Foundations of Data Science Lecture 5

Rumi Chunara, PhD

CS3943/9223

So Far…

- What is Data Science?
- Intro to R
- Data cleaning, sampling, processing
- Intro to ML – what is it
- Two Basic Algorithms
  - kNN
  - Linear Regression
- Time-series Analyses
  - Regression and lagged data in R

Today

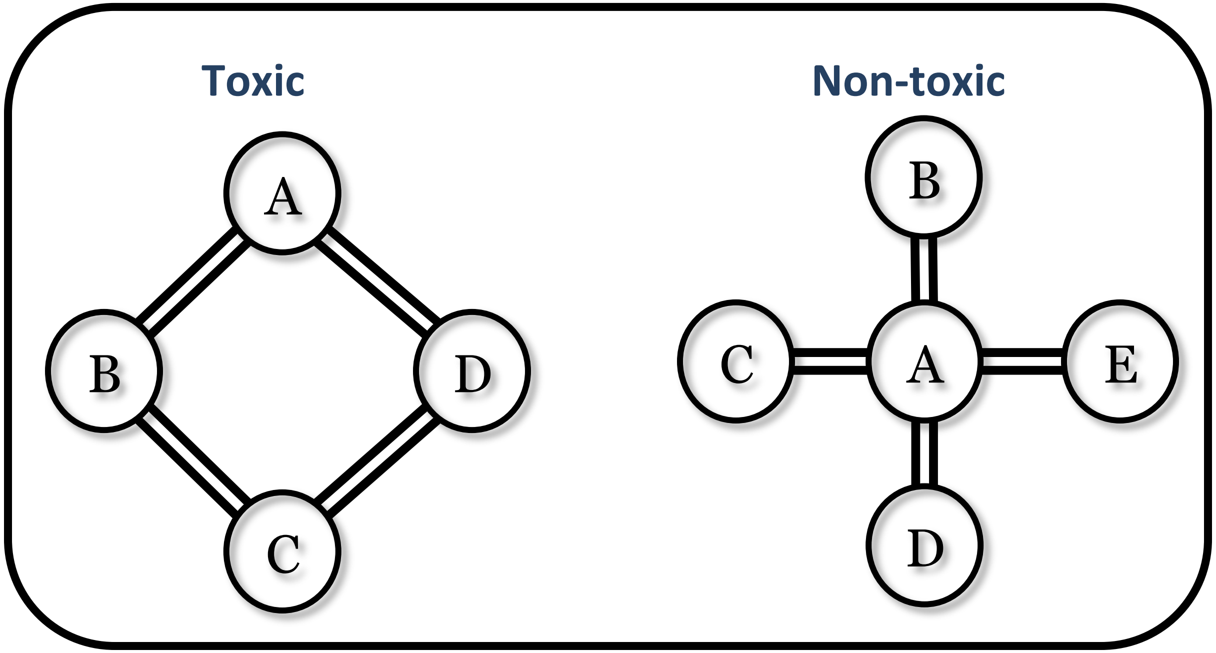
- Intro to Classification
  - Support Vector Machines

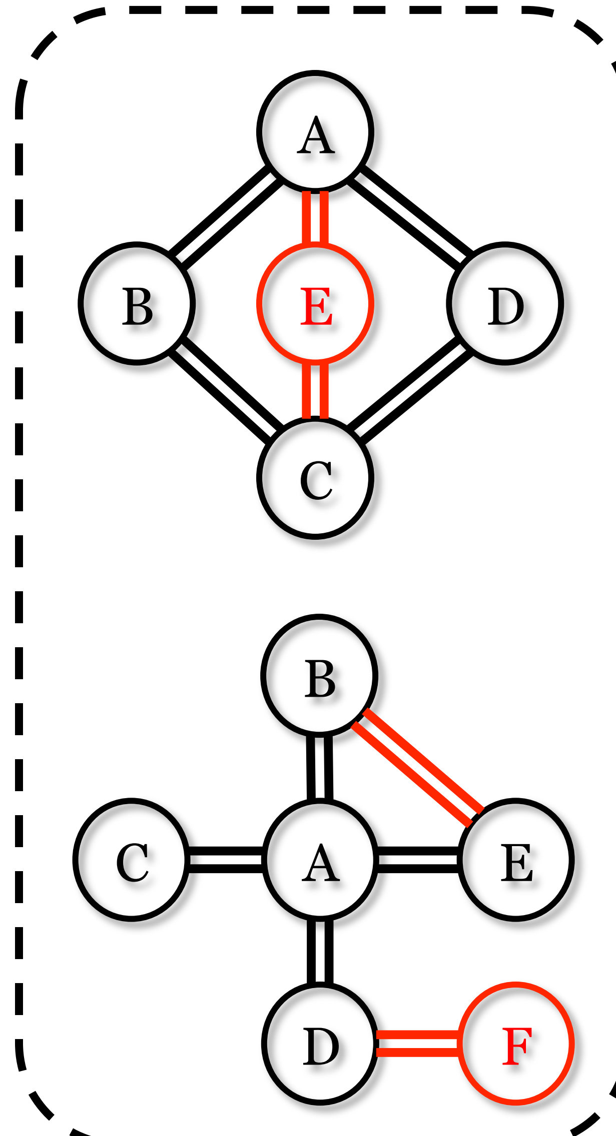
Thanks to: Andrew W. Moore, Professor School of Computer Science CMU

Example: Molecular Structures

Unknown

Known





Task: predict whether molecules are toxic, given set of known examples

Solution: Machine Learning

- Computationally *discover* and/or *predict* properties of interest of a set of data
- Two Approaches:
  - **Unsupervised**: discover discriminating properties among groups of data (Example: Clustering)
  - **Supervised**: known properties, categorize data with unknown properties (Example: Classification)

Introduction

- Support Vector Machines (SVMs) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis.
- A SVM model is a representation of the examples as points in space , mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

Why SVM?

- Easy to use
- Often has good generalization performance
- Same algorithm solves a variety of problems with little tuning.

Applications of SVM

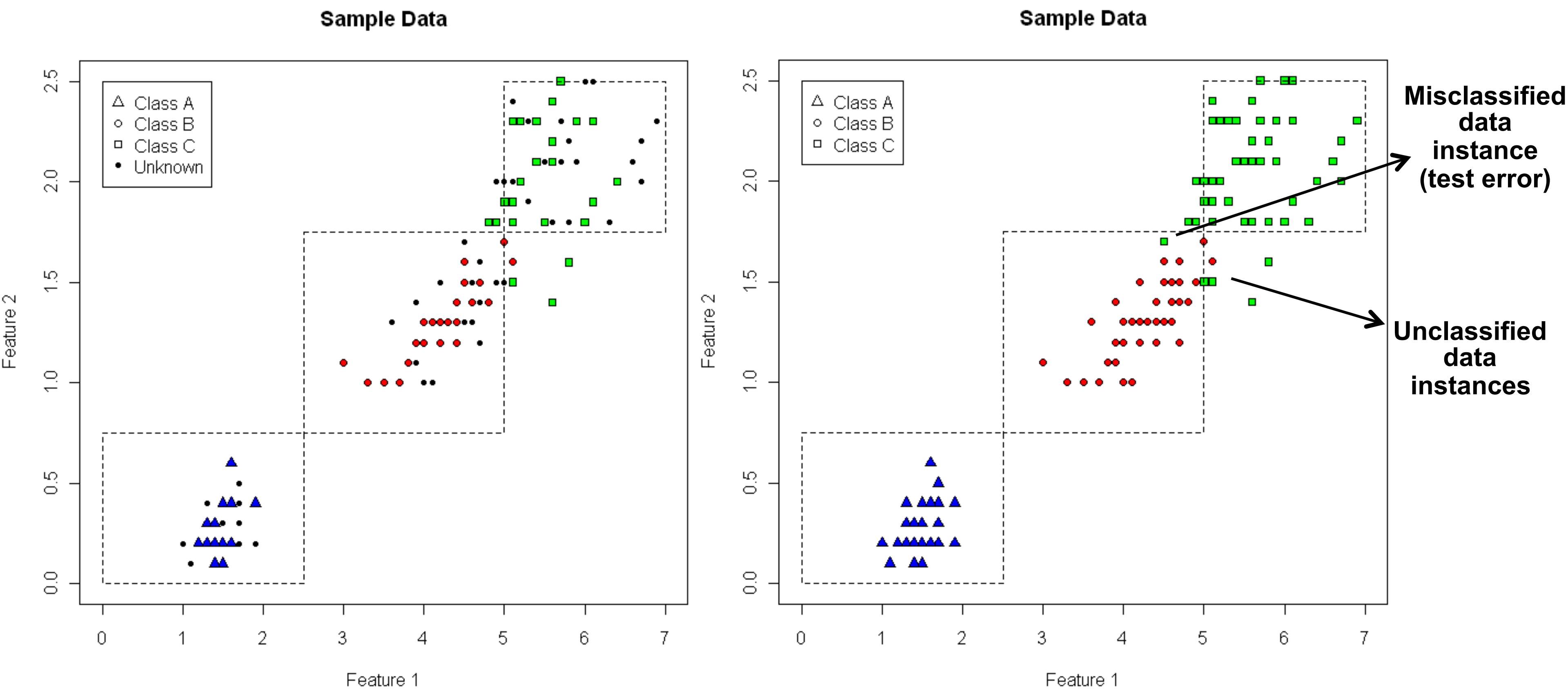
- Hand written characters can be recognized using SVM
- Used in medical science to classify proteins with up to 90% of the compounds classified correctly
- Text and image classification

Rough Set

- It is a mathematical tool to deal with un-integrality and uncertain knowledge
- It can effectively analyze and deal with all kinds of fuzzy, conflicting and incomplete information, and finds out the connotative knowledge from it, and reveals its underlying rules
- It finds the lower and upper approximation of the original set (given a pair of sets)
- Applications in many fields including data mining and artificial intelligence

Classification

- ***Classification***: **The task of assigning class labels in a discrete class label set Y to input instances in an input space X**
- **Ex: Y = {** ***toxic, non-toxic*** **}, X = {*****valid molecular structures*****}**

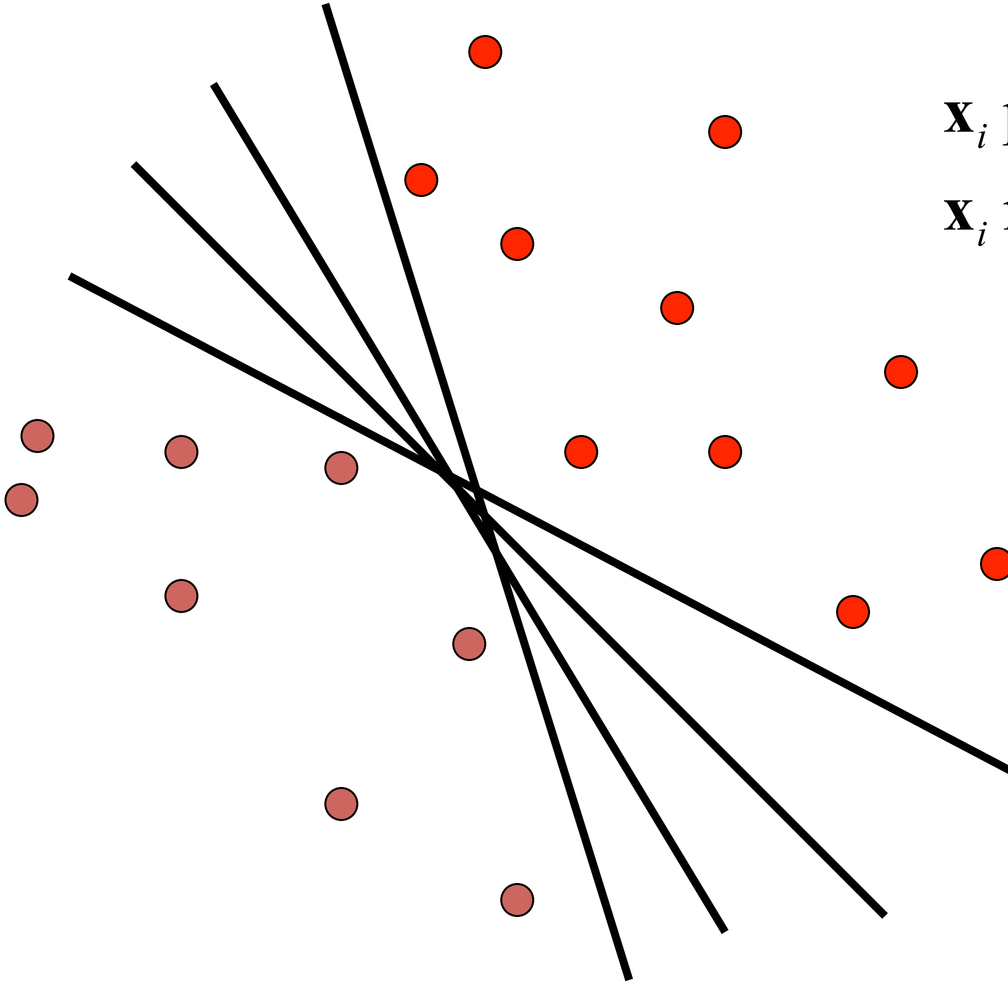


Training the classification model using the training data

Assignment of the unknown (test) data to appropriate class labels using the model

Linear classifiers

- Find a linear function (*hyperplane*) to separate positive and negative examples

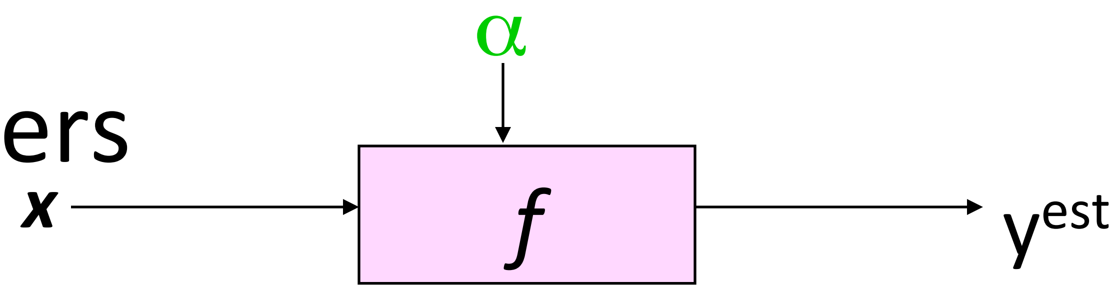


$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

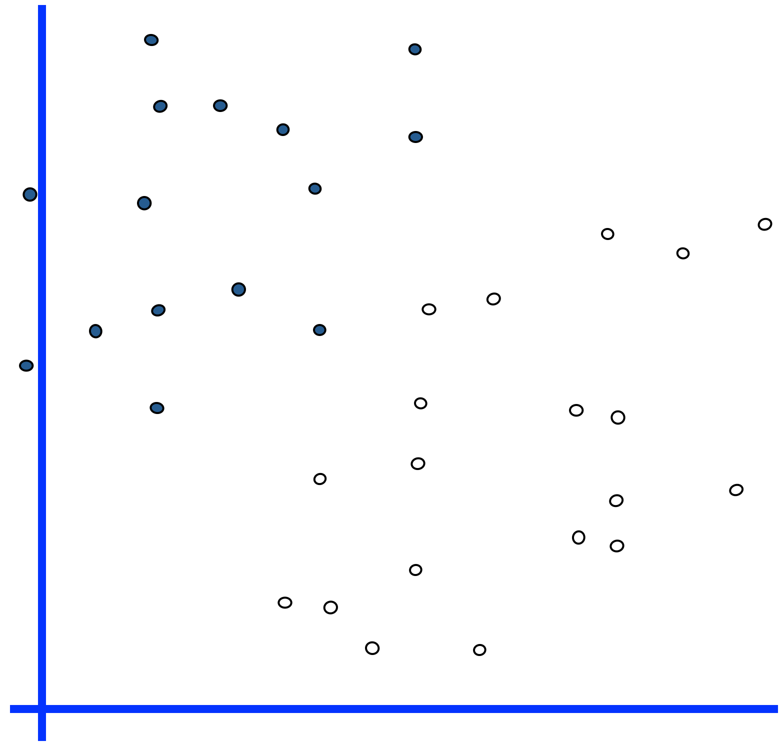
Which hyperplane is best?

Linear Classifiers



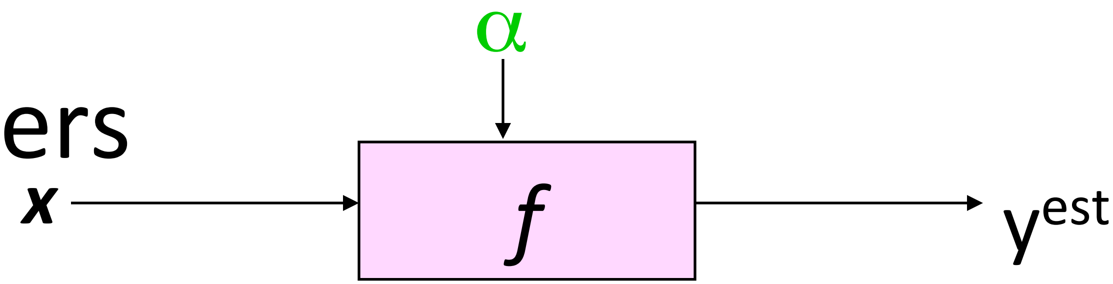
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- ● denotes +1
- ○ denotes -1

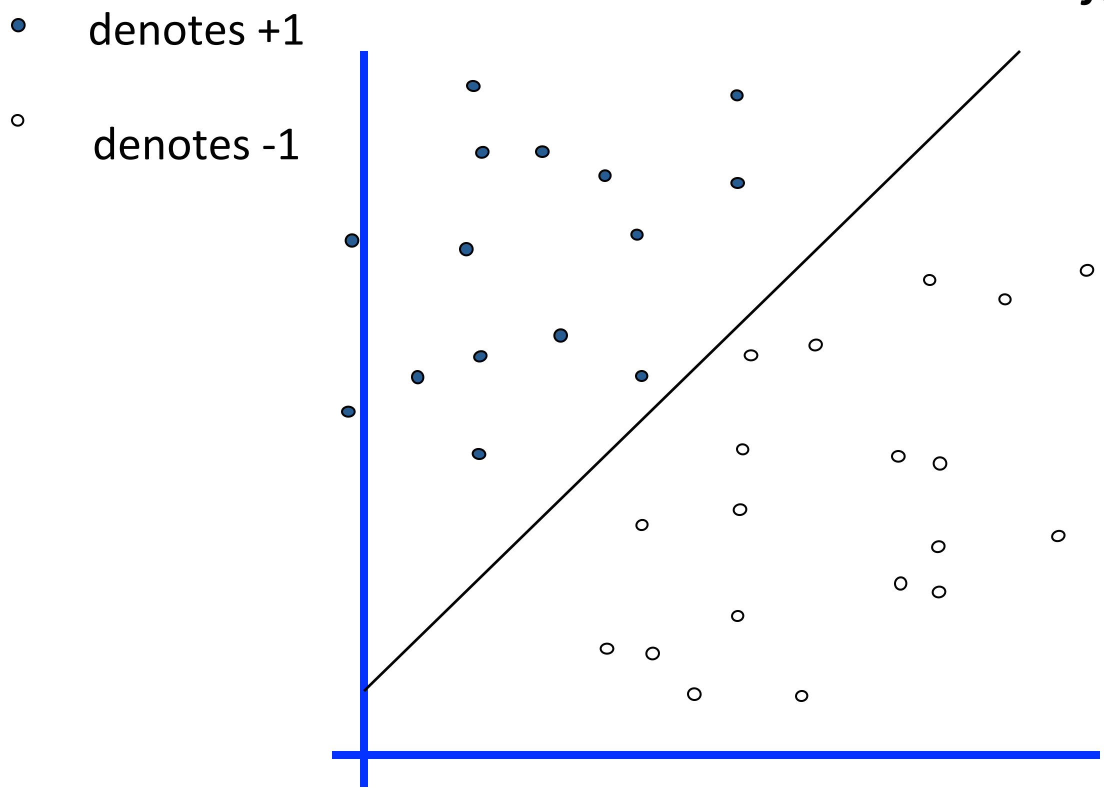


How would you classify this data?

Linear Classifiers

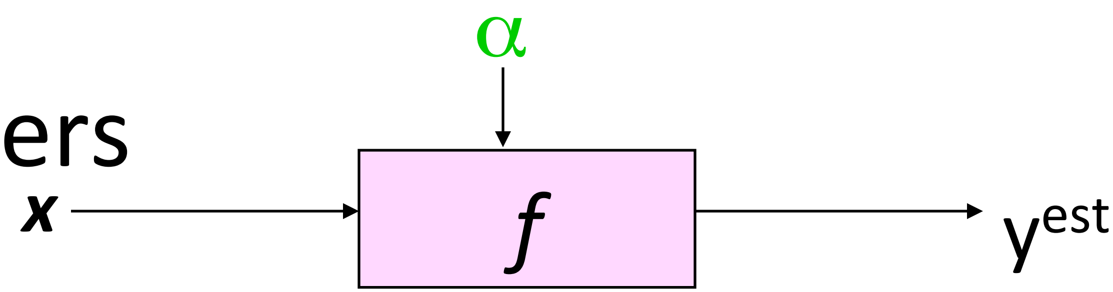


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$



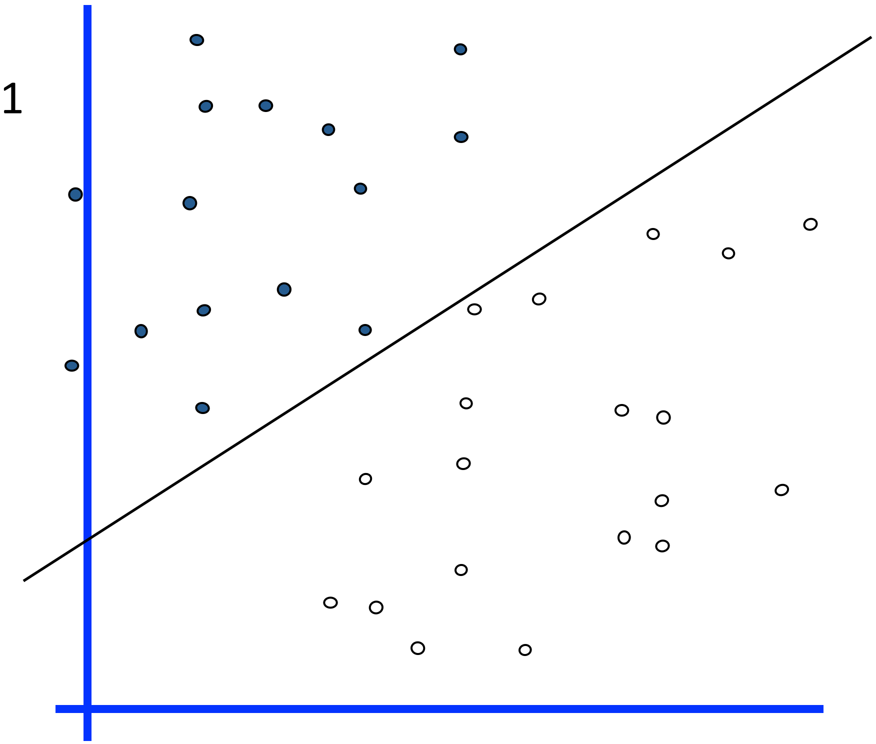
How would you classify this data?

Linear Classifiers



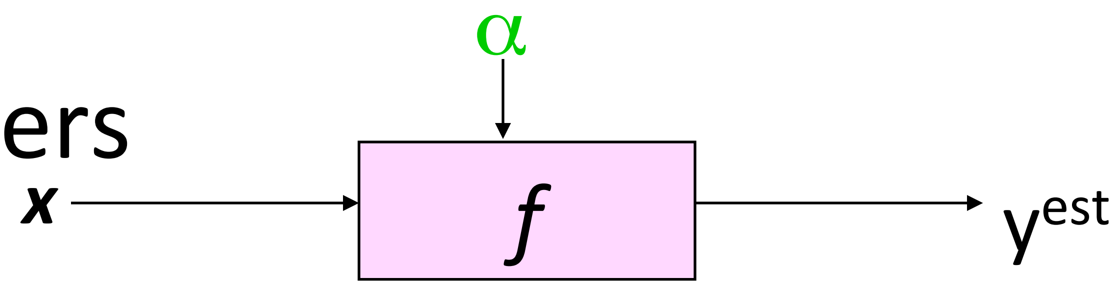
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\cdot \mathbf{x} - b)$

- denotes +1
- denotes -1



How would you classify this data?

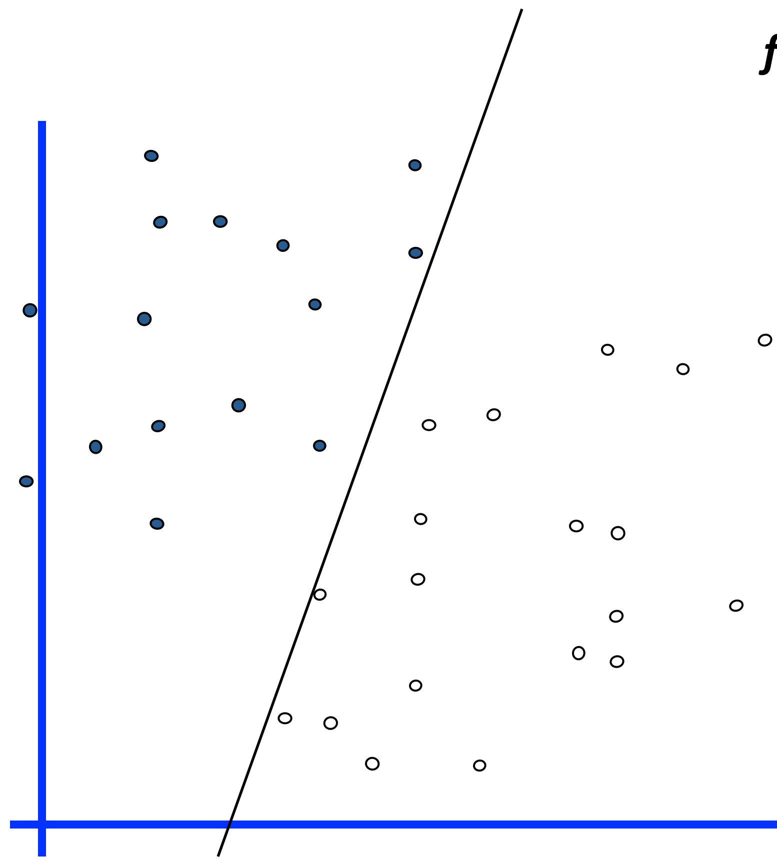
Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

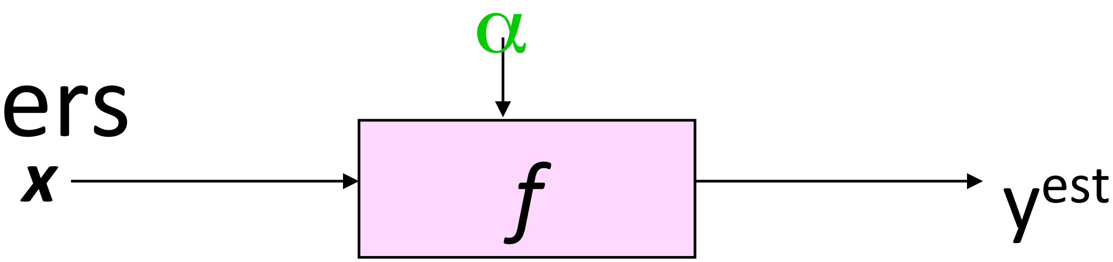
● denotes +1

○ denotes -1



How would you classify this data?

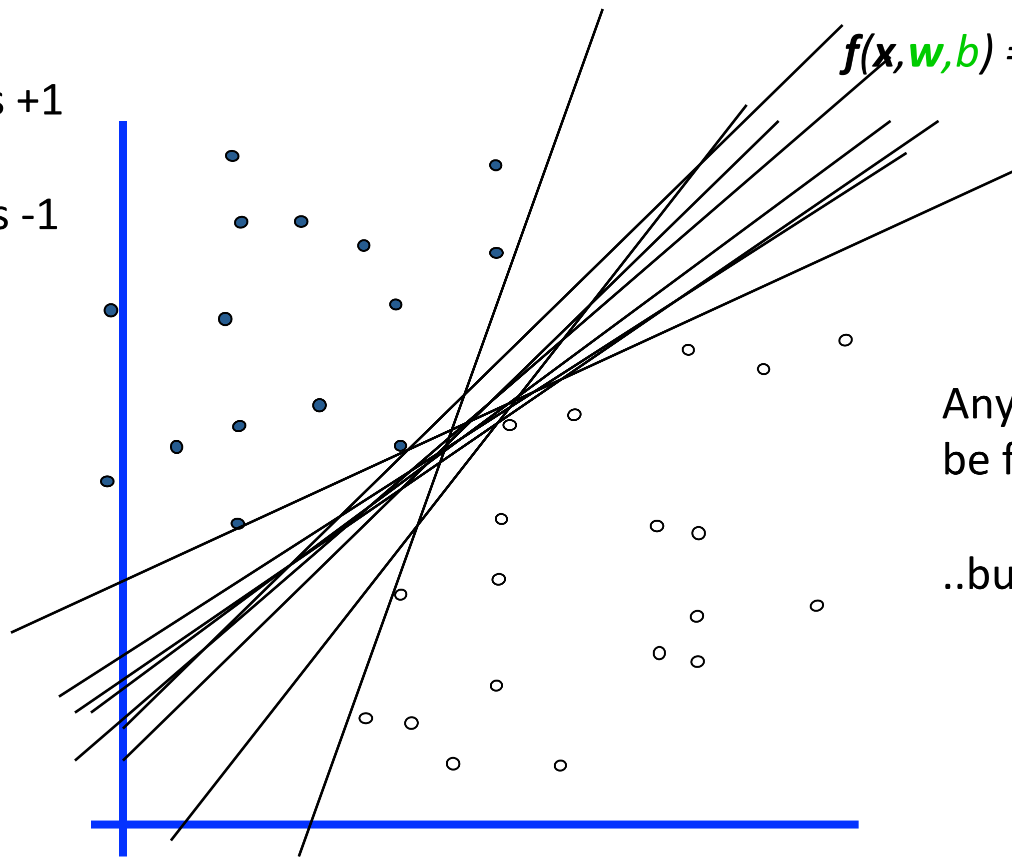
Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\cdot \mathbf{x} - b)$

● denotes +1

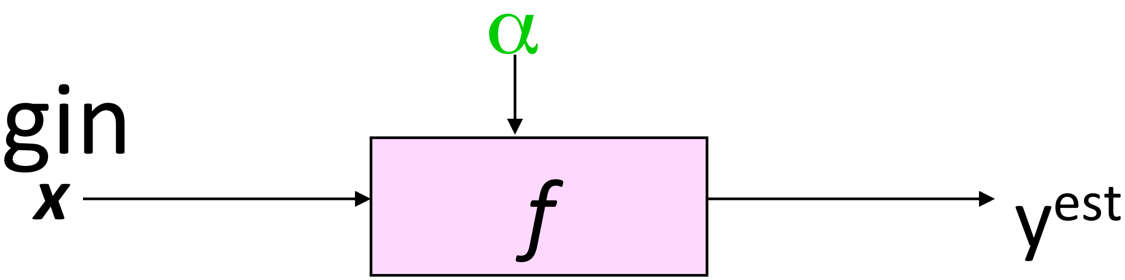
○ denotes -1



Any of these would be fine..

..but which is best?

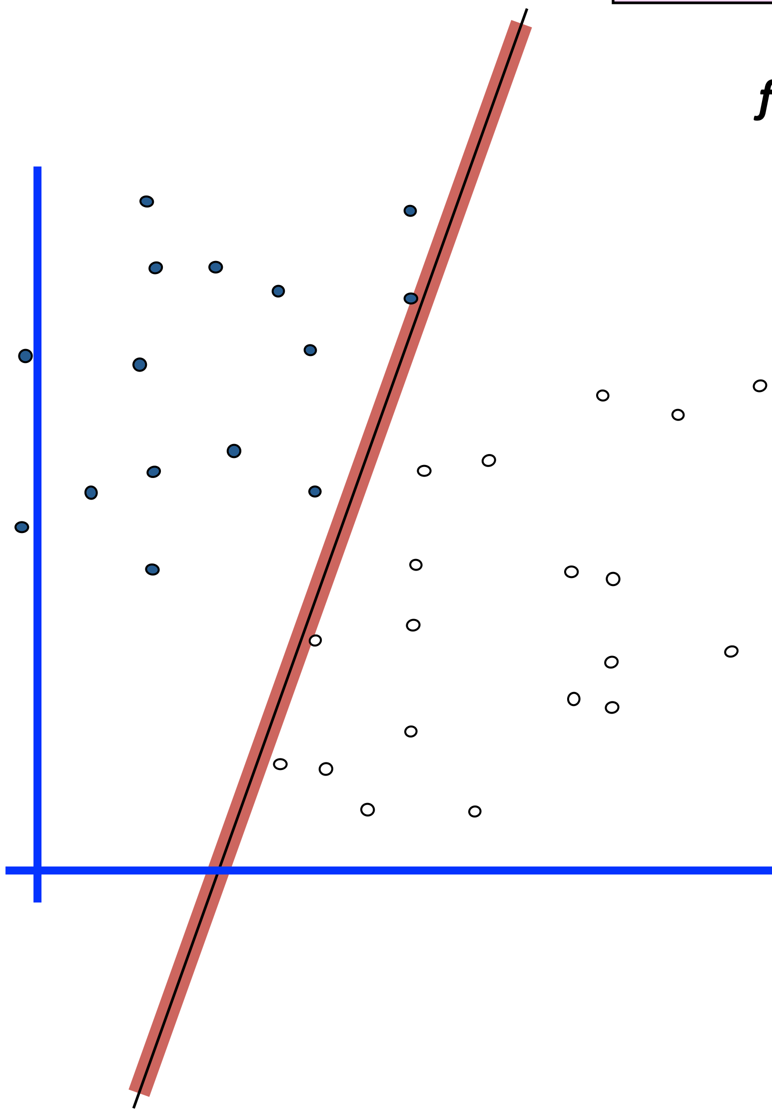
Classifier Margin



$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}.\ \boldsymbol{x} - b)$

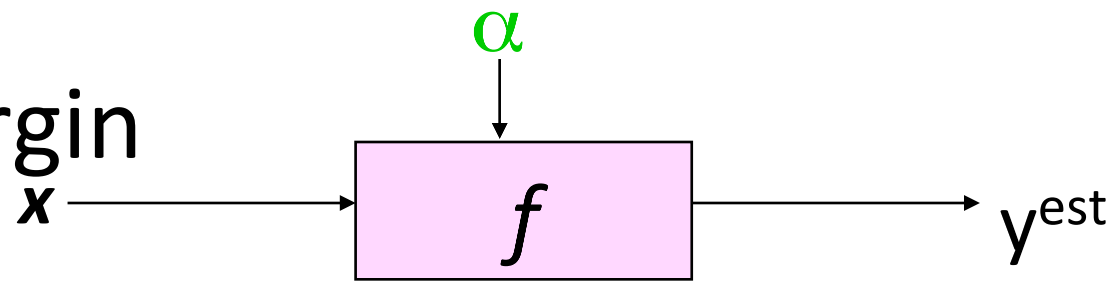
● denotes +1

○ denotes -1

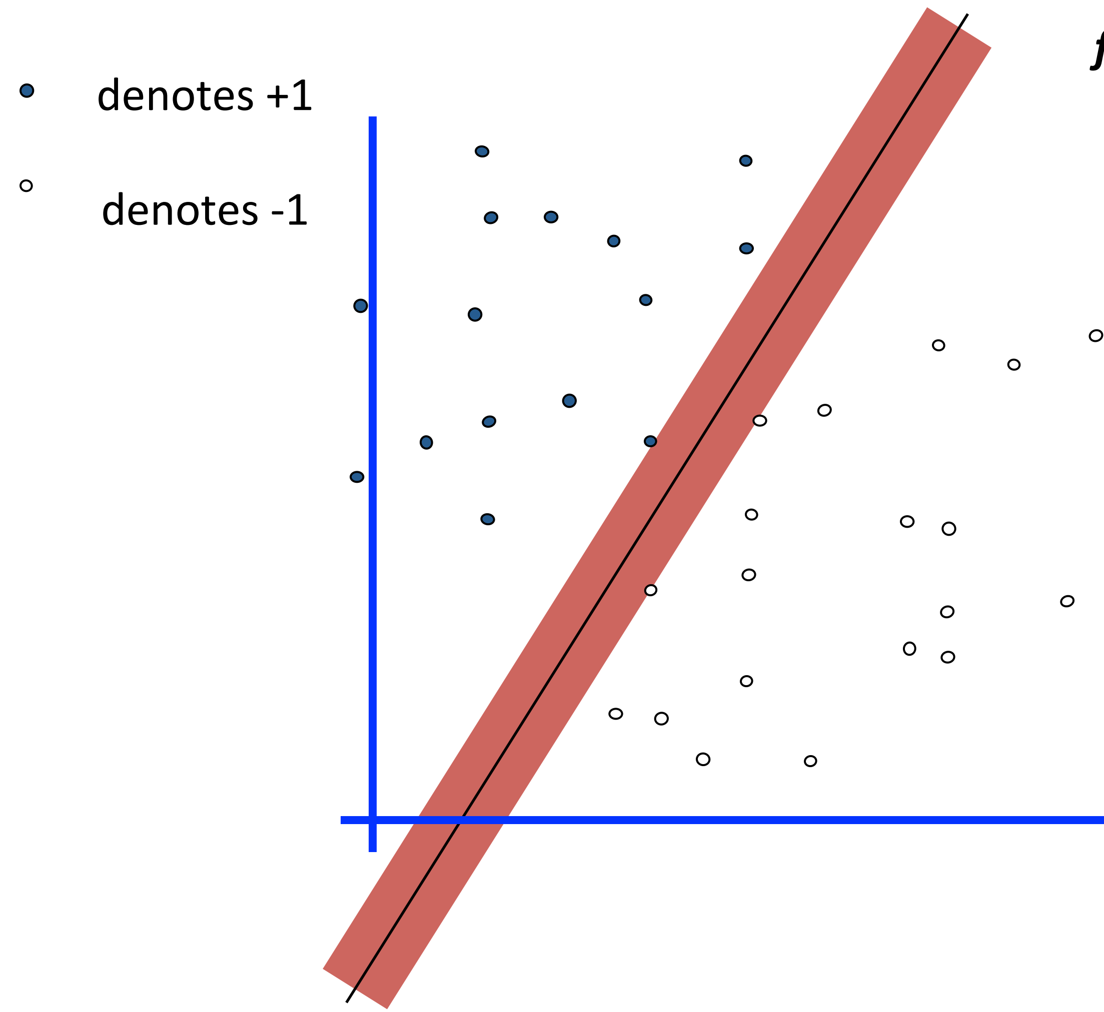


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



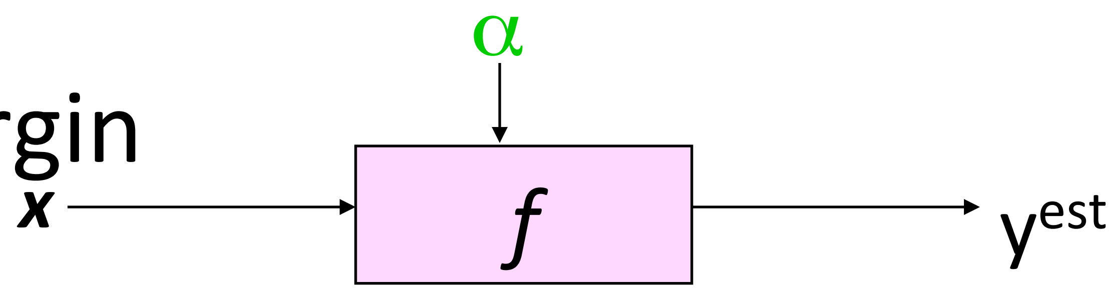
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$



The maximum margin linear classifier is the linear classifier with the, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

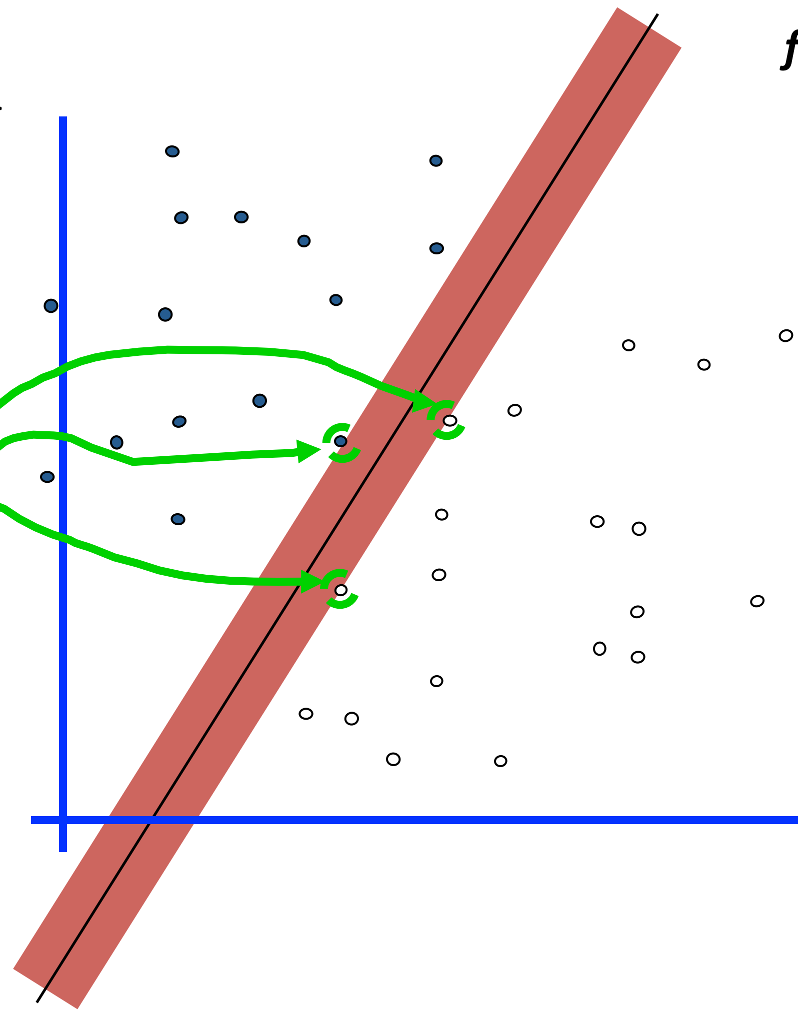
Maximum Margin



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

● denotes +1

○ denotes -1



Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

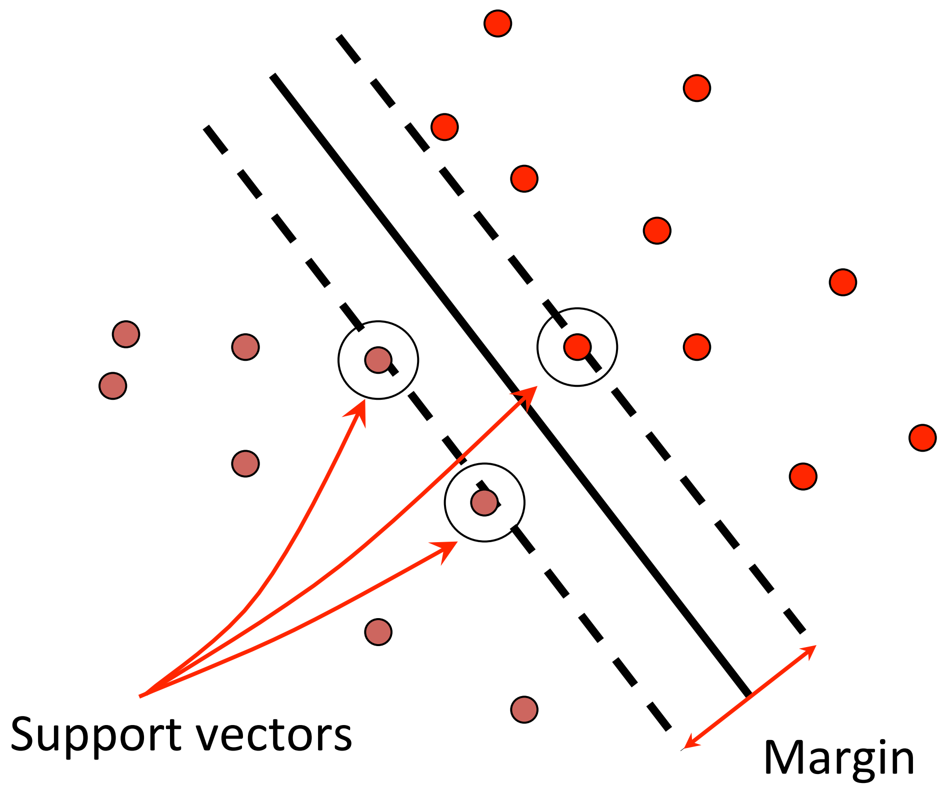
Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support vector machines

- Find a hyperplane that maximizes the *margin* between the positive and negative examples



$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\quad \dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the maximum margin hyperplane

1. Maximize the margin $2/\|\mathbf{w}\|$
2. Correctly classify all training examples:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \qquad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

- *Equivalent Quadratic programming problem*:

Maximize M
$\beta_0, \beta_1, \beta_2, \ldots, \beta_P$

Subject to $\sum_{j=1}^{p} \beta_j^2 = 1$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \qquad \forall i = 1, \ldots, n$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support Vector *Classification*

- Observations in two classes are not necessarily separable by a hyperplane
- Even if a separating plane exists, there are instances in which a classifier based on a hyperplane would not be desirable
- Thus consider a classifier based on a hyperplane that does *not* perfectly separate the two classes

Support Vector Classifiers

- Tuning parameter (or “regulization parameter”): *C*

Now problem is:

Maximize *M*

$$\beta_0, \beta_1, \beta_2, \ldots., \beta_P$$

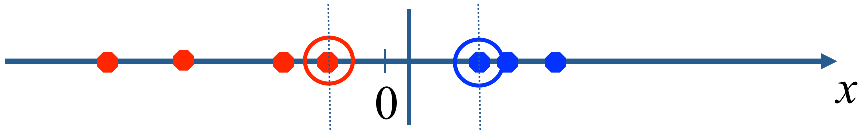
Subject to: $\sum_{j=1}^{p} \beta_j^2 = 1$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \varepsilon_i) \qquad \varepsilon_i \geq 0, \sum_{i=1}^{n} \varepsilon_i \leq C$$

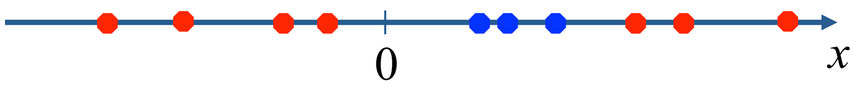
- Controls how much misclassification you will allow.
- For larger values of *C*, the optimization chooses a smaller-margin hyperplane *(desired)*

Nonlinear SVMs

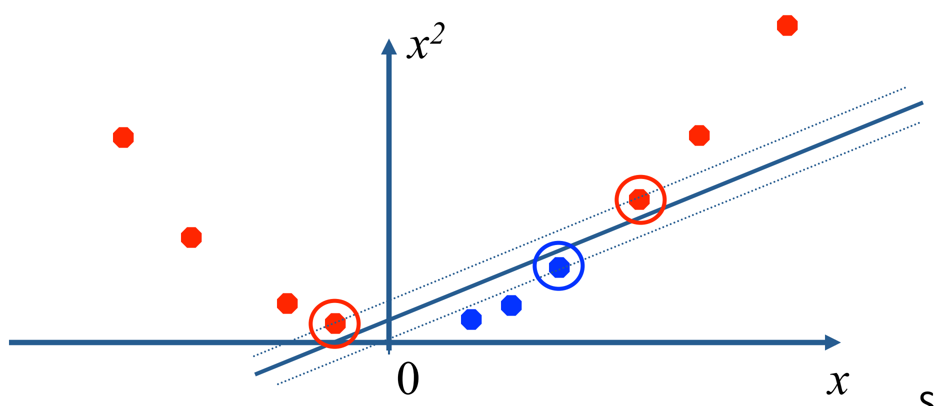
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



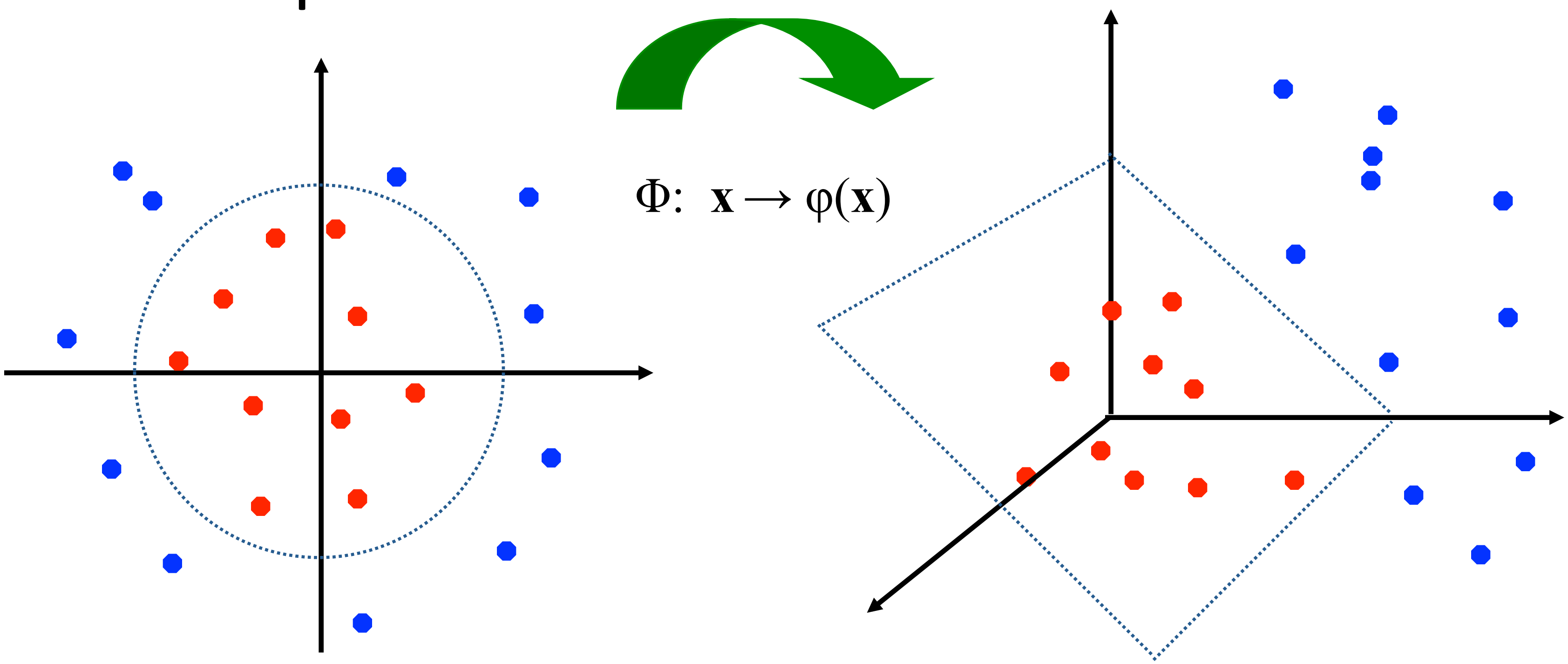
- We can map it to a higher-dimensional space:



Slide credit: Andrew Moore

Nonlinear SVMs

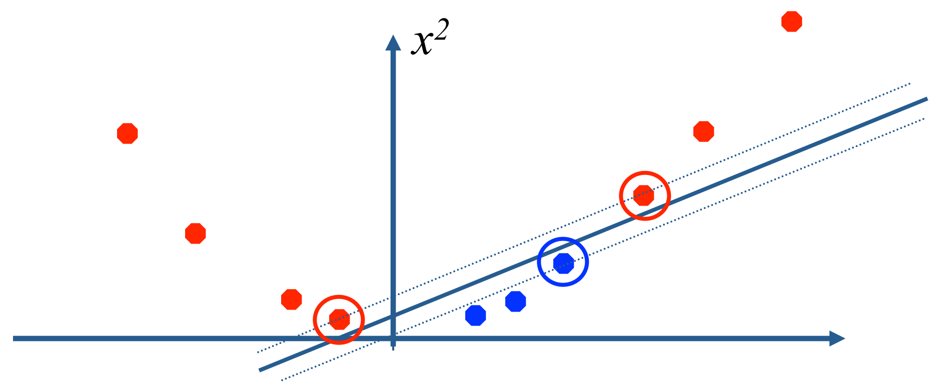
- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

Nonlinear kernel: Example

- Consider the mapping $\varphi(x) = (x, x^2)$



$$\varphi(x) \cdot \varphi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2$$

$$K(x, y) = xy + x^2 y^2$$

Polynomial kernel

Kernel Selection of SVM

- There are many kernel functions in SVM and selecting a good kernel function is a research issue.
- Popular kernel functions are:

Linear kernel : $K(x_i, x_j) = x_i^T x_j$

Polynomial kernel : $K(x_i, x_j) = (y x_i^T x_j + r)^d$, $y>0$

RBF kernel : $K(x_i, x_j) = \exp(-y||x_i - x_j||^2)$, $y>0$

Sigmoid kernel : $K(x_i, x_j) = \tanh(y x_i^T x_j + r)$

Model selection of SVM

- Model selection is also an important issue in SVM. Its success depends on the tuning of several parameters which affect the generalization error.
- If we use the linear SVM, we only need to tune the cost parameter C.
- As many problems are non-linearly separable, we need to select the cost parameter C and kernel parameters y, d.

What about multi-class SVMs?

- Unfortunately, there is no "definitive" multi-class SVM formulation
- In practice, we have to obtain a multi-class SVM by combining multiple binary (two-class) SVMs
- One vs. all
  - Training: learn an SVM for each class vs. the others
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- One vs. one
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM "votes" for a class to assign to the test example

SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages:
  - Kernel-based framework is very powerful, flexible
  - SVMs work very well in practice, even with very small training sample sizes

- Cons
  - No "direct" multi-class SVM, must combine two-class SVMs
  - Computation, memory
    - During training time, must compute a matrix of kernel values between all pairs of examples (training time >= O(n^2))
    - Training SVMs can take a very long time for large-scale problems (actually kernel SVMs can typically handle ~100K training examples on a single machine)

Data Imbalance

- If the amount of positive class samples differs greatly from the negative class in a dataset, then the feature of majority class will be much more and significant, but the feature of minority class will be very blur.
- Classifiers based on this kind of highly imbalance dataset will easily misclassify a new unknown minority sample to the majority class.

Addressing imbalance dataset

- Addressing imbalance dataset classification problem can be divided into two main directions:

- ***Sampling approaches*** – include methods that over-sample the minority class to match the size of the majority class and methods that under-sample the majority class to match the size of the minority class.
- ***Algorithmic-based*** – designed to improve a classifier's performance based on their inherent characteristics.

SVM algorithms for imbalance datasets

- ***Under sampling classification*** – It is similar to oversampling classification. It transfers imbalance dataset into balance dataset first and then use traditional SVM method. It randomly selects majority class samples to match minority class samples.
- ***Random classification*** – It uses cross-validation function to obtain training and testing datasets first and then adopt SVM train function on training dataset to build a classifier model.

Algorithm Evaluation measures

	Predicted Positive	**Predicted Negative**
Actual Positive	TP (True Positive)	FN (False Negative)
Actual Negative	FP (False Positive)	TN (True Negative)

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

F-measure = 2*Precision*Recall/(Precision + Recall)

Example

- Functional classifications of Yeast genes based on DNA microarray expression data.
- Training dataset
  - genes that are known to have the same function f
  - genes that are known to have a different function than f

Data files

- <u>Sample</u> (tab-delimited, same for testing)

gene	alpha_0X	alpha_7X	alpha_14X	alpha_21X	…
YMR300C	-0.1	0.82	0.25	-0.51	…
YAL003W	0.01	-0.56	0.25	-0.17	…
YAL010C	-0.2	-0.01	-0.01	-0.36	…
…					

- <u>Training data</u>

gene	Respiration_chain_complexes.mipsfc
YMR300C	-1
YAL003W	1
YAL010C	-1

Test data output

gene	classification	discriminant
YKL197C	-1	-3.349
YGL022W	-1	-4.682
YLR069C	-1	-2.799
YJR121W	1	0.7072

SVC in R

- There are (at least) five packages that implement SVM in R:
  - e1071
  - Kernlab
  - klaR
  - svmpath
  - Shogun

Implementation Steps

Find data, and portion with assigned labels e.g.

```
> y <- factor(c(1,-1,-1,1))
```

Input parameters like: kernal type, degree (if polynomial), regularization coefficient, number of labels

Training the support vector classifier

```
> model <- svm(x,y,type="C-classification")
```

Test the original data again using the output model

```
> predict(model,x)
```

R output

- Coefficients and intercept of hyperplane
- # support vectors

The Real World

- Gee, I'm building a text classifier for real, now!
- What should I do?

- How much training data do you have?
  - None
  - Very little
  - Quite a lot
  - A huge amount and its growing

Manually written rules

- No training data, adequate editorial staff?
- Never forget the hand-written rules solution!
  - If (wheat or grain) and not (whole or bread) then
    - Categorize as grain
- In practice, rules get a lot bigger than this
- With careful crafting (human tuning on development data) performance is high, but:
- Amount of work required is huge
  - Estimate days per class … plus maintenance

Very little data?

- If you're just doing supervised classification, you should stick to something high bias
  - There are theoretical results that Naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
- The interesting theoretical answer is to explore semi-supervised training methods:
  - Bootstrapping, EM over unlabeled documents, …
- The practical answer is to get more labeled data as soon as you can
  - How can you insert yourself into a process where humans will be willing to label data for you??

A reasonable amount of data?

- Perfect!
- We can use all our clever classifiers
- Roll out the SVM!

- But if you are using an SVM/NB etc., you should probably be prepared with the "hybrid" solution where there is a Boolean overlay
  - Or else to use user-interpretable Boolean-like models like decision trees
  - Users like to hack, and management likes to be able to implement quick fixes immediately

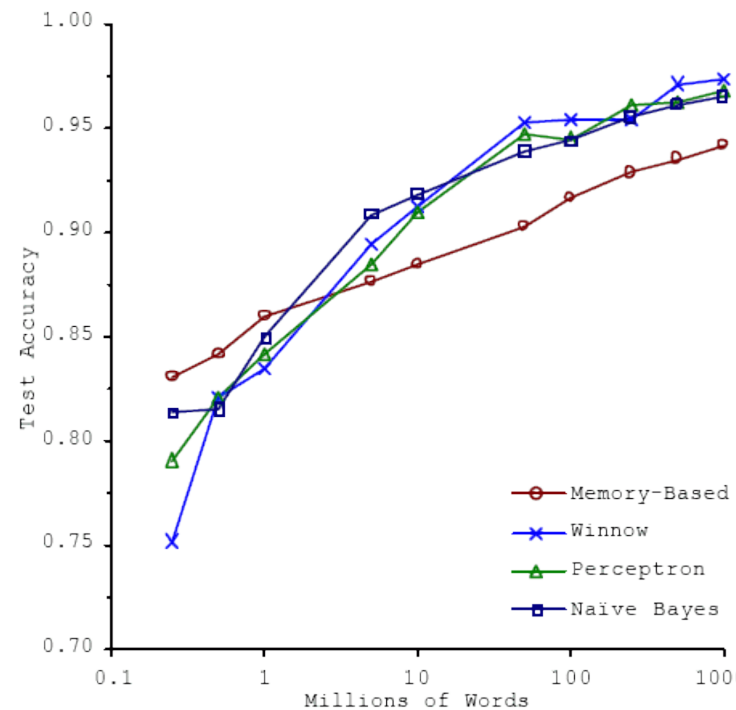
A huge amount of data?

- This is great in theory for doing accurate classification…
- But it could easily mean that expensive methods like SVMs (train time) or kNN (test time) are quite impractical

- Naïve Bayes can come back into its own again!
  - Or other advanced methods with linear training/test complexity like regularized logistic regression (though much more expensive to train)

Accuracy as a function of data size

- With enough data the choice of classifier may not matter much, and the best choice may be unclear
  - Data: Brill and Banko on context-sensitive spelling correction
- But the fact that you have to keep doubling your data to improve performance is a little unpleasant



How many categories?

- A few (well separated ones)?
  - Easy!
- A zillion closely related ones?
  - Think: Yahoo! Directory, Library of Congress classification, legal applications
  - Quickly gets difficult!
    - Classifier combination is always a useful technique
      - Voting, bagging, or boosting multiple classifiers
    - Much literature on hierarchical classification
      - Mileage fairly unclear, but helps a bit (Tie-Yan Liu et al. 2005)
    - May need a hybrid automatic/manual solution

How can one tweak performance?

- Aim to exploit any domain-specific useful features that give special meanings or that zone the data
  - E.g., an author byline or mail headers
- Aim to collapse things that would be treated as different but shouldn't be.
  - E.g., part numbers, chemical formulas

- Does putting in "hacks" help?
  - You bet!
    - Feature design and non-linear weighting is *very* important in the performance of real-world systems

Upweighting

- You can get a lot of value by differentially weighting contributions from different document zones:
- That is, you count as two instances of a word when you see it in, say, the abstract
  - Upweighting title words helps (Cohen & Singer 1996)
    - Doubling the weighting on the title words is a good rule of thumb
  - Upweighting the first sentence of each paragraph helps (Murata, 1999)
  - Upweighting sentences that contain title words helps (Ko *et al,* 2002)

Text Summarization techniques in text classification

- Text Summarization: Process of extracting key pieces from text, normally by features on sentences reflecting position and content
- Much of this work can be used to suggest weightings for terms in text categorization
    - See: Kolcz, Prabakarmurthi, and Kalita, CIKM 2001: Summarization as feature selection for text categorization
  - Categorizing purely with title,
  - Categorizing with first paragraph only
  - Categorizing with paragraph with most keywords
  - Categorizing with first and last paragraphs, etc.

Measuring Classification Figures of Merit

- Not just accuracy; in the real world, there are economic measures:
  - Your choices are:
    - Do no classification
      - That has a cost (hard to compute)
    - Do it all manually
      - Has an easy-to-compute cost if doing it like that now
    - Do it all with an automatic classifier
      - Mistakes have a cost
    - Do it with a combination of automatic classification and manual review of uncertain/difficult/"new" cases
  - Commonly the last method is most cost efficient and is adopted

A common problem: Concept Drift

- Categories change over time
- Example: “president of the united states”
  - 1999: clinton is great feature
  - 2010: clinton is bad feature
- One measure of a text classification system is how well it protects against concept drift.
  - Favors simpler models like Naïve Bayes
- Feature selection: can be bad in protecting against concept drift

Rules of thumb

- Radial basis kernel usually performs better.
- Scale your data. scale each attribute to [0,1] or [-1,+1] to avoid over-fitting.
- Try different penalty parameters C for two classes in case of unbalanced data.

Why RBF kernel?

- RBF is the main kernel function because of the following reasons
- The RBF kernel nonlinearly maps samples into a higher dimensional space unlike to linear kernel.
- The RBF kernel has less hyper parameters than the polynomial kernel.
- The RBF kernel has less numerical difficulties.

Summary: Classification, so far

- Nearest-neighbor and k-nearest-neighbor classifiers
  - Euclidian distance, Cosine distance, Jaccard distance, etc.
- Support vector machines
  - Linear classifiers
  - Margin maximization
  - The kernel trick
  - Multi-class
- Of course, there are many other classifiers out there
  - Neural networks, boosting, decision trees, …
  - Deep neural networks, convolutional neural networks: jointly learning feature representation and classifiers
- Real world: exploit domain specific structure!