STAT 215A Fall 2022 Week 12

Theo Saarinen

Thanks to Tiffany Tang and past GSIs for sharing their slides

Outline for today

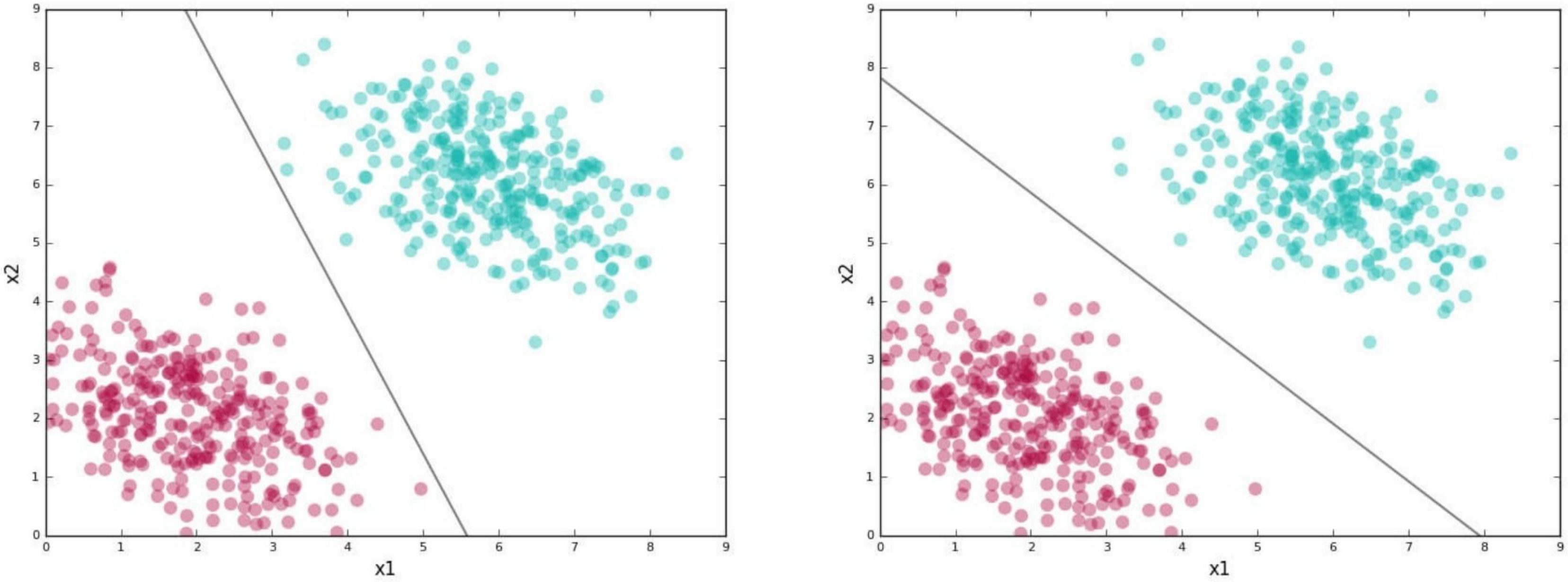
- More classification algorithms
  - SVM
  - Random forest
- Evaluation of classification performance

Support vector machines

- Intuition: https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93
- More in-depth discussion of the math:
  - https://towardsdatascience.com/understanding-support-vector-machine-part-1-lagrange-multipliers-5c24a52ffc5e
  - https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d
- Elements of Statistical Learning
  - Section 4.5 and Chapter 12
- We'll focus on intuition

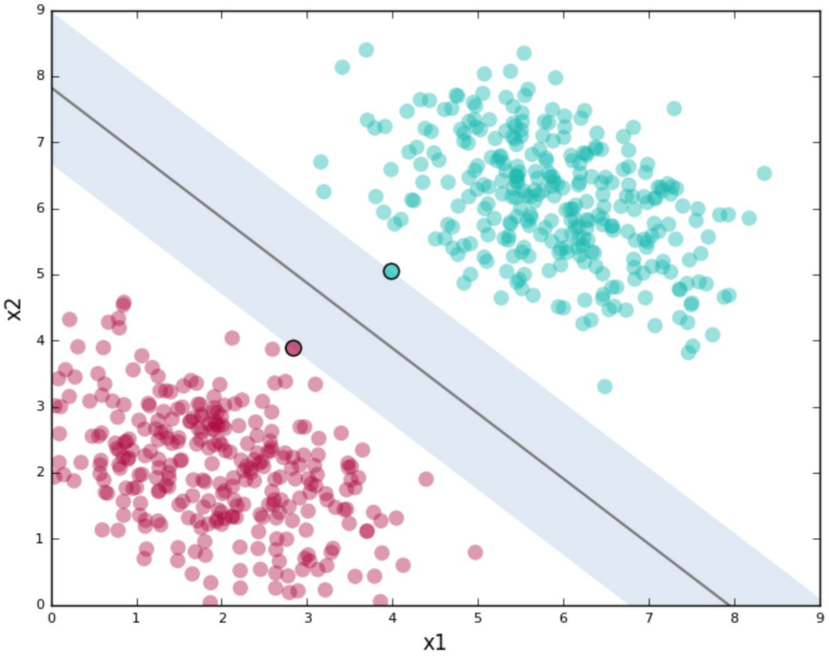
Support vector machines (SVM)

Which is better?



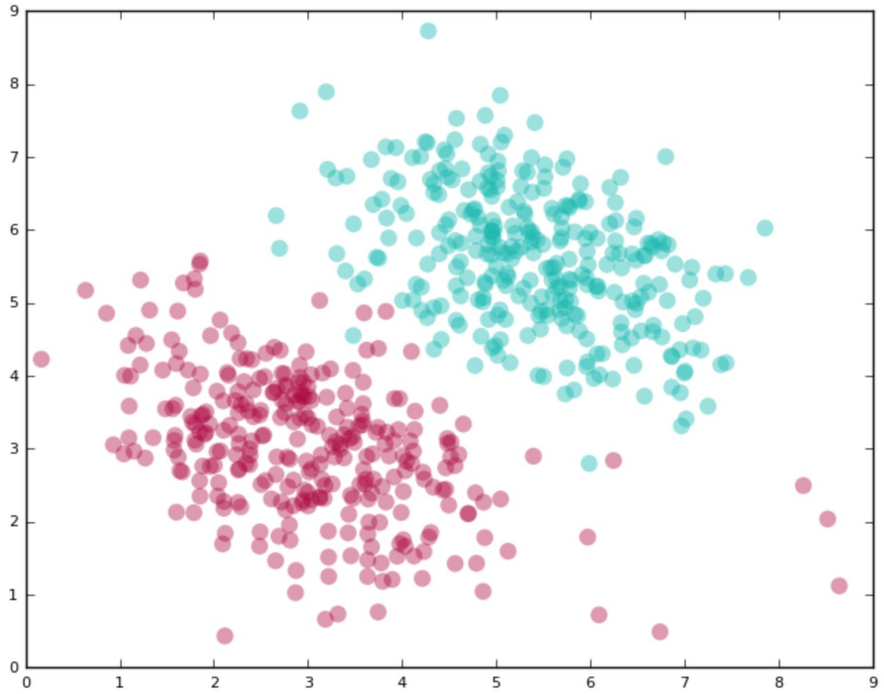
Support vector machines (SVM)

An idea: maximize space between two hyperplanes that separate the classes
“Maximum margin” classifier



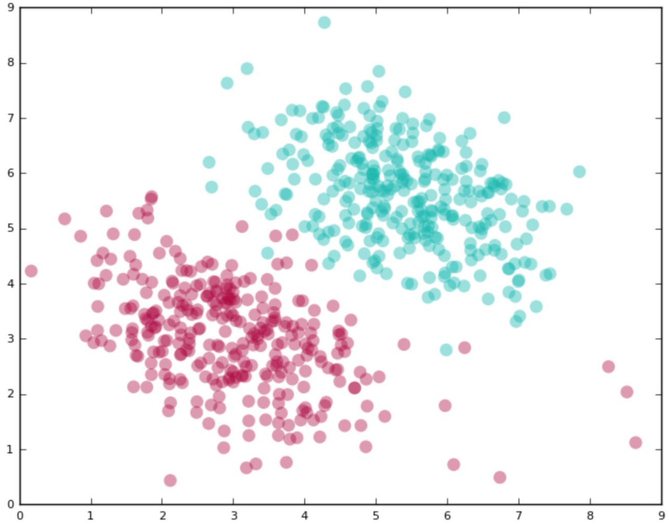
Support vector machines (SVM)

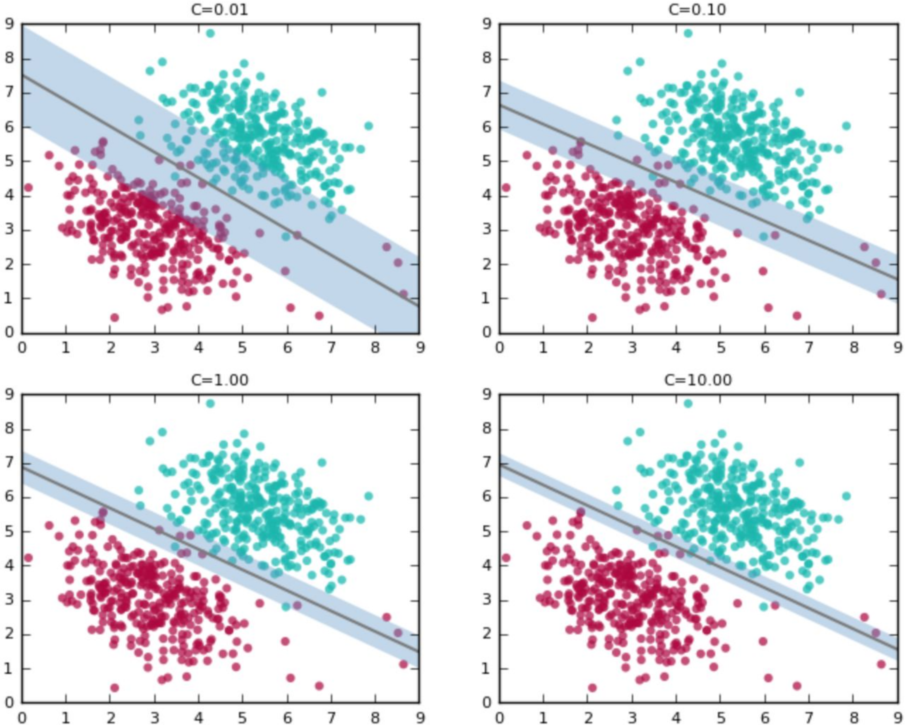
What about when the two classes are overlapping?



Support vector machines (SVM)

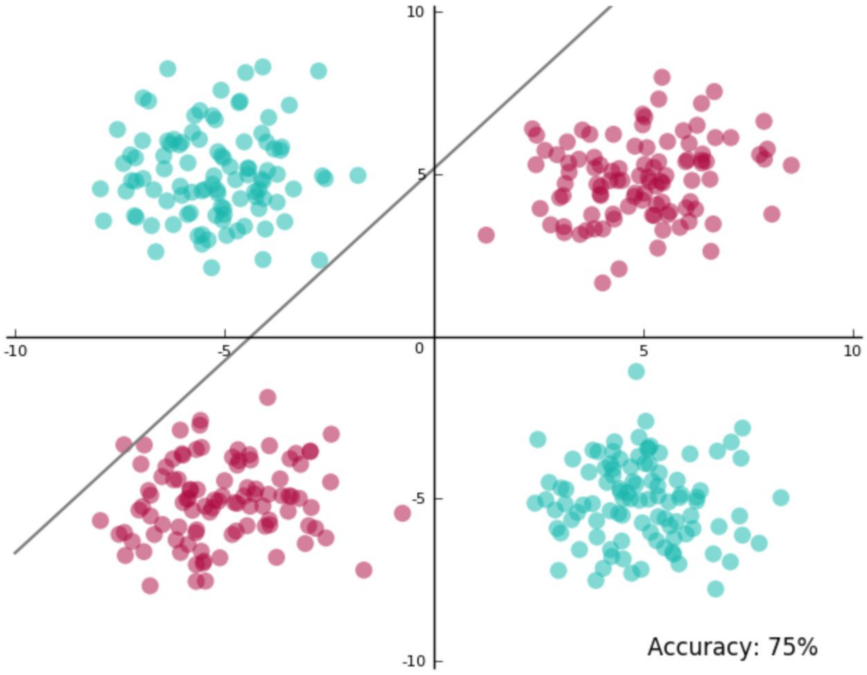
Another idea: Allow for some “slack”





Support vector machines (SVM)

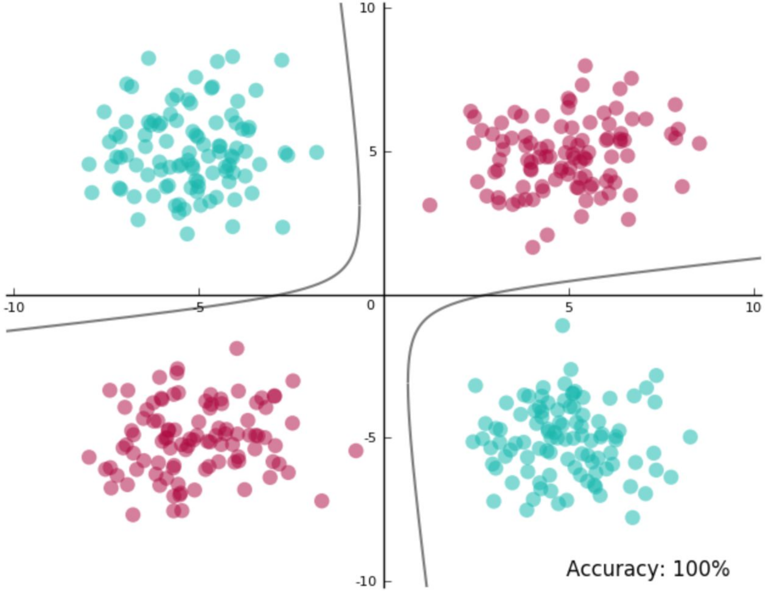
What if there is no good separating hyperplane?

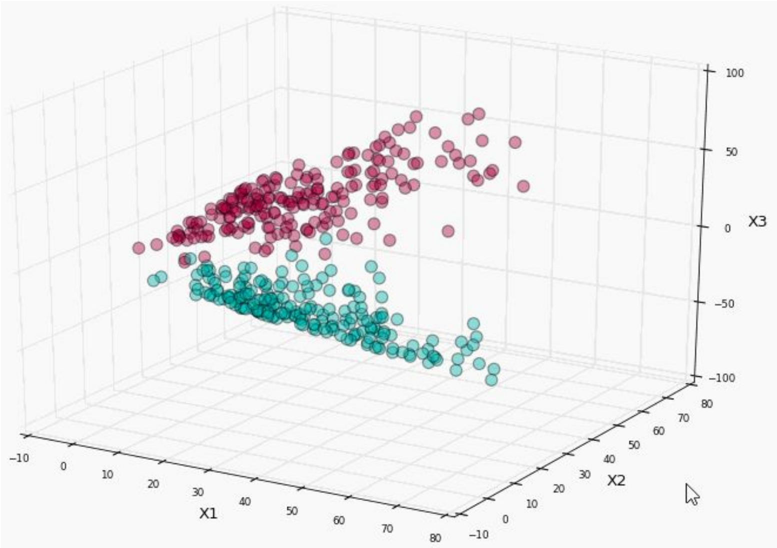


Support vector machines (SVM)

Idea: Find a higher-dimensional representation of the data where it becomes linearly separable

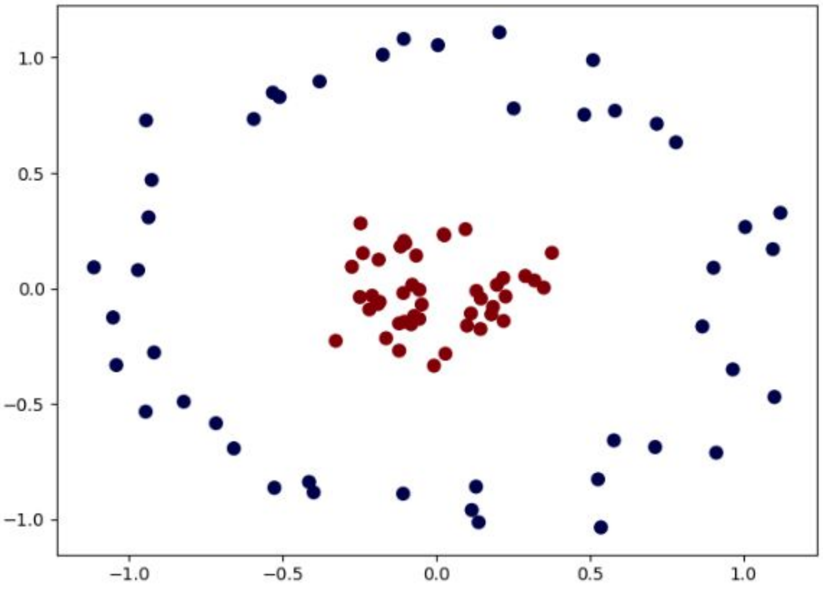
$$X_1 = x_1^2$$
$$X_2 = x_2^2$$
$$X_3 = \sqrt{2}x_1x_2$$

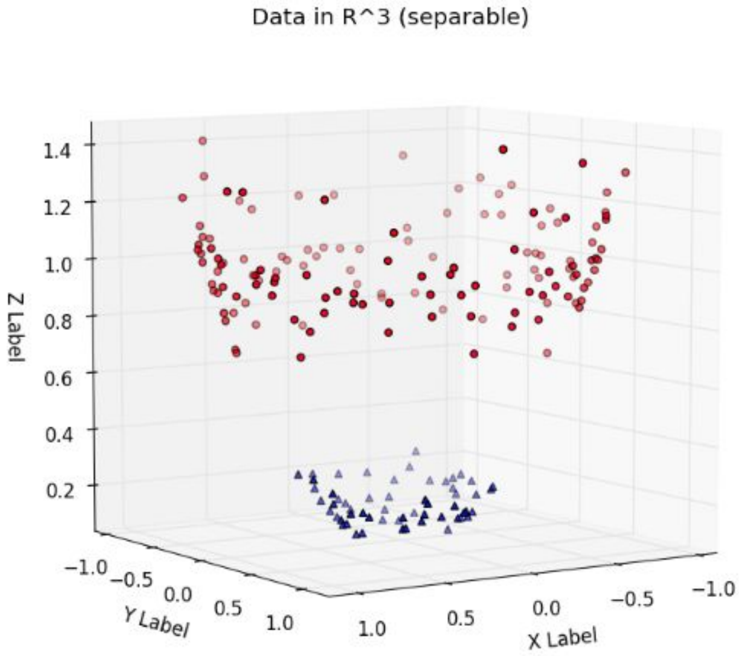




Support vector machines (SVM)

Another example of a higher dimensional representation that is linearly separable





Support vector machines (SVM)

So how do we perform this “lifting” to higher dimensions trick in a computationally feasible way? The answer: the **kernel trick**.

- Can show that by maximizing the margins while allows for slack, SVM solves the following maximization problem:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \overbrace{x_i^T x_k}^{\text{Inner product between two of the data points}}$$

$$\text{subject to } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0$$

Kernel trick

Why not replace the usual inner product $x_i^\top x_k$ with

$$\varphi(x_i)^\top \varphi(x_k)$$

where φ is some map from $\mathbb{R}^p$ to a higher-dimensional space (possible even infinite dimensional).

- The trick: don't need to know what φ actually is.
  - Good news: we don't have to compute an infinite dimensional map.
- Instead, we find the kernel function:

$$K(x_i, x_k) = \varphi(x_i)^\top \varphi(x_k)$$

Kernel trick

Some common kernel functions:

- Linear kernel: $K(x_i, x_k) = x_i^\top x_k$
- Naive polynomial kernel: $K(x_i, x_k) = (x_i^\top x_k)^d$
- Polynomial kernel: $K(x_i, x_k) = (1 + x_i^\top x_k)^d$
- Gaussian kernel: $K(x_i, x_k) = \exp\left\{-\frac{1}{2}\|x_i - x_k\|_2^2\right\}$
- Radial basis kernel: $K(x_i, x_k) = \exp\{-\gamma\|x_i - x_k\|_2^2\}$
- Sigmoid kernel: $K(x_i, x_k) = \tanh(\eta x_i^\top x_j + \nu)$

Kernel trick

An example: polynomial kernels for 2-dimensional data

$$k(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}^T\mathbf{y})^2 = (1+x_1\,y_1+x_2\,y_2)^2 =$$
$$= 1 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1y_1 + 2x_2y_2 + 2x_1x_2y_1y_2$$

This is an inner product between two 6-dimensional vectors:

$$\varphi(x) = \varphi(x_1,x_2) = \left(1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2\right)$$
$$\varphi(y) = \varphi(y_1,y_2) = (1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1y_2)$$
$$\Longrightarrow \quad K(x,y) = \varphi(x)^T\varphi(y)$$

What happens if we use the naive polynomial kernel? $K(x,y) = (x^Ty)^2$

Kernel trick

Another example: write the Gaussian kernel as an inner product.

$$
\begin{aligned}
K(x,z) = e^{-\frac{1}{2\sigma^2}(x-z)^2} &= e^{-\frac{x^2+z^2}{2\sigma^2}} e^{\frac{xz}{\sigma^2}} \\
&= e^{-\frac{x^2+z^2}{2\sigma^2}} \left(\sum_{n=0}^{\infty} \frac{(xz)^n}{\sigma^{2n} n!} \right) \\
&= e^{-\frac{x^2+z^2}{2\sigma^2}} \left(\sum_{n=0}^{\infty} \sqrt{\frac{1}{\sigma^{2n} n!}} x^n \cdot \sqrt{\frac{1}{\sigma^{2n} n!}} z^n \right) \\
&= e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{z^2}{2\sigma^2}} \left[1 \cdot 1 + \sqrt{\frac{1}{\sigma^2 1!}} x \cdot \sqrt{\frac{1}{\sigma^2 1!}} z + \sqrt{\frac{1}{\sigma^4 2!}} x^2 \cdot \sqrt{\frac{1}{\sigma^4 2!}} z^2 + \ldots \right] \\
&= \phi(x)^\top \phi(z)
\end{aligned}
$$

Kernel trick

Another example: write the Gaussian kernel as an inner product.

$$
\begin{aligned}
K(x,z) = e^{-\frac{1}{2\sigma^2}(x-z)^2} &= e^{-\frac{x^2+z^2}{2\sigma^2}} e^{\frac{xz}{\sigma^2}} \\
&= e^{-\frac{x^2+z^2}{2\sigma^2}} \left(\sum_{n=0}^{\infty} \frac{(xz)^n}{\sigma^{2n} n!} \right) \\
&= e^{-\frac{x^2+z^2}{2\sigma^2}} \left(\sum_{n=0}^{\infty} \sqrt{\frac{1}{\sigma^{2n} n!}} x^n \cdot \sqrt{\frac{1}{\sigma^{2n} n!}} z^n \right) \\
&= e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{z^2}{2\sigma^2}} \left[1 \cdot 1 + \sqrt{\frac{1}{\sigma^2 1!}} x \cdot \sqrt{\frac{1}{\sigma^2 1!}} z + \sqrt{\frac{1}{\sigma^4 2!}} x^2 \cdot \sqrt{\frac{1}{\sigma^4 2!}} z^2 + \ldots \right] \\
&= \phi(x)^\top \phi(z)
\end{aligned}
$$

Takeaway: by replacing the usual inner product with the Gaussian kernel it’s as if we’re projecting the data into an infinite dimensional space and finding a separating hyperplane there.

Recap of SVMs + kernel trick

- **Idea:** find a separating hyperplane that maximizes margins (with some slack) between classes
- This becomes an optimization problem:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k K(x_i, x_k)$$

$$\text{subject to } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0$$

- The maximum depends on the data only through the inner product, so we can use the kernel trick to “lift” the data into a higher-dimensional space which hopefully helps us to find a separating hyperplane

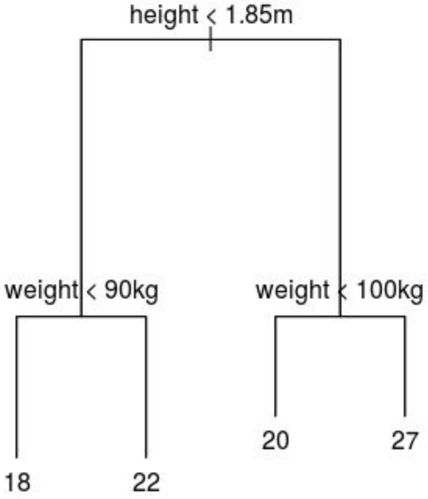
SVM in practice

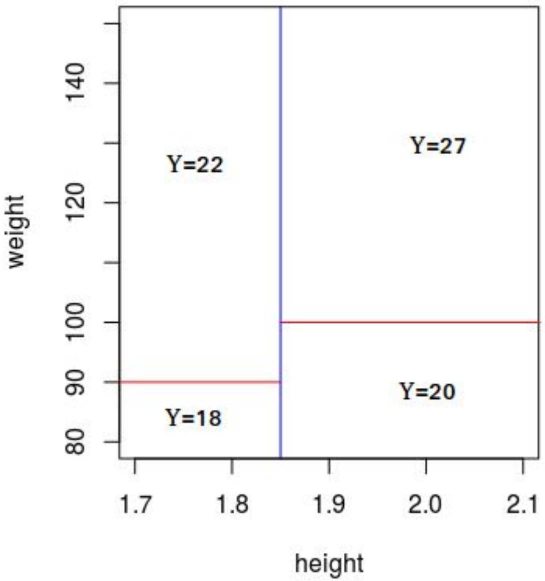
- Kernel trick allows for extreme flexibility
- However, with this greater flexibility comes a greater danger of overfitting, especially if p is large
- Lots of other methods based upon this kernel trick:
  - Kernel PCA
  - Kernel ridge regression
  - Spectral clustering
  - etc.

Trees and random forests

CART (Classification and Regression Trees)

Idea: recursively partition the data space via binary splits and fit a simple model for each result region





CART (Classification and Regression Trees)

- At each split in the tree, how to choose what variable (*j*) to split on and what threshold (*t*)?
- For now, assume we have a regression problem: at each split, we want to optimize L^2 loss

$$\min_{j,t}\left\{\min_{\mu_L}\sum_{i:x_{ij}\leq t}(y_i-\mu_L)^2+\min_{\mu_R}\sum_{i:x_{ij}>t}(y_i-\mu_R)^2\right\}$$

- Can find global minimum for each split via a brute force search, but not necessarily for the entire tree

CART (Classification and Regression Trees)

Brute force search algorithm:

- For each feature j:
  - Sort X: $x_{1j} \leq \ldots \leq x_{nj}$ => $O(n \log n)$
  - Scan from left to right and threshold t_j that minimizes L^2 loss => $O(n)$
- Out of the p possible splits, take the best t_j

Total complexity: $O(pn \log n + pnK)$ where K is the number of splits

- For classification, can replace L^2 loss with classification error, Gini index, etc.

CART (Classification and Regression Trees)

Advantages:

- Can deal with continuous, categorical, binary, count features all at the same time
- Doesn't depend on scale of $\mathbf{X}$
- Easily interpretable, fairly flexible, and fast

Disadvantages:

- Potentially too simple
- Not a great balance between bias-variance tradeoff
  - As depth of tree increases, overfits to the training data, resulting in high variance and no bias
  - If tree is too shallow, underfits and we have the opposite problem

Random forest

Idea: Increase bias of trees to reduce variance of forest.

- Introduce bias in each tree of the forest by downsampling the m_{try} out of p variables randomly to search and potentially split on (introduces bias as we sometimes do not split on features that are informative).
    - Why do this? This reduces the correlation between the predictions of trees in the forest.
- To reduce the variance:
    - Grow many trees (e.g., 500 trees) using bootstrap samples of the data and average predictions over this ”forest”.
    - Since each tree is i.i.d. the variance of the forest is a function of the variance of each tree and the correlation between the predictions of the trees. Decreasing this correlation decrease the variance of the forest.

Random forest algorithm

Inputs: number of trees to grow (B), number of variables to randomly select a each split (m_{try}), number of leaf/terminal nodes (M)

For each tree, $b = 1, \ldots, B$:

- Bootstrap data: $\mathbf{X}^{*b}$
- Grow decision (CART) tree T^{b} such that:
  - At each split in the tree, randomly choose m_{try} out of p variables to try and potentially split on
  - Grow until tree has M leaf / terminal nodes
- Make prediction: $\hat{y}(x) = \frac{1}{B}\sum_{b=1}^{B} T^{b}(x)$

Random Forest in Practice

- Because we are bootstrapping the data before constructing each tree, we essentially have a “test set” for each tree that we can exploit
    - We call this left out data due to bootstrapping the **out-of-bag (OOB)** data, from which we can compute the OOB error
    - OOB error can be used like CV error to tune parameters like m_{try}
- Can obtain marginal feature importances from RF

Random Forest in Practice

Advantages:

- Doesn’t depend on scale of $\mathbf{X}$
- Great prediction for lots of problems
- Reduces bias and variance simultaneously unlike CART

Disadvantages:

- May not be optimal with correlated features or $p >> n$
- No longer easily interpretable

In R: `ranger` (https://github.com/imbs-hl/ranger) and `Rforestry` (https://github.com/forestry-labs/Rforestry)

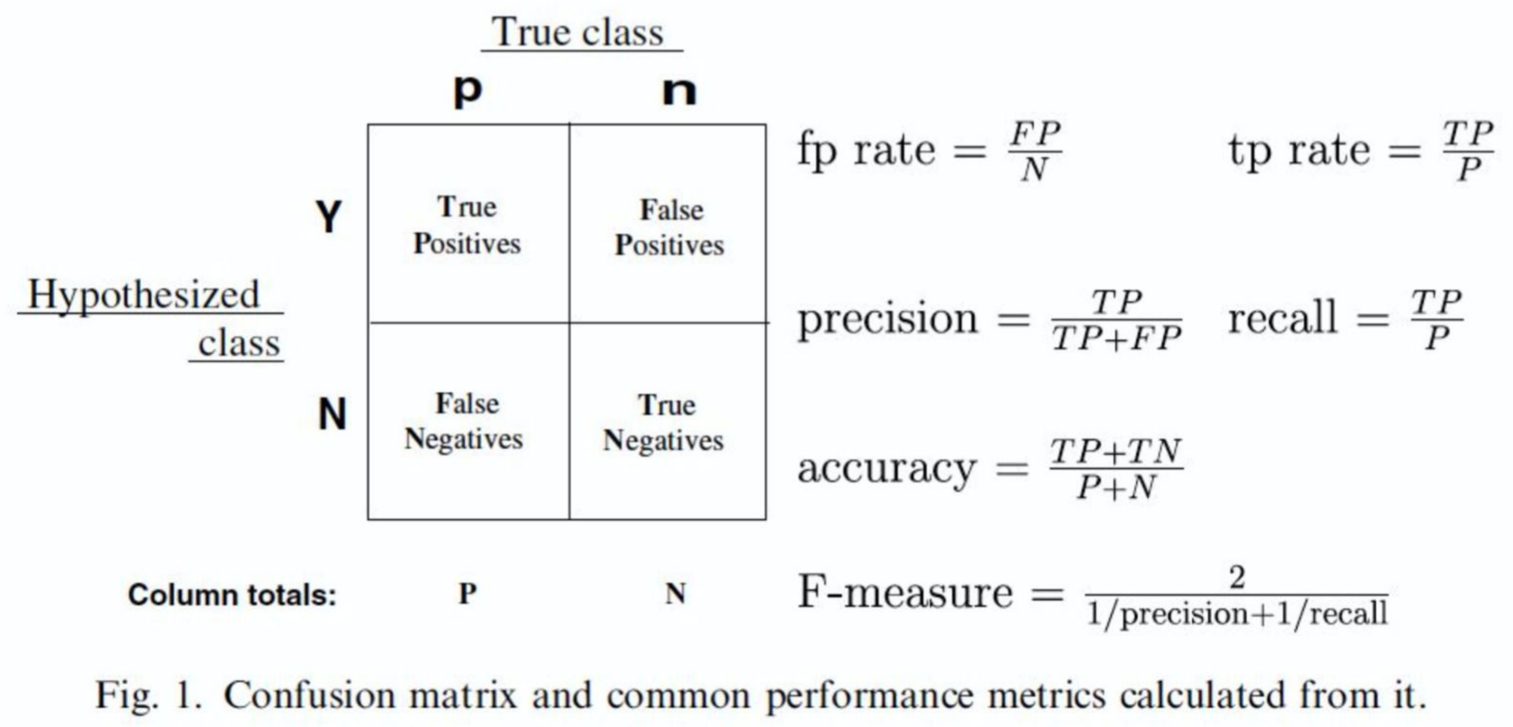
- `Rforestry` created by a former student of Bin, Sören Künzel, maintained by Theo

Evaluation metrics for classification

How to evaluate your classification methods?

- Going beyond classification error
- What if we have class imbalance?
  - For example, if we take a sample of 100 people and only 10 have the disease, then always predicting healthy gives 90% classification accuracy!
  - We can do better.

Confusion matrix



True class

Hypothesized class

	p	n
Y	True Positives	False Positives
N	False Negatives	True Negatives
Column totals:	P	N

$$\text{fp rate} = \frac{FP}{N} \qquad \text{tp rate} = \frac{TP}{P}$$

$$\text{precision} = \frac{TP}{TP+FP} \qquad \text{recall} = \frac{TP}{P}$$

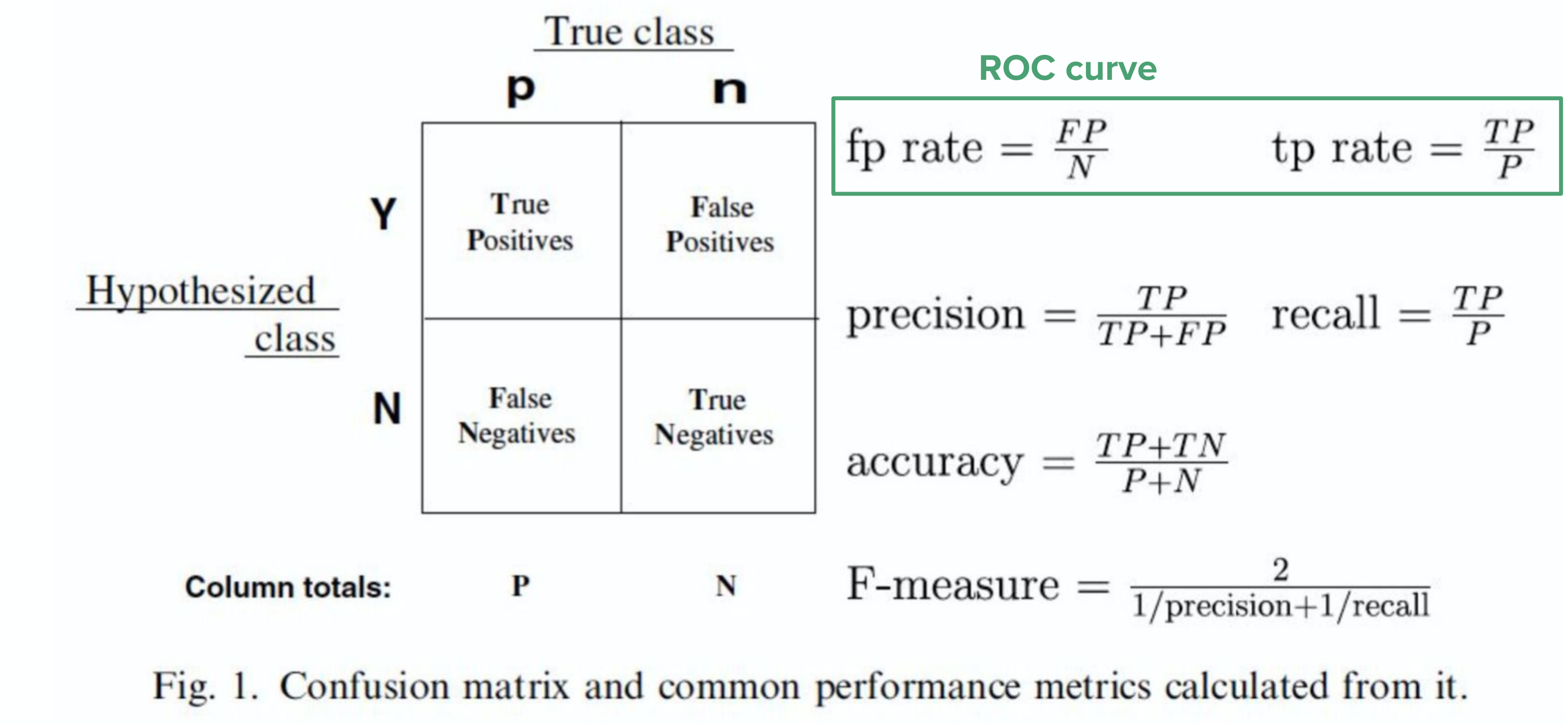
$$\text{accuracy} = \frac{TP+TN}{P+N}$$

$$\text{F-measure} = \frac{2}{1/\text{precision}+1/\text{recall}}$$

Fig. 1. Confusion matrix and common performance metrics calculated from it.

Source: Fawcett (2005)

Confusion matrix



		True class	
		p	n
Hypothesized class	Y	True Positives	False Positives
	N	False Negatives	True Negatives
Column totals:		P	N

ROC curve

$$\text{fp rate} = \frac{FP}{N} \qquad \text{tp rate} = \frac{TP}{P}$$

$$\text{precision} = \frac{TP}{TP+FP} \qquad \text{recall} = \frac{TP}{P}$$

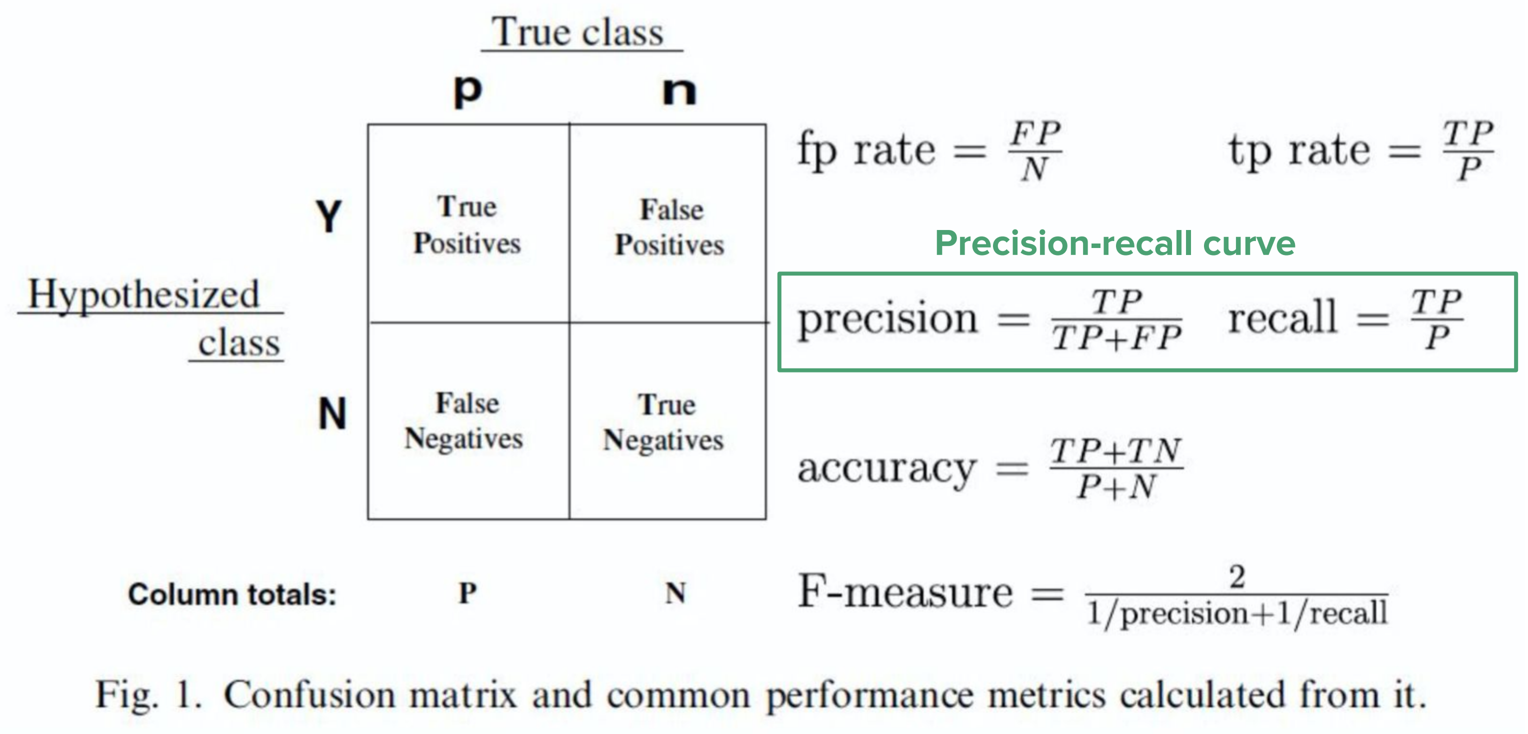
$$\text{accuracy} = \frac{TP+TN}{P+N}$$

$$\text{F-measure} = \frac{2}{1/\text{precision}+1/\text{recall}}$$

Fig. 1. Confusion matrix and common performance metrics calculated from it.

Source: Fawcett (2005)

Confusion matrix



True class

		p	n
Hypothesized class	Y	True Positives	False Positives
	N	False Negatives	True Negatives
Column totals:		P	N

$$\text{fp rate} = \frac{FP}{N} \qquad \text{tp rate} = \frac{TP}{P}$$

Precision-recall curve

$$\text{precision} = \frac{TP}{TP+FP} \qquad \text{recall} = \frac{TP}{P}$$

$$\text{accuracy} = \frac{TP+TN}{P+N}$$

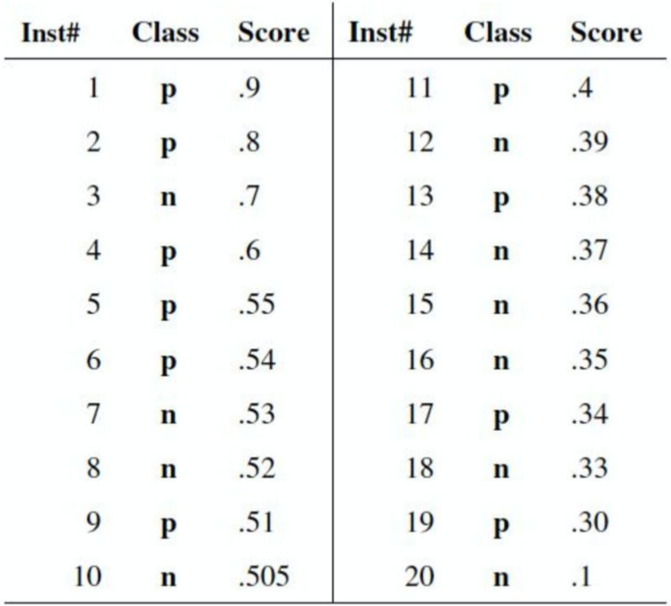
$$\text{F-measure} = \frac{2}{1/\text{precision}+1/\text{recall}}$$

Fig. 1. Confusion matrix and common performance metrics calculated from it.

Source: Fawcett (2005)

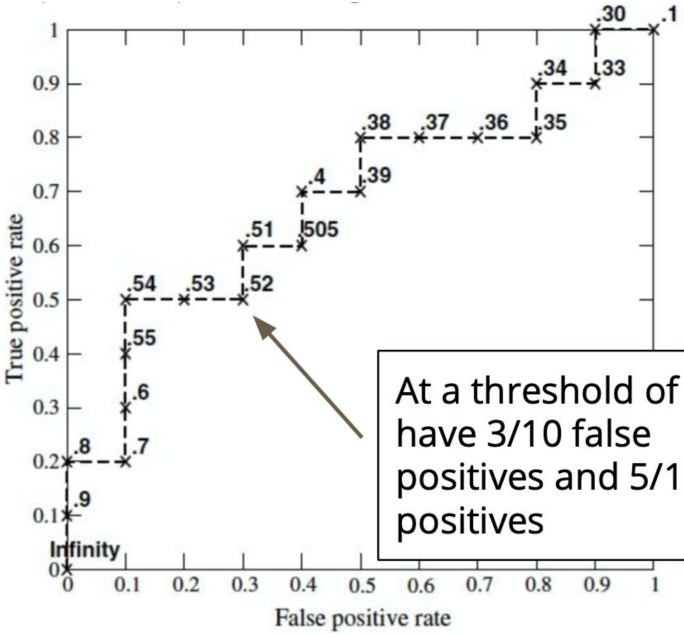
Receiver operating characteristics (ROC) curve

We can generate an ROC curve when the output of a classifier is a probability and we must choose a threshold for the final predicted class



Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

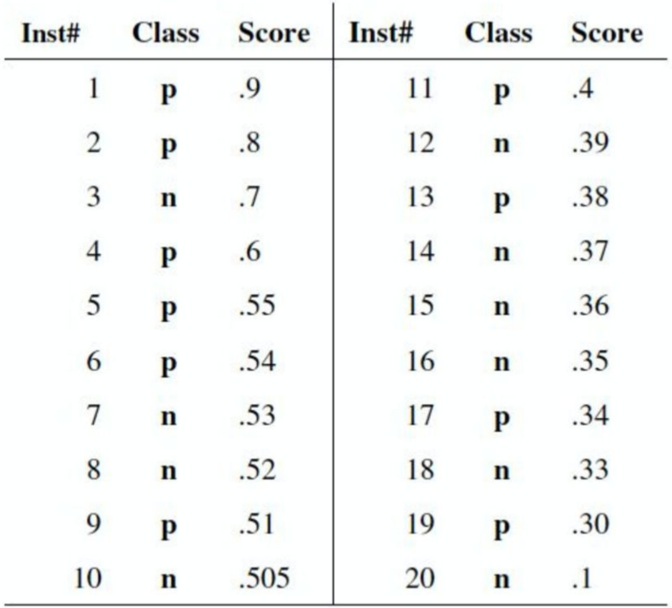
Source: Fawcett (2005)



At a threshold of .52 we have 3/10 false positives and 5/10 true positives

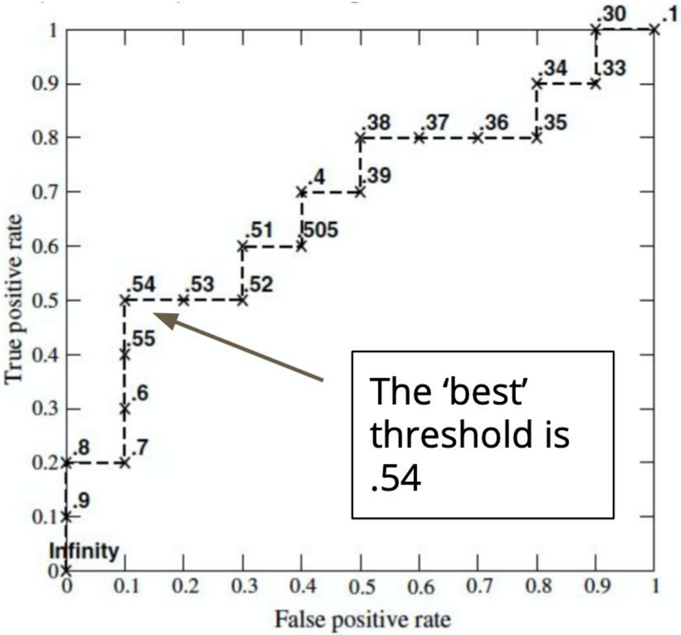
Receiver operating characteristics (ROC) curve

We can generate an ROC curve when the output of a classifier is a probability and we must choose a threshold for the final predicted class



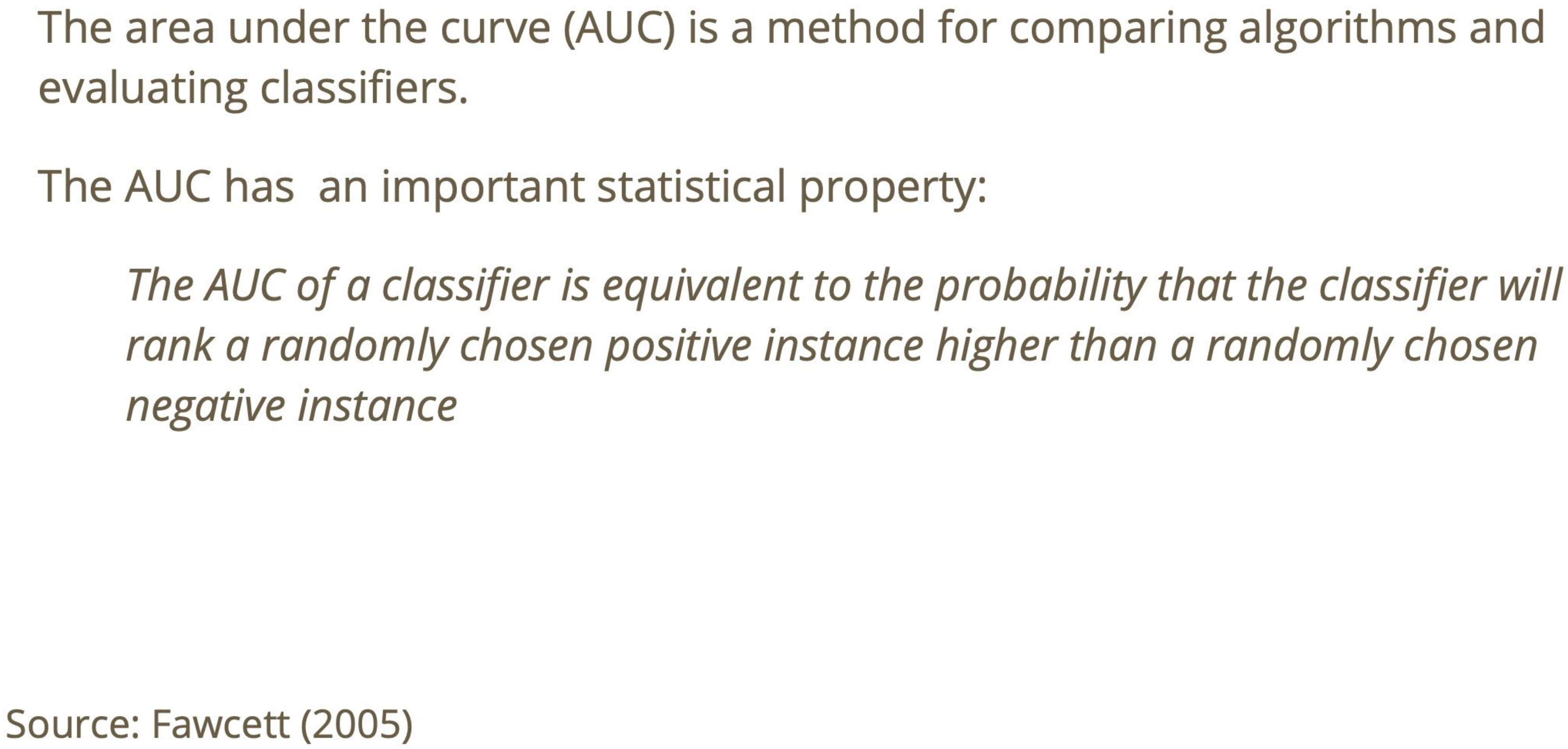
Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1

Source: Fawcett (2005)



The ‘best’ threshold is .54

Area under the curve



The area under the curve (AUC) is a method for comparing algorithms and evaluating classifiers.

The AUC has an important statistical property:

> *The AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance*

Source: Fawcett (2005)

Area under the curve

Care should be taken when using ROC curves to compare classifiers

- The ROC graph is often used to select the best classifiers simply by graphing them in ROC space and seeing which one dominates.
- This is misleading: it is analogous to taking the maximum of a set of accuracy figures from a single test set.
- Without a measure of **variance** we cannot compare classifiers

It is a good idea to the average of multiple ROC curves (e.g. via cross validation)

See Fawcett (2005) for examples on how to average

Source: Fawcett (2005)

ROC vs Precision-Recall (PR) Curves

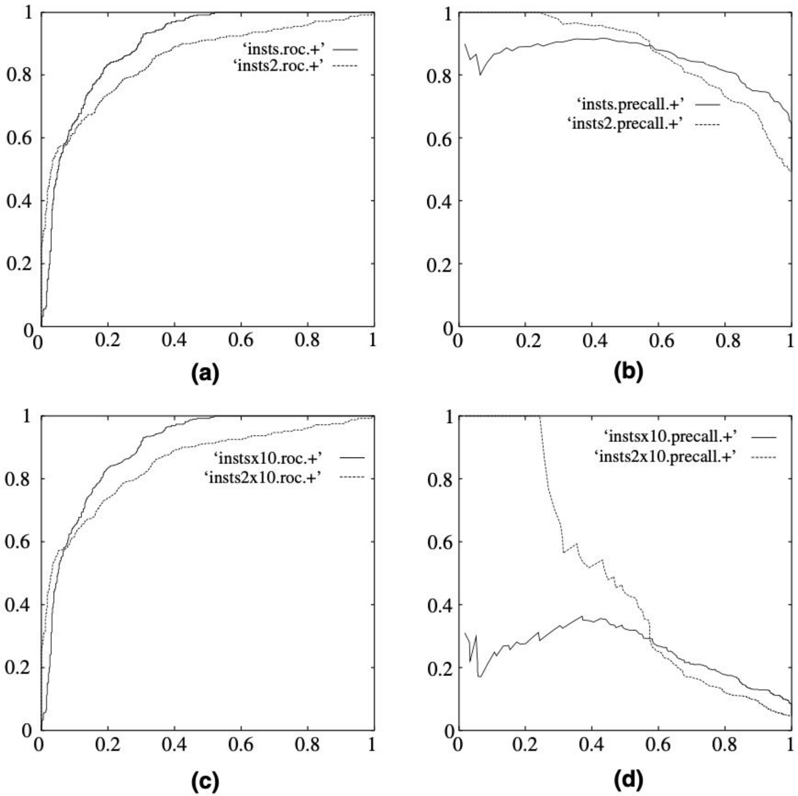


Fig. 5. ROC and precision-recall curves under class skew. (a) ROC curves, 1:1; (b) precision-recall curves, 1:1; (c) ROC curves, 1:10 and (d) precision-recall curves, 1:10.

ROC vs PR curves

- Generally, precision-recall curves are preferred when there is class imbalance
- ROC curves tend to paint an overly optimistic view of the model on datasets with class imbalance
- PR calculations do not involve the true negatives rate and hence do not typically present such an optimistic view

tidymodels
rsample
recipes
parsnip
TUNE
yardstick
www.rstudio.com