Machine Learning

CS 165B

Prof. Matthew Turk

Wednesday, April 27, 2016

Today

- Linear learning models (cont.)
- Midterm review

Notes

- This lecture is being recorded, so be sure to use the PowerPoint version to play the lecture (slides and audio)!
- Reminder: my office hours this week are Thursday 9:30-11:30am
- Today:
  - Finish least-squares concepts
  - Midterm review
  - Begin perceptron coverage

Least-squares regression for classification

- We can use regression techniques to learn a binary classifier by encoding the two classes as real numbers and thresholding the output function
  - Label positive examples with +1 and negative examples with –1
  - I.e., $y_i \epsilon \{+1, -1\}$

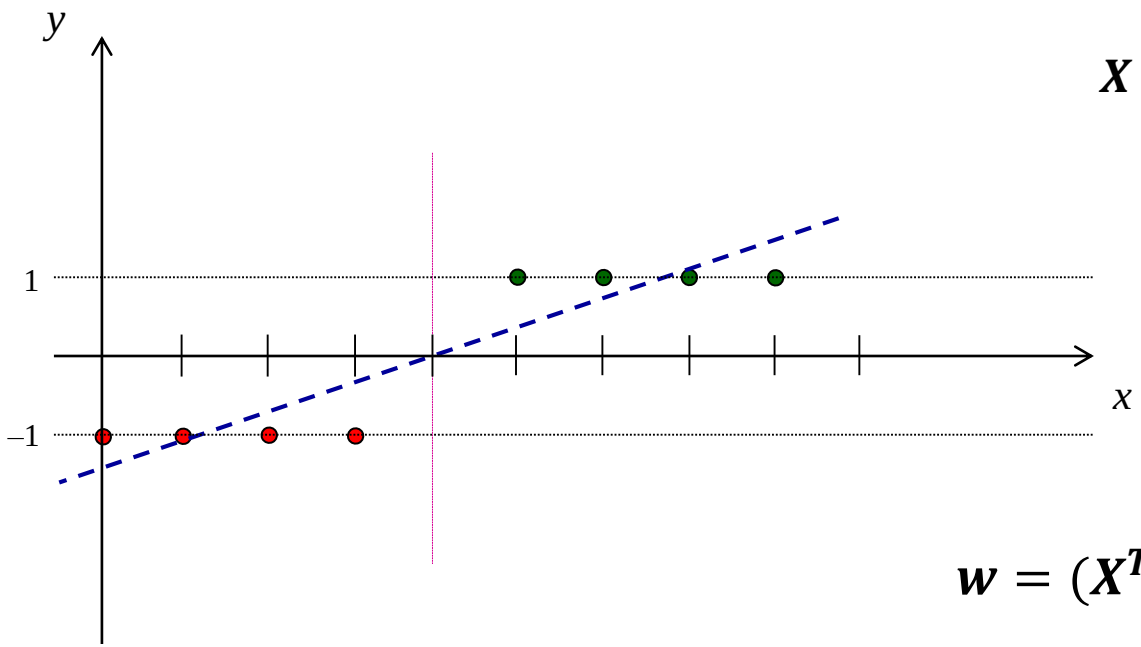
$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- Assign class $\hat{y} = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x})$

$$\text{sgn}(x) = \text{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Least-squares regression for classification

Training data: $\{(x_i, y_i)\} = \{ (0, -1), (1, -1), (2, -1), (3, -1), (5, 1), (6, 1), (7, 1), (8, 1) \}$



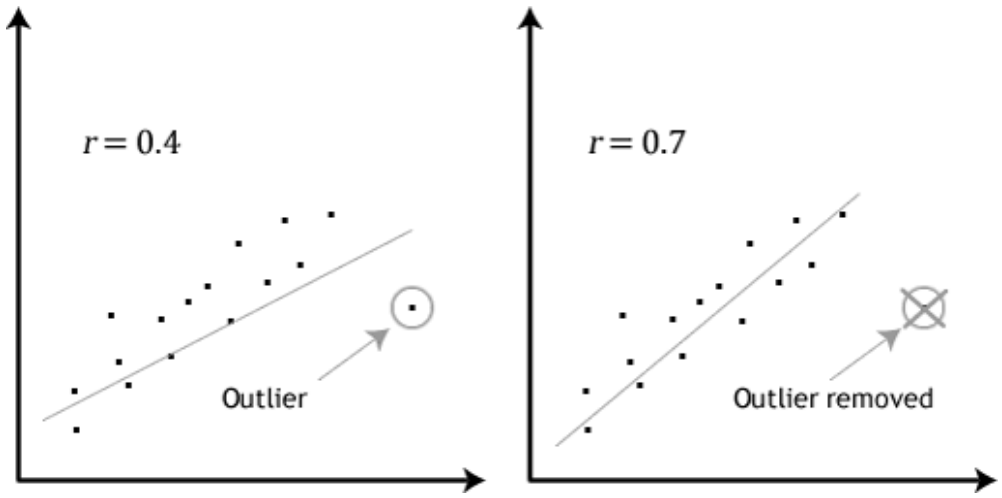
$$\boldsymbol{X} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 5 & 1 \\ 6 & 1 \\ 7 & 1 \\ 8 & 1 \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \begin{bmatrix} 0.33 \\ -1.33 \end{bmatrix}$$

Test data point $\boldsymbol{x} = \begin{bmatrix} x \\ 1 \end{bmatrix}$ ⇨ $\hat{y} = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x}) = \text{sign}(0.33x - 1.33)$

Outliers

- An outlier is a measurement/observation that is distant from other observations
  - Could be due to measurement error or "heavy-tailed distribution" events
  - In other words, experimental anomalies
- In some machine learning problems we're very interested in such outliers (e.g., anomaly detection)
  - But in linear regression, they can be problematic – linear regression is sensitive to outliers



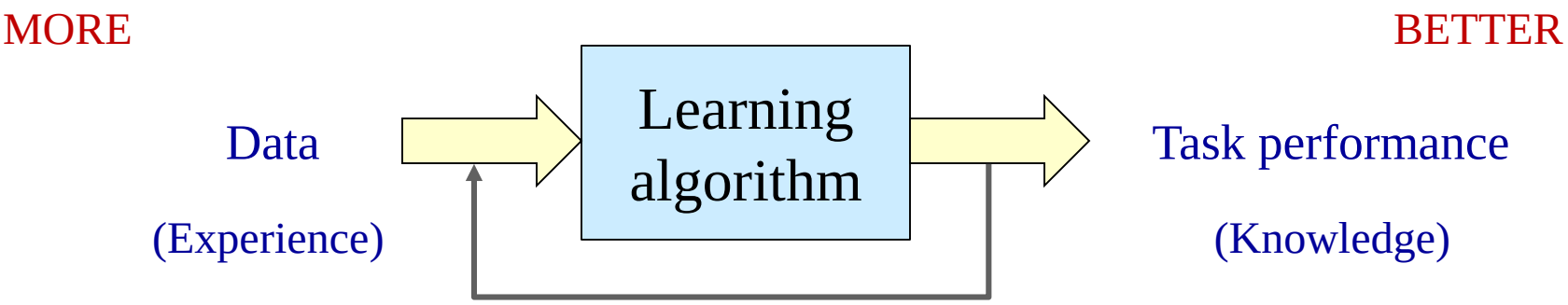
There is a lot of research on reducing sensitivity to outliers.

E.g., robust loss functions, probabilistic modeling methods such as RANSAC

A little review for the midterm

Introduction to machine learning

Machine learning is *the design and analysis of algorithms that improve their performance at some task with experience*



The learning may take place offline (before the execution of the task) – e.g., face detection, spam detection

It may take place online (as a task progresses) – e.g., an adaptive interface

It may do both – e.g., speech recognition systems

What is a learning problem?

- Learning involves improving performance
  - at some task ***T***
  - with experience ***E***
  - evaluated in terms of performance measure ***P***

- Example: learn to play checkers
  - Task ***T***: playing checkers well
  - Experience ***E***: playing against itself
  - Performance ***P***: percent of games won against humans

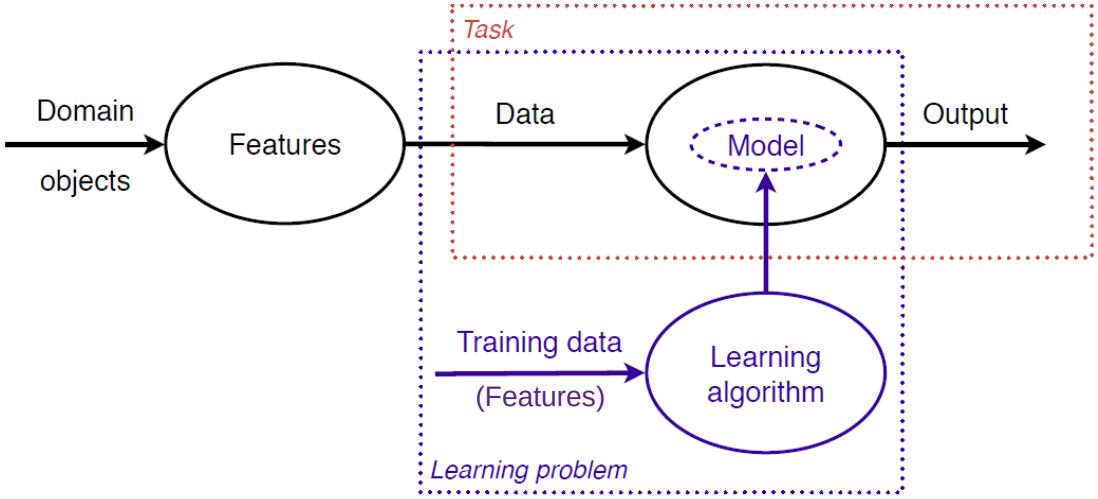
- What exactly should be learned?
  - How might this be represented?
  - What specific algorithm(s) should be used?

Components of a learning problem

- Task: the behavior or task that's being improved, e.g., classification, object recognition, acting in an environment
- Data: the experiences that are being used to improve performance in the task
- Measure of performance: How can the improvement be measured? Examples:
  - Provide more accurate solutions (e.g., increasing the accuracy in prediction)
  - Cover a wider range of problems
  - Obtain answers more economically (e.g., improved speed)
  - Simplify codified knowledge
  - New skills that were not presented initially

Machine learning

Machine learning is about using the right features to build the right models that achieve the right tasks



Features – how we describe our data objects

Model – a mapping from data points to outputs:

Output = f (Data)

This is what machine learning produces!

Task – an abstract representation of the problem we want to solve

Some machine learning problems

Training data includes desired outputs (input/output pairs)

Training data does not include desired outputs

Outputs	Supervised learning	Unsupervised learning
Discrete	Classification / labeling Binary / multi-class	Clustering
Continuous	Regression Real-valued prediction	Dimensionality reduction

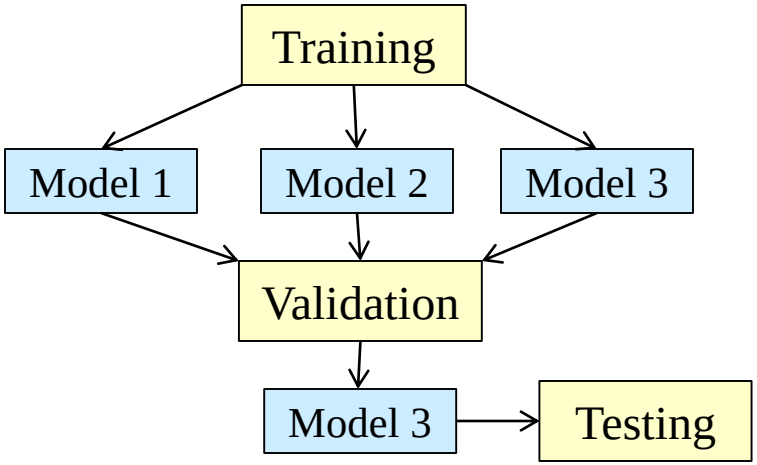
Also semi-supervised learning, reinforcement learning, etc.

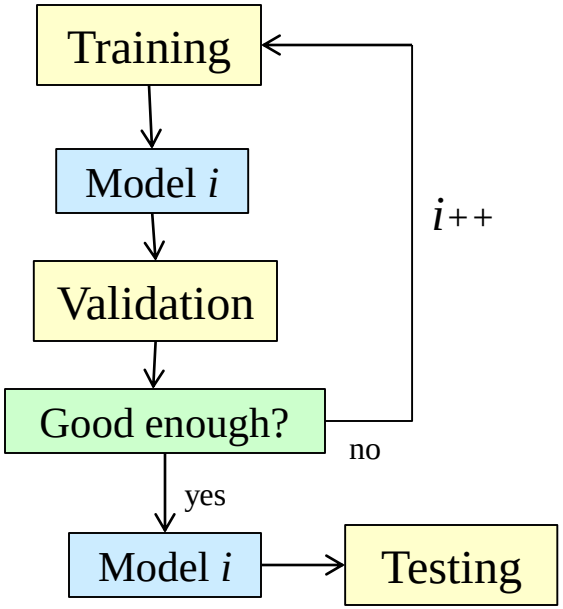
Training data includes *some* desired outputs

Learning through trial-and-error interactions with the environment

Training, validation, and test datasets

- We typically divide the dataset into three subsets:
  - Training data is used for learning the parameters of the models
  - Validation data is used to decide which model to employ
  - Test data is used to get a final, unbiased estimate of how well the model works





Sometimes reduced to training and testing a single mode (no validation step)

Homogeneous coordinates

- Instead of writing $\boldsymbol{x} \cdot \boldsymbol{w} > t$, let's use homogeneous coordinates to simplify the decision rule to $\boldsymbol{x}^{\circ} \cdot \boldsymbol{w}^{\circ} > 0$

$$\boldsymbol{x}^{\circ} = \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

$$\boldsymbol{w}^{\circ} = \begin{bmatrix} \boldsymbol{w} \\ -t \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ -t \end{bmatrix}$$

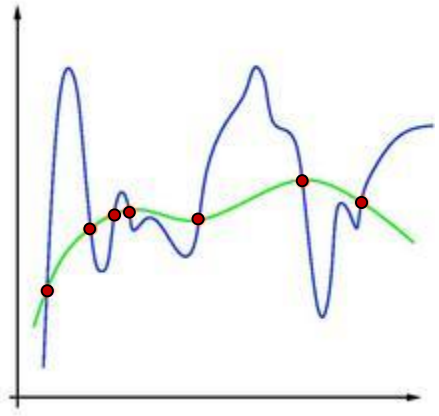
Note: I represent homogeneous coordinates a little differently from the textbook!

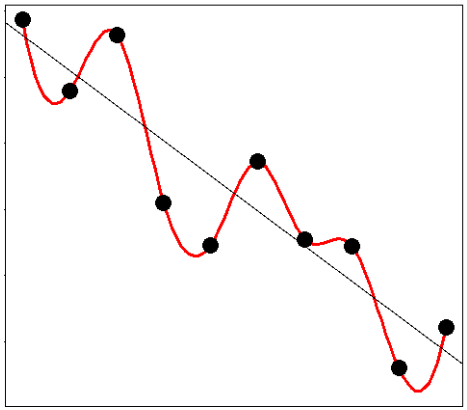
- Homogeneous coordinates embeds an N-dimensional representation in an N+1-dimensional space
- Advantage: The decision boundary passes through the origin of the extended coordinate system
  - Simplifies the math…

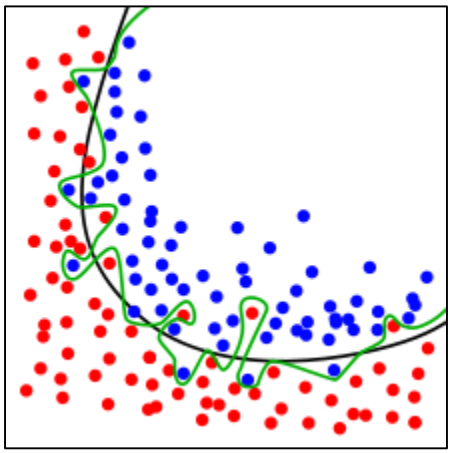
Overfitting

- *Overfitting* and *generalization* are important concepts in machine learning

- Overfitting: Learning that results in good performance on the training data but poor performance on the real task
  - Example: Memorization or lookup table
  - Example: Fitting a model to the data that has more parameters then needed







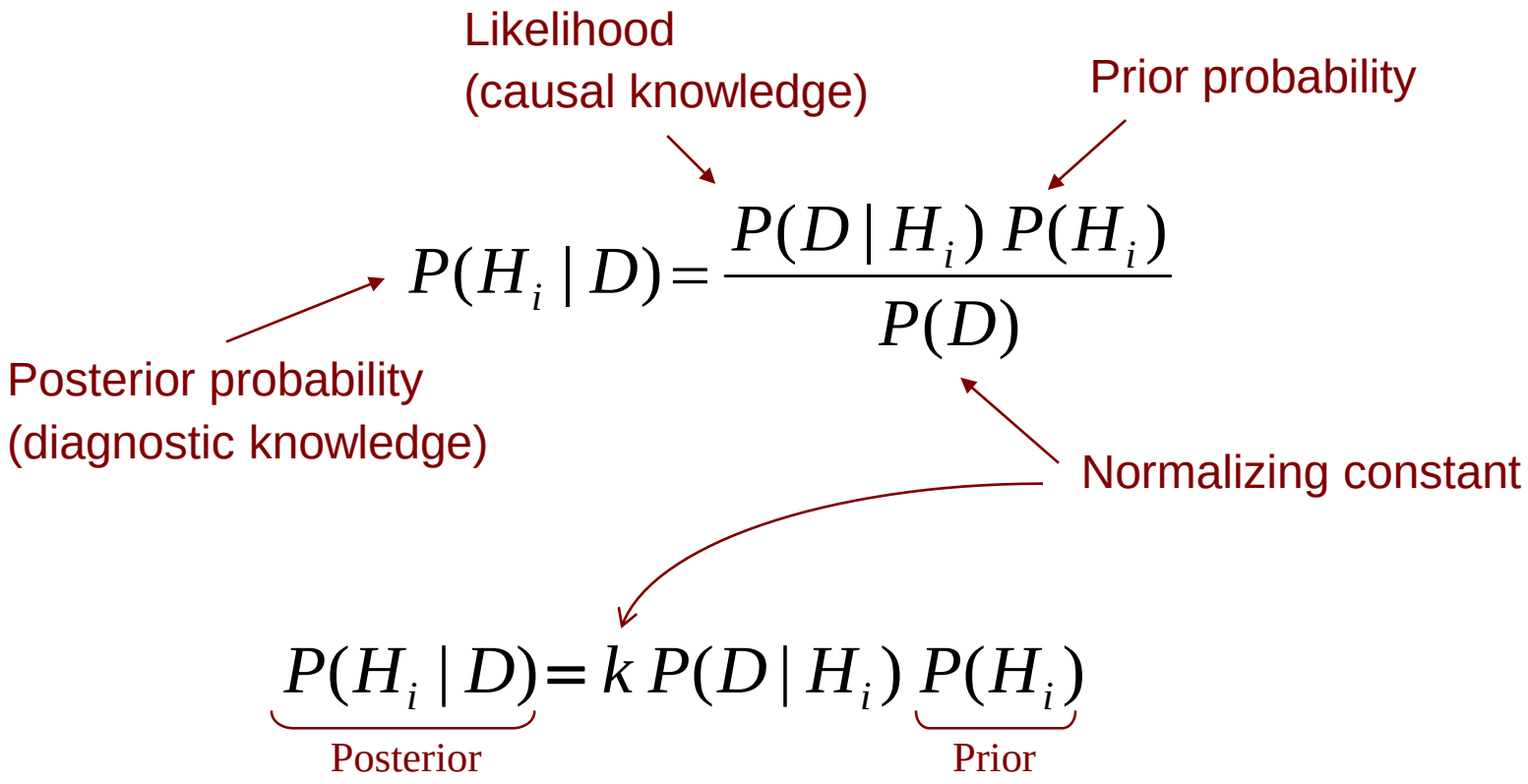
Generalization

- We want machine learning solutions that ***generalize*** to the range of inputs/data that will be seen – not just solutions that work well on the training data
- The real aim of machine learning is to do well on test data that is not known during learning
- Choosing values for the parameters that minimize the error on the training data is not necessarily the best policy.

- We want the learning machine to model the true regularities in the data and to ignore the noise in the data
  - But the learning machine does not know which regularities are real and which are accidental quirks of the particular set of training examples we happen to have! So we have to help….

Bayes' Rule

- This simple equation is very useful in practice
  - Usually framed in terms of hypotheses (H) and data (D)
    - Which of the hypotheses is best supported by the data?



Probabilistic models

- In general, probabilistic models aim to model the relationship between the feature values **X** and the target variables **Y** using probability distributions
- Predict **Y** based on **X** and the *posterior distribution* **P(Y | X)**
- Using Bayes' Rule

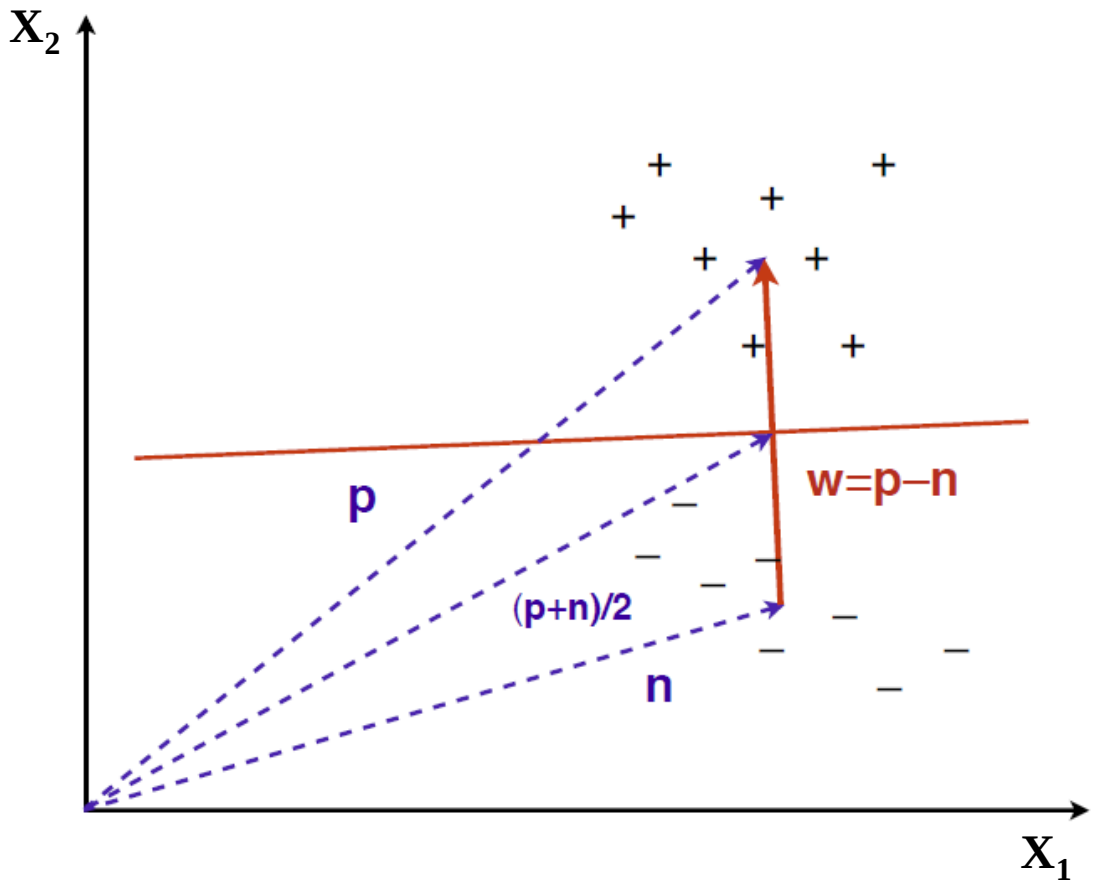
Likelihood, Prior, Posterior

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- Decision rule: Choose **Y** that maximizes the value of **P(Y | X)**
  - Known as the *maximum a posteriori (MAP)* rule, or *MAP estimation*
- Decision rule: Choose **Y** that maximizes the value of **P(X | Y)**
  - Known as the *maximum likelihood (ML)* rule, or *maximum likelihood estimation*

Basic linear classifier

Constructs a linear decision boundary halfway between the positive and negative centers of mass of the two classes



$$f(x) = 1 \;\; if \;\; \boldsymbol{x} \cdot \boldsymbol{w} > t$$
$$0 \;\; otherwise$$

How to compute t ?

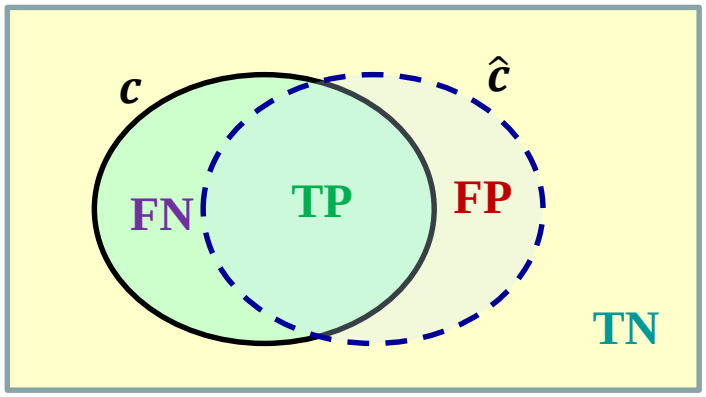
Dimensionality reduction

- Let's say we build a ML system to predict someone's occupation. The 12 features given to us are:
  - First name, last name, SSN, father's occupation, mother's occupation, highest educational degree, last year's salary, federal taxes paid last year, home address, model of car, miles driven last year, miles flown last year
- Some of these features may be useless, with no relevant information
- There may be redundancies – correlations among features
- Can we transform this 12-dimensional classification problem to a lower-dimensional problem?
  - Perhaps easier, computationally simpler…
- Yes, through dimensionality reduction

TP – true positives
FP – false positives
TN – true negatives
FN – false negatives

Correct
Errors

Instance space



Actual class
c

Predicted class $\hat{c}$	1	0	
1	TP	FP Type I Error	*Estimated positive* $\hat{P}$
0	FN Type II Error	TN	*Estimated negative* $\hat{N}$
	Positives P	*Negatives* N	TOTAL

Key terminology

False positive rate (FPR) = $\frac{FP}{N}$ = α

False negative rate (FNR) = $\frac{FN}{P}$ = β

True positive rate (TPR) = $\frac{TP}{P}$ = Sensitivity = Recall = 1 - β

True negative rate (TNR) = $\frac{TN}{N}$ = Specificity = 1 - α

Accuracy = $\frac{TP+TN}{P+N} = \left(\frac{P}{P+N}\right) TPR + \left(\frac{N}{P+N}\right) TNR$

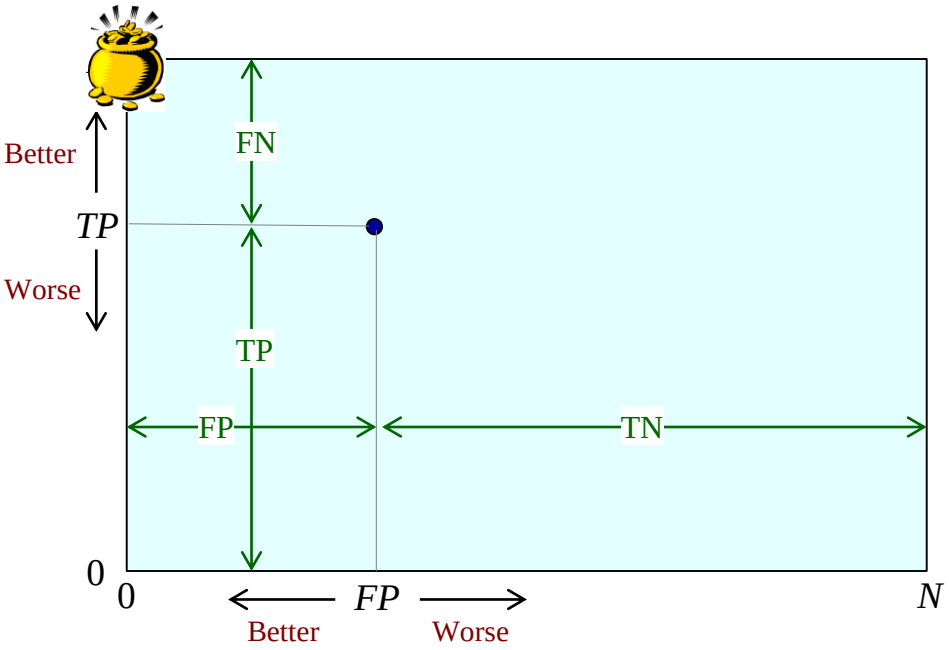
Error rate = $\frac{FP+FN}{P+N}$

Precision = $\frac{TP}{\hat{P}}$

Predicted class $\hat{c}$ \ Actual class c	1	0	
1	TP	FP	*Estimated positive* $\hat{P}$
0	FN	TN	*Estimated negative* $\hat{N}$
	Positives P	*Negatives* N	TOTAL

Coverage plot

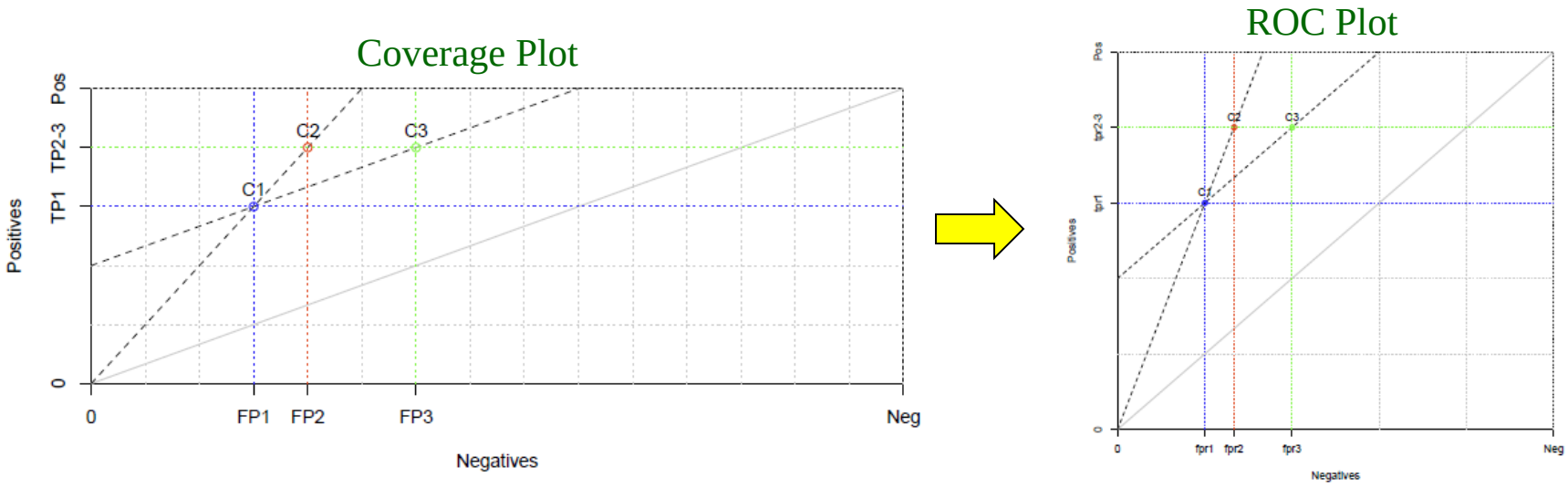
- It's very important to understand contingency tables and the values derived from them (false positive rate, accuracy, error rate, precision, etc.)
- The coverage plot provides a way to visualize classifier performance: { P, N, TP, FP }
  - A contingency table becomes a single point in a coverage plot



$\hat{c}$ \ c	1	0	
1	TP	FP	*Estimated positive* $\hat{P}$
0	FN	TN	*Estimated negative* $\hat{N}$
	Positives P	*Negatives* N	TOTAL

ROC plot

- If we normalize the coverage plot to a square, with each axis ranging from 0 to 1, we can plot TPR and FPR (instead of TP and FP)
- This gives us an ROC plot
  - ROC – "receiver operating characteristic"
    - Comes from signal detection theory
  - In an ROC plot, line segments with slope 1 connect classifiers with the same average recall



Classifier margin and loss function

- True class function $c(x) = \begin{cases} +1 \text{ for positive examples} \\ -1 \text{ for negative examples} \end{cases}$
- The scoring classifier assigns a margin $z(x)$ to each instance x:

$$z(x) = c(x)\hat{s}(x)$$

  - Positive if the estimate $\hat{s}(x)$ is correct
  - Negative if $\hat{s}(x)$ is incorrect
    - Since $\hat{s} > 0$ indicates positive estimate and $\hat{s} < 0$ negative
  - Large positive margins mean the classifier is "strongly correct"
  - Large negative margins are bad – they mean the classifier screwed up!
- In learning a classifier, we'd like to reward large *positive* margins and penalize large *negative* margins by the use of a loss function **L(z)** that maps the margin to an associated loss

$$\mathrm{L}: \mathbb{R} \rightarrow [0, \infty)$$

Ranking classifier

- The scores from a scoring classifier may not be particularly meaningful – they are not derived from any "true" scores – so it may be preferable to ignore the magnitude and just keep the **order** of the scores on a set of instances
  - This is less sensitive to outliers – i.e., more robust to noise/errors
- All positive examples should (ideally) be ranked higher than all negative examples
  - Exceptions to this are ranking errors
  - Count the ranking errors (*err*): For all (pos, neg) example pairs, how many rank neg higher than pos?
    - Ties count ½

Ranking error rate: $\textit{rank-err} = {}^{err}/_{PN}$

Ranking accuracy: $\textit{rank-acc} = 1 - \textit{rank-err}$

Empirical probabilities

- In machine learning, we often calculate *empirical probabilities* – i.e., calculate relative frequencies from the available data

  N_i instances of k classes C_i in the training data S:

  $$\text{Relative frequency} = \frac{N_i}{|S|} = \hat{p}_i$$

- But this can be problematic, especially with small amounts of training data
  - Probabilities of 0 and 1 generally should be avoided
- There are various common ways to smooth or correct the relative frequencies to avoid 0 and 1
  - E.g., Laplace correction and m-estimate:

Add a pseudo-count to each class

$$\text{Laplace correction} = \frac{N_i+1}{|S|+k}$$

Choose number of pseudo-counts m and their class distribution π_i

$$\text{m-estimate} = \frac{N_i+m\pi_i}{|S|+m} \qquad \sum_i \pi_i = 1$$

Regression – another predictive ML task

- In the classification tasks we've been discussing, the label space was a discrete set of classes
  - Classification, scoring, ranking, probability estimation
- Regression learns a function (the regressor) that is a mapping $\hat{f} : \mathcal{X} \rightarrow \mathbb{R}$ from examples – $f(x_i)$
  - I.e., the target variable (output) is real-valued
- Assumption: the examples will be noisy, so watch out for overfitting – need to capture the general trend or shape of the function, not exactly match every data point
  - E.g., if fitting an N-degree polynomial to the training data (thus N+1 parameters to estimate), choose one as low degree as possible
- The number of data points should be much greater than the number of parameters to be estimated!
  - How much data is needed? An open question in ML….

Concept learning

- In concept learning, we want to learn a Boolean function over a set of attributes+values
  - I.e., derive a Boolean function from training examples
    - Positive and negative examples of the concept
      - Positive: Temperature = high ∧ Coughing = yes ∧ Spots = yes
      - Negative: Temperature = medium ∧ Coughing = no ∧ Spots = yes
  - This is our hypothesis
- The target concept $\boldsymbol{c}$ is the true concept
  - We want the hypothesis to be a good estimate of the true concept
  - Thus we wish to find $\boldsymbol{h}$ (or $\hat{\boldsymbol{c}}$) such that $\boldsymbol{h} = \boldsymbol{c}$ (or $\hat{\boldsymbol{c}} = \boldsymbol{c}$)
- The hypothesis is a Boolean function over the features
  - E.g., some combinations of {Temperature, Coughing, Spots} are <u>in</u> the concept, and others are <u>not in</u> the concept

The hypothesis space

- Using a set of features, what concepts can possibly be learned?
- The space of all possible concepts is called the hypothesis space
  - What is the hypothesis space for a given problem?
- First, how many possible instances are there for a given set of features?
  - In set theory, the Cartesian product of all the features
  - $F_1 \times F_2 \times \ldots \times F_N$
  - All combinations of feature values
  - UCSB courses: Quarter (4), Dept (40), courselevel (2), topic (500)
  - 4×40×2×500 = 160,000 possible instances
    - E.g., (spring, CS, ugrad, ML), (fall, Music, grad, StringTheory), …

The conjunctive hypothesis space

- Let's limit our hypothesis space to conjunctive concepts – i.e., hypotheses that can be expressed as a conjunction of literals (features)

  Quarter=? ∧ Dept=? ∧ courselevel=? ∧ topic=?

- We add "absence" or "don't care" to each feature, so now the total number of combinations is 5×41×3×501 = 308,115
  - That's a lot, but much better than $2^{160,000}$! (between 2^{18} and 2^{19})
- The most general hypothesis is (X, X, X, X), which includes all possible instances
  - (fall, X, X, X) is the concept of all fall quarter courses
  - (fall, CS, grad, X) is the concept of all CS graduate courses in the fall
- In this conjunctive hypothesis space, we can't represent concepts like "all courses in AI or Graphics"

Complete and consistent concepts

"Cover" – classify as positive

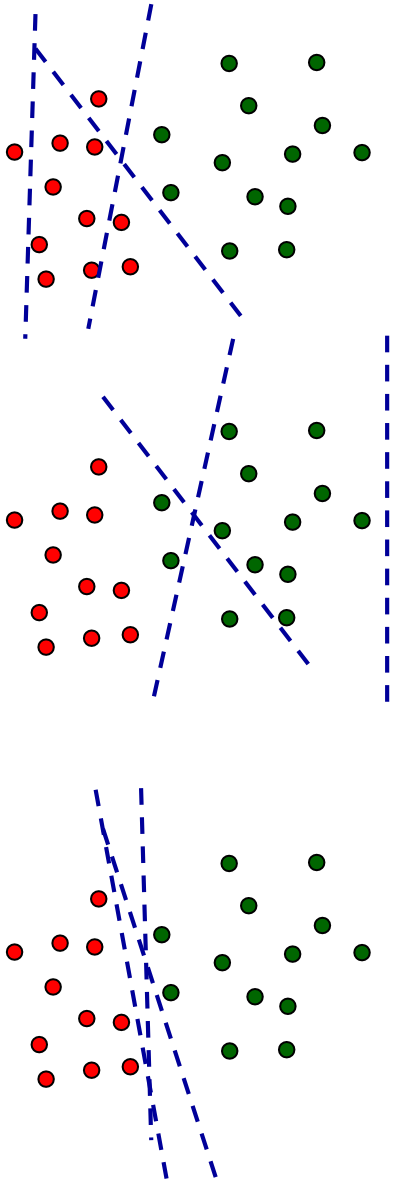
A concept is **complete** if it covers all positive examples

- FNR = 0

A concept is **consistent** if it covers none of the negative examples

- FPR = 0

The **version space** is the set of all concepts that are both complete and consistent



Decision trees

Note: This is very similar to ID3, a well-known DT algorithm

- Tree models can be used for classification, ranking, probability estimation, regression, and clustering
- Recursive generic tree learning procedure:

Algorithm GrowTree(D, F) – grow a feature tree from training data.

Input : data D; set of features F.
Output : feature tree T with labelled leaves.
if Homogeneous(D) **then return** Label(D);
$S \leftarrow$BestSplit(D, F) ; // e.g., BestSplit-Class (Algorithm 5.2)
split D into subsets D_i according to the literals in S;
for each i **do**
  if $D_i \neq \emptyset$ **then** $T_i \leftarrow$GrowTree(D_i, F) ;
  else T_i is a leaf labelled with Label(D);
end
return a tree whose root is labelled with S and whose children are T_i

Most useful feature

Divide-and-conquer approach: build a tree for each subset of the data, then merge into a single tree

Impurity

- In the binary case, we have P positives and N negatives in the data
  - The best split would be a feature that divides the data D into two pure partitions: D_1 with the P positives and D_2 with the N negatives
- So a measure of partition impurity should be minimum when the data are 100% positives or negatives, and maximum when 50/50
- Like with empirical probabilities, we can estimate impurity by counting. We define the proportion of positives in D_i as:

$$\dot{p} = \frac{P}{P + N}$$

- Impurity is a function of $\dot{p}$
  - Should be zero when $\dot{p}$ = 0 or 1, and maximum when $\dot{p}$ = 0.5
  - We can write impurity as Imp(D) or Imp($\dot{p}$)

Linear least-squares regression

- Regression learns a function (the regressor) that is a mapping $\hat{f} : \mathcal{X} \rightarrow \mathbb{R}$; it's learned from examples, $(x_i, f(x_i))$
  - I.e., the target variable (output $\hat{f}(x)$) is real-valued
- Linear regression – the function is linear
  - Fit a line/plane/hyperplane to the data
- The difference between f and $\hat{f}$ is known as the residual ϵ

$$\epsilon_i = f(x_i) - \hat{f}(x_i)$$

- The least squares method minimizes the sum of the squared residuals – i.e., find $\hat{f}$ that minimizes $\sum_i {\epsilon_i}^2$ on the training data
- Univariate or multivariate regression
  - Univariate – one input variable
  - Multivariate – multiple input variables

Multivariate least-squares in matrix form

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon}$$

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} \quad \Leftarrow \quad \text{Least-squares solution } \hat{\boldsymbol{w}}$$

$$= \boldsymbol{S}^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$

Scatter matrix for $\boldsymbol{X}$

$$\boldsymbol{S} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$$

Note: Often $\boldsymbol{X}$ is written transposed from how it's defined here, so

$$\boldsymbol{y} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{w} + \boldsymbol{\epsilon}$$

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}})^{-1}\boldsymbol{X}\boldsymbol{y}$$

$$\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$$

Need to understand in context

Linear regression function

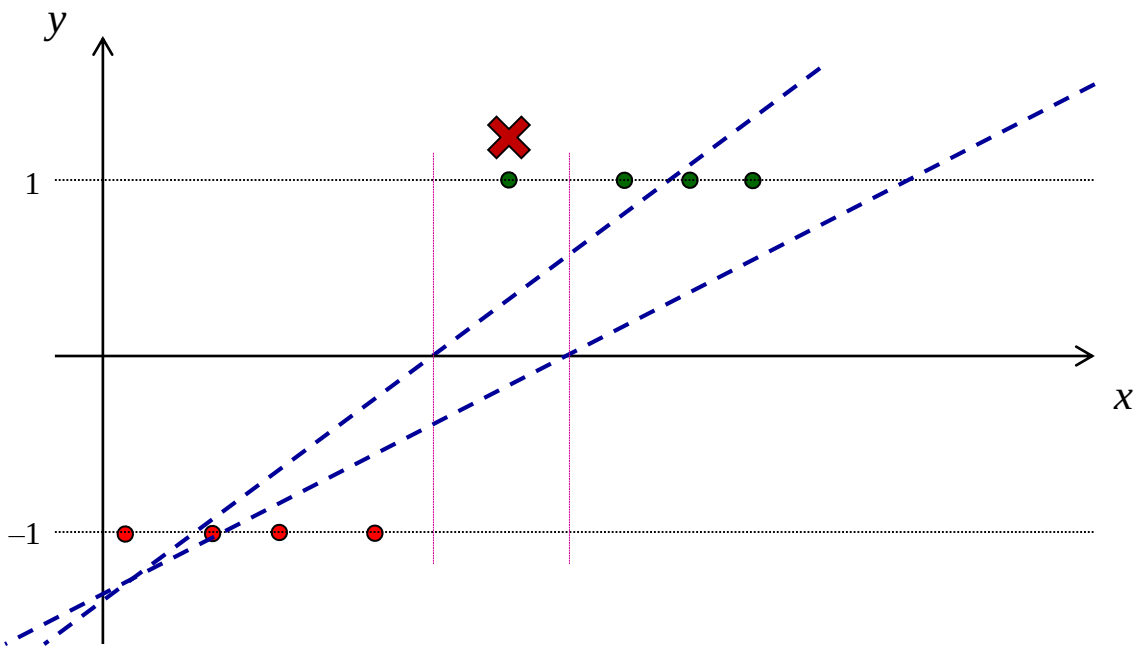
$$y(\boldsymbol{x}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{w}$$

Using homogeneous coordinates

Moving on…

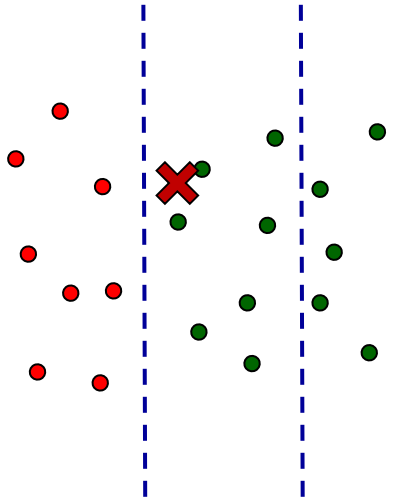
The perceptron

- A least squared classifier is not guaranteed to find a perfect decision boundary for linearly separable data



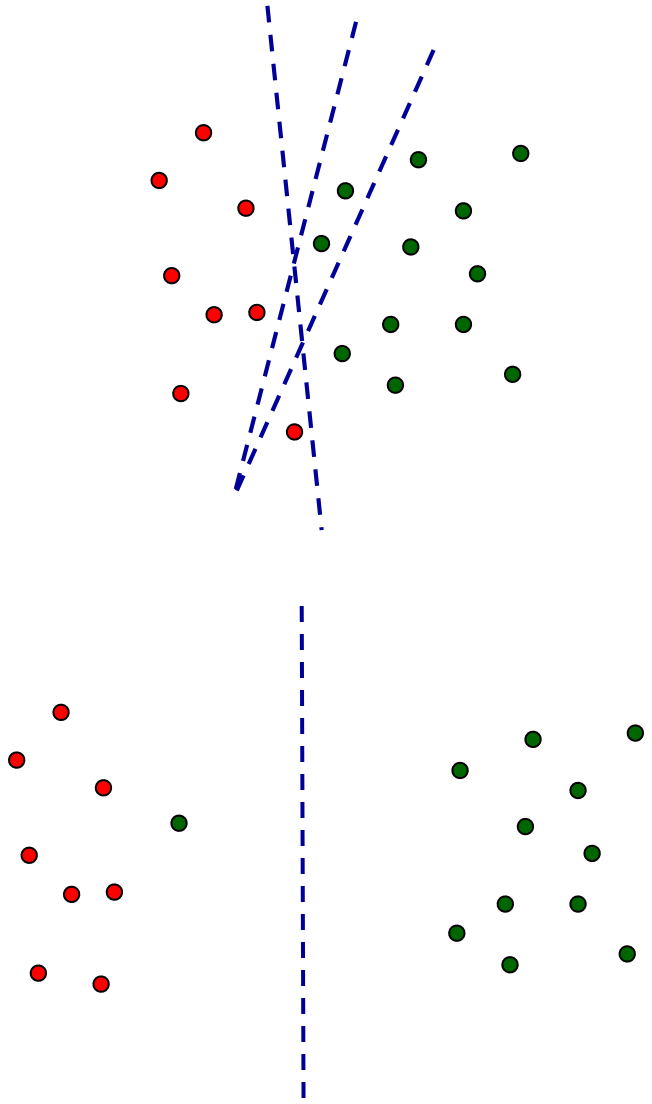
The perceptron

- A least squared classifier is not guaranteed to find a perfect decision boundary for linearly separable data



- The perceptron model is a linear classifier that will achieve perfect separation on linearly separable data
- A perceptron iterates over the training data, updating ***w*** every time it encounters an incorrectly classified example

The perceptron



How to move the boundary for a misclassified example?

How much to move it?

Update rule:

$$\boldsymbol{w}' = \boldsymbol{w} + \eta y_i \boldsymbol{x_i}$$

η is the learning rate

$0 < \eta \leq 1$

Iterate through the training examples (each pass over the data is called an epoch) until all examples are correctly classified

Guaranteed to converge if the training data is linearly separable – but won't converge otherwise