Machine Learning

CS 165B

Prof. Matthew Turk

Monday, April 11, 2016

Today

- Classification (cont.)
  Chapters 2-3

Notes

- HW#1 answers posted on GauchoSpace today
  - Can go over any questions at tomorrow's discussion sessions
  - Main point was to get you to think through several of the concepts important to machine learning
  - Also probability background
    - P(X, Y) vs. P(X | Y)
    - Marginalize P(X, Y) to get P(X)
- HW#2
  - Posted by this Friday, due following Friday (April 22)
- CS seminar
  - Today at 3:30pm, CS Conference Room (1132 HFH)
  - Sameer Singh, University of Washington
  - "Interactive Machine Learning for Information Extraction"

Classifier margin and loss function

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

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

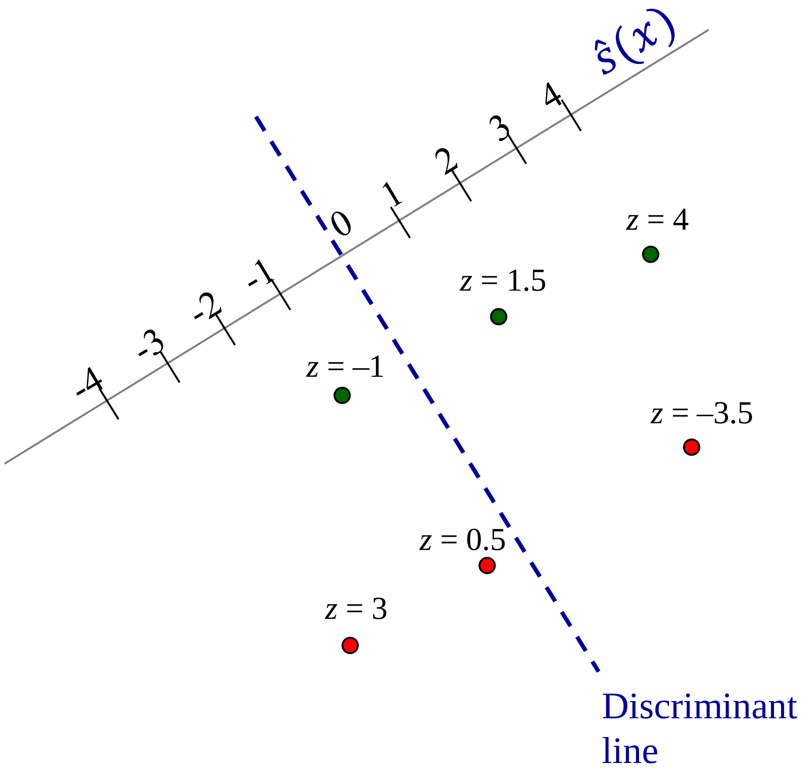
  - Positive if the estimate $\hat{s}(x)$ is correct
  - Negative if $\hat{s}(x)$ is incorrect
    - Since $\hat{s} > 0$ indicates positive estimate and $\hat{s} < 0$ negative
  - Large positive margins mean the classifier is "strongly correct"
  - Large negative margins are bad – they mean the classifier screwed up!

Classifier margin and loss function

Training data:

Positive class: $c = +1$

Negative class: $c = -1$



Score $\hat{s}(x)$

True class function $c(x)$

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

How should each training data point impact the classifier learned from this data?

The loss function L(z) will determine this

At this point in the iterative classifier training algorithm, which training data points are the most important?

Classifier margin and loss function

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

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

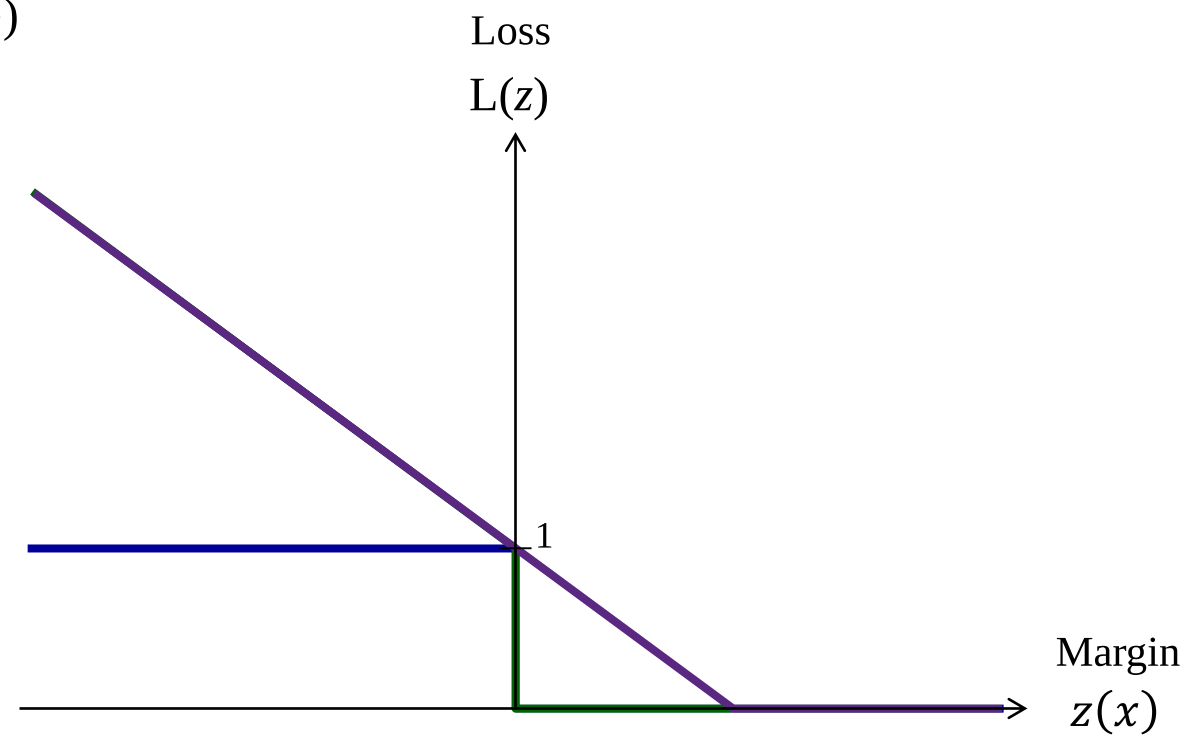
  - Positive if the estimate $\hat{s}(x)$ is correct
  - Negative if $\hat{s}(x)$ is incorrect
    - Since $\hat{s} > 0$ indicates positive estimate and $\hat{s} < 0$ negative
  - Large positive margins mean the classifier is “strongly correct”
  - Large negative margins are bad – they mean the classifier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

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

The loss function, L(z)

What should the loss function look like?

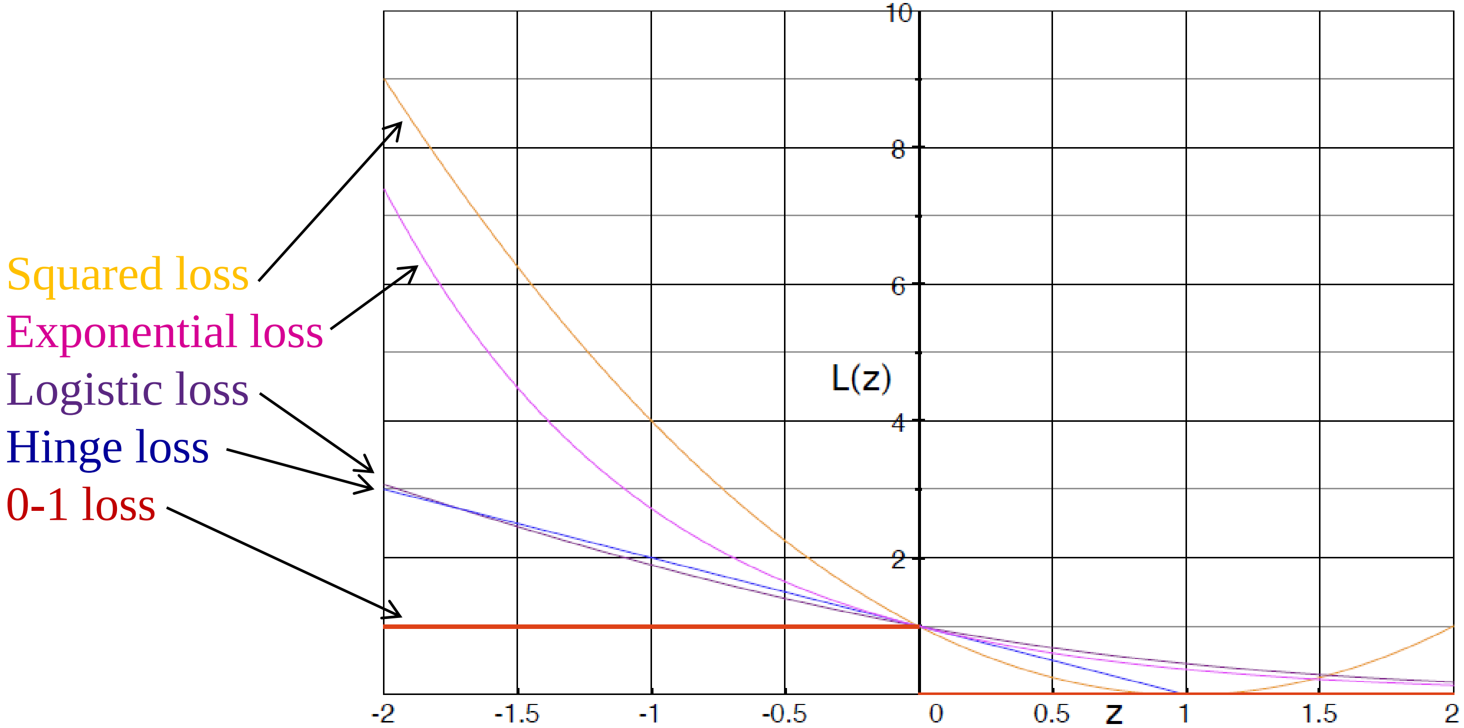
$$\mathrm{L} : \mathbb{R} \rightarrow [0, \infty)$$



Penalize wrong classifications more

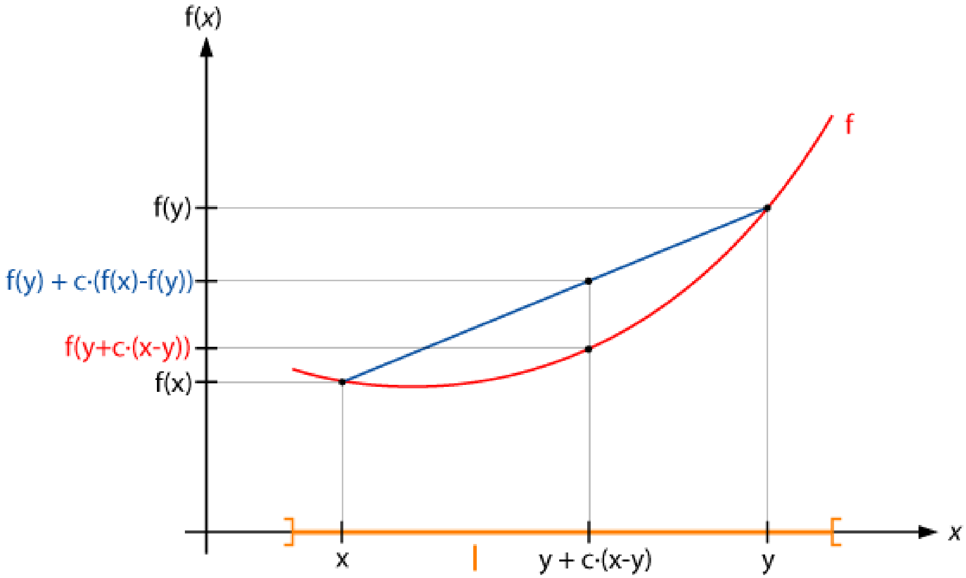
The loss function, L(z)

- Characteristics of the loss function:
  - For an example on the decision boundary, L(0) = 1
  - L(z) ≥ 1 for z < 0
  - 0 ≤ L(z) < 1 for z > 0



The loss function, L(*z*)

- Loss functions are often used in optimization problems (to minimize a function) that lead to modifying weights in training
  - Typically it is squared – thus the mapping to $[0, \infty)$
- To help make this solvable, the loss function is often chosen to be convex, since optimizing a convex function is computationally more tractable



A convex function lies below the line connecting any two points on the function

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: *rank-err* = ${}^{err}/_{PN}$

Ranking accuracy: *rank-acc* = 1 – *rank-err*

Ranking classifier performance

Score: Low  High

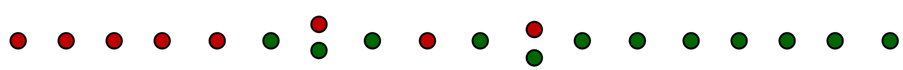
# of ranking errors? Ranking error rate? Ranking accuracy?

2 2/(5)(5) = 0.08 0.92

Low  High

# of ranking errors? Ranking error rate? Ranking accuracy?

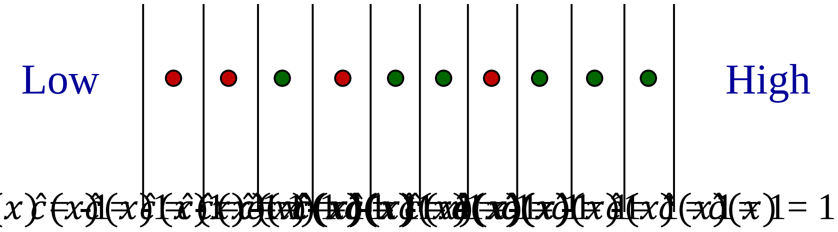
9 9/(8)(12) = 0.09375 0.90625

Low  High

# of ranking errors? Ranking error rate? Ranking accuracy?

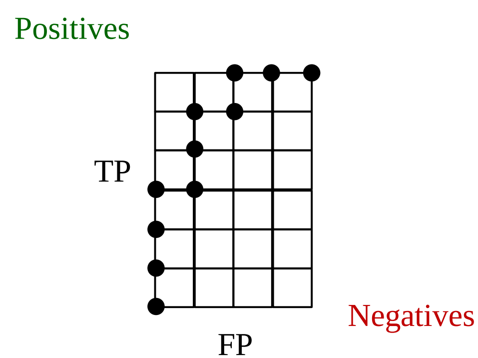
9 9/(8)(12) = 0.09375 0.90625

Ranking classifier and the coverage curve



Move the decision line and count:

FP = ?   TP = ?



This is the coverage curve

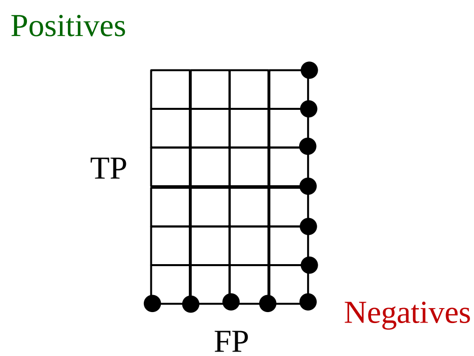
If we normalize to a square graph, we get the ROC curve

The area under the ROC curve is the ranking accuracy

Ranking classifier and the coverage curve

Low  High

What about this case? Looks like the ranking is terrible!



What is the ranking accuracy? Zero (0%)

Q: What would the ranking accuracy be of this ranking?

Low  High

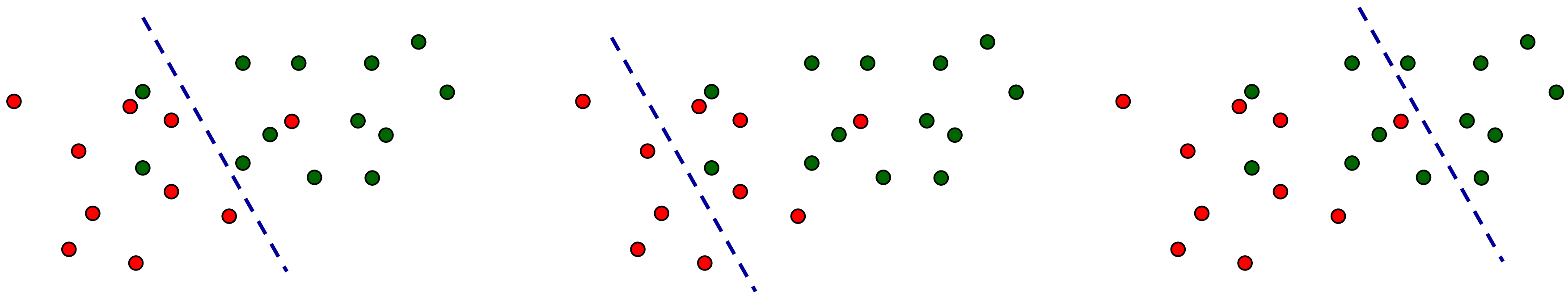
One (100%)

Classifier design – operating point

- You, as a classifier designer, can often move decision boundaries (modify thresholds) to make the false positive rate as high or as low as you wish
  - A very high threshold (don't let anything through!) results in no false positives – but lots of false negatives
  - A very low threshold (let everything through!) results in no false negatives – but lots of false positives
- This doesn't necessarily make the classifier better or worse – it just changes the operating point of the classifier
- This is often application-specific:
  - When might false positives be especially undesirable?
  - When might false negatives be especially undesirable?
  - We can encode these preferences in a cost function to compute an optimal threshold, given this information

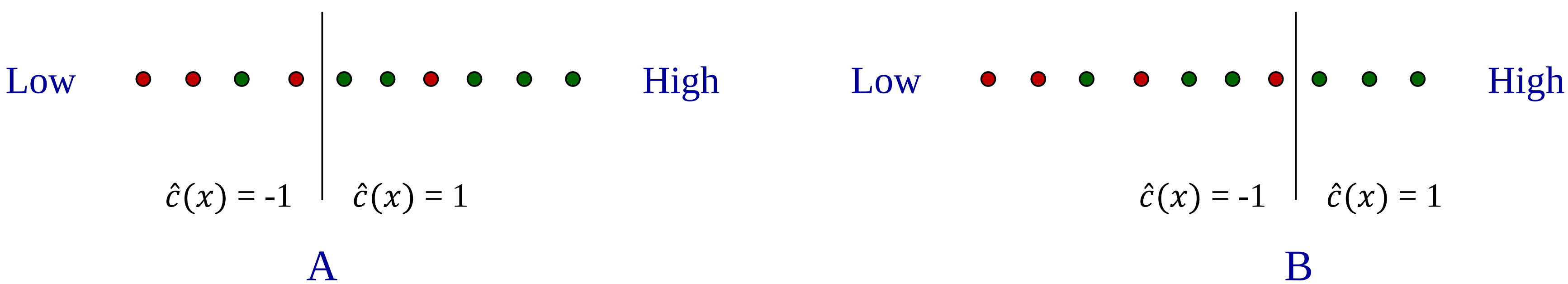
Classifier design

Which classifier is best: A, B, or C?



A B C

A or B?



<u>It depends</u> on what you want to optimize:

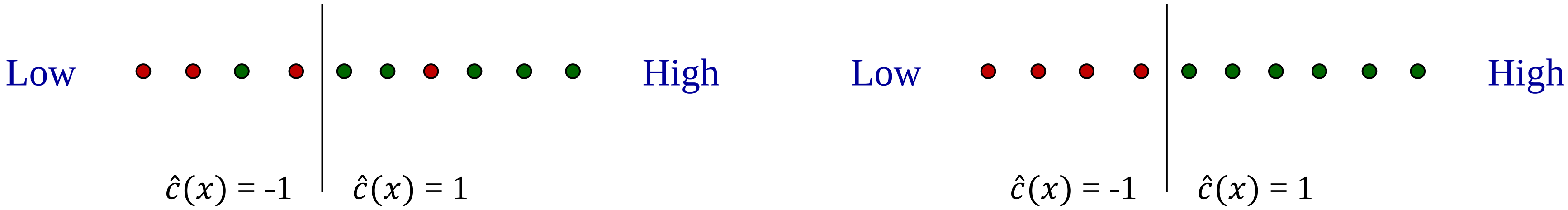
– TPR, FPR, error rate, accuracy, precision, …

Classifier design

A or B?

A B

A or B?



A B

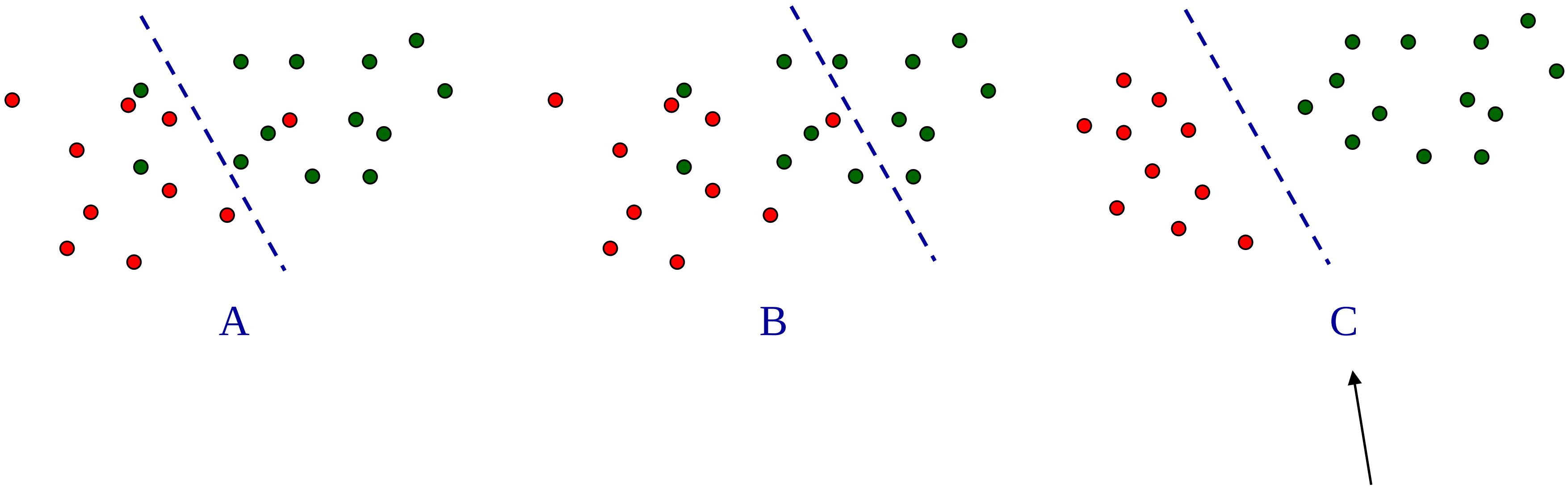
It also depends on how good your features are!

- Either *raw* features or *constructed* features

Feature separation vs. classifier design

Placing the separating boundary = classifier design

Increasing the feature separation = feature construction



We can design a better classifier starting with the features in C!

Class probability estimation

A class probability estimator is a scoring classifier that outputs probabilities over the k classes – i.e., a mapping:

$$\hat{p} : \mathcal{X} \rightarrow [0,1]^k$$

where

$$\sum_{i=1}^{k} \hat{p}_i(x) = 1$$

A key issue here is that we generally do not have access to the true probabilities for training data.

- E.g., an email is either spam or ham – it doesn't have a probability of being spam!
- So how can we train to learn such probabilities?

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:

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

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

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

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

Empirical probabilities

Training data set S

- C_1: 7 instances
- C_2: 14
- C_3: 0
- C_4: 4

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

$|S| = 25$

$\hat{p}_1$: 7/25 = 0.28
$\hat{p}_2$: 14/25 = 0.56
$\hat{p}_3$: 0/25 = 0.0
$\hat{p}_4$: 4/25 = 0.16

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

$\hat{p}_1$: 8/29 = 0.28
$\hat{p}_2$: 15/29 = 0.52
$\hat{p}_3$: 1/29 = 0.03
$\hat{p}_4$: 5/29 = 0.17

m-estimate = $\dfrac{N_i+m\pi_i}{|S|+m}$ $\quad \sum_i \pi_i = 1$

$m = 40$: {10, 10, 10, 10}

$\hat{p}_1$: 17/65 = 0.26
$\hat{p}_2$: 24/65 = 0.37
$\hat{p}_3$: 10/65 = 0.15
$\hat{p}_4$: 14/65 = 0.22

$m = 40$: {10, 0, 18, 12}

$\hat{p}_1$: 17/65 = 0.26
$\hat{p}_2$: 14/65 = 0.22
$\hat{p}_3$: 18/65 = 0.28
$\hat{p}_4$: 16/65 = 0.25

Multi-class classification

- Many classification problems involve multiple classes
- Performance can be described with the multi-class contingency table
  - Not including the marginals, also known as the **confusion matrix**
  - We can compute accuracy, per-class precision, per-class recall…

	Predicted			
	15	2	3	20
Actual	7	15	8	30
	2	3	45	50
	24	20	56	100

Accuracy = (15+15+45)/100 = 0.75

Class 1 precision = 15/24 = 0.63

Class 1 recall = 15/20 = 0.75

Etc.