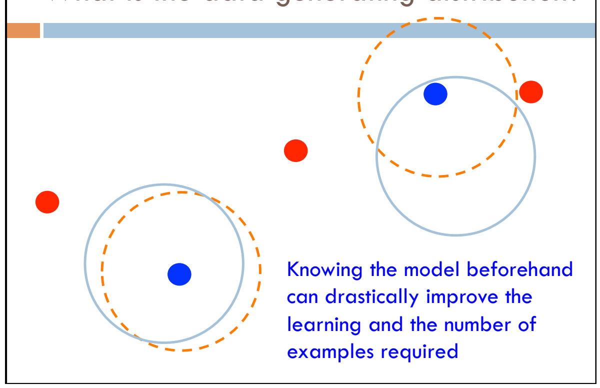
What is the data generating distribution?



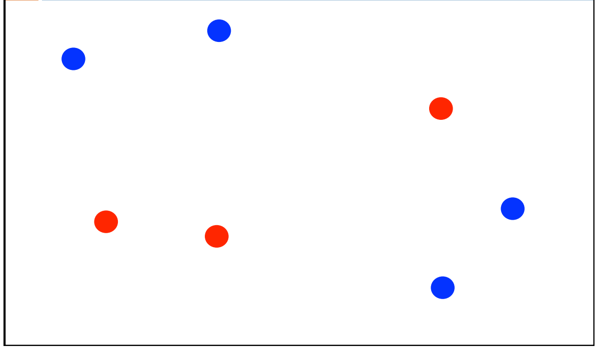
Actual model



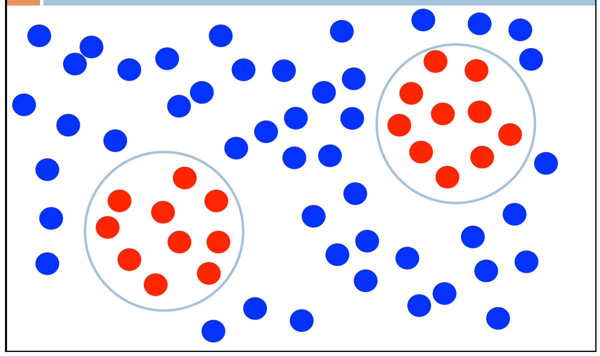
What is the data generating distribution?



What is the data generating distribution?



Make sure your assumption is correct, though!

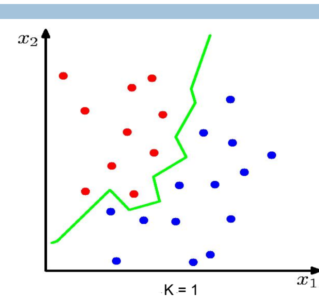


Machine learning models

What were the *model* assumptions (if any) that *k*-NN and NB made about the data?

Are there training data sets that could never be learned correctly by these algorithms?

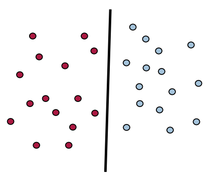
k-NN model



Linear models

A strong assumption is *linear separability*:
- in 2 dimensions, you can separate labels/classes by a line
- in higher dimensions, need hyperplanes

A *linear model* is a model that assumes the data is linearly separable



Hyperplanes

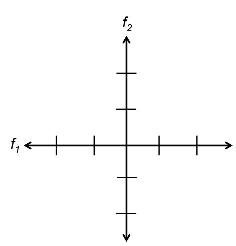
A hyperplane is line/plane in a high dimensional space

What defines a line?
What defines a hyperplane?

Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$



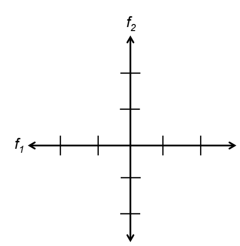
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

What does this line look like?



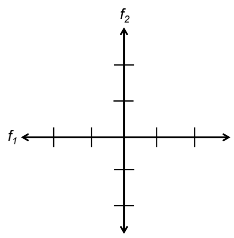
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

-2	1
-1	0.5
0	0
1	-0.5
2	-1



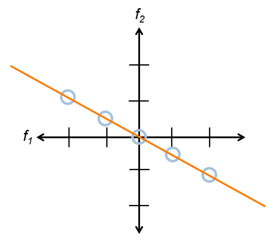
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

-2	1
-1	0.5
0	0
1	-0.5
2	-1



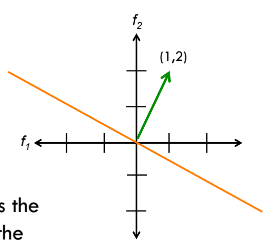
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

w=(1,2)

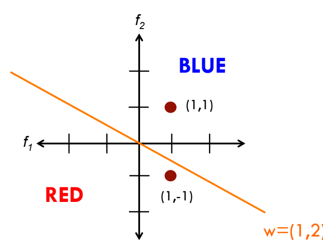


We can also view it as the line perpendicular to the *weight vector*

Classifying with a line

Mathematically, how can we classify points based on a line?

$$0 = 1 f_1 + 2 f_2$$



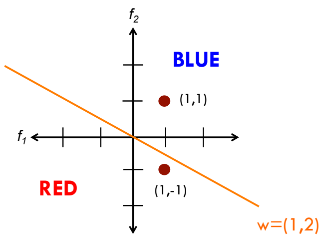
Classifying with a line

Mathematically, how can we classify points based on a line?

$0 = 1f_1 + 2f_2$

(1,1): $1*1+2*1 = 3$

(1,-1): $1*1+2*-1 = -1$



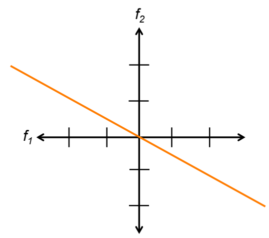
The sign indicates which side of the line

Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$0 = w_1 f_1 + w_2 f_2$

$0 = 1f_1 + 2f_2$



How do we move the line off of the origin?

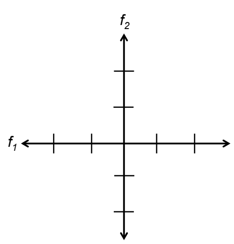
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$a = w_1 f_1 + w_2 f_2$

$-1 = 1f_1 + 2f_2$

-2
-1
0
1
2



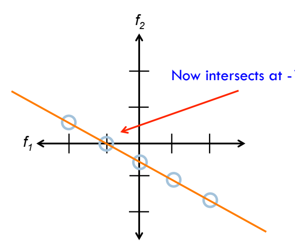
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$a = w_1 f_1 + w_2 f_2$

$-1 = 1f_1 + 2f_2$

-2	0.5
-1	0
0	-0.5
1	-1
2	-1.5



Linear models

A linear model in *n*-dimensional space (i.e. *n* features) is define by *n*+1 weights:

In two dimensions, a line:

$$0 = w_1 f_1 + w_2 f_2 + b$$ (where b = -a)

In three dimensions, a plane:

$$0 = w_1 f_1 + w_2 f_2 + w_3 f_3 + b$$

In *n*-dimensions, a *hyperplane*

$$0 = b + \sum_{i=1}^{n} w_i f_i$$

Classifying with a linear model

We can classify with a linear model by checking the sign:

$f_1, f_2, \ldots, f_m$ → classifier

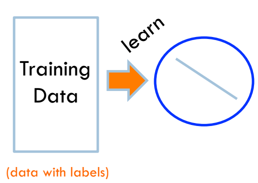
$b + \sum_{j=1}^{m} w_j f_j > 0$ Positive example

$b + \sum_{j=1}^{m} w_j f_j < 0$ Negative example

Learning a linear model

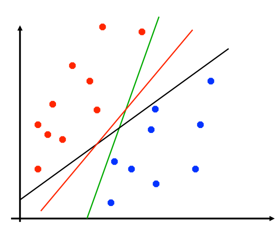
Geometrically, we know what a linear model represents

Given a linear model (i.e. a set of weights and b) we can classify examples

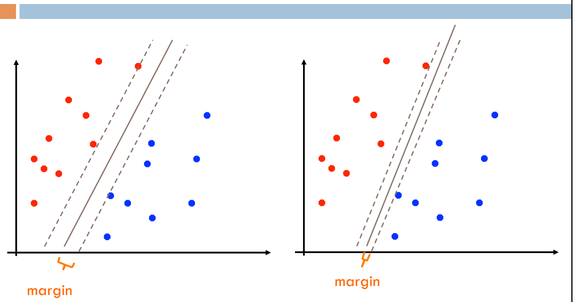


How do we learn a linear model?

Which hyperplane would you choose?

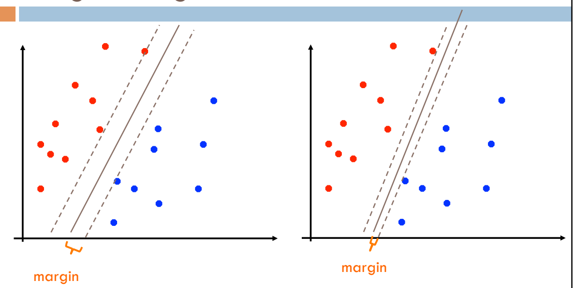


Large margin classifiers



Choose the line where the distance to the nearest point(s) is as large as possible

Large margin classifiers



The margin of a classifier is the distance to the closest points of either class

Large margin classifiers attempt to maximize this

Large margin classifier setup

Select the hyperplane with the largest margin where the points are classified correctly!

Setup as a constrained optimization problem:

$$\max_{w,b} \quad \text{margin}(w,b)$$

subject to:

$$y_i(w \cdot x_i + b) > 0 \quad \forall i$$

what does this say?

y_i: label for example i, either 1 (positive) or -1 (negative)
x_i: our feature vector for example i

Large margin classifier setup

$$\max_{w,b} \quad \text{margin}(w,b)$$

subject to:

$$y_i(w \cdot x_i + b) > 0 \quad \forall i$$

$$\max_{w,b} \quad \text{margin}(w,b)$$

subject to:

$$y_i(w \cdot x_i + b) \ge c \quad \forall i$$
$$c > 0$$

Are these equivalent?

Large margin classifier setup

$$\max_{w,b} \ \ \text{margin}(w,b)$$

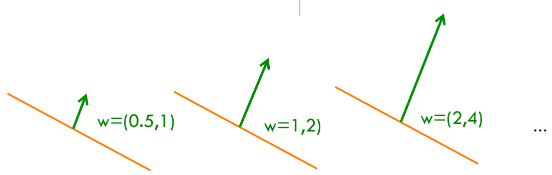
subject to:

$$y_i(w \cdot x_i + b) > 0 \ \ \forall i$$

$$\max_{w,b} \ \ \text{margin}(w,b)$$

subject to:

$$y_i(w \cdot x_i + b) \geq c \ \ \forall i$$
$$c > 0$$



Large margin classifier setup

$$\max_{w,b} \ \ \text{margin}(w,b)$$

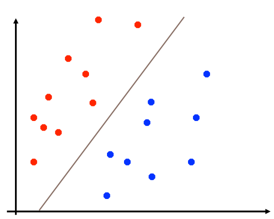
subject to:

$$y_i(w \cdot x_i + b) \geq 1 \ \ \forall i$$

We'll assume c =1, however, any c > 0 works

Measuring the margin

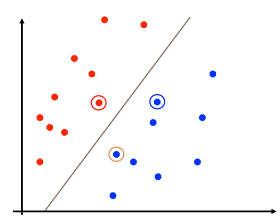
How do we calculate the margin?



Support vectors

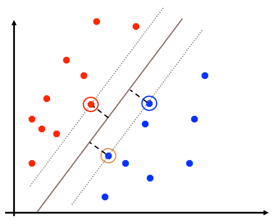
For any separating hyperplane, there exist some set of "closest points"

These are called the support vectors



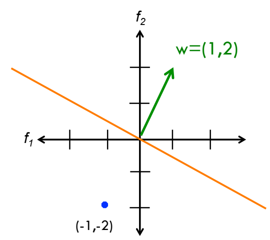
Measuring the margin

The margin is the distance to the support vectors, i.e. the "closest points", on either side of the hyperplane



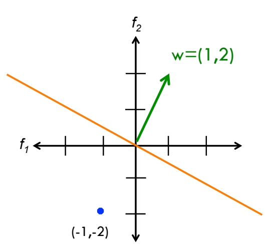
Distance from the hyperplane

How far away is this point from the hyperplane?



Distance from the hyperplane

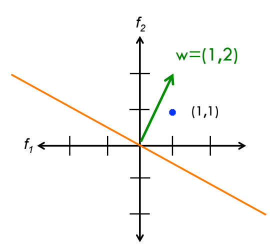
How far away is this point from the hyperplane?



$$d = \sqrt{1^2 + 2^2} = \sqrt{5}$$

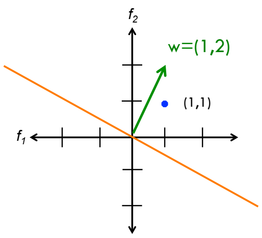
Distance from the hyperplane

How far away is this point from the hyperplane?



Distance from the hyperplane

How far away is this point from the hyperplane?

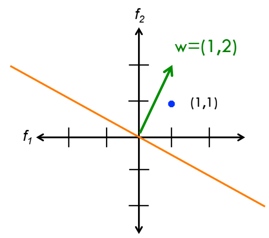


$$d(x) = \frac{w}{\|w\|} \cdot x + b$$

length normalized weight vectors

Distance from the hyperplane

How far away is this point from the hyperplane?



$$d(x) = \frac{w}{\|w\|} \cdot x + b$$

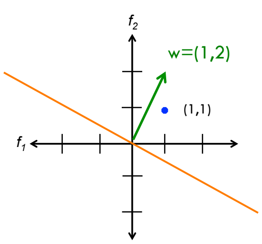
$$= \frac{1}{\sqrt{5}}(w_1 x_1 + w_2 x_2) + b$$

$$= \frac{1}{\sqrt{5}}(1*1 + 1*2) + 0$$

$$= 1.34$$

Distance from the hyperplane

Why length normalized?

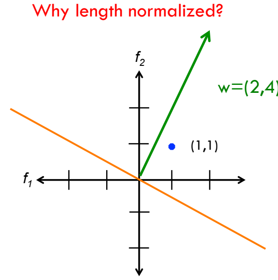


$$d(x) = \frac{w}{\|w\|} \cdot x + b$$

length normalized weight vectors

Distance from the hyperplane

Why length normalized?

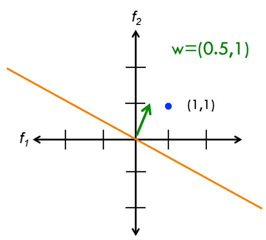


$$d(x) = \frac{w}{\|w\|} \cdot x + b$$

length normalized weight vectors

Distance from the hyperplane

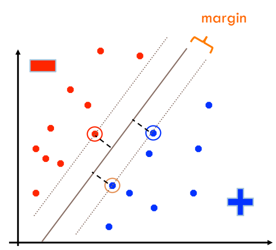
Why length normalized?



$$d(x) = \frac{w}{\|w\|} \cdot x + b$$

length normalized weight vectors

Measuring the margin



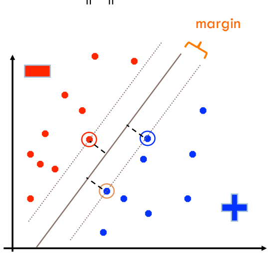
Thought experiment:

Someone gives you the optimal support vectors

Where is the max margin hyperplane?

Measuring the margin

$$d(x) = \frac{w}{\|w\|} \cdot x + b$$



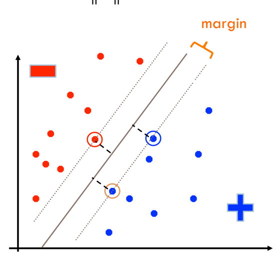
Margin = $(d^+ - d^-)/2$

Max margin hyperplane is halfway in between the positive support vectors and the negative support vectors

Why?

Measuring the margin

$$d(x) = \frac{w}{\|w\|} \cdot x + b$$



Margin = $(d^+ - d^-)/2$

Max margin hyperplane is halfway in between the positive support vectors and the negative support vectors

- All support vectors are the same distance
- To maximize, hyperplane should be directly in between

Measuring the margin

$$d(x) = \frac{w}{\|w\|} \cdot x + b$$

Margin = (d$^+$-d$^-$)/2

$$\text{margin } = \frac{1}{2}\left(\frac{w}{\|w\|} \cdot x^+ + b - \left(\frac{w}{\|w\|} \cdot x^- + b\right)\right)$$

What is *wx+b* for support vectors?

Hint:

$$\max_{w,b} \quad \text{margin}(w,b)$$

subject to:

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$

Measuring the margin

$$\max_{w,b} \quad \text{margin}(w,b)$$

subject to:

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$

The support vectors have $y_i(w \cdot x_i + b) = 1$

Otherwise, we could make the margin larger!

Measuring the margin

$$d(x) = \frac{w}{\|w\|} \cdot x + b$$

Margin = (d$^+$-d$^-$)/2

$$\text{margin } = \frac{1}{2}\left(\frac{w}{\|w\|} \cdot x^+ + b - \left(\frac{w}{\|w\|} \cdot x^- + b\right)\right)$$

$$= \frac{1}{2}\left(\frac{1}{\|w\|} - \frac{-1}{\|w\|}\right)$$

negative example

$$= \frac{1}{\|w\|}$$

Maximizing the margin

$$\max_{w,b} \quad \frac{1}{\|w\|}$$

subject to:

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$

Maximizing the margin is equivalent to minimizing ||w||!
(subject to the separating constraints)

Maximizing the margin

$$\min_{w,b} \quad \|w\|$$

subject to:

$$y_i(w \cdot x_i + b) \ge 1 \quad \forall i$$

Maximizing the margin is equivalent to minimizing ||w||!
(subject to the separating constraints)

Maximizing the margin

The minimization criterion wants w to be as small as possible

$$\min_{w,b} \quad \|w\|$$

subject to:

$$y_i(w \cdot x_i + b) \ge 1 \quad \forall i$$

The constraints:

1. make sure the data is separable
2. encourages w to be larger (once the data is separable)

Maximizing the margin: the real problem

$$\min_{w,b} \quad \|w\|^2$$

subject to:

$$y_i(w \cdot x_i + b) \ge 1 \quad \forall i$$

What's the difference?

Maximizing the margin: the real problem

$$\min_{w,b} \quad \|w\|^2$$

subject to:

$$y_i(w \cdot x_i + b) \ge 1 \quad \forall i$$

Why the squared?

Maximizing the margin: the real problem

$$\min_{w,b} \quad \|w\| = \sqrt{\sum_i w_i^{\,2}}$$

subject to:

$$y_i(w \cdot x_i + b) \ge 1 \quad \forall i$$

$$\min_{w,b} \quad \|w\|^2 = \sum_i w_i^{\,2}$$

subject to:

$$y_i(w \cdot x_i + b) \ge 1 \quad \forall i$$

Minimizing $||w||$ is equivalent to minimizing $||w||^2$

The sum of the squared weights is a convex function!

Support vector machine problem

$$\min_{w,b} \quad \|w\|^2$$

subject to:

$$y_i(w \cdot x_i + b) \ge 1 \quad \forall i$$

This is a version of a quadratic optimization problem

Maximize/minimize a quadratic function

Subject to a set of linear constraints

Many, many variants of solving this problem (we'll see one in a bit)

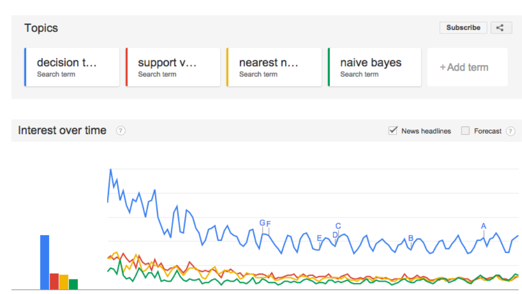
Support vector machines

One of the most successful (if not the most successful) classification approach:

decision tree	About 2,240,000 results (0.32 sec)
Support vector machine	About 2,180,000 results (0.36 sec)
k nearest neighbor	About 844,000 results (0.33 sec)
Naïve Bayes	About 71,300 results (0.32 sec)

Google scholar

Trends over time



Other successful classifiers in NLP

Perceptron algorithm
- Linear classifier
- Trains "online"
- Fast and easy to implement
- Often used for tuning parameters (not necessarily for classifying)

Logistic regression classifier (aka Maximum entropy classifier)
- Probabilistic classifier
- Doesn't have the NB constraints
- Performs very well
- More computationally intensive to train than NB

Resources

SVM
- SVM light: http://svmlight.joachims.org/
- Others, but this one is awesome!

Maximum Entropy classifier
- http://nlp.stanford.edu/software/classifier.shtml

General ML frameworks:
- Python: scikit-learn, MLpy
- Java: Weka (http://www.cs.waikato.ac.nz/ml/weka/)
- Many others…