

STAT 215A Fall 2019

Week 12

Tiffany Tang

11/15/19

Announcements

- ▶ Lab 4 Clarifications:
 - ▶ Everyone in the group submits the **same** lab4 report/files, but each person needs to push the files to their individual private repos
- ▶ Plan for the rest of the semester
 - ▶ Next week:
 - ▶ Introduce final project and discuss expectations for report
 - ▶ Additional topics?
 - ▶ December 6: Rebecca Barter guest lecture; research talk

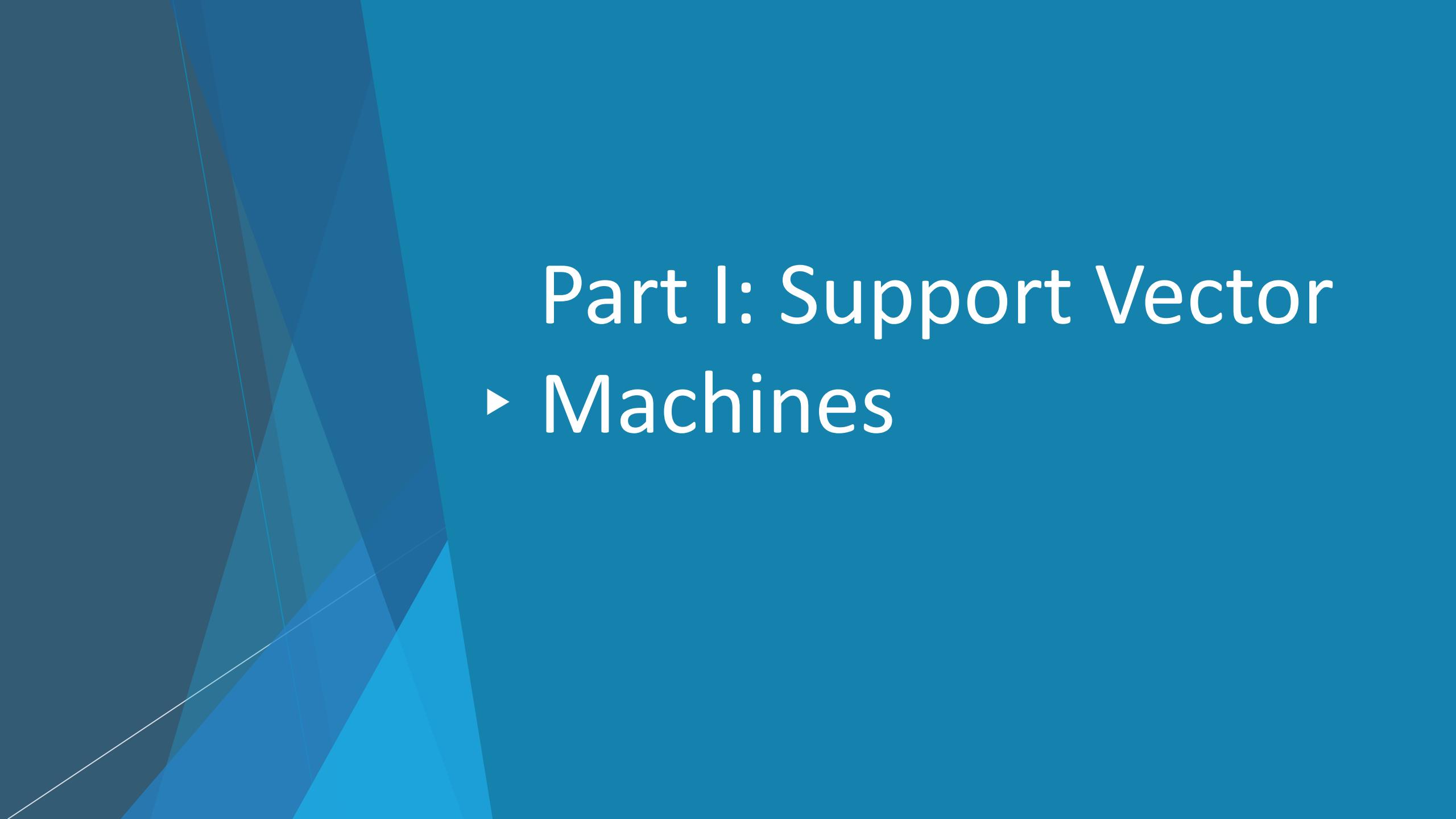
Review: Classification methods thus far

	Logistic	Naïve Bayes	LDA	QDA
Pros	<ul style="list-style-type: none">Can do inference (with all the caveats)	<ul style="list-style-type: none">Can choose any likelihood model	<ul style="list-style-type: none">Convenient visualizationsLinearly separable	<ul style="list-style-type: none">Quadratic decision boundaries
Cons	<ul style="list-style-type: none">Problems when $p > n$ (a solution: regularized logistic regression)Model misspecification?	<ul style="list-style-type: none">Assumes that features are independent (a very strong assumption)Model misspecification?	<ul style="list-style-type: none">Problems when $p > n$ (a solution: RDA)Model misspecification? Non-normal or non-linear decision boundaries?	<ul style="list-style-type: none">Problems when $p > n$ (a solution: RDA)Requires larger n to estimate more parameters adequately (compared to LDA)Model misspecification? Non-normal or non-linear decision boundaries?

► What about more flexible, non-parametric methods?

Additional tools for your tool belt

- ▶ A step towards more flexible classification: k nearest neighbors
- ▶ Plan for today:
 - ▶ SVMs + Kernel Trick
 - ▶ Trees and Random Forests
- ▶ Evaluation?

The background features a large, abstract graphic on the left side composed of overlapping blue triangles of varying shades of teal and cyan. It has a clean, modern look with sharp edges.

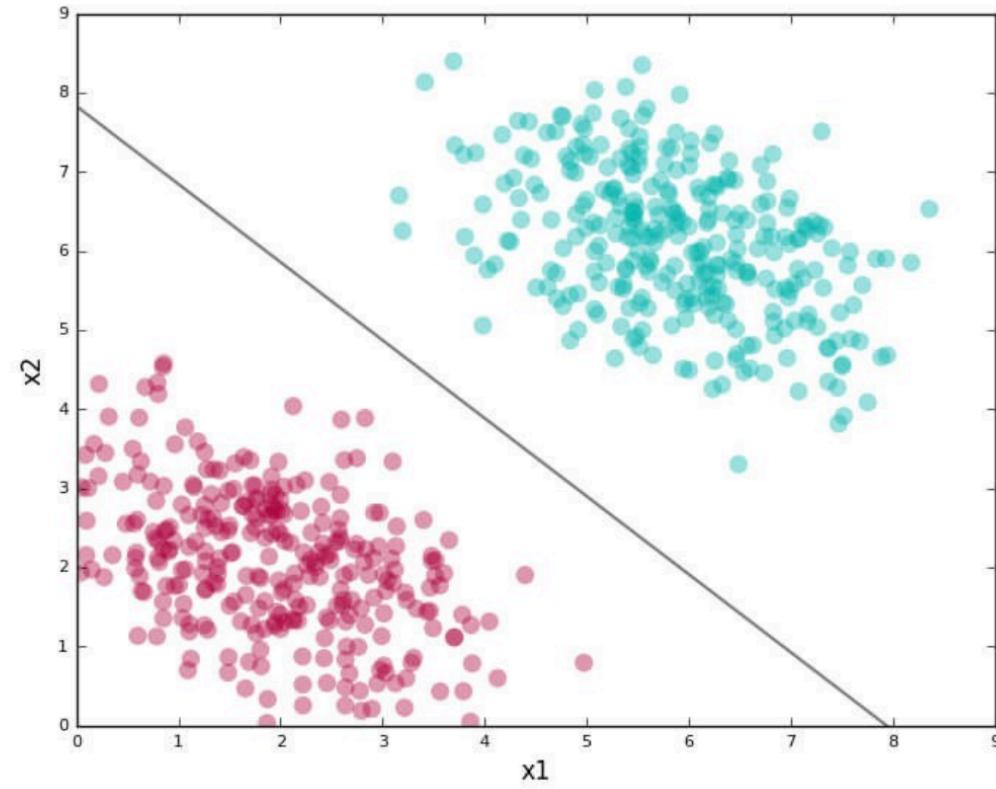
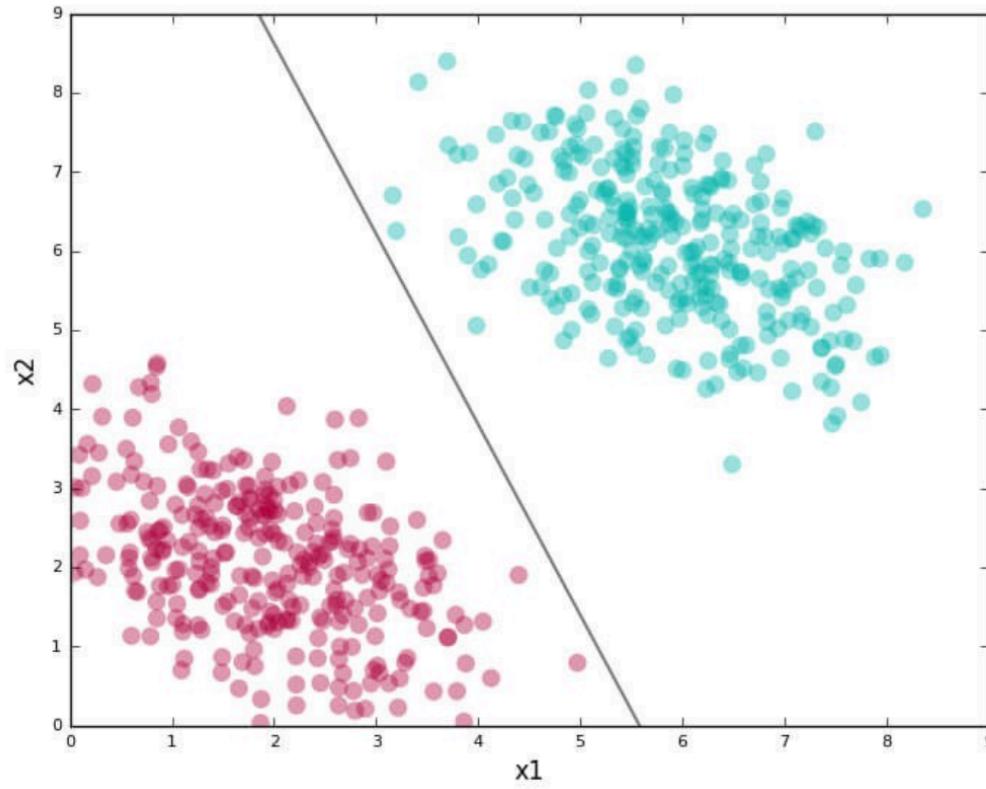
Part I: Support Vector Machines

Support Vector Machines (SVM)

- ▶ One can do a lot of math here...
- ▶ Some resources
 - ▶ Gives great intuition: <https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93>
 - ▶ Some discussion of the math behind SVMs:
 - ▶ <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>
 - ▶ In Elements of Statistical Learning, see p. 132 (section 4.5.2) and p. 419 (ch 12.2)
 - ▶ We will focus on the intuition

Support Vector Machines (SVM)

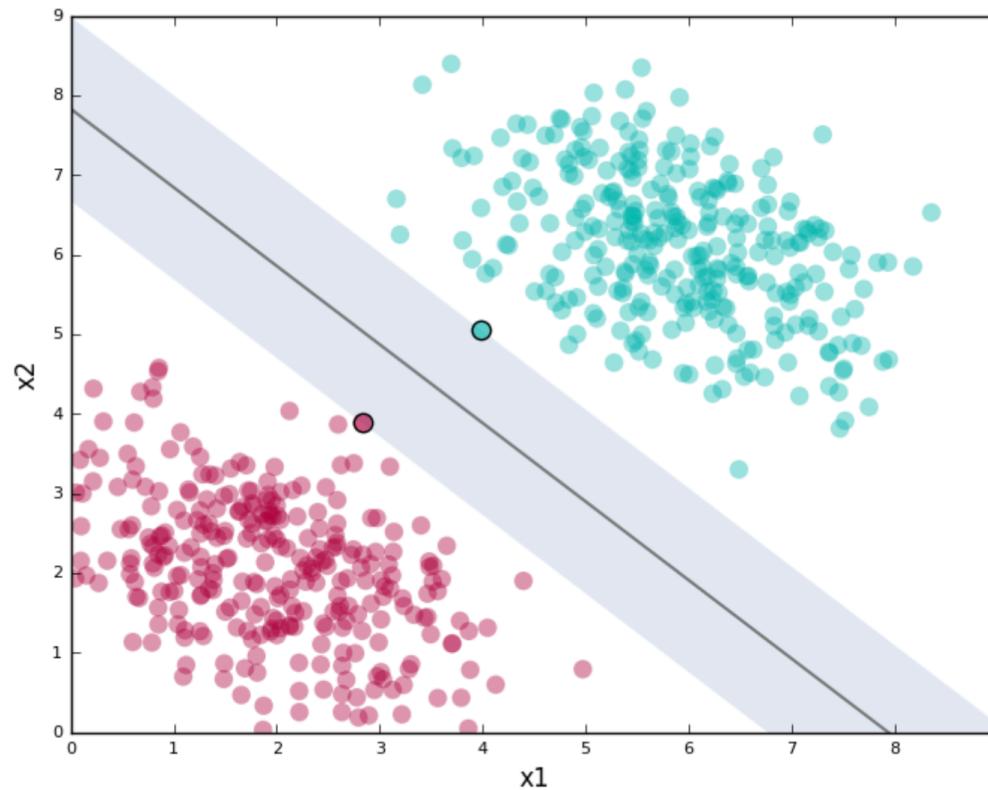
- ▶ Which classifier is better?



Support Vector Machines (SVM)

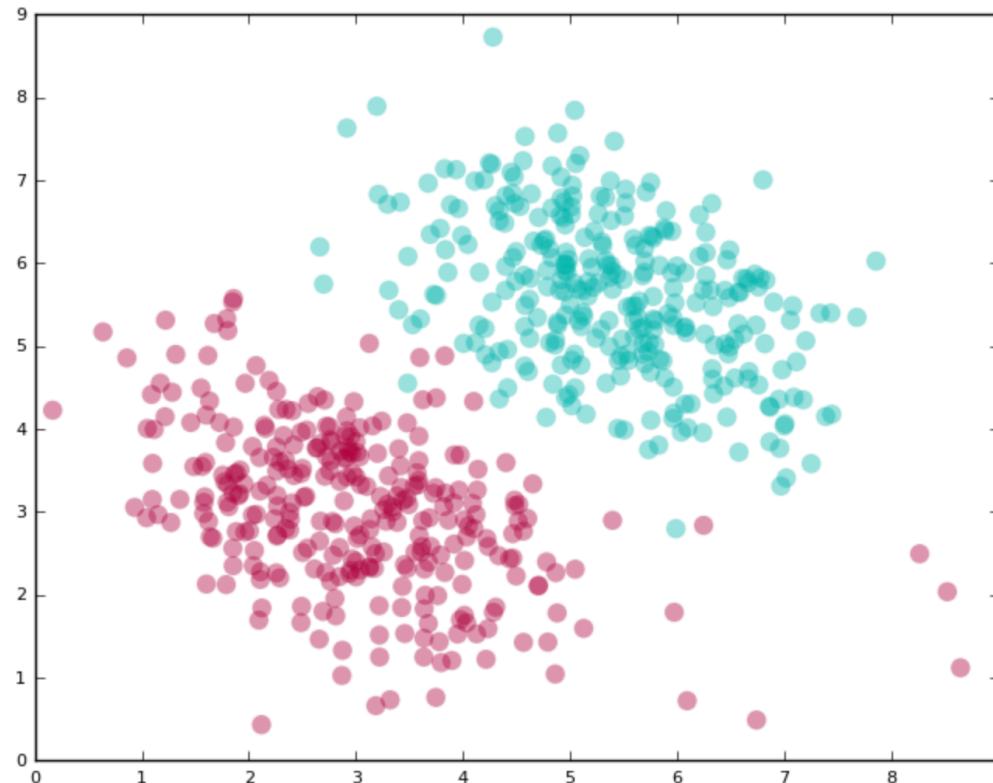
- ▶ **Basic Idea:**

- ▶ Maximize the space between two hyperplanes that separate the classes
 - ▶ Called a maximum margin classifier



Support Vector Machines (SVM)

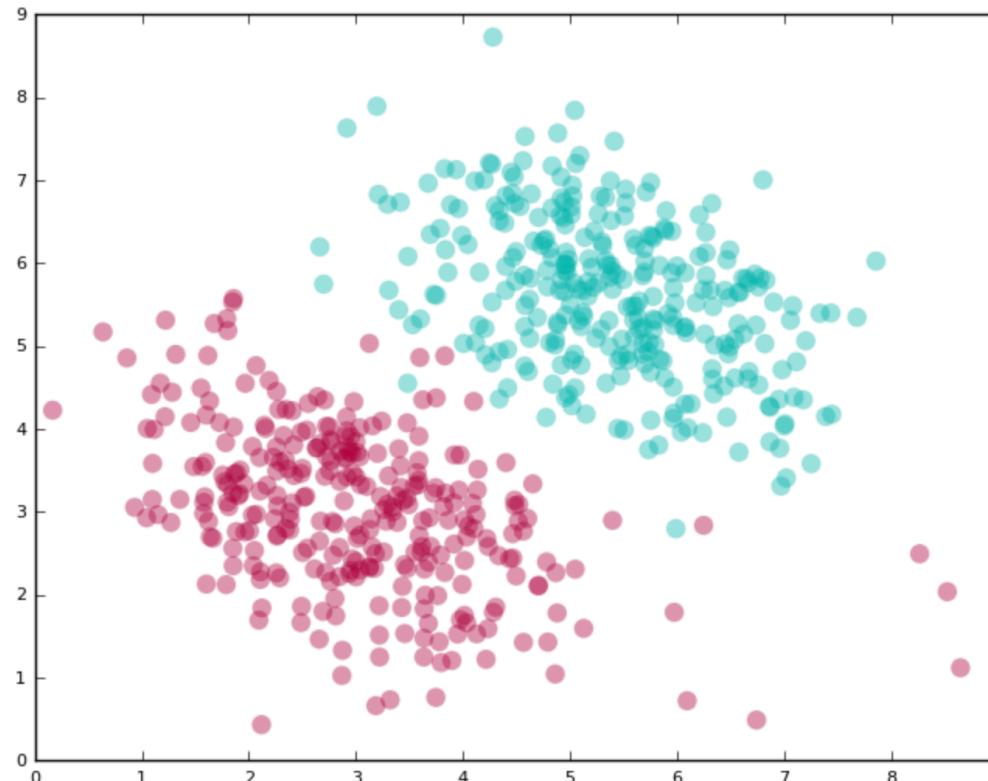
- ▶ What about the case where things aren't perfectly separable?



Support Vector Machines (SVM)

- ▶ **Basic Idea:**

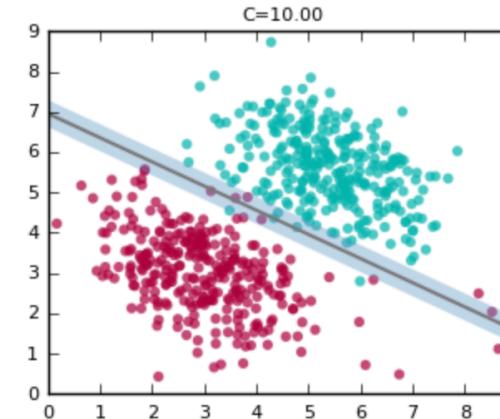
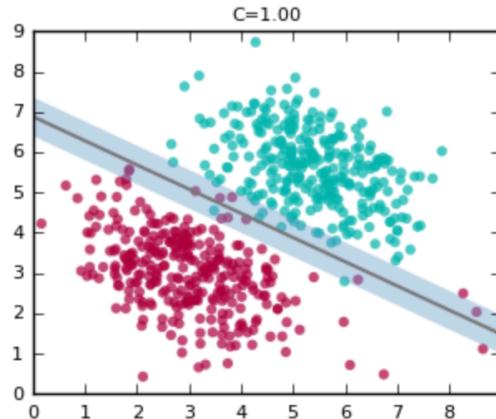
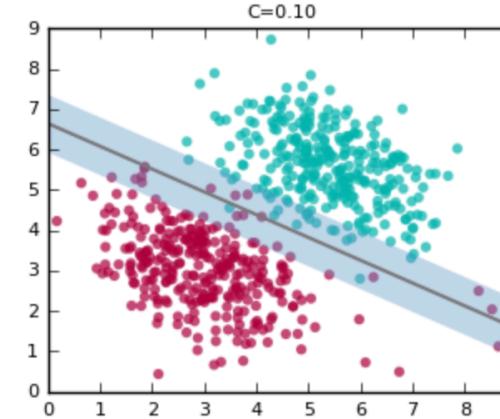
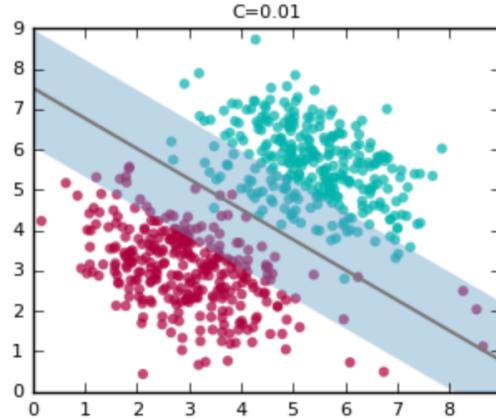
- ▶ Maximize the space between two hyperplanes that separate the classes while allowing for some “slack”



Support Vector Machines (SVM)

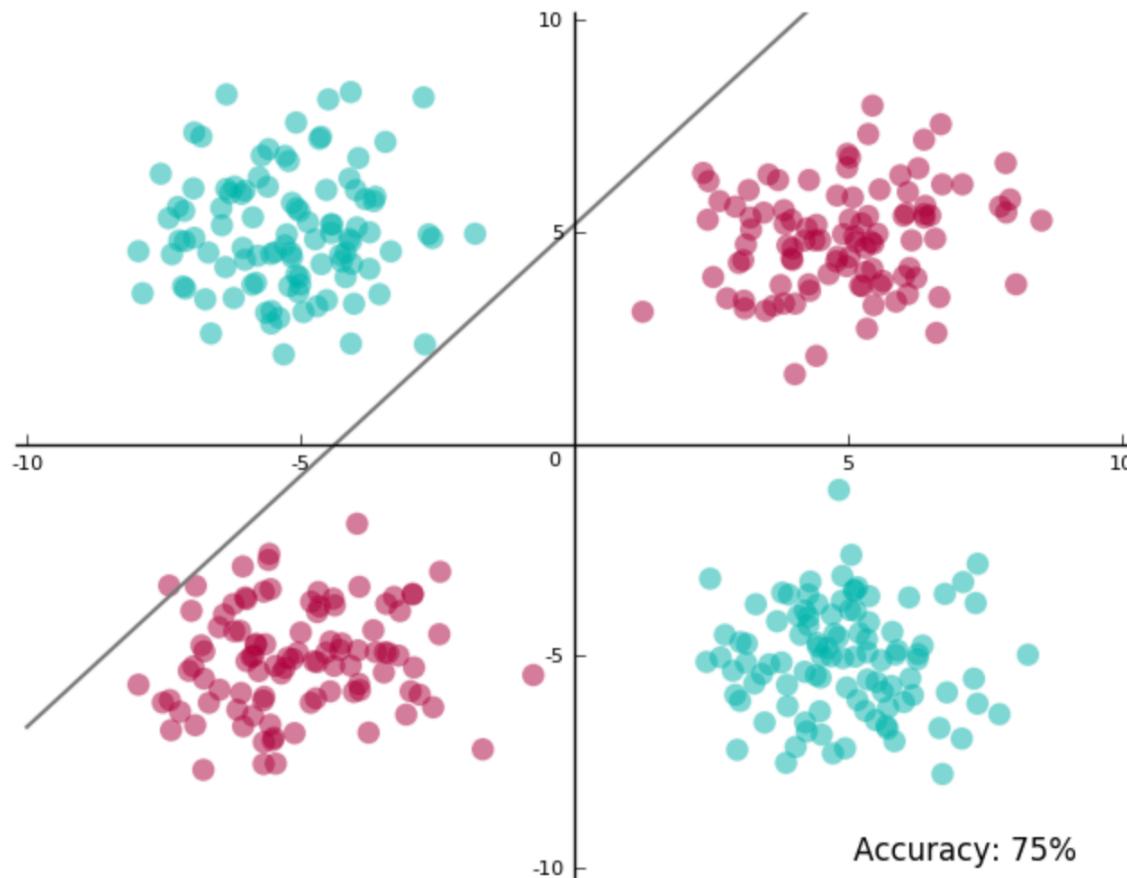
► Basic Idea:

- Maximize the space between two hyperplanes that separate the classes while allowing for some “slack”



Support Vector Machines (SVM)

- ▶ So far so good with the linearly separable case, but what about non-linearly separable data



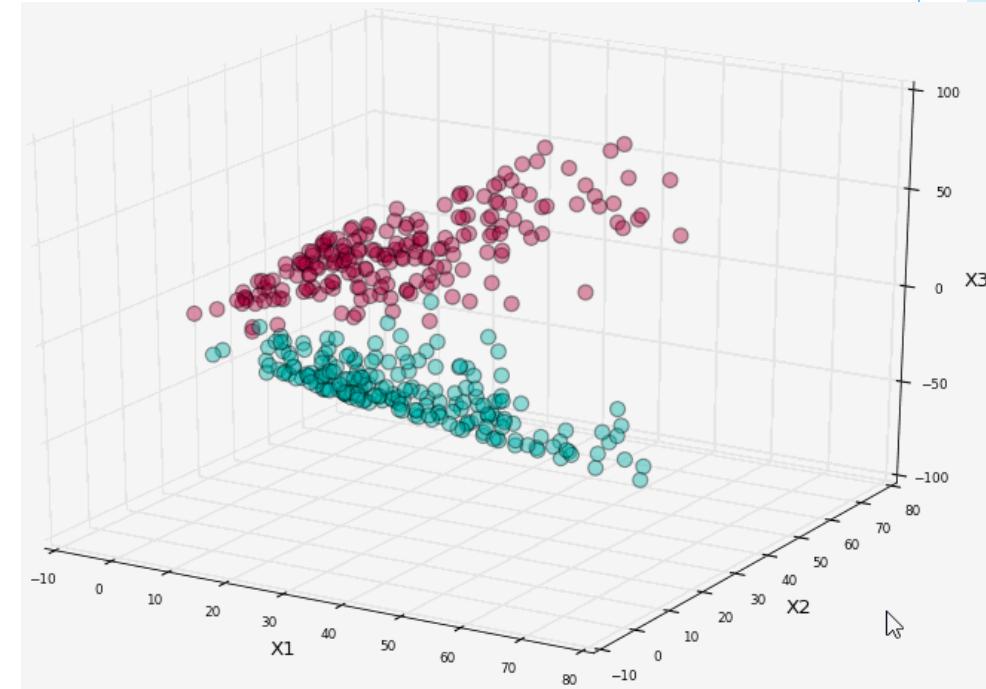
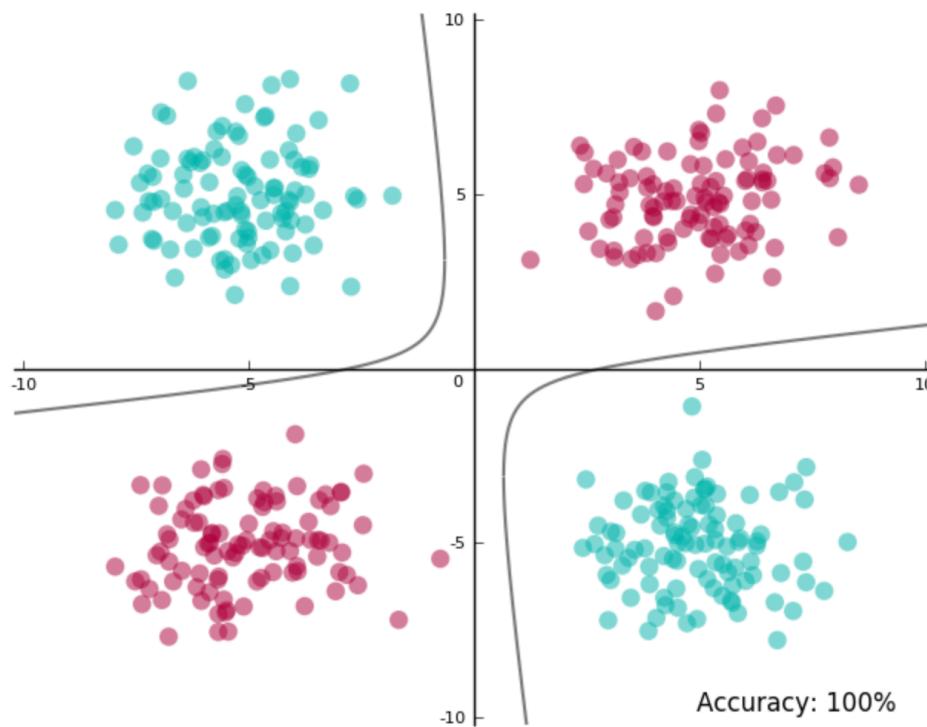
Support Vector Machines (SVM)

- **Idea:** find a higher-dimensional projection such that in that higher-dimensional space, the data becomes linearly separable

$$X_1 = x_1^2$$

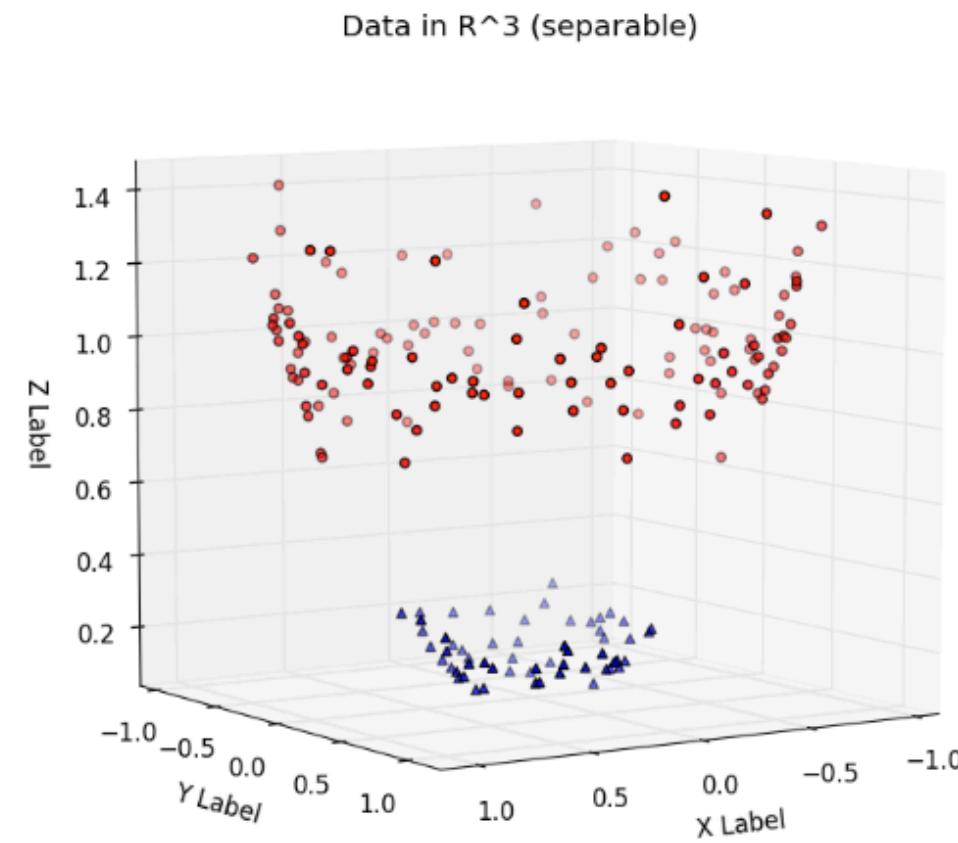
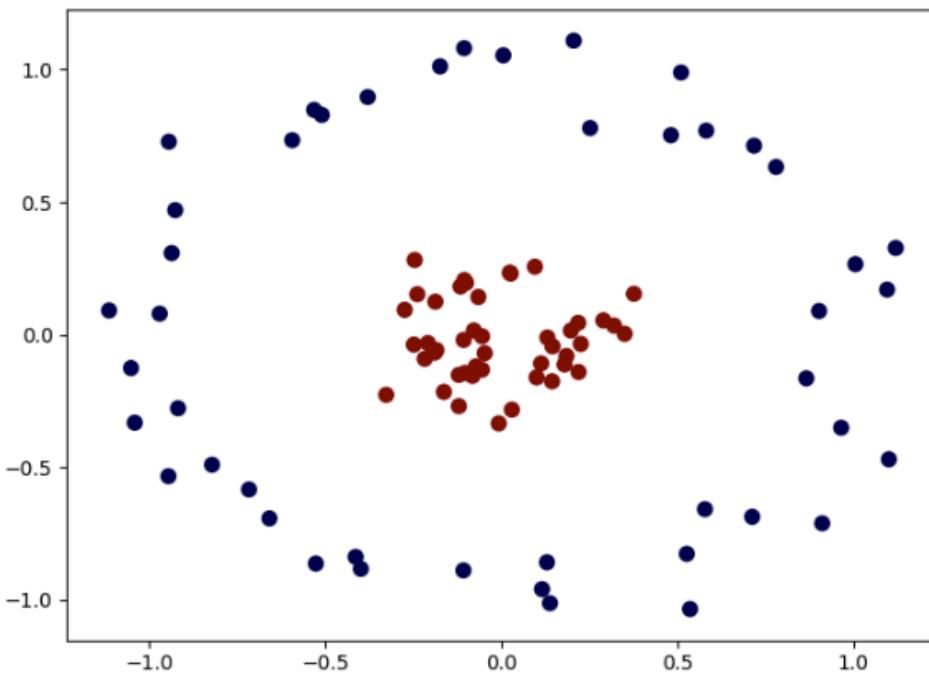
$$X_2 = x_2^2$$

$$X_3 = \sqrt{2}x_1x_2$$



Support Vector Machines (SVM)

- **Idea:** find a higher-dimensional projection such that in that higher-dimensional space, the data becomes linearly separable



Support Vector Machines (SVM)

- ▶ How to perform this “lifting” trick in a computationally feasible way?
SVMs take advantage of the **kernel trick**
- ▶ Can show that by maximizing the margins while allowing for slack,
SVM solves the following optimization problem:

$$\text{maximize } \mathcal{L}_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\begin{aligned} \text{subject to } & \sum_i \alpha_i y_i = 0 \\ & \alpha_i \geq 0 \quad \forall i \end{aligned}$$

- ▶ Main point: this is a function of the inner product between data points

Kernel Trick

- ▶ SVM simply needs the inner product between data points: $x_i^T x_j$
- ▶ Leveraging the idea of projecting onto a higher dimensional space, we can replace the usual inner product $x_i^T x_j$ with

$$\varphi(x_i)^T \varphi(x_j)$$

where φ is some map from R^p to a higher-dimensional space (possibly even infinite dimensional)

- ▶ What is the trick? No need to even figure out what φ is
 - ▶ This is great news for us! Imagine trying to compute an infinite dimensional function...
- ▶ Instead of explicitly computing φ , just need kernel function:

$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$$

Kernel Trick

Some common kernel functions: $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$

- ▶ Linear kernel: $K(x, z) = x^T z$
- ▶ Naïve polynomial kernel: $K(x, z) = (x^T z)^d$
- ▶ Polynomial kernel: $K(x, z) = (1 + x^T z)^d$
- ▶ Gaussian kernel: $K(x, z) = e^{-\frac{1}{2\sigma^2}||x - z||_2^2}$
- ▶ Radial basis kernel: $K(x, z) = e^{-\gamma||x - z||_2^2}$
- ▶ Sigmoid kernel: $K(x, z) = \tanh(\eta \cdot x^T z + \nu)$

Kernel Trick

- ▶ Let's show that we can write the Gaussian kernel as

$$K(x, z) = e^{-\frac{1}{2\sigma^2}(x-z)^2} = \phi(x_i)^T \phi(x_j)$$

$$\begin{aligned} e^{\frac{-1}{2\sigma^2}(x_i-x_j)^2} &= e^{\frac{-x_i^2-x_j^2}{2\sigma^2}} \left(1 + \frac{2x_i x_j}{1!} + \frac{(2x_i x_j)^2}{2!} + \dots \right) \\ &= e^{\frac{-x_i^2-x_j^2}{2\sigma^2}} \left(1 \cdot 1 + \sqrt{\frac{2}{1!}} x_i \cdot \sqrt{\frac{2}{1!}} x_j + \sqrt{\frac{(2)^2}{2!}} (x_i)^2 \cdot \sqrt{\frac{(2)^2}{2!}} (x_j)^2 + \dots \right) \\ &= \phi(x_i)^T \phi(x_j) \end{aligned} \tag{1.25}$$

$$\text{where, } \phi(x) = e^{\frac{-x^2}{2\sigma^2}} \left(1, \sqrt{\frac{2}{1!}} x, \sqrt{\frac{2^2}{2!}} x^2, \dots \right)$$

- ▶ **Takeaway:** by replacing the usual inner product $x_i^T x_j$ with the Gaussian kernel, it's as if we were projecting the data into an infinite dimensional space and finding a separating hyperplane in that infinite dimensional space!

Kernel Trick

- ▶ Can do something similar with other kernels: e.g., polynomial kernels for 2-dimensional data

$$\begin{aligned} k(\mathbf{x}, \mathbf{y}) &= (1 + \mathbf{x}^T \mathbf{y})^2 = (1 + x_1 y_1 + x_2 y_2)^2 = \\ &= 1 + x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 \end{aligned}$$

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

$$\varphi(x) = \varphi(x_1, x_2) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2)$$

$$\varphi(y) = \varphi(y_1, y_2) = (1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1 y_2)$$

$$\Rightarrow K(x, y) = \varphi(x)^T \varphi(y)$$

- ▶ What happens if we use the kernel $K(x, y) = (x^T y)^2$

Recap of SVMs + Kernel Trick

- ▶ **Basic idea:** find separating hyperplane that maximizes margins (with some slack) between classes
- ▶ Turns out the optimization problem corresponding to this idea is:

$$\text{maximize } \mathcal{L}_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\begin{aligned} \text{subject to } & \sum_i \alpha_i y_i = 0 \\ & \alpha_i \geq 0 \quad \forall i \end{aligned}$$

- ▶ Because the optimization problem depends only on the inner product $x_i^T x_j$, can use the kernel trick to “lift” the data into a higher-dimensional space to hopefully find an optimal separating hyperplane

Practical Notes on SVM

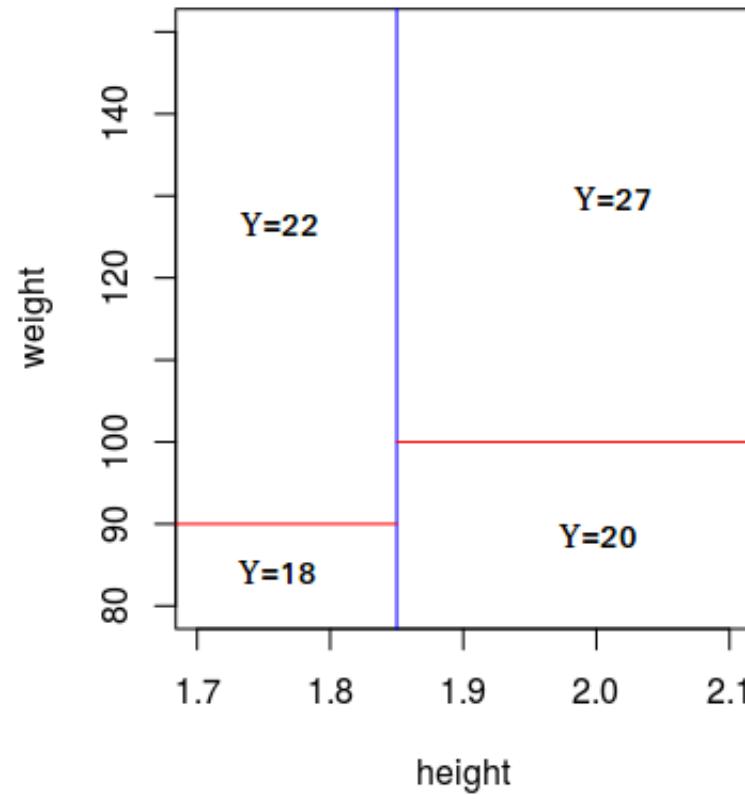
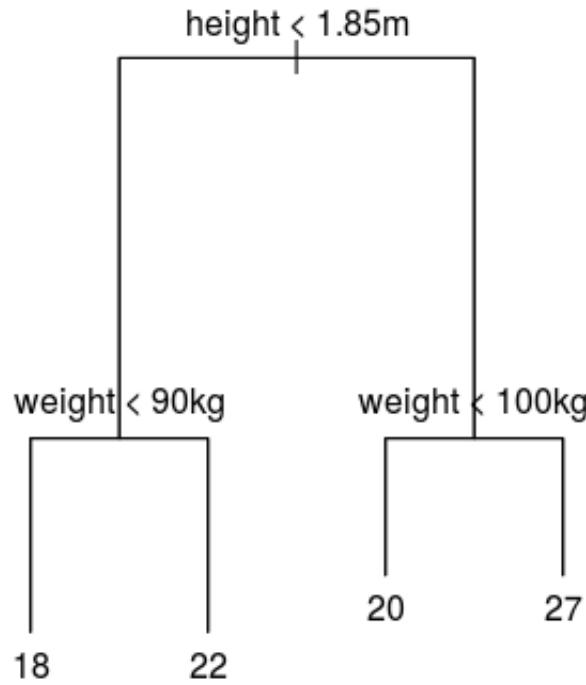
- ▶ Kernel trick allows for extreme flexibility
- ▶ But with flexibility, there is a greater danger of overfitting, especially if p is large
- ▶ Lots of other methods based upon this kernel trick: kernel PCA, kernel ridge regression, etc.



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
 - ▶ Note: we cannot necessarily find the global minimum for the entire tree

CART (Classification and Regression Trees)

- ▶ Brute force search algorithm:
 - ▶ For each feature j :
 - ▶ Sort X according to Y : $x_{(1)j} \leq \dots \leq x_{(n)j} \rightarrow O(n \log n)$
 - ▶ Scan from left to right and threshold t_j that minimizes L_2 loss $\rightarrow O(n)$
 - ▶ Out of the p possible splits, take the best t_j
- ▶ Total complexity: $O(pn \log n + pnK)$ where $K = \text{number of splits} \rightarrow \text{Not too bad!}$
- ▶ 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 X!!
- ▶ Easily interpretable, fairly flexible, and fast

► Disadvantages:

- ▶ Potentially too simple
- ▶ Not a great balance between bias-variance tradeoff
 - ▶ As depth of tree increases, massively overfit to the training data → no bias, high variance
 - ▶ If tree is too shallow, massively underfit → high bias, low variance

Random Forest

- ▶ **Idea:** try to reduce both the bias and variance using decision trees
- ▶ To reduce the bias, grow deep trees (i.e., grow trees to purity so that in each of the leaf nodes, we have 1-3 observations left)
- ▶ To reduce the variance,
 - ▶ Grow many trees (e.g., 500 trees) using bootstrap samples of the data and average over this "forest"
 - ▶ Try to force these trees to be close to iid: at each split, select m_{try} out of p variables randomly to search and potentially split on

Random Forest Algorithm

- ▶ **Inputs:** number of trees to grow (B), number of variables to randomly select at each split (m_{try}), number of leaf/terminal nodes (M)
- ▶ For each tree $b = 1, \dots, B$:
 - ▶ Bootstrap data $\rightarrow X^{*b}$
 - ▶ 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 X
 - ▶ Great prediction for lots of problems
 - ▶ Reduces bias and variance simultaneously unlike CART
- ▶ May not be optimal with correlated features or $p \gg n$
- ▶ No longer easily interpretable
- ▶ Two useful packages:
 - ▶ `library(randomForest)`
 - ▶ `library(ranger)`
 - ▶ `ranger` is much faster than `randomForest`

Another useful tool in practice

- ▶ Caret package
- ▶ See tutorial created by Rebecca Barter in the week12 folder

The background features a large, abstract graphic on the left side composed of overlapping blue triangles of varying shades of teal and cyan. It has thin white lines separating the triangles.

Evaluation Metrics for Classification

How to evaluate your classification methods

- ▶ Going beyond classification error...
- ▶ What if we have class imbalance?
 - ▶ Ex. sample of 100 people and only 10 have the disease → always predict healthy and get 90% classification accuracy!!

Confusion Matrix

		<u>True class</u>	
		p	n
<u>Hypothesized class</u>	Y	True Positives	False Positives
	N	False Negatives	True Negatives
Column totals:	P	N	F-measure = $\frac{2}{\text{precision} + \text{recall}}$

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

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

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

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

Source: Fawcett (2005)

Confusion Matrix

		<u>True class</u>		ROC curve	
		p	n	fp rate = $\frac{FP}{N}$	tp rate = $\frac{TP}{P}$
<u>Hypothesized class</u>	Y	True Positives	False Positives	precision = $\frac{TP}{TP+FP}$	recall = $\frac{TP}{P}$
	N	False Negatives	True Negatives		
Column totals:		P	N	accuracy = $\frac{TP+TN}{P+N}$	F-measure = $\frac{2}{1/\text{precision}+1/\text{recall}}$

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

Confusion Matrix

		<u>True class</u>	
		p	n
<u>Hypothesized class</u>	Y	True Positives	False Positives
	N	False Negatives	True Negatives
Column totals:	P	N	$\text{fp rate} = \frac{FP}{N}$ $\text{tp rate} = \frac{TP}{P}$ precision-recall curve
			$\text{precision} = \frac{TP}{TP+FP}$ $\text{recall} = \frac{TP}{P}$
			$\text{accuracy} = \frac{TP+TN}{P+N}$
			$\text{F-measure} = \frac{2}{\text{1/precision}+\text{1/recall}}$

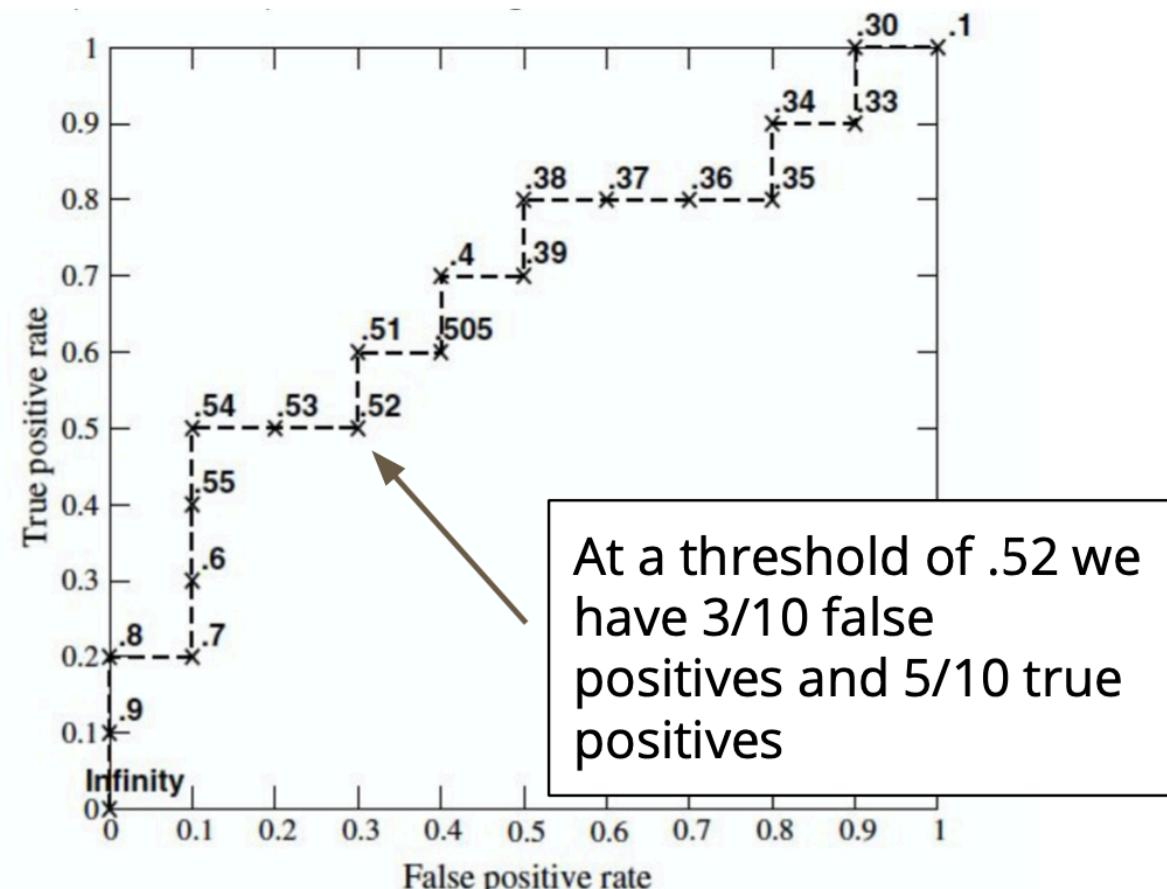
Fig. 1. Confusion matrix and common performance metrics calculated from it.

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

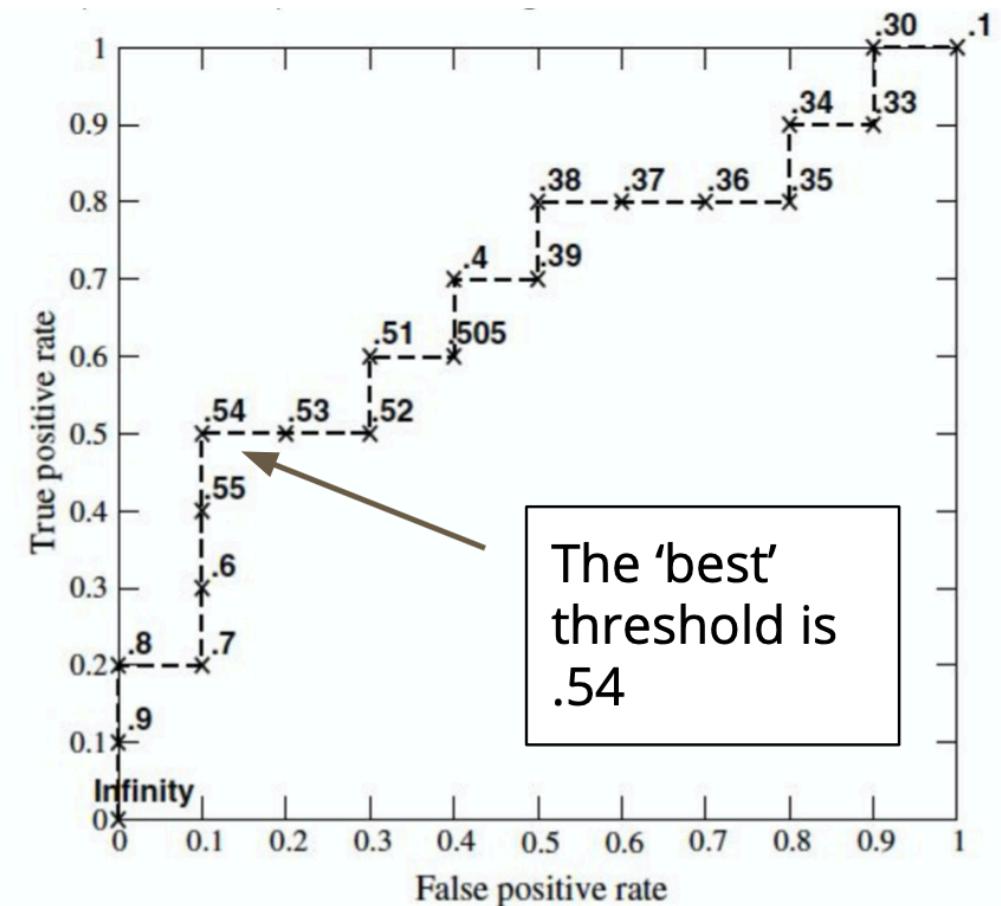
Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1



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)

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

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

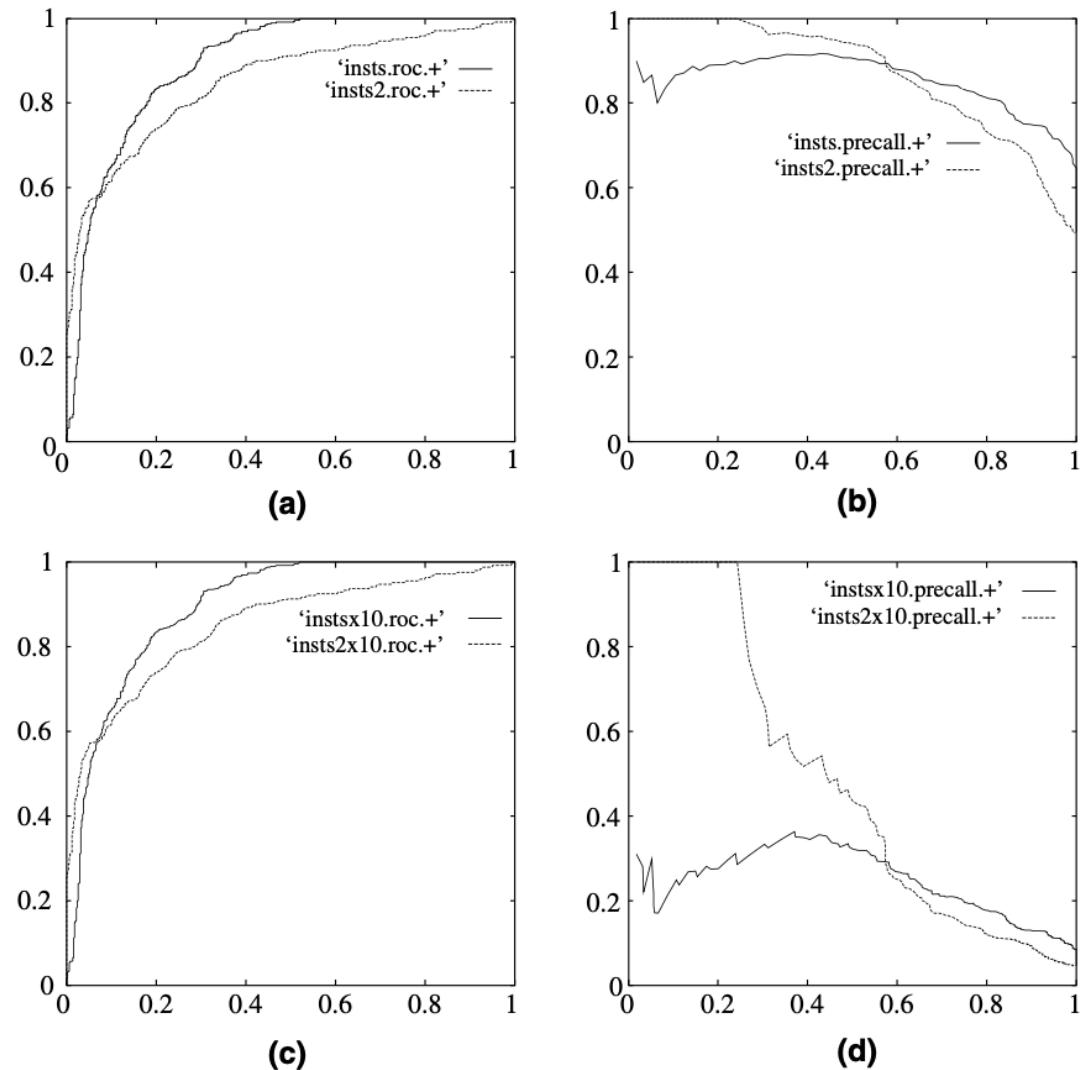


Fig. 5. ROC and precision-recall curves under class skew. (a) ROC curves, 1:1; (b) precision-recall curves, 1:1; (c) ROC curves, 1:10 and (d) precision-recall curves, 1:10.

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