UVA CS 6316/4501 – Fall 2016 Machine Learning

Lecture 10: Supervised Classification with Support Vector Machine

Dr. Yanjun Qi

University of Virginia

Department of
Computer Science

Where are we ? → Five major sections of this course

- ❑ ~~Regression (supervised)~~
- ❑ Classification (supervised)
- ❑ Unsupervised models
- ❑ Learning theory
- ❑ Graphical models

Today

- ❑ Supervised Classification
- ❑ Support Vector Machine (SVM)

e.g. SUPERVISED LEARNING

- Find function to map input space X to output space Y

$$f : X \longrightarrow Y$$

- So that the difference between *y* and *f(x)* of each example ***x*** is small.

e.g.

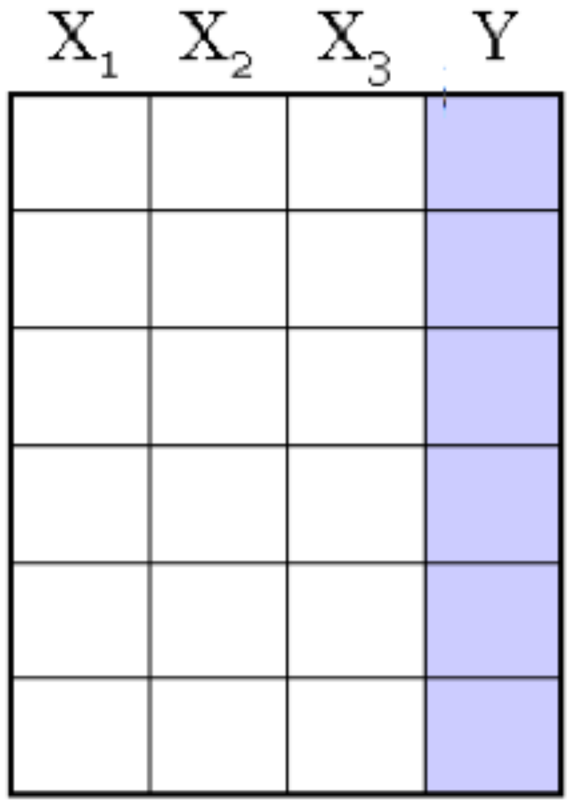
x	I believe that this book is not at all helpful since it does not explain thoroughly the material . it just provides the reader with tables and calculations that sometimes are not easily understood …

→

y	-1

Input X : e.g. a piece of English text

Output Y: {1 / Yes , -1 / No }
e.g. Is this a positive product review ?



A Dataset for classification

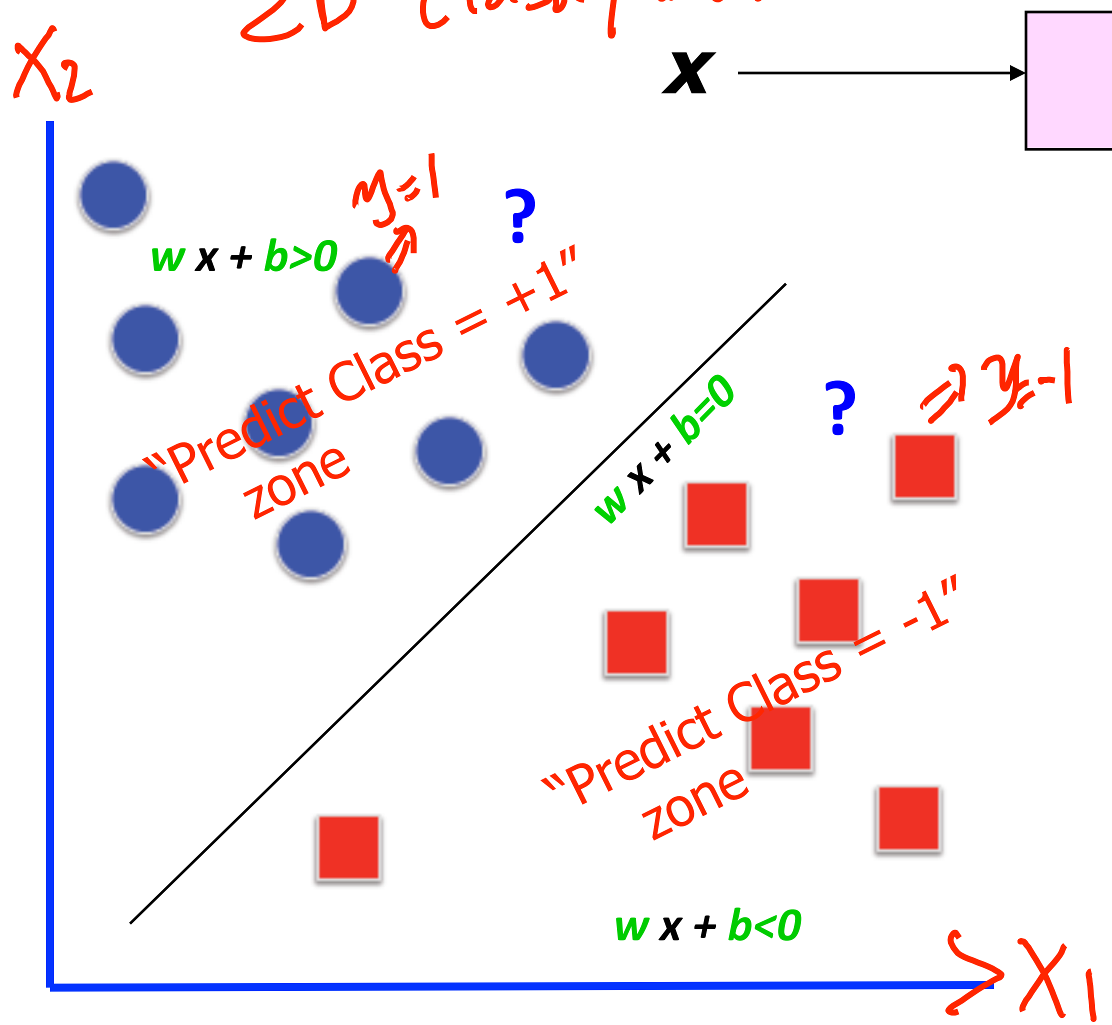
$$f : \boxed{X} \longrightarrow \boxed{Y}$$

Output Class: categorical variable

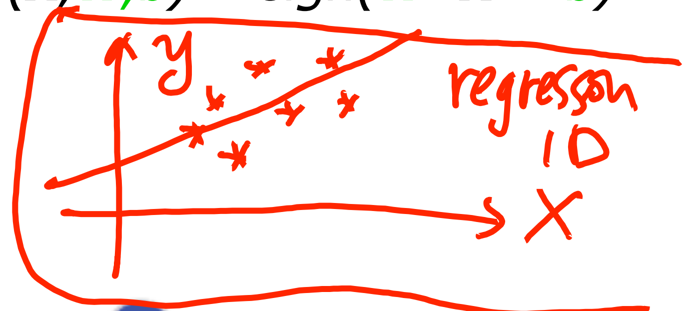
- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

e.g. SUPERVISED Linear Binary Classifier

2D classification



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}^T\mathbf{x} + b)$



denotes +1 point

denotes -1 point

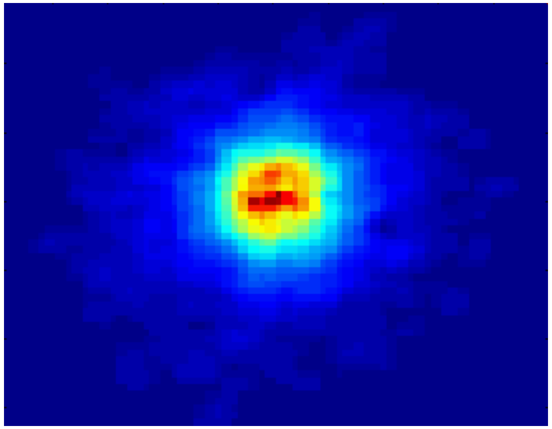
? denotes future points

Courtesy slide from Prof. Andrew Moore's tutorial

Application 1: Classifying Galaxies

Courtesy: http://aps.umn.edu

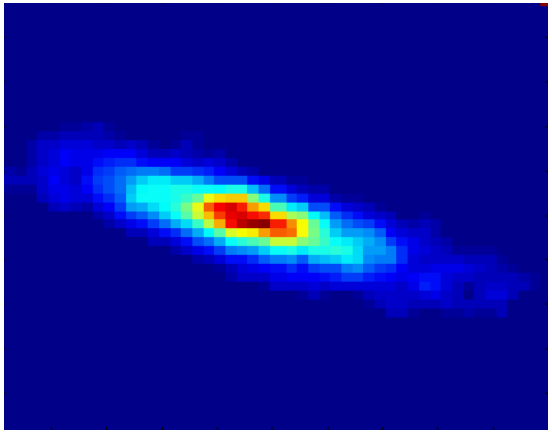
Early



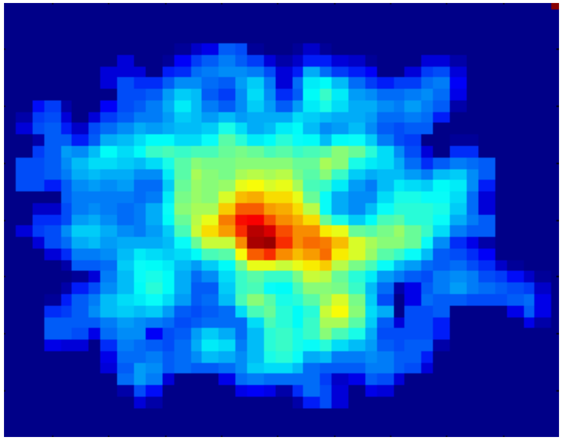
Class:
- Stages of Formation

Attributes:
- Image features,
- Characteristics of light waves received, etc.

Intermediate



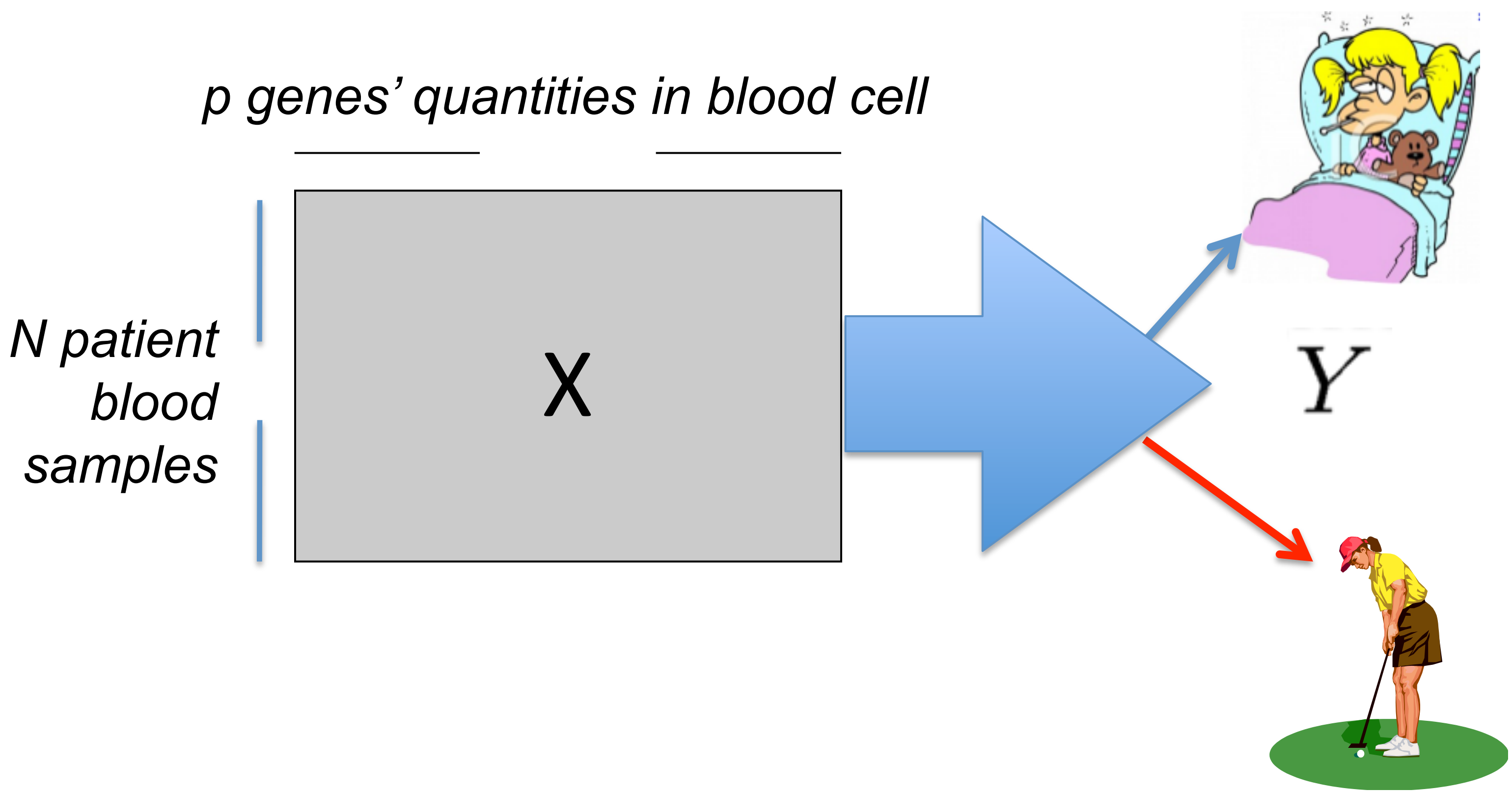
Late



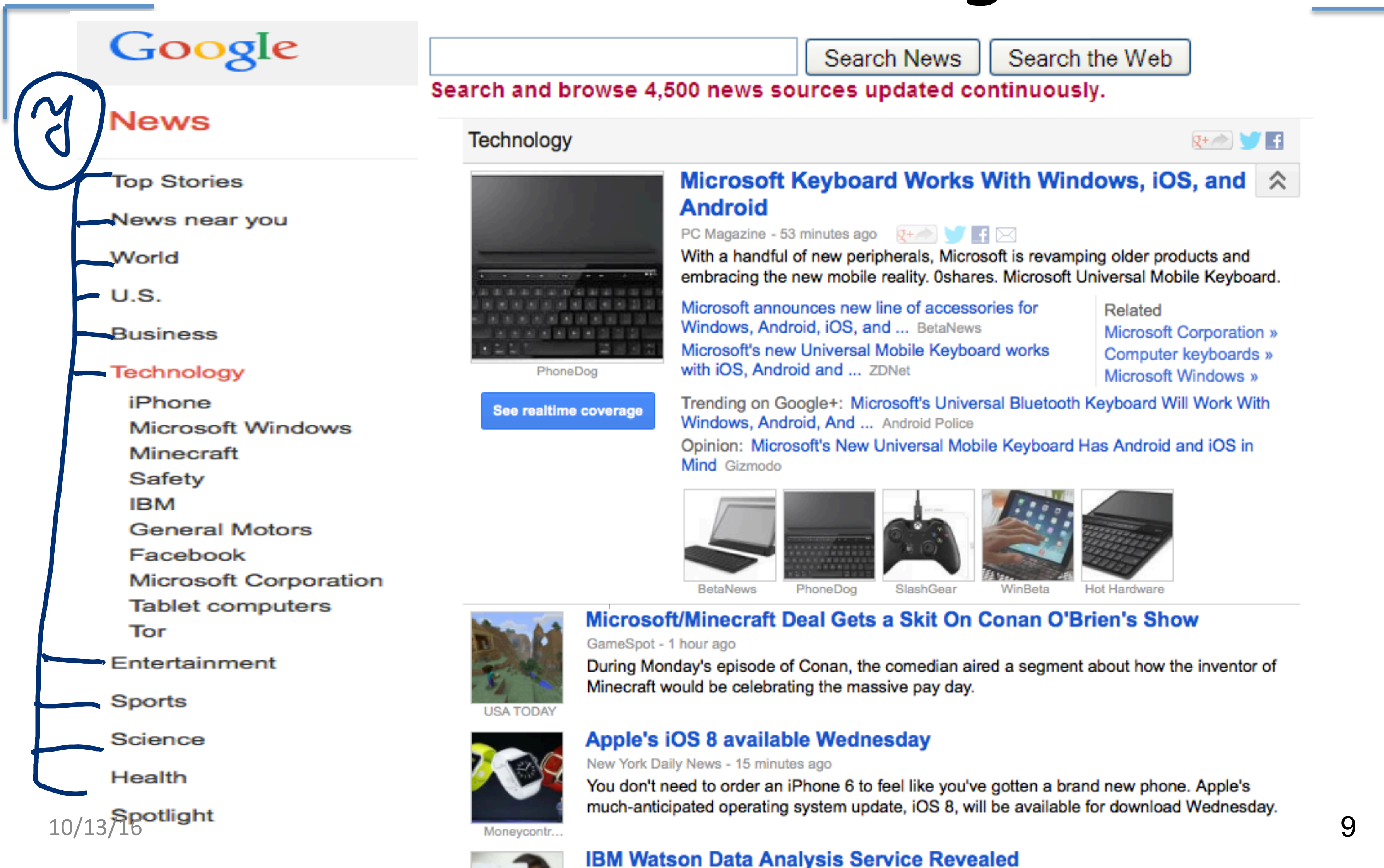
Data Size:
- 72 million stars, 20 million galaxies
- Object Catalog: 9 GB
- Image Database: 150 GB

From [Berry & Linoff] Data Mining Techniques, 1997

Application 2: **Cancer Classification using gene expression**



Application 3: – Text Documents, e.g. *Google News*



Text Document Representation

[word]

- Each document becomes a `term' vector,
  - each term is an (attribute) of the vector,
  - the value of each describes the number of times the corresponding term occurs in the document.

w_1 w_2 .... w_D

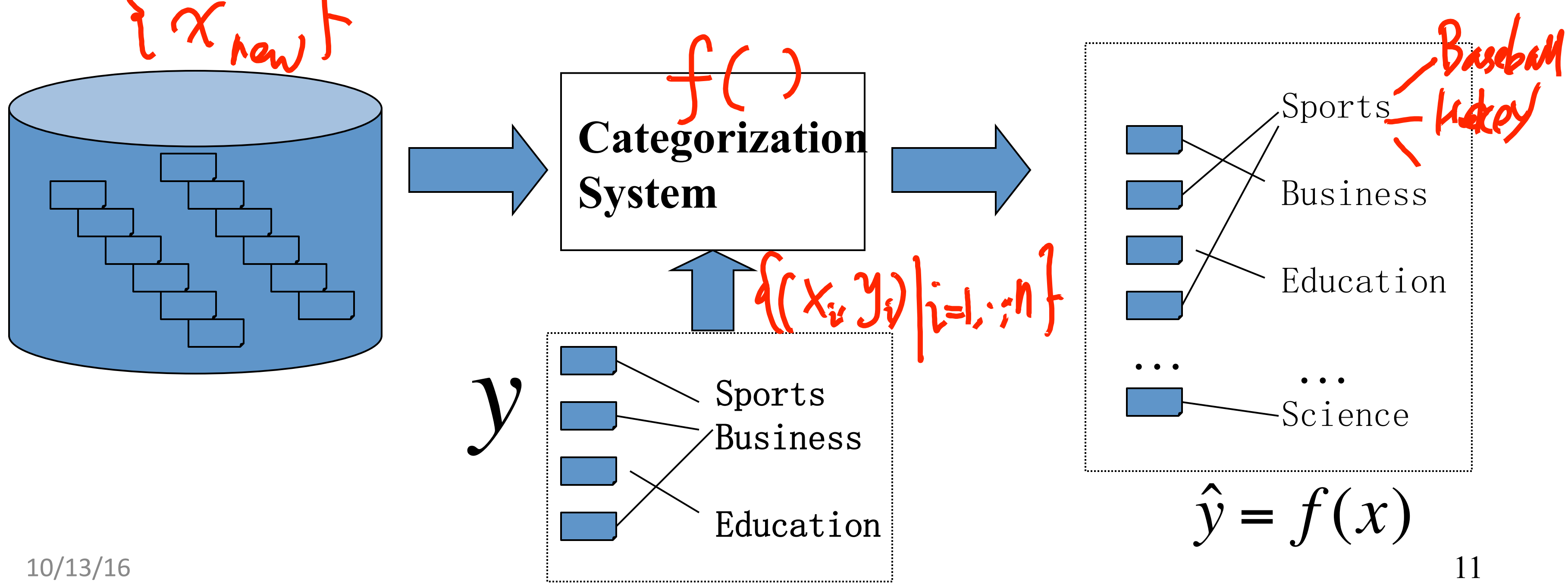
Bag of 'words'

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

Text Categorization

$\{y_{tree}\}$

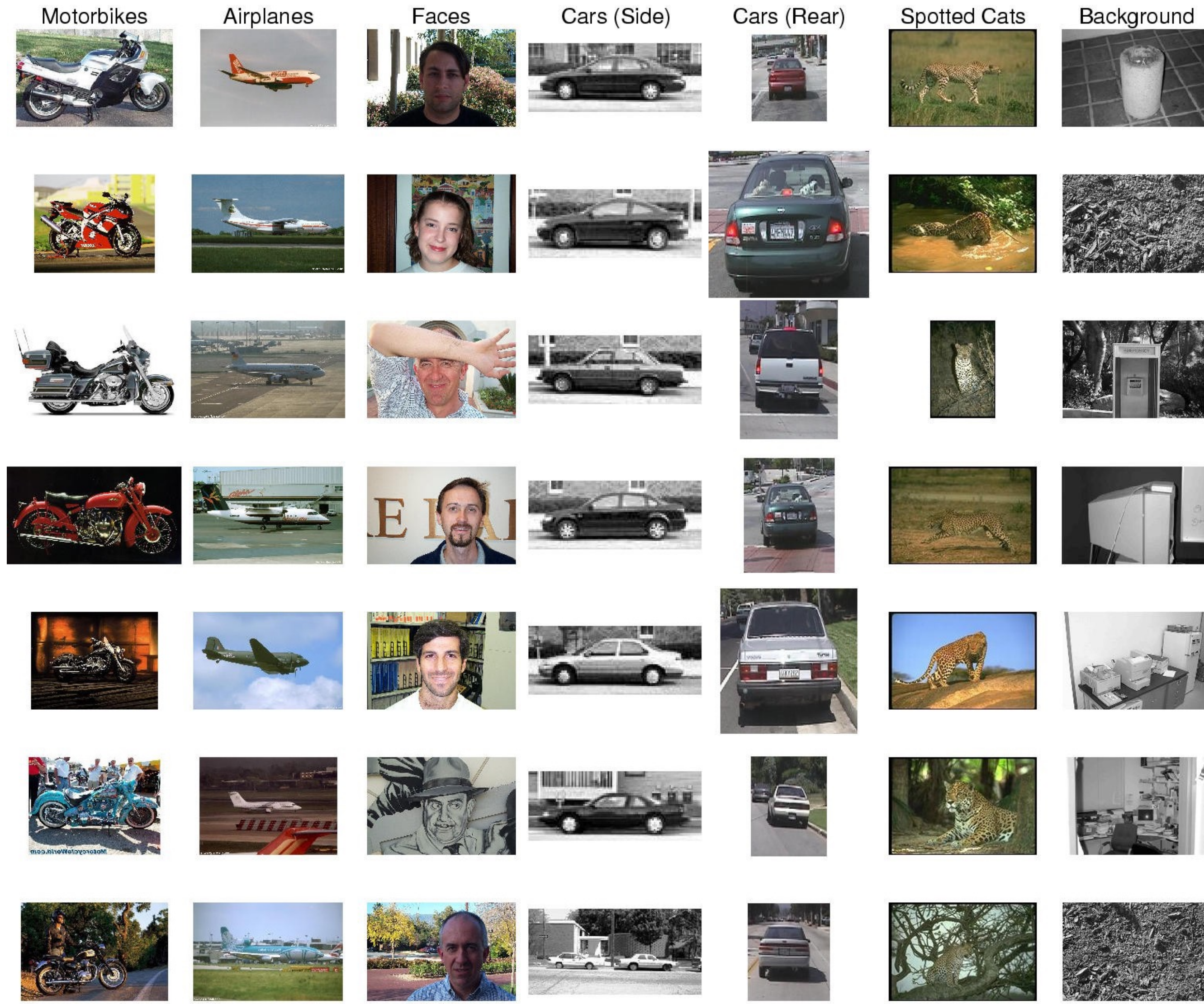
- Pre-given categories and labeled document examples (Categories may form hierarchy)
- Classify new documents
- A standard supervised learning problem



Examples of Text Categorization

- News article classification
- Meta-data annotation
- Automatic Email sorting
- Web page classification

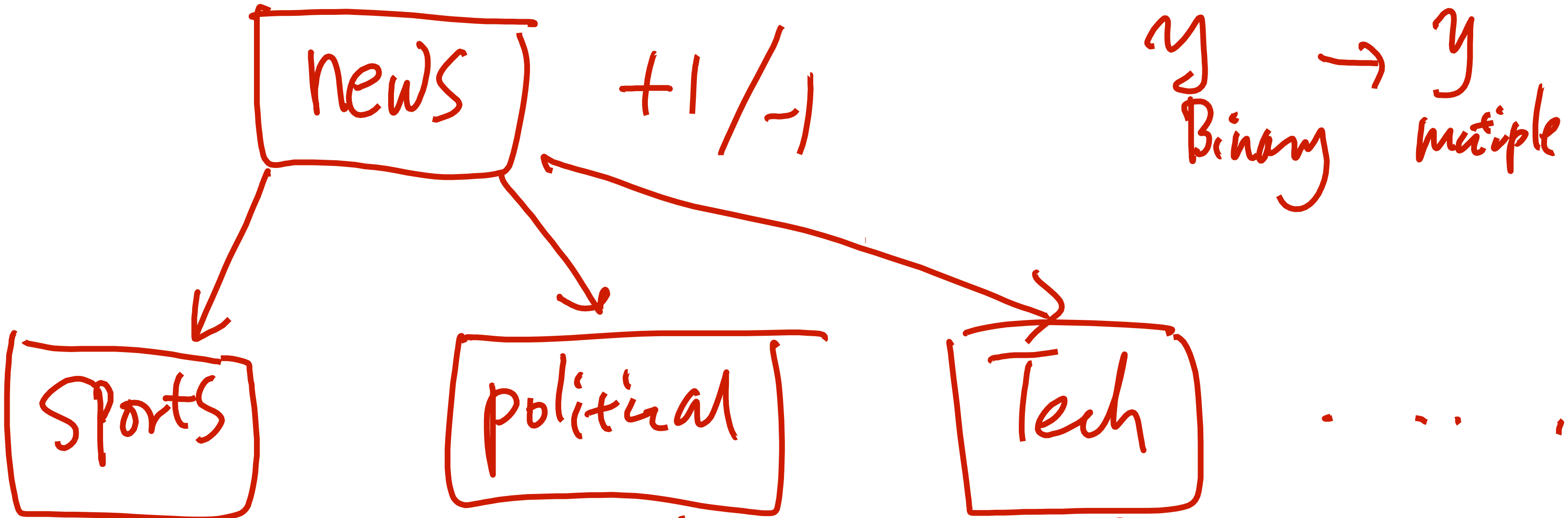
Application 4: – Objective recognition / Image Labeling (Label Images into predefined classes)



$\{x, \{y_{tree}\}\}$

Hierachical supervised classification

$y_{\text{Binary}} \rightarrow y_{\text{multiple}} \rightarrow y_{\text{tree}}$



$f_1(\ )$ $f_2(\ )$

① $X \rightarrow$ news $+1/-1$ $\rightarrow$ Tech $+/-$

② $X \rightarrow$ Tech news $+/1$

$f_3(\ )$

LATER

Image Representation for – Objective recognition

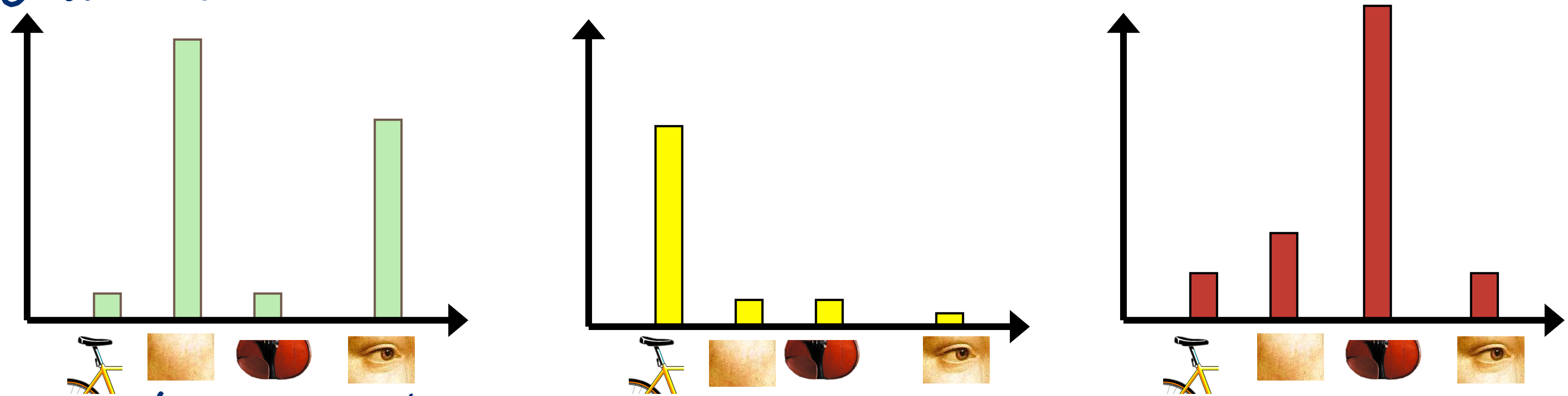
- Image representation ➔ bag of "visual words"

visual

Object → Bag of 'words'

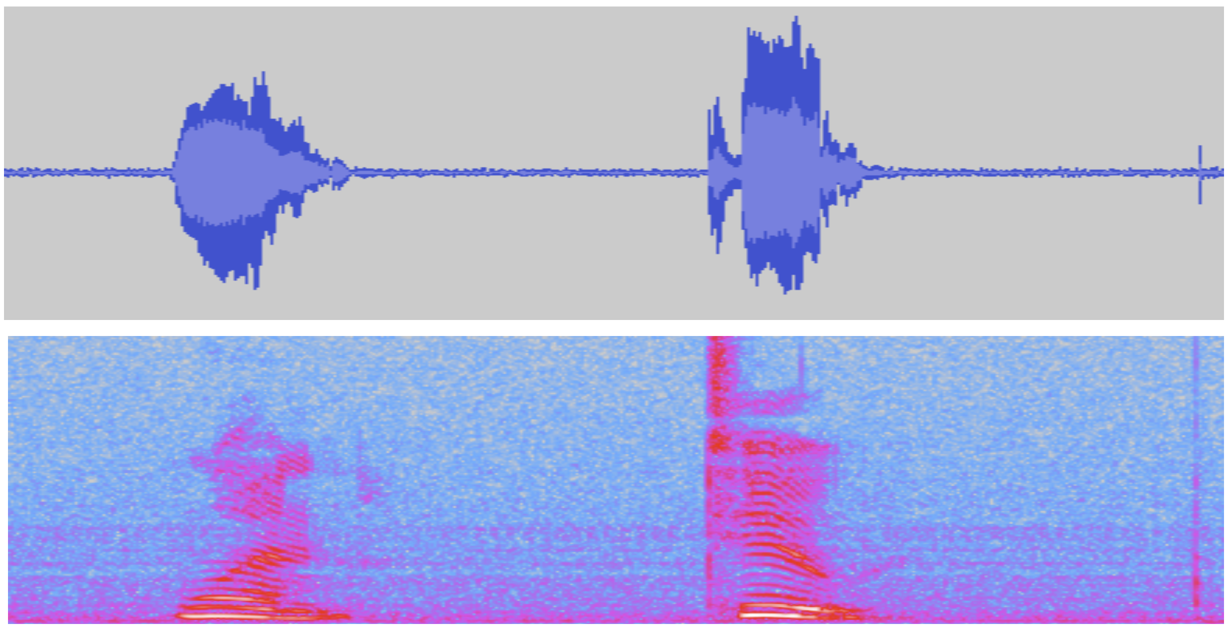
- An object image: histogram of visual vocabulary – a numerical vector of D dimensions.

Occurrence



W1 W2 W3.. Wn

Application 5: – Audio Classification



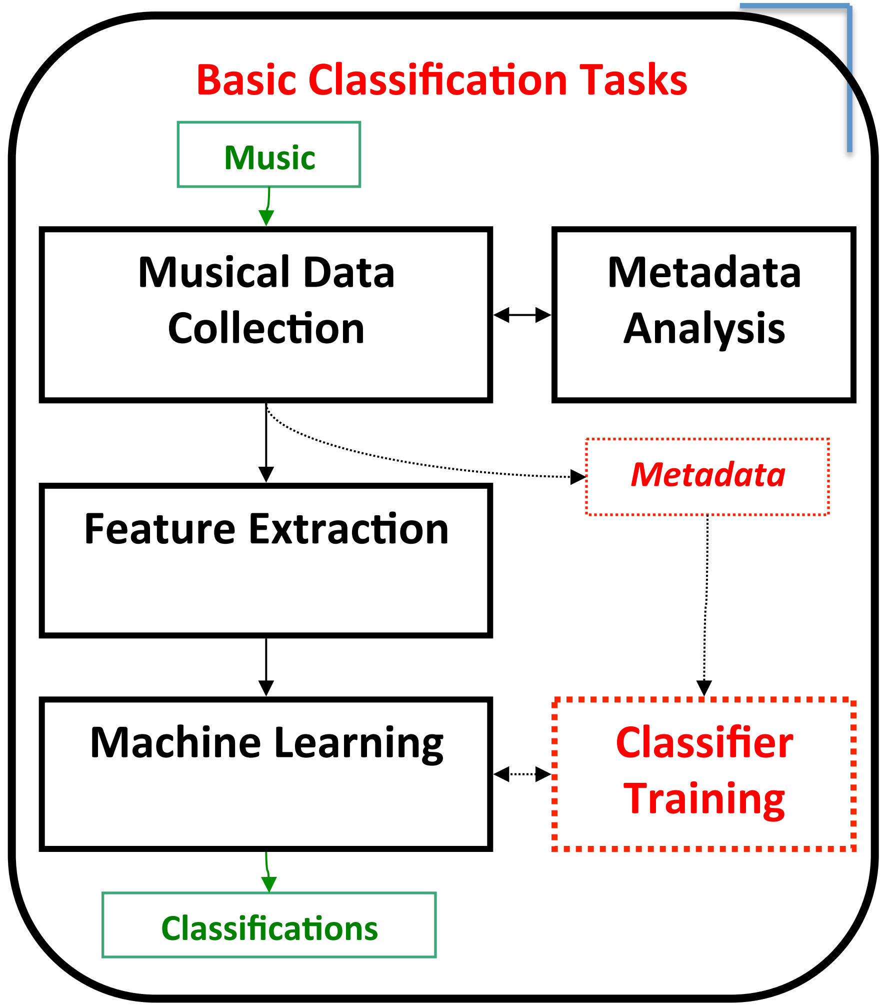
- Real-life applications:
  - Customer service phone routing
  - Voice recognition software

Music Information Retrieval Systems e.g., Automatic Music Classification

- Many areas of research in music information retrieval (MIR) involve using computers to classify music in various ways
  - Genre or style classification
  - Mood classification
  - Performer or composer identification
  - Music recommendation
  - Playlist generation
  - Hit prediction
  - Audio to symbolic transcription
  - etc.
- Such areas often share similar central procedures

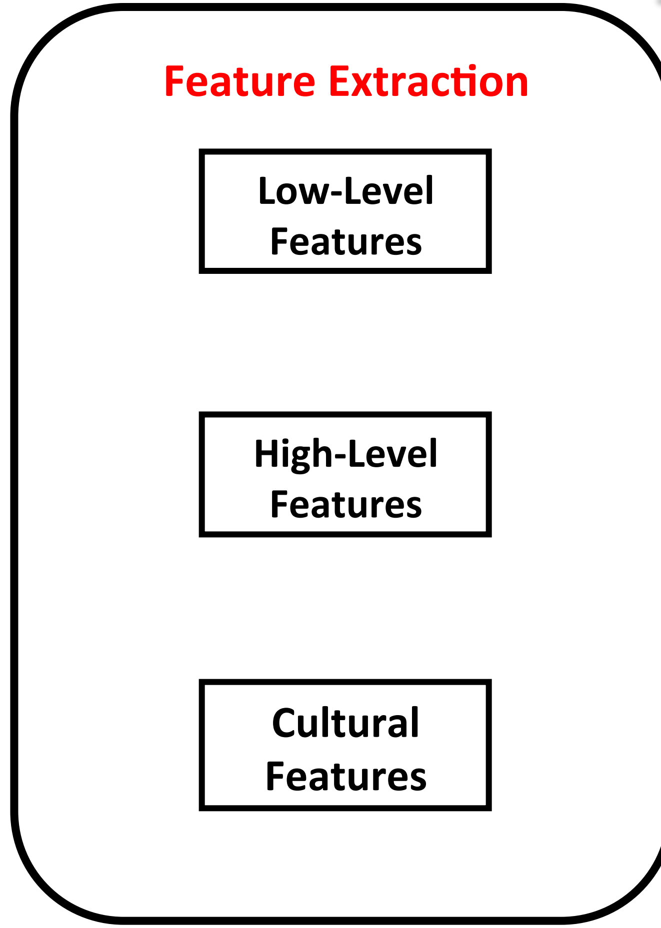
Music Information Retrieval Systems e.g., Automatic Music Classification

- Musical data collection
  - The instances (basic entities) to classify
  - Audio recordings, scores, cultural data, etc.
- Feature extraction
  - Features represent characteristic information about instances
  - Must provide sufficient information to segment instances among classes (categories)
- Machine learning
  - Algorithms ("classifiers" or "learners") learn to associate feature patterns of instances with their classes



Audio, Types of features

- Low-level
  - Associated with signal processing and basic auditory perception
  - e.g. spectral flux or RMS
  - Usually not intuitively musical
- High-level
  - Musical abstractions
  - e.g. meter or pitch class distributions
- Cultural
  - Sociocultural information outside the scope of auditory or musical content
  - e.g. playlist co-occurrence or purchase correlations



Where are we ? → Three major sections for classification

- We can divide the large variety of classification approaches into roughly three major types

1. Discriminative
   - directly estimate a decision rule/boundary
   - e.g., support vector machine, decision tree, logistic regression

2. Generative:
   - build a generative statistical model
   - e.g., Bayesian networks, Naïve Bayes classifier

3. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K nearest neighbors

ICML '06 Proceedings of the 23rd international conference on Machine learning

A study comparing Classifiers

An Empirical Comparison of Supervised Learning Algorithms

Rich Caruana CARUANA@CS.CORNELL.EDU
Alexandru Niculescu-Mizil ALEXN@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

A number of supervised learning methods have been introduced in the last decade. Unfortunately, the last comprehensive empirical evaluation of supervised learning was the Statlog Project in the early 90's. We present a large-scale empirical comparison between ten supervised learning methods: SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps. We also examine the effect that calibrating the models via Platt Scaling and Isotonic Regression has on their performance. An important aspect of our study is

This paper presents results of a large-scale empirical comparison of ten supervised learning algorithms using eight performance criteria. We evaluate the performance of SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps on eleven binary classification problems using a variety of performance metrics: accuracy, F-score, Lift, ROC Area, average precision, precision/recall break-even point, squared error, and cross-entropy. For each algorithm we examine common variations, and thoroughly explore the space of parameters. For example, we compare ten decision tree styles, neural nets of many sizes, SVMs with many kernels, etc.

Because some of the performance metrics we examine

A study comparing Classifiers
→ 11 binary classification problems

PROBLEM	#ATTR	TRAIN SIZE	TEST SIZE	%POZ
ADULT	14/104	5000	35222	25%
BACT	11/170	5000	34262	69%
COD	15/60	5000	14000	50%
CALHOUS	9	5000	14640	52%
COV_TYPE	54	5000	25000	36%
HS	200	5000	4366	24%
LETTER.P1	16	5000	14000	3%
LETTER.P2	16	5000	14000	53%
MEDIS	63	5000	8199	11%
MG	124	5000	12807	17%
SLAC	59	5000	25000	50%

Ratio among label

A study comparing Classifiers

➔ 11 binary classification problems / 8 metrics

ACC ROC

Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774

Ratio of Positive Class (binary case)

- Class imbalance issue num $\underbrace{AP}_{\text{actual positive}} \ll$ num $\underbrace{AN}_{\text{actual neg}}$
- Balanced accuracy: $= \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right)$

$P \rightarrow TP+FP$ $\quad N \rightarrow TN+FN$

Confusion Matrix

Labeled

	actual + (AP)	actual − (AN)
predicted+	TP	FP
predicted−	FN	TN

precision

Recall

TP: true positive

FP: false positive

$$\text{Accuracy} = \frac{\#\text{Correct Predicted}}{\#\text{all test Examples}}$$

$$= \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{Precision\_Pos} = \frac{TP}{P}$$

$$\text{Recall\_Pos} = \frac{TP}{TP + FN}$$

$$\Rightarrow F_1 = \frac{2\,\text{Rec} \times \text{Prec}}{\text{Rec} + \text{Prec}}$$

Ratio of Positive Class (binary case)

(1, 99)
pos neg

If $\frac{\text{Actual P}}{AP + AN}$ very small (e.g. < 1%)

⇒ a classifier can predict every example as Neg

⇒ 1

	AP 1	AN 99
predict P	0	0
predict N	1	99

⇒ Accuracy = $\frac{99}{100}$ = 0.99

⇒ Balanced Acc =

Bad - neg - classifier

① Balanced Acc $= \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right)$

$= \frac{1}{2}\left(\frac{0}{0+\varepsilon} + \frac{99}{100}\right) = 0.495$

$[0, 1]$

② another classifier

	AP	AN
PP	1	0
PN	0	99

Balanced Acc $= \frac{1}{2}\left(\frac{1}{1} + \frac{99}{99}\right) = 1$

Acc $= \frac{1+99}{1+0+99+0} = 1$

Today

- ❑ Supervised Classification
- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Linearly Non-separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Multiclass SVM

History of SVM

Young / theoretically sound / Impactful

- SVM is inspired from statistical learning theory [3]
- SVM was first introduced in 1992 [1]
- SVM becomes popular because of its success in handwritten digit recognition (1994)
  - 1.1% test error rate for SVM. This is the same as the error rates of a carefully constructed neural network, LeNet 4.
    - Section 5.11 in [2] or the discussion in [3] for details
- SVM is now regarded as an important example of "kernel methods", arguably the hottest area in machine learning 20 years ago

[1] B.E. Boser *et al*. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.
[2] L. Bottou *et al*. Comparison of classifier methods: a case study in handwritten digit recognition. Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 2, pp. 77-82, 1994.
[3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.

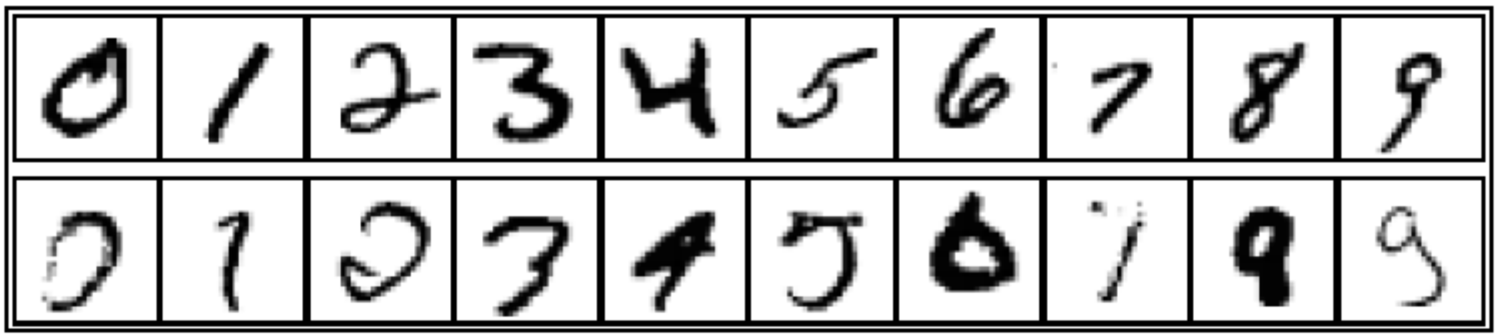
Applications of SVMs

- Computer Vision
- Text Categorization
- Ranking (e.g., Google searches)
- Handwritten Character Recognition
- Time series analysis
- Bioinformatics
- ……….

→Lots of very successful applications!!!

Handwritten digit recognition

→ MNIST (SVM) 1994



3-nearest-neighbor = 2.4% error

400–300–10 unit MLP = 1.6% error

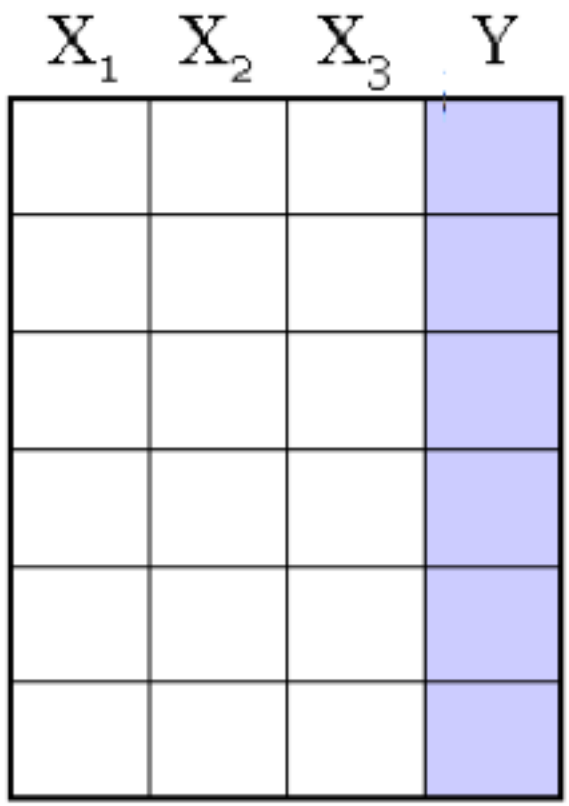
LeNet: 768–192–30–10 unit MLP = 0.9% error

In 90s, SVM achieves the

...t best (kernel machines, vision algorithms) $\approx$ 0.6% error

> ImageNet (Deep(NN)) 2012

A Dataset for binary/ classification



$$f : \boxed{X} \longrightarrow \boxed{Y}$$

Output as Binary Class: only two possibilities

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/ predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

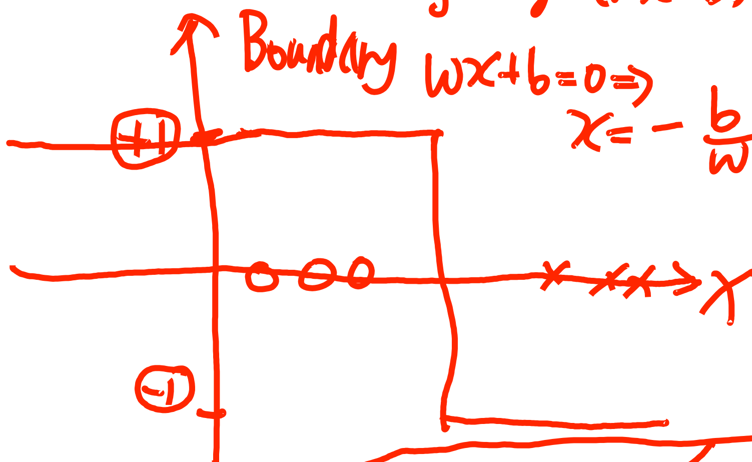
Binary Classification

$y \in \{-1, 1\}$

if ⇒ 1D [$x \in R$]

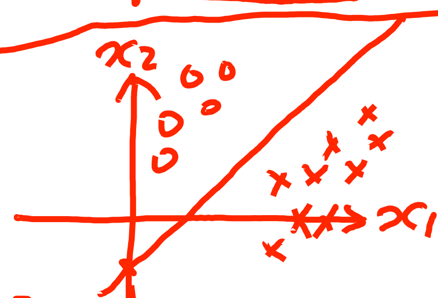
$y = sign(wx+b)$

Boundary $wx+b=0 \Rightarrow x = -\frac{b}{w}$



if ⇒ 2D [$x \in R^2$]

Boundary $w^Tx+b=0$



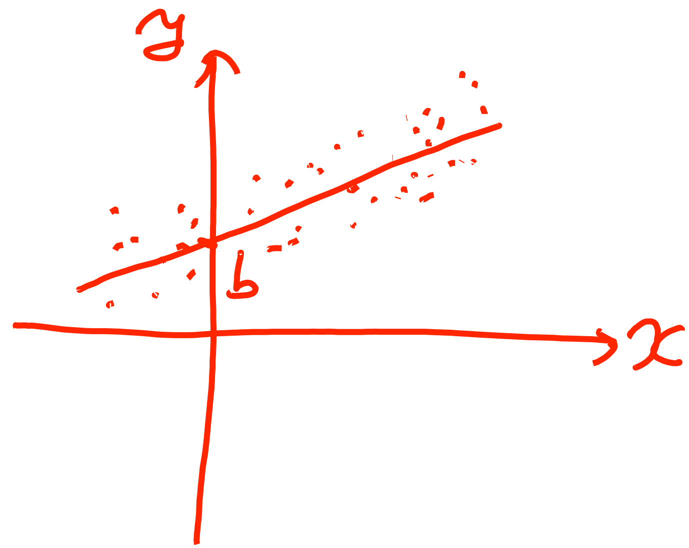
if ⇒ p dim [$x \in R^p$]

Boundary : hyperplane

Regression

$y \in R$

if ⇒ 1D $x \in R$



$y = wx+b$

$y = w^Tx+b$ if $x \in R^p$

Affine hyperplanes

- https://en.wikipedia.org/wiki/Hyperplane
- any hyperplane can be given in coordinates as the solution of a single linear (algebraic) equation of degree 1.

$[a_1 x_1 + a_2 x_2 + a_3 x_3 + \cdots + a_p x_p = b]$, at least one $a_i \neq 0$

$\Rightarrow$ e.g. classification Boundary $w^T x + b = 0$

$\begin{cases} x \in R^p \\ b \in R \end{cases}$

Review : Vector Product, Orthogonal, and Norm

For two vectors *x* and *y*,

$$x^T y$$

is called the *(inner) vector product.*

x and *y* are called *orthogonal* if

$$x^T y = 0$$

The square root of the product of a vector with itself,

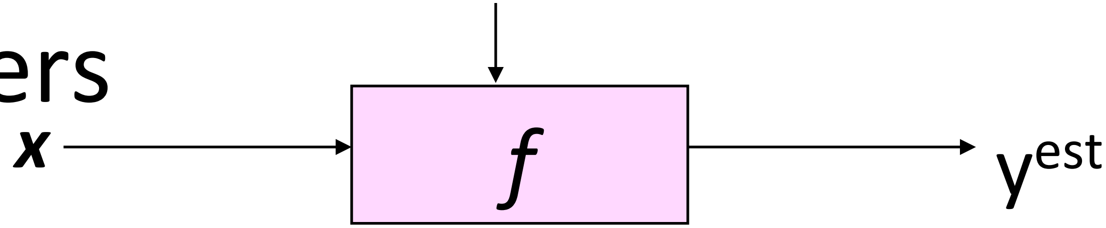
$$\sqrt{x^T x}$$

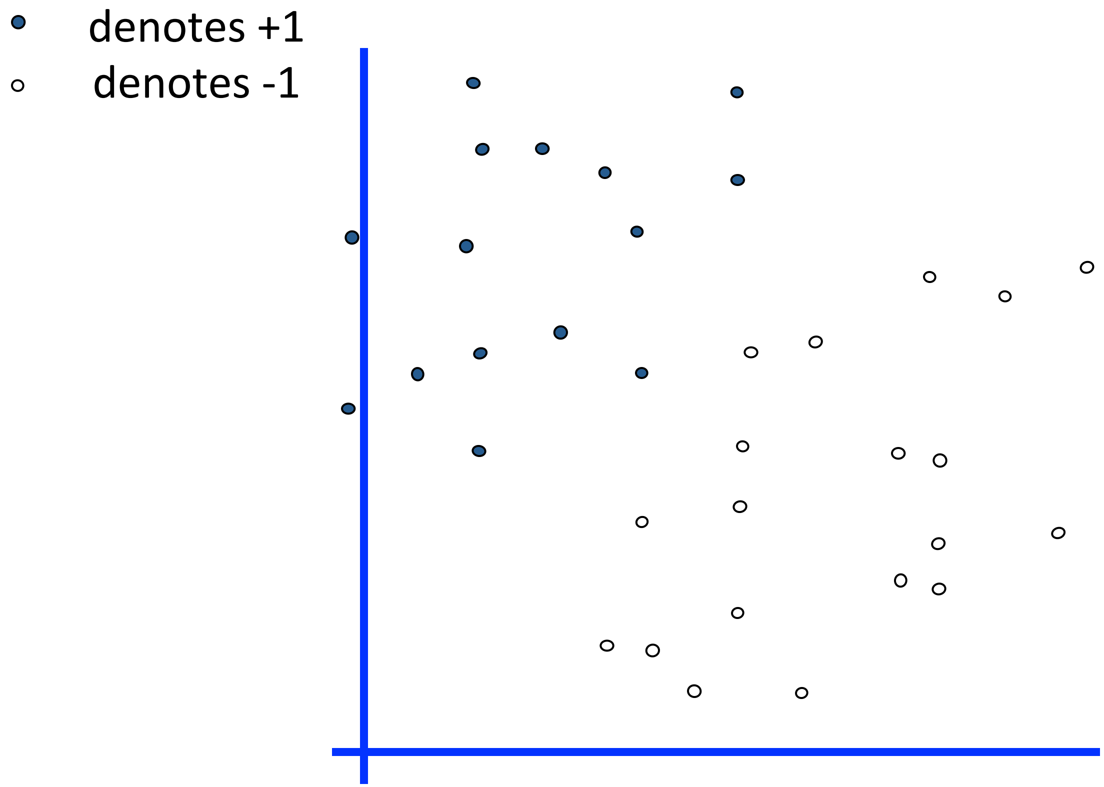
is called the *2-norm* ($|x|_2$), can also write as $|x|$

Today

- ❑ Supervised Classification
- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Linearly Non-separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Multiclass SVM

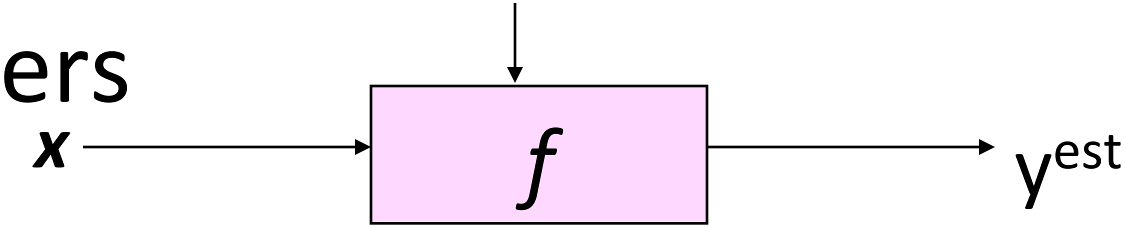
Linear Classifiers

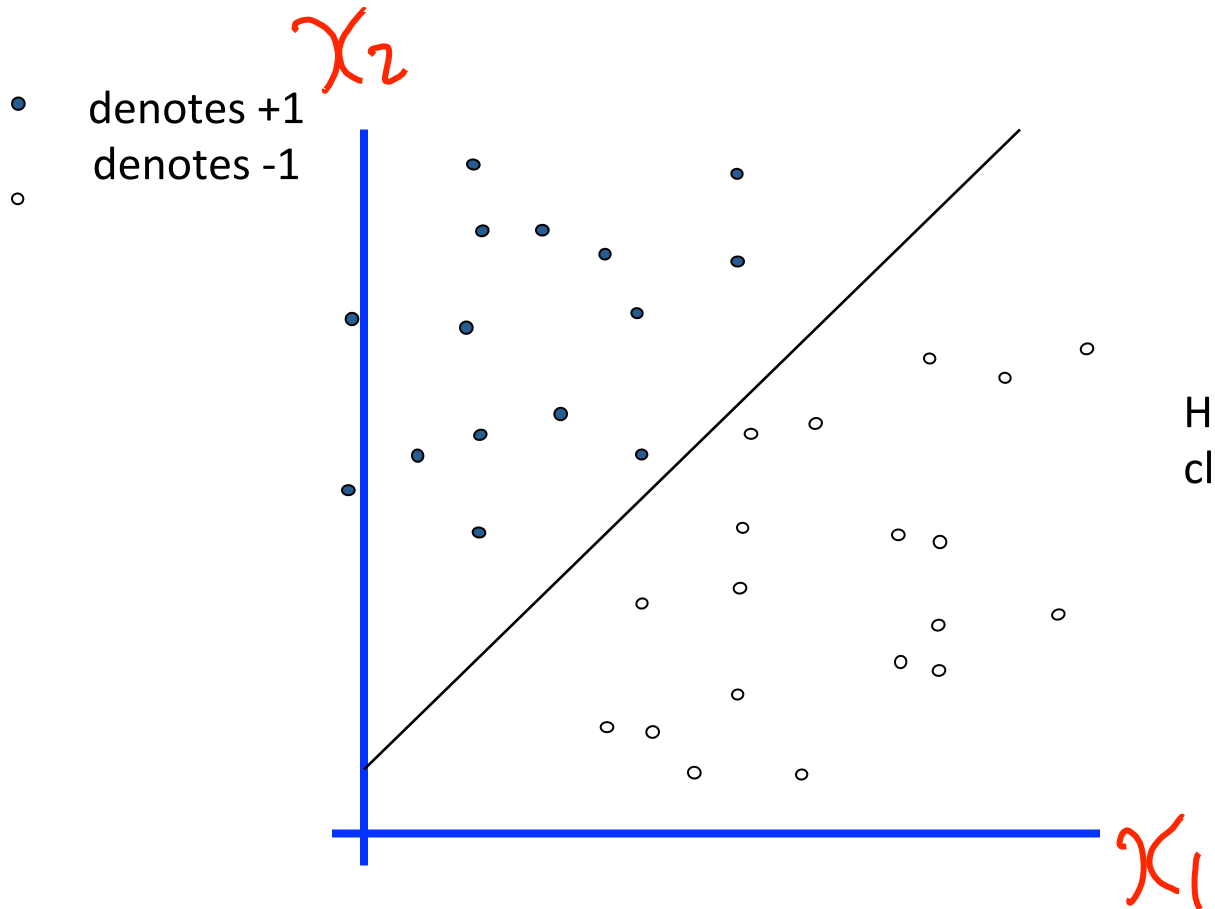




How would you classify this data?

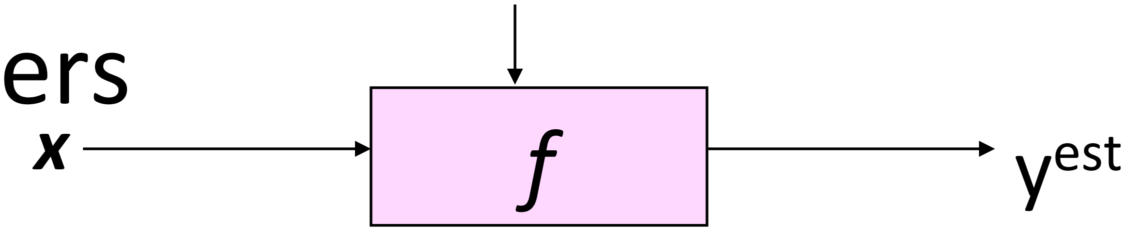
Linear Classifiers



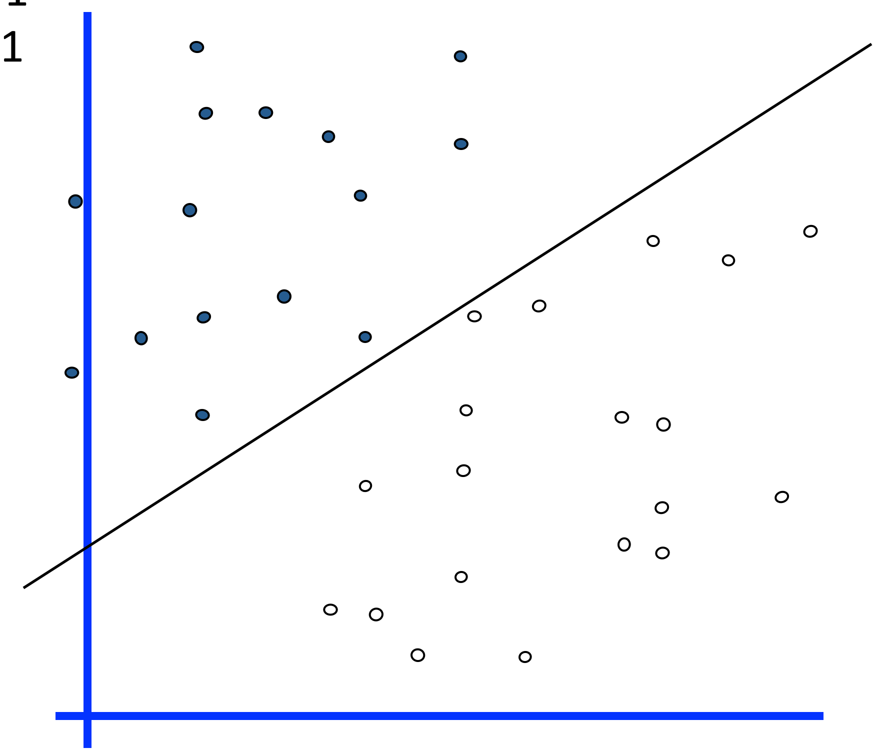


How would you classify this data?

Linear Classifiers

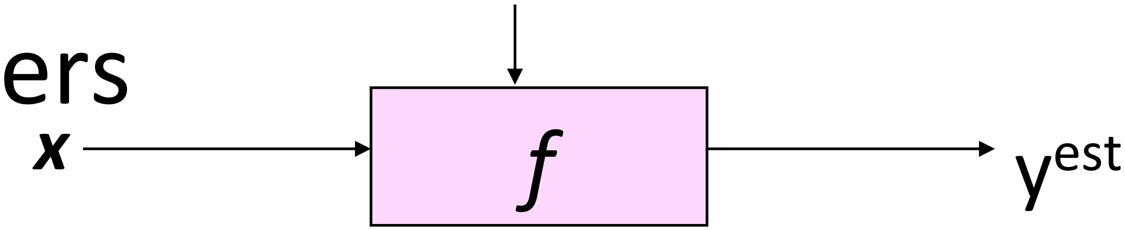


- denotes +1
- denotes -1

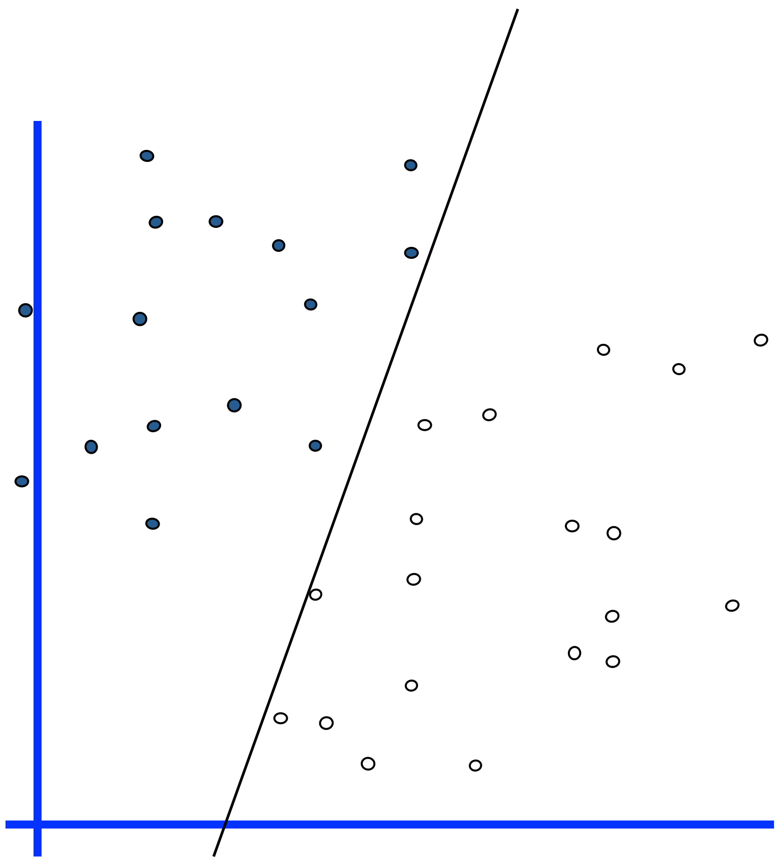


How would you classify this data?

Linear Classifiers

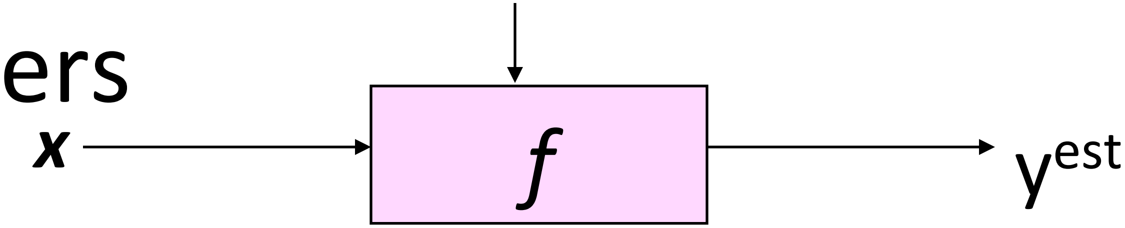


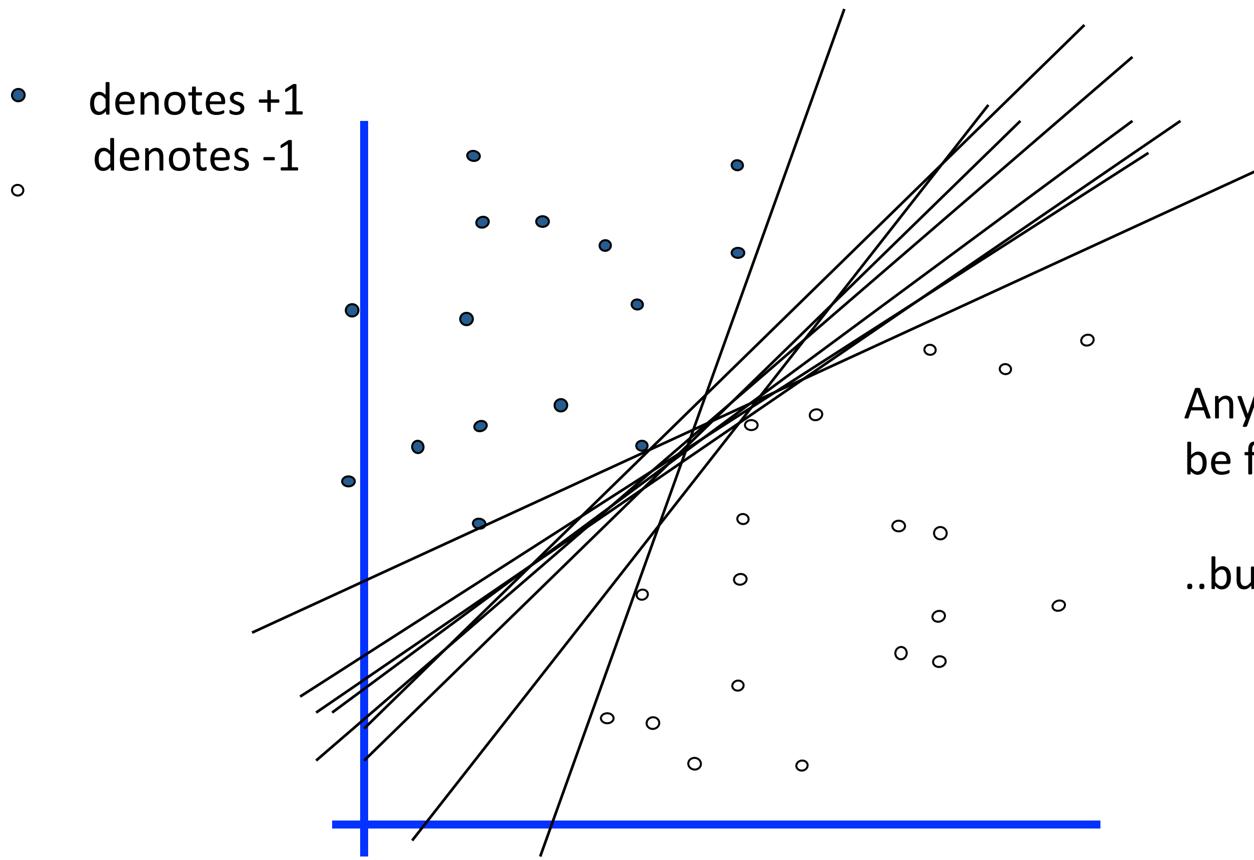
• denotes +1
○ denotes -1



How would you classify this data?

Linear Classifiers

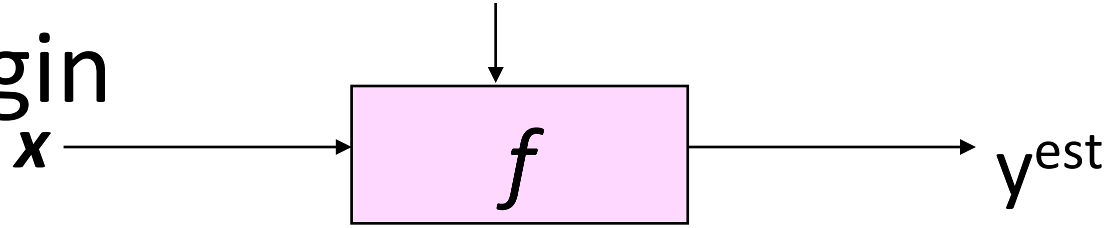


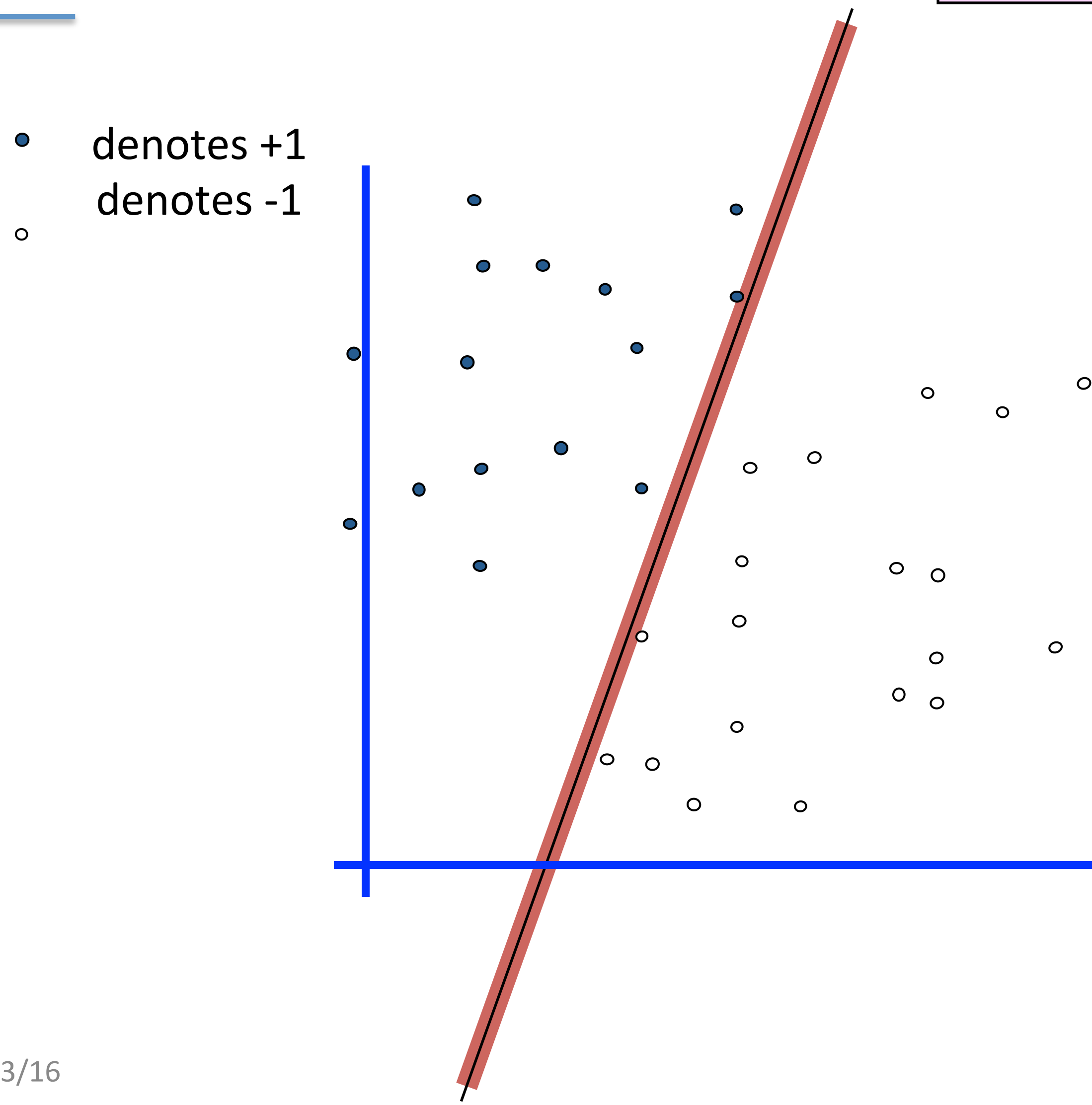


Any of these would be fine..

..but which is best?

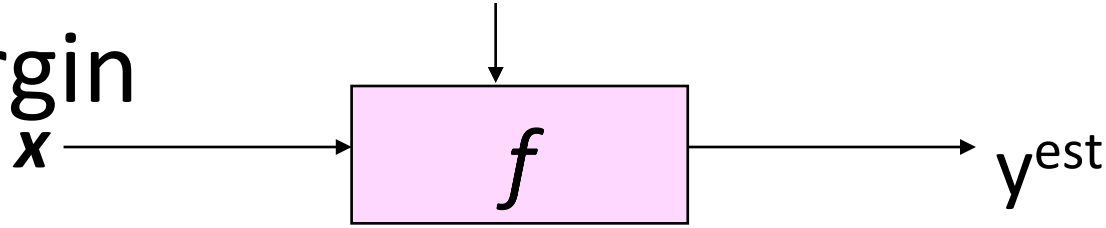
Classifier Margin

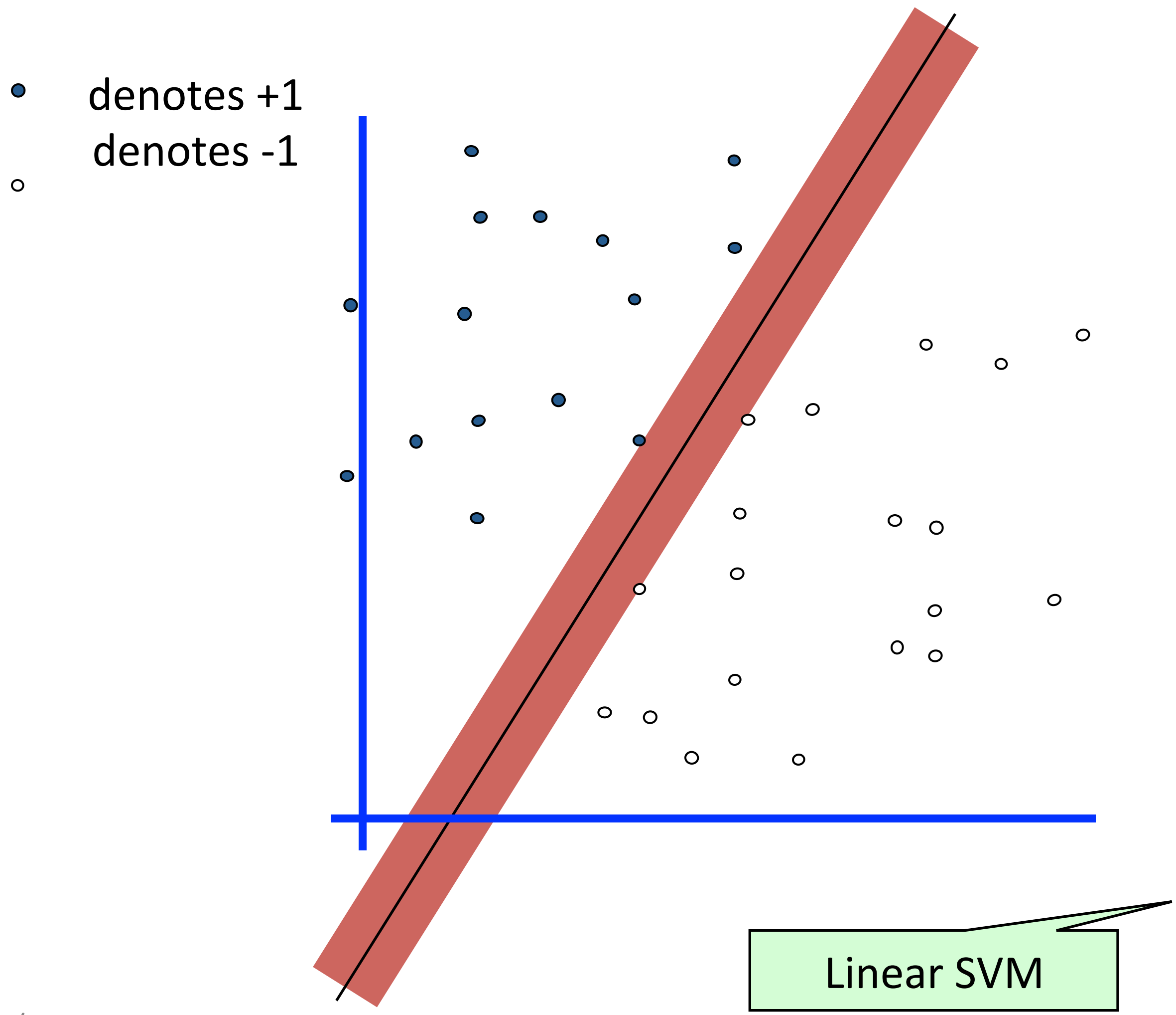




Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

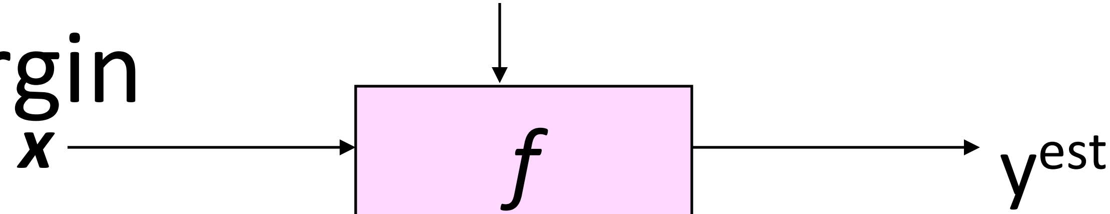
Maximum Margin

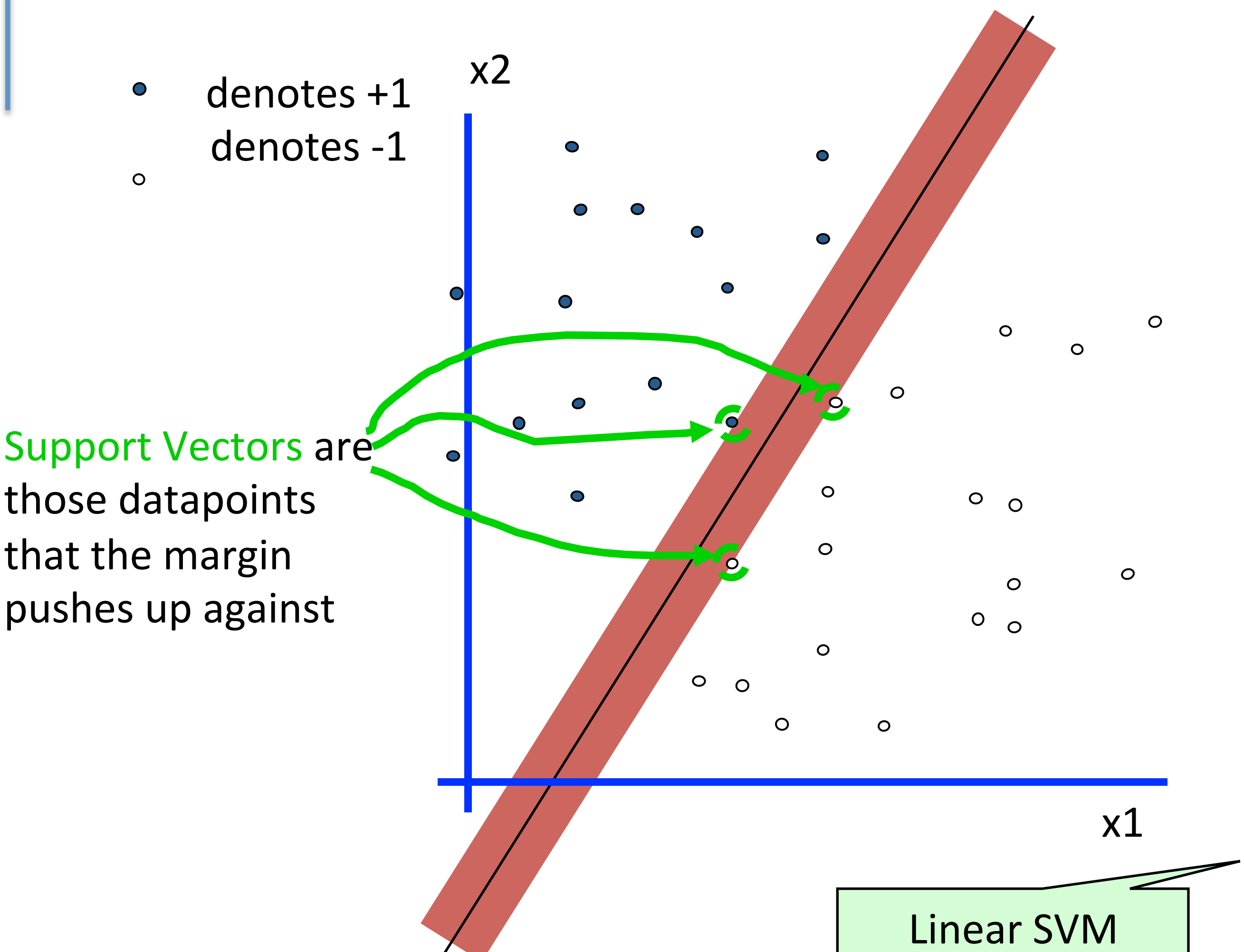




The maximum margin linear classifier is the linear classifier with the, maximum margin.
This is the simplest kind of SVM (Called an LSVM)

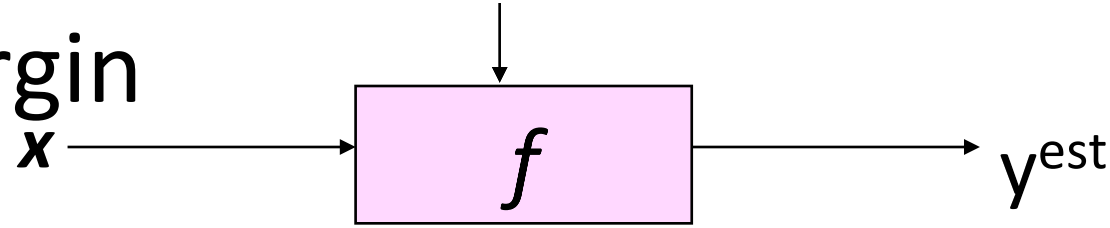
Maximum Margin



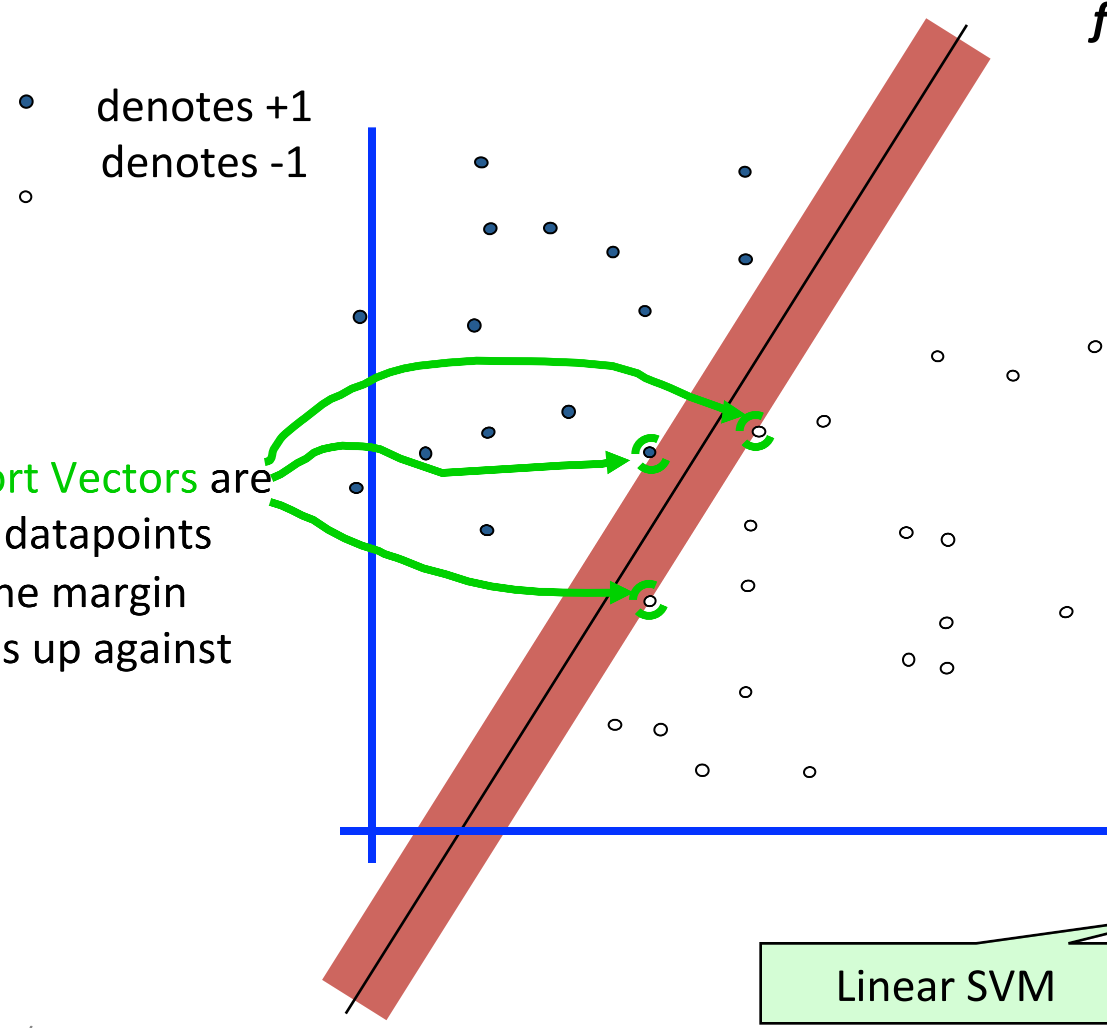


The maximum margin linear classifier is the linear classifier with the, maximum margin.
This is the simplest kind of SVM (Called an LSVM)

Maximum Margin



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}^T\mathbf{x} + b)$



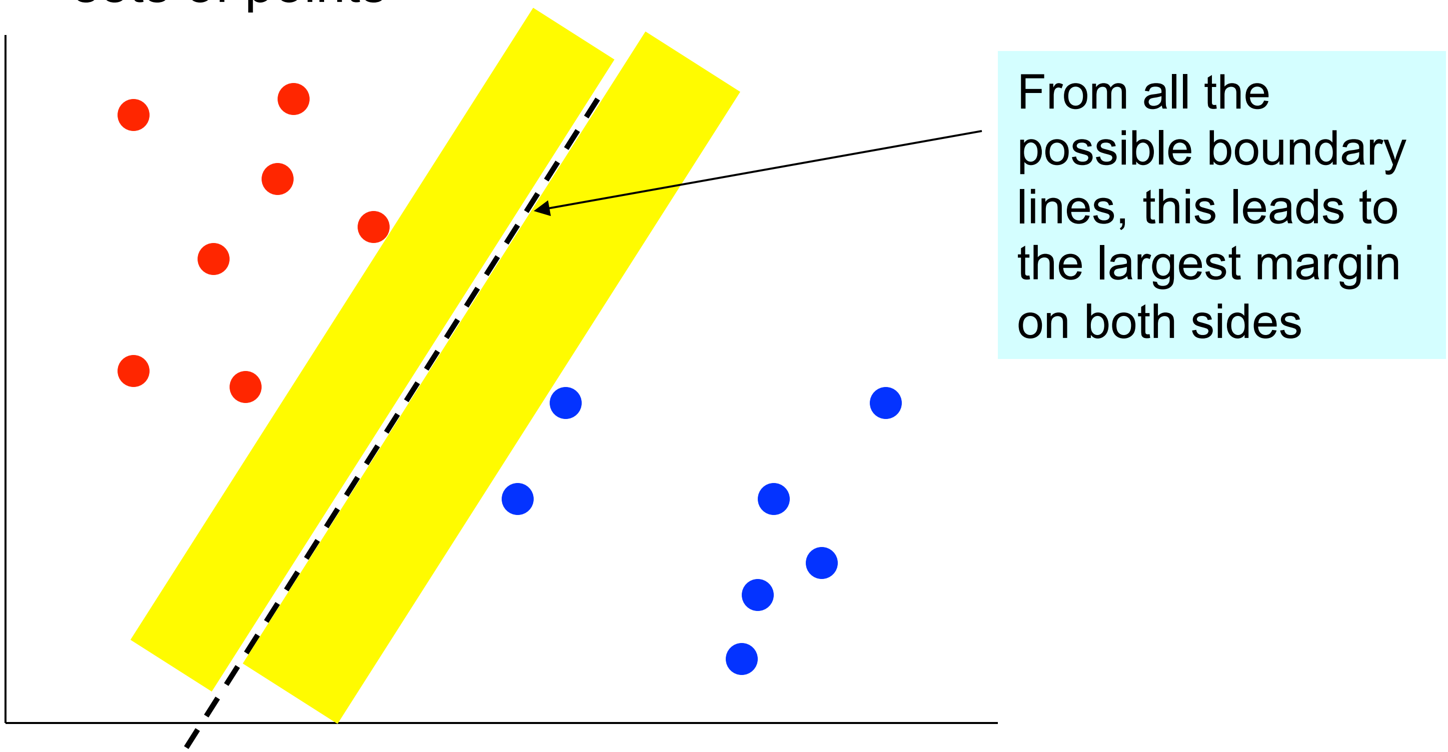
Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the, maximum margin.
This is the simplest kind of SVM (Called an LSVM)

Linear SVM

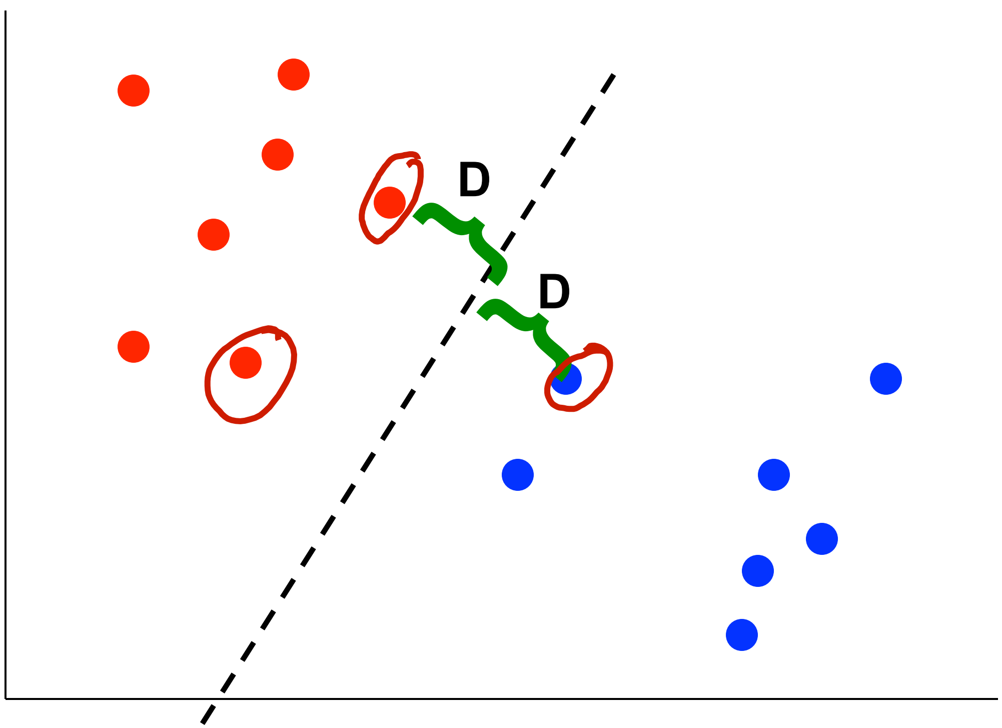
Max margin classifiers

- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest margin from both sets of points



Max margin classifiers

- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest margin from points on both sides

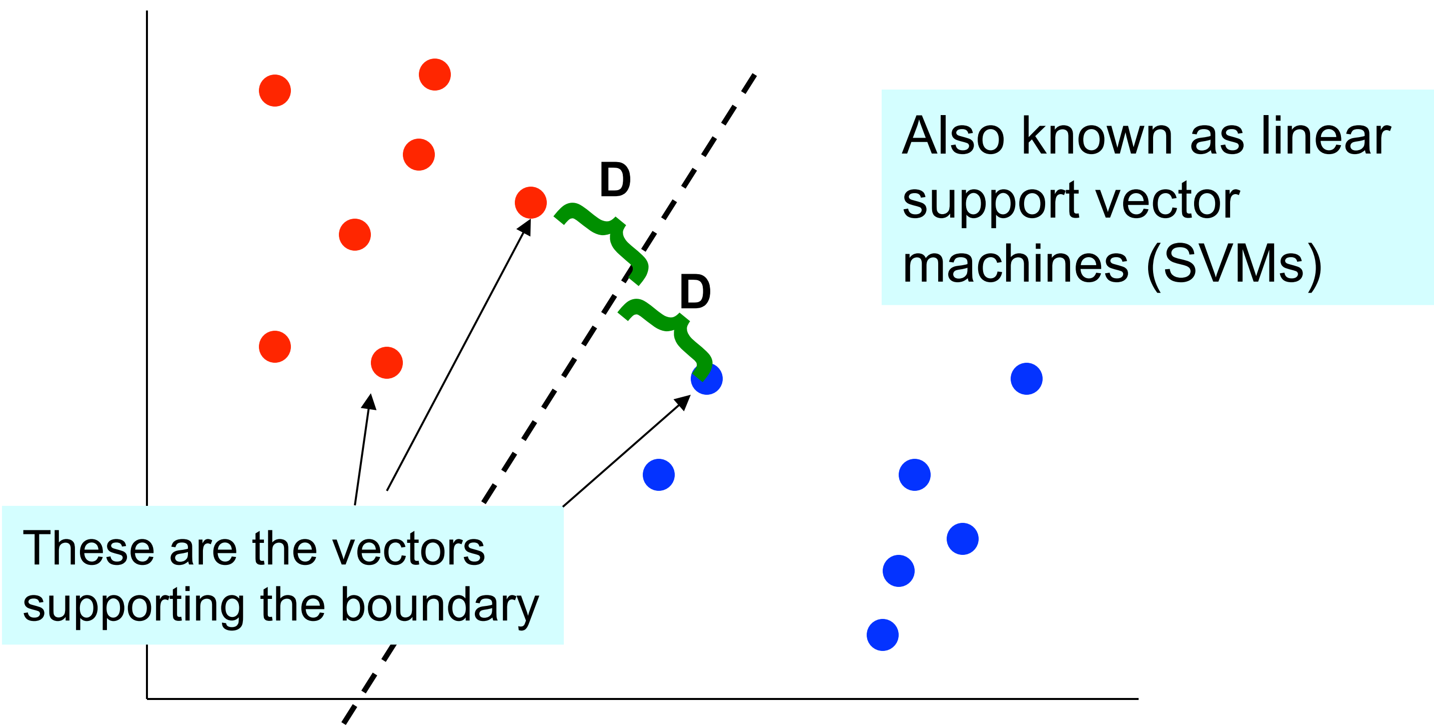


Why?

- Intuitive, 'makes sense'
- Some theoretical support
- Works well in practice

Max margin classifiers

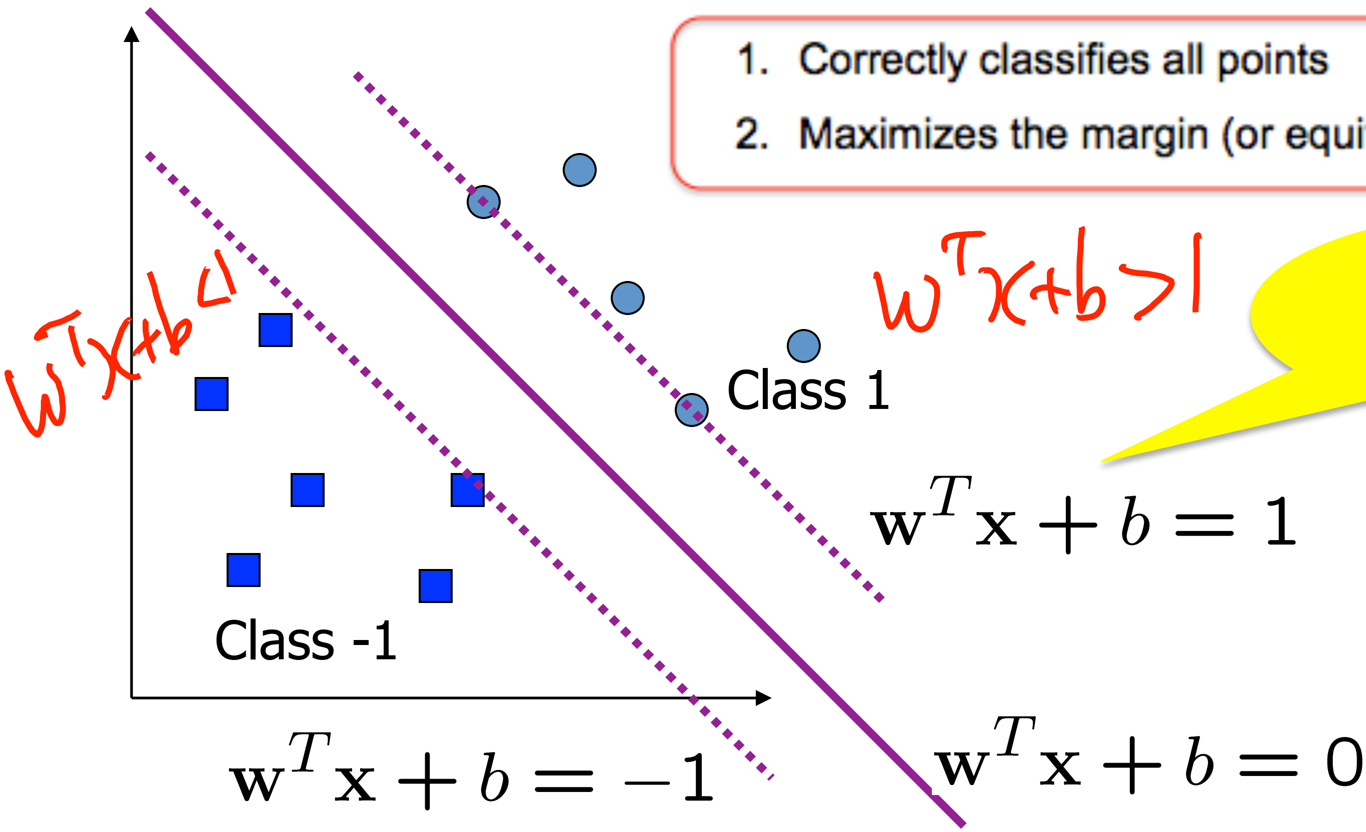
- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest margin from points on both sides



Max-margin & Decision Boundary

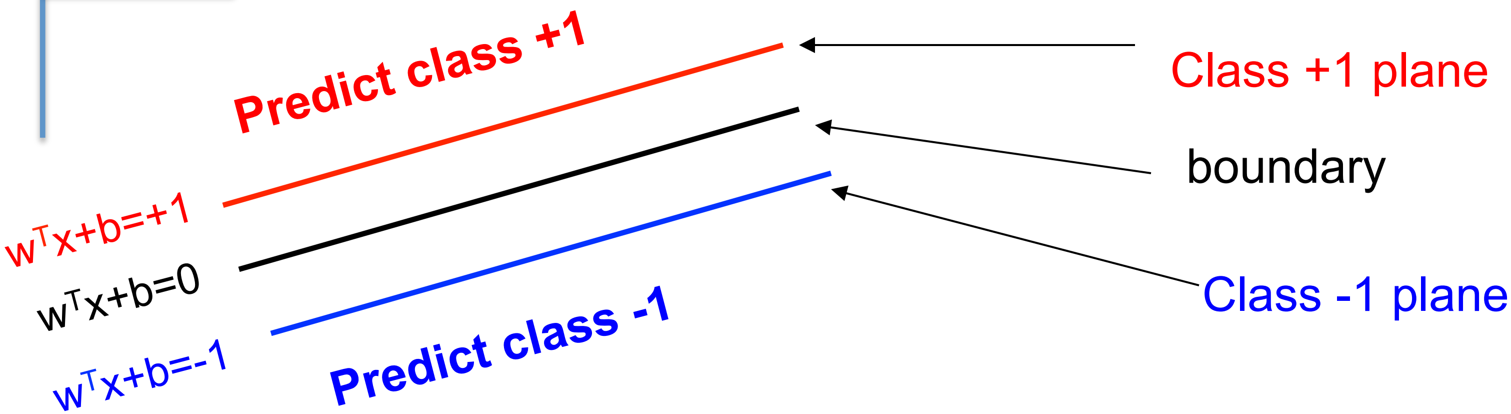
- The decision boundary should be as far away from the data of both classes as possible

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes w^Tw)



W is a p-dim vector; b is a scalar

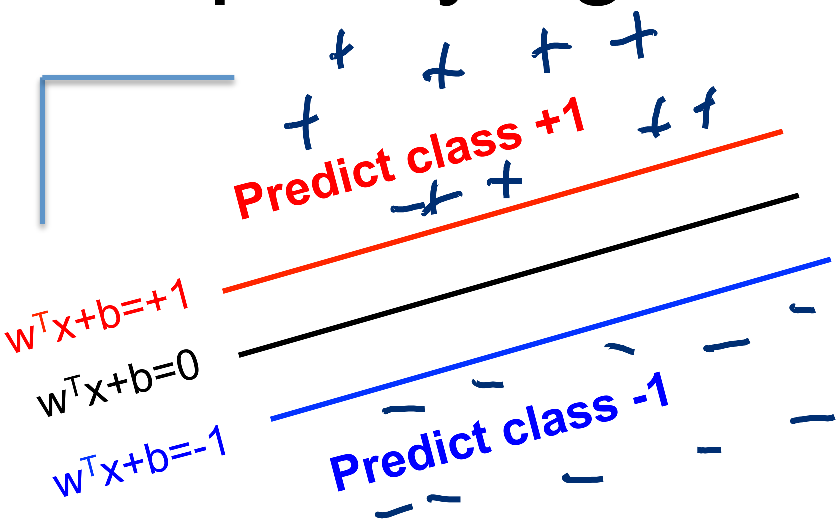
Specifying a max margin classifier



Classify as +1	if	$w^Tx+b >= 1$
Classify as -1	if	$w^Tx+b <= -1$
Undefined	if	$-1 < w^Tx+b < 1$

my formulation

Specifying a max margin classifier



Is the linear separation assumption realistic?

We will deal with this shortly, but lets assume it for now

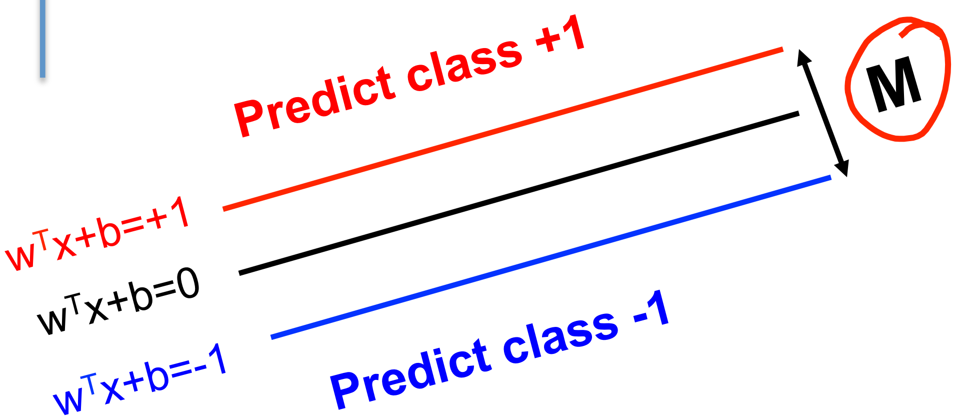
Classify as +1	if	$w^Tx+b >= 1$
Classify as -1	if	$w^Tx+b <= -1$
Undefined	if	$-1 < w^Tx+b < 1$

Now Assuming such lines exist in our train

Today

- ❑ Supervised Classification
- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Linearly Non-separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Multiclass SVM

Maximizing the margin



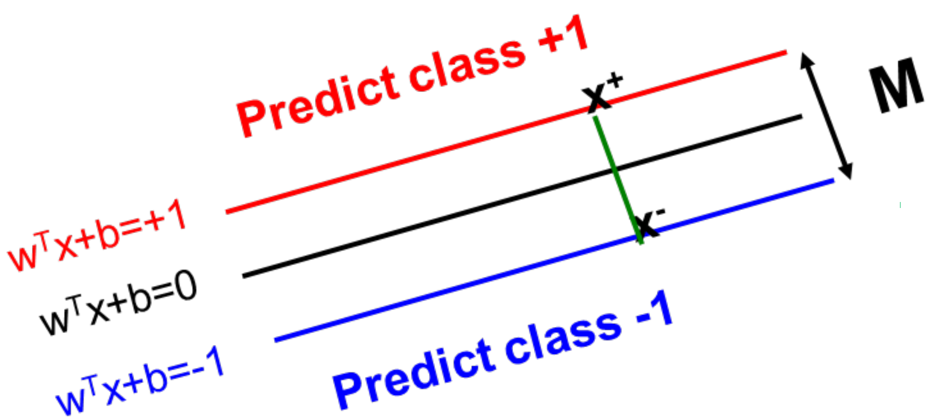
Classify as +1 if $w^Tx+b >= 1$

Classify as -1 if $w^Tx+b <= -1$

Undefined if $-1 < w^Tx+b < 1$

- Lets define the width of the margin by M
- How can we encode our goal of maximizing M in terms of our parameters (w and b)?
- Lets start with a few obsevrations

Margin M



Classify as +1 if $w^Tx+b >= 1$

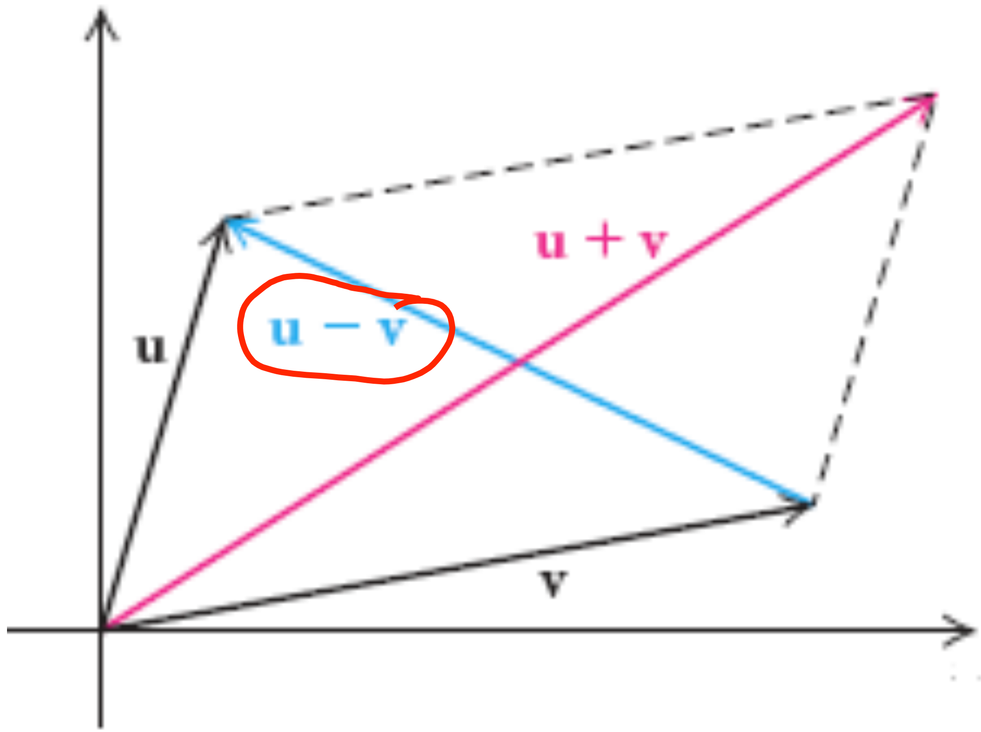
Classify as -1 if $w^Tx+b <= -1$

Undefined if $-1 < w^Tx+b < 1$

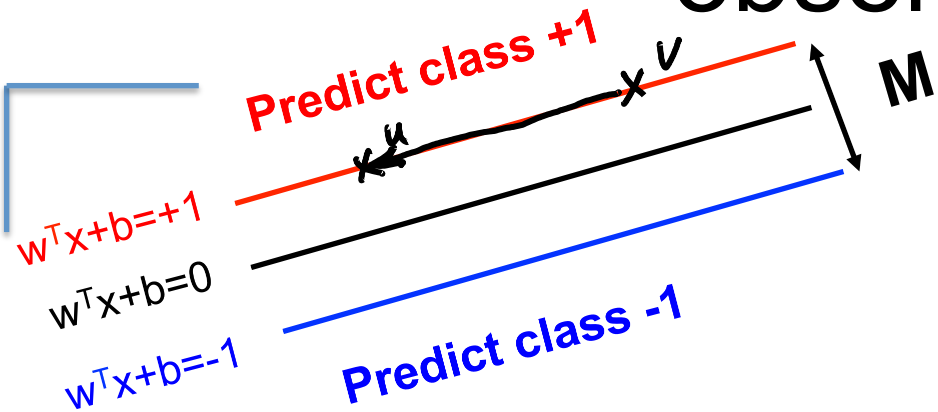
$M = |x^+ - x^-|$ length of vector $(x^+ - x^-)$

$\Rightarrow$ How to represent $(x^+ - x^-)$???

➔ Review :Vector Subtraction



Maximizing the margin: observation-1



Classify as +1 if $w^Tx+b >= 1$

Classify as -1 if $w^Tx+b <= -1$

Undefined if $-1 < w^Tx+b < 1$

- **Observation 1: the vector w is orthogonal to the +1 plane**
- Why?

→ vector (u-v) shown above

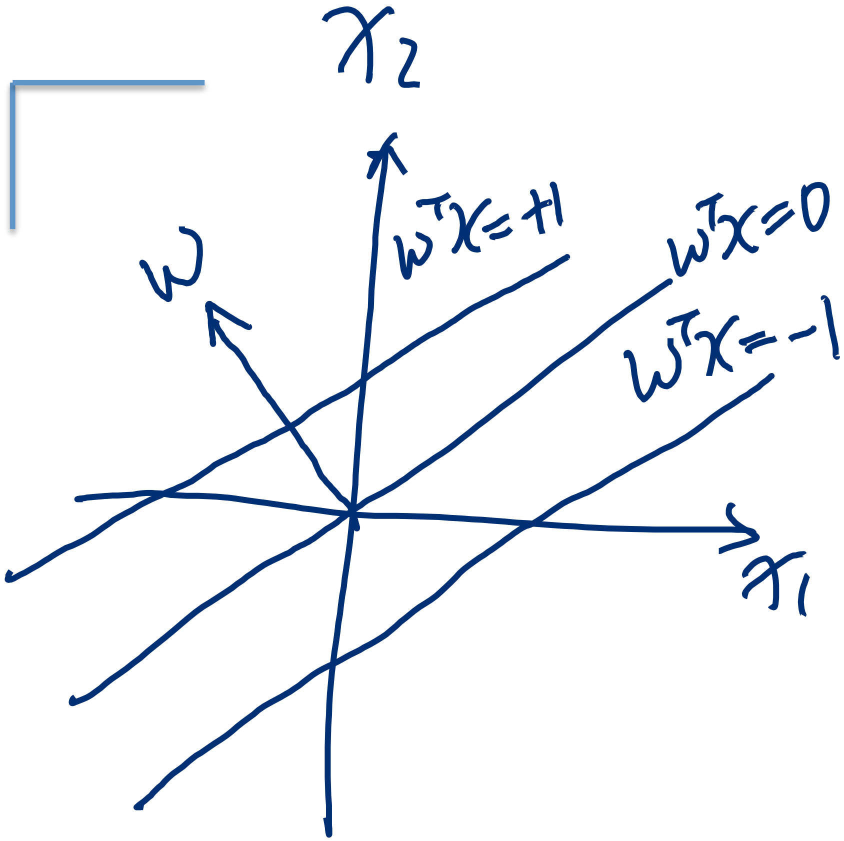
→ $w^T(u-v) = w^Tu - w^Tv = (1-b) - (1-b) = 0$

$\Rightarrow$ w orthogonal to (u-v)

Let u and v be two points on the +1 plane, then for the vector defined by u and v we have $w^T(u-v) = 0$

Corollary: the vector w is orthogonal to the -1 plane

[Classification]



[Regression]

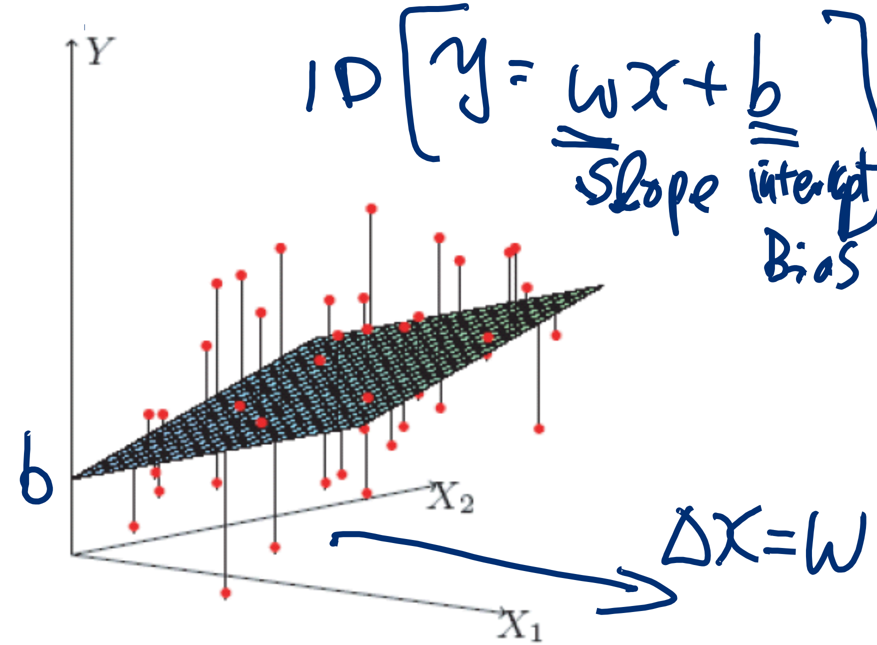
$$y = w^T x + b$$

$$\Delta X = \frac{\partial y}{\partial x} = w$$

1D [$y = \underline{w}x + \underline{b}$]

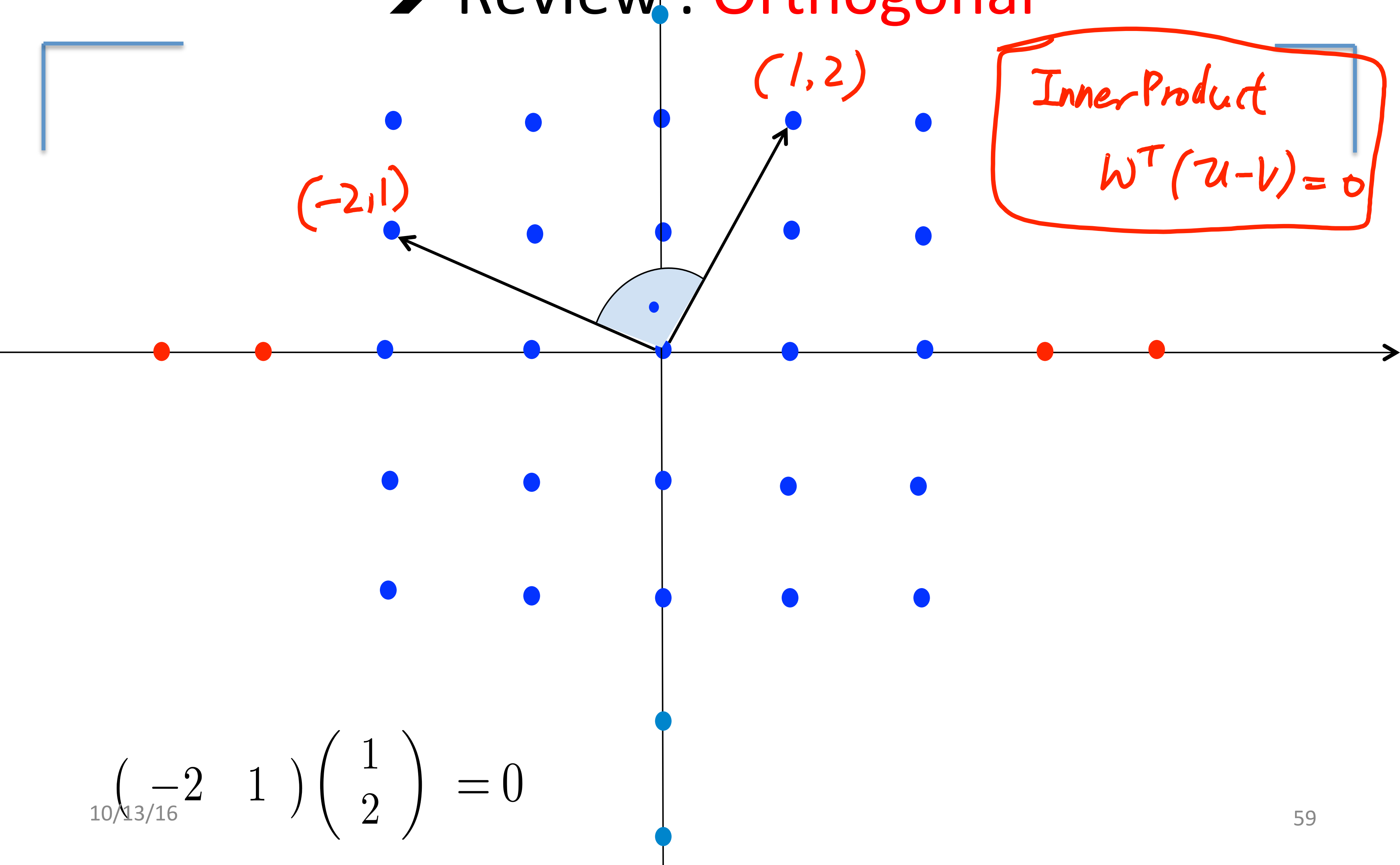
slope intercept

Bias



The gradient points in the direction of the greatest rate of increase of the function and its magnitude is the slope of the graph in that direction

Observation 1
➔ Review : Orthogonal



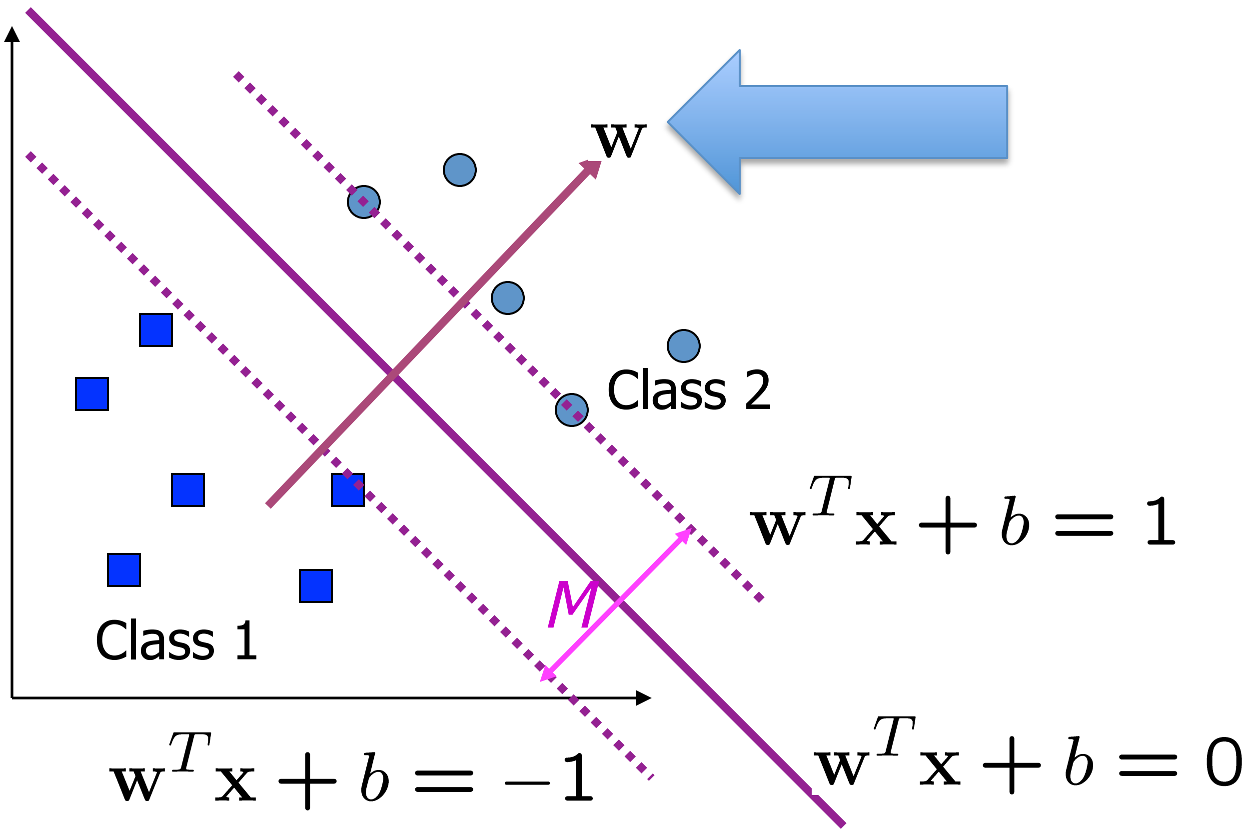
$$\begin{pmatrix} -2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = 0$$

Maximizing the margin: observation-1

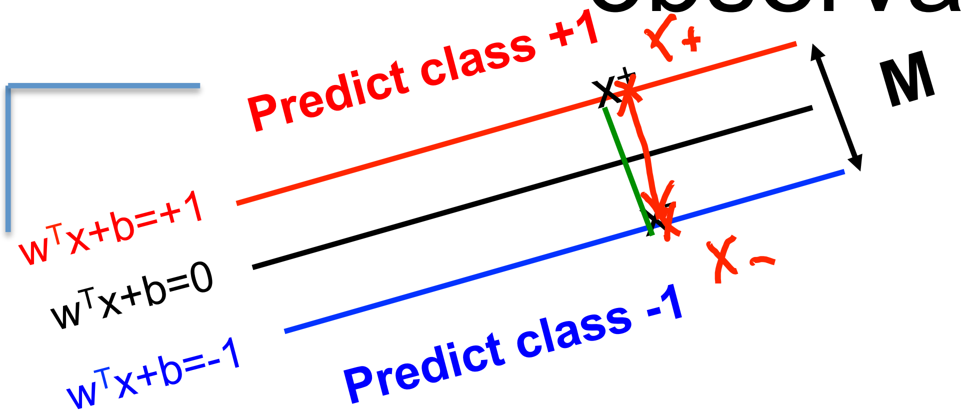
- **Observation 1: the vector w is orthogonal to the +1 plane**

-1 plane

0 -plane



Maximizing the margin: observation-2



Classify as +1 if $w^Tx+b >= 1$

Classify as -1 if $w^Tx+b <= -1$

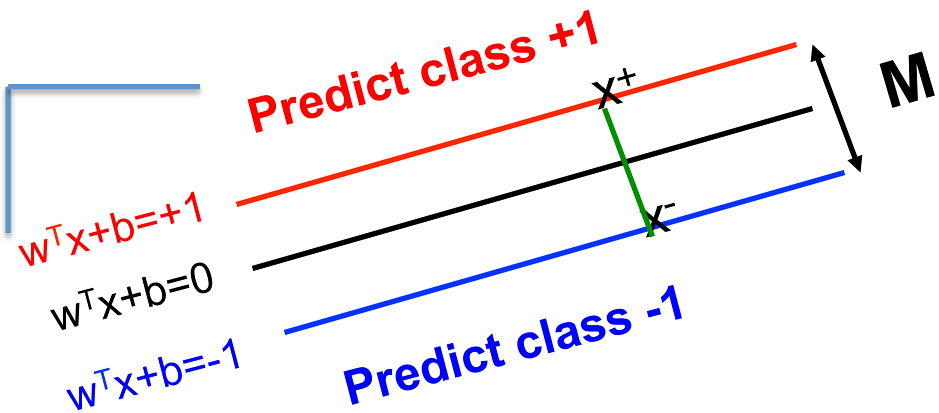
Undefined if $-1 < w^Tx+b < 1$

- Observation 1: the vector w is orthogonal to the +1 and -1 planes
- Observation 2: if x^+ is a point on the +1 plane and x^- is the closest point to x^+ on the -1 plane then

$$x^+ = \lambda w + x^-$$

Since w is orthogonal to both planes we need to 'travel' some distance along w to get from x^+ to x^-

Putting it together



- $w^T x^+ + b = +1$
- $w^T x^- + b = -1$
- $x^+ = \lambda w + x^-$
- $| x^+ - x^- | = M$

We can now define M in terms of w and b

$$M = |x^+ - x^-|$$
$$= |\lambda w|$$
$$= \lambda |w|$$
$$= \lambda \sqrt{w^T w}$$
$$= \frac{2}{w^T w} \sqrt{w^T w}$$
$$\left[= \frac{2}{\sqrt{w^T w}} \right]$$

$$w^T x^+ + b = 1$$

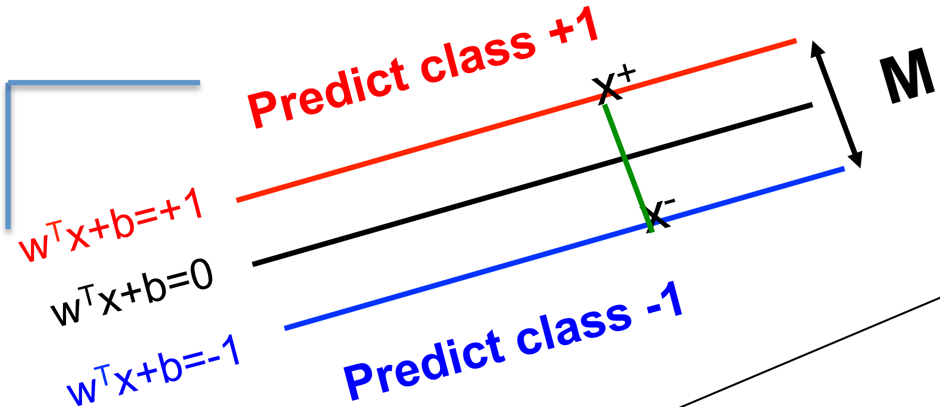
$$w^T(\lambda w + x^-) + b = +1$$

$$\lambda w^T w + \underbrace{w^T x^- + b}_{\Rightarrow -1} = 1$$

$$\lambda w^T w = 2$$

$$\Rightarrow \quad \lambda = \frac{2}{w^T w}$$

Putting it together



- $w^T x^+ + b = +1$
- $w^T x^- + b = -1$
- $x^+ = \lambda w + x^-$
- $| x^+ - x^- | = M$

We can now define M in terms of w and b

$w^T x^+ + b = +1$

=>

$w^T (\lambda w + x^-) + b = +1$

=>

$w^T x^- + b \ + \lambda w^T w = +1$

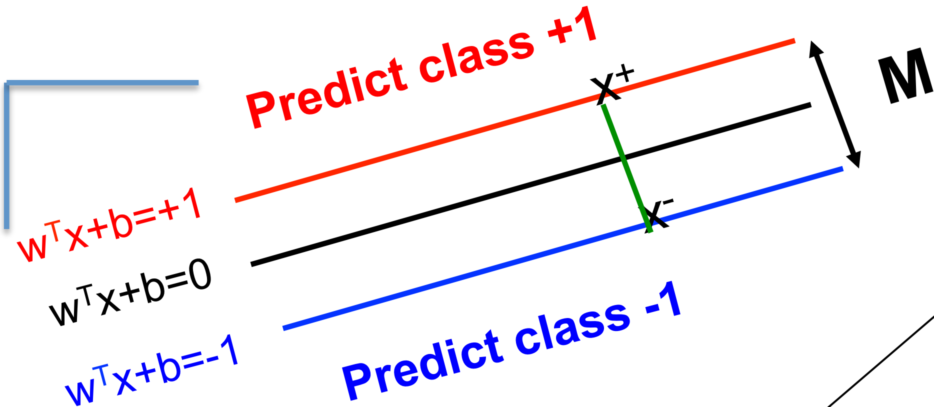
=>

$-1 \ + \lambda w^T w = +1$

=>

$\lambda = 2/w^T w$

Putting it together



- $w^T x^+ + b = +1$
- $w^T x^- + b = -1$
- $x^+ = \lambda w + x^-$
- $| x^+ - x^- | = M$
- $\lambda = 2/w^T w$

$M = |x^+ - x^-|$

=>

$$M = |\lambda w| = \lambda |w| = \lambda \sqrt{w^T w}$$

=>

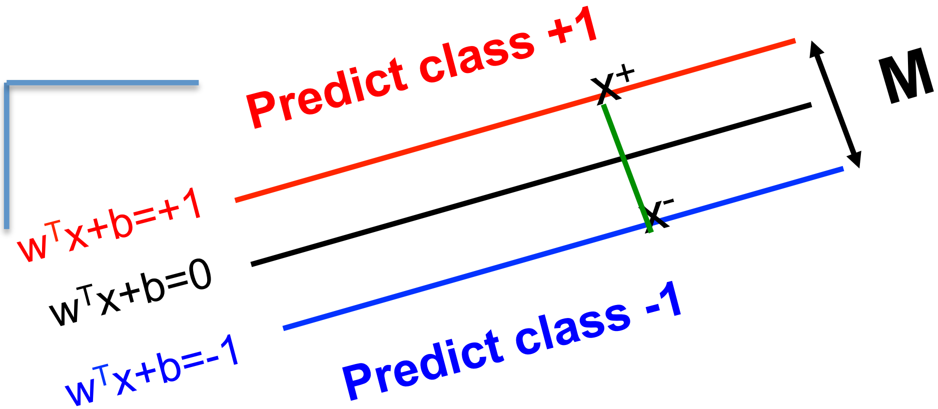
$$M = 2\frac{\sqrt{w^T w}}{w^T w} = \frac{2}{\sqrt{w^T w}}$$

We can now define M in terms of w and b

$\max(M)$

$\Rightarrow \min(w^T w)$

Finding the optimal parameters



$$M = \frac{2}{\sqrt{w^T w}} = \frac{2}{||\mathbf{w}||}$$

We can now search for the optimal parameters by finding a solution that:

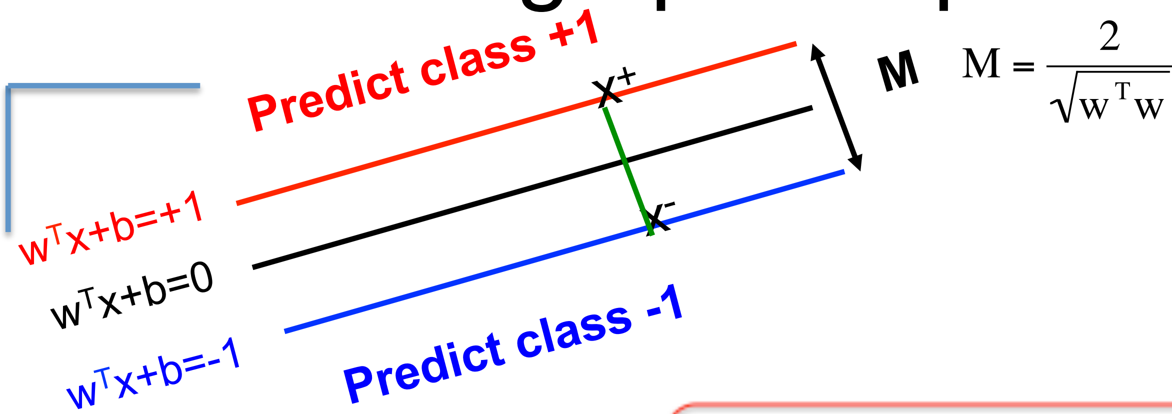
1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

Several optimization methods can be used: Gradient descent, simulated annealing, EM etc.

Today

- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Linearly Non-separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide

Optimization Step
i.e. learning optimal parameter for SVM



1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

Min $(w^T w)/2$

subject to the following constraints:

For all x in class + 1

$w^T x+b >= 1$

For all x in class - 1

$w^T x+b <= -1$

A total of n constraints if we have n input samples

min $(w^T w/2)$

s.t. $\begin{cases} w^T x_i + b \geq 1 & \text{pos} \\ w^T x_j + b \leq -1 & \text{neg} \end{cases}$

Support Vector Machine

Task	classification
↓	↓
Representation	Kernel Func K(xi, xj)
↓	↓
Score Function	Margin + Hinge Loss (optional)
↓	↓
Search/Optimization	QP with Dual form
↓	↓
Models, Parameters	Dual Weights

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

$$w = \sum_i \alpha_i x_i y_i$$

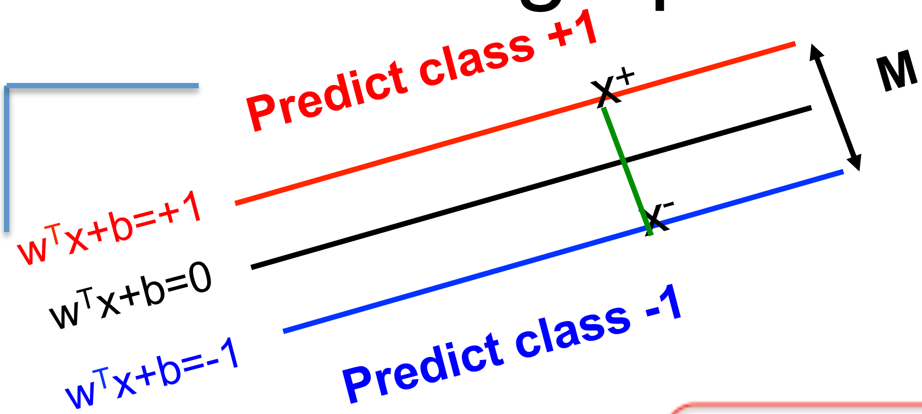
$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + C\sum_{i=1}^{n} \varepsilon_i$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1 - \varepsilon_i$$

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

Optimization Step
i.e. learning optimal parameter for SVM



$$M = \frac{2}{\sqrt{w^T w}}$$

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

Min $(w^T w)/2$

subject to the following constraints:

For all x in class + 1

$w^T x + b >= 1$ $y_i = 1$

For all x in class - 1

$w^T x + b <= -1$ $y_i = -1$

A total of n constraints if we have n input samples

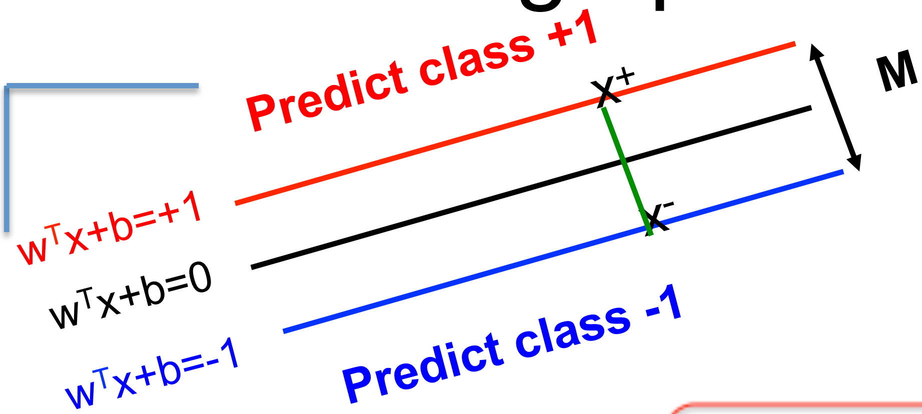
→ Pos $y_i = 1$, $w^T x_i + b \geq 1$

$y_i(w^T x_i + b) \geq 1$

→ neg $y_i = -1$, $w^T x_i + b \leq -1$

$y_i(w^T x_i + b) \geq 1$

Optimization Step i.e. learning optimal parameter for SVM



$$M = \frac{2}{\sqrt{w^T w}}$$

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes w^Tw)

Min $(w^Tw)/2$

subject to the following constraints:

For all x in class + 1

$w^Tx+b >= 1$

For all x in class - 1

$w^Tx+b <= -1$

A total of n constraints if we have n input samples

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1$

$w^T x_i$

Optimization Review: Ingredients

- Objective function
- Variables
- Constraints

**Find values of the variables
that minimize or maximize the objective function
while satisfying the constraints**

Optimization with Quadratic programming (QP)

Quadratic programming solves optimization problems of the following form:

$$\min_U \boxed{\frac{u^T R u}{2}} + d^T u + c$$

$u \rightarrow$ variable

$f(u) \rightarrow$ object

Quadratic term

subject to n inequality constraints:

$$a_{11}u_1 + a_{12}u_2 + \ldots \leq b_1$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$a_{n1}u_1 + a_{n2}u_2 + \ldots \leq b_n$$

$g_i(u) \rightarrow$ constraints

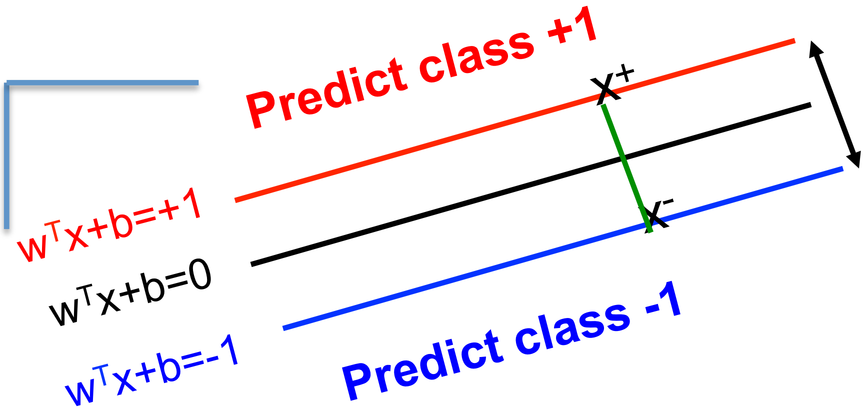
When a problem can be specified as a QP problem we can use solvers that are better than gradient descent or simulated annealing

and k equivalency constraints:

$$a_{n+1,1}u_1 + a_{n+1,2}u_2 + \ldots = b_{n+1}$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$a_{n+k,1}u_1 + a_{n+k,2}u_2 + \ldots = b_{n+k}$$

SVM as a QP problem

R as I matrix, d as zero vector, c as 0 value



$$M = \frac{2}{\sqrt{w^T w}}$$

$$\min_U \frac{u^T R u}{2} + d^T u + c$$

subject to n inequality constraints:

$$a_{11}u_1 + a_{12}u_2 + \ldots \le b_1$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$a_{n1}u_1 + a_{n2}u_2 + \ldots \le b_n$$

and k equivalency constraints:

$$a_{n+1,1}u_1 + a_{n+1,2}u_2 + \ldots = b_{n+1}$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$a_{n+k,1}u_1 + a_{n+k,2}u_2 + \ldots = b_{n+k}$$

Min $(w^Tw)/2$

subject to the following inequality constraints:

For all x in class + 1

$w^Tx+b >= 1$

For all x in class - 1

$w^Tx+b <= -1$

A total of n constraints if we have n input samples

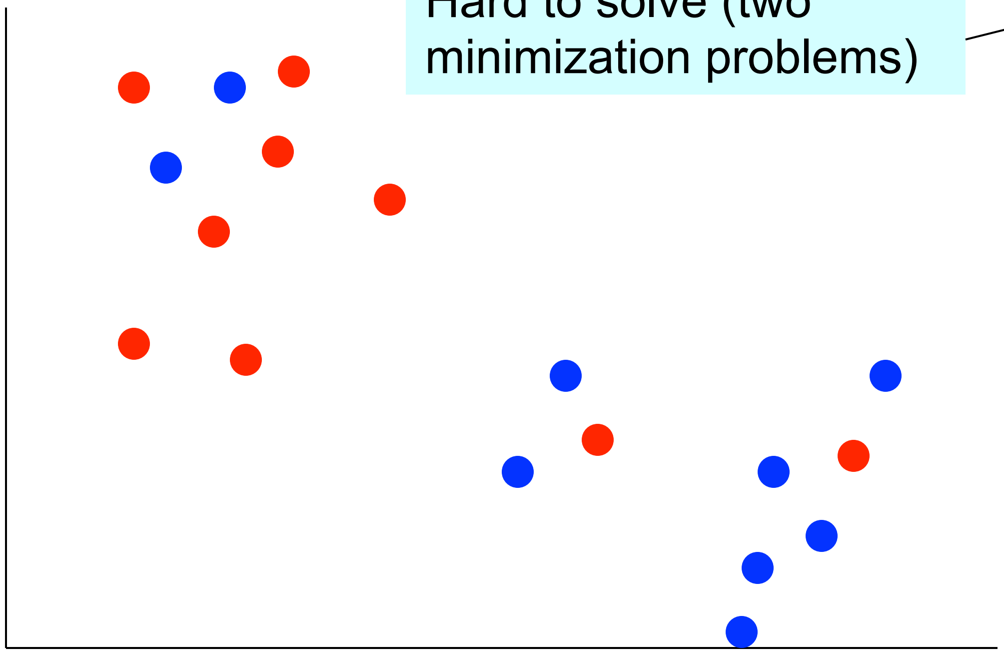
Today

- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Linearly Non-separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide

Linearly Non separable case

- So far we assumed that a linear plane can perfectly separate the points
- But this is not usally the case

 - noise, outliers



Hard to solve (two minimization problems)

How can we convert this to a QP problem?

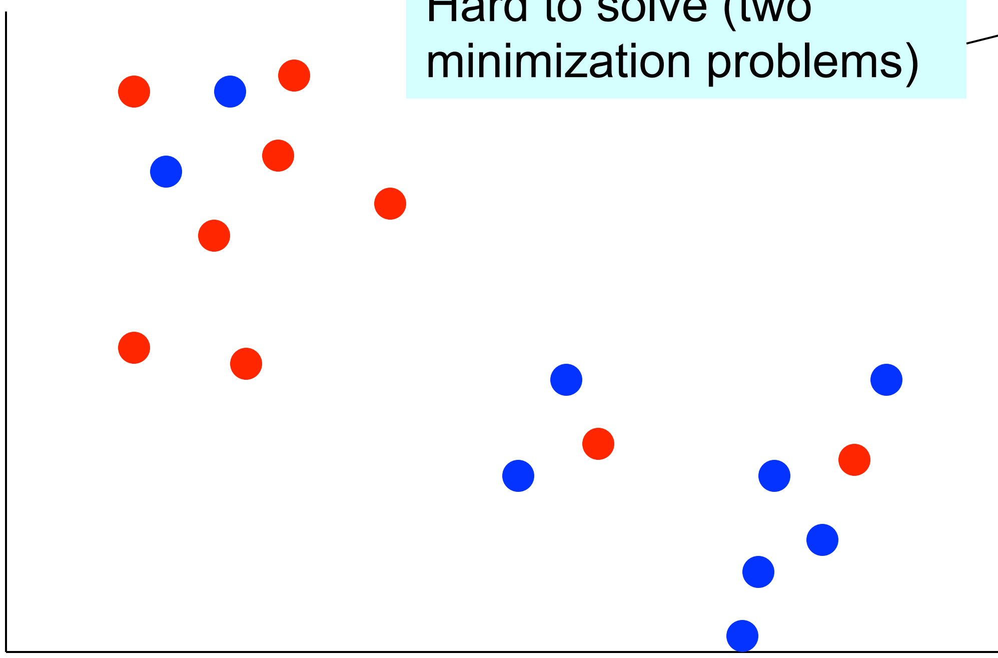
- Minimize training errors?

$$\begin{cases} \min\ w^T w \\ \min\ \#errors \end{cases}$$

Linearly Non separable case

- So far we assumed that a linear plane can perfectly separate the points
- But this is not usally the case

 - noise, outliers



Hard to solve (two minimization problems)

How can we convert this to a QP problem?

- Minimize training errors?

 min $w^T w$

 min #errors

- Penalize training errors:

 min $w^T w + C^*(\#errors)$

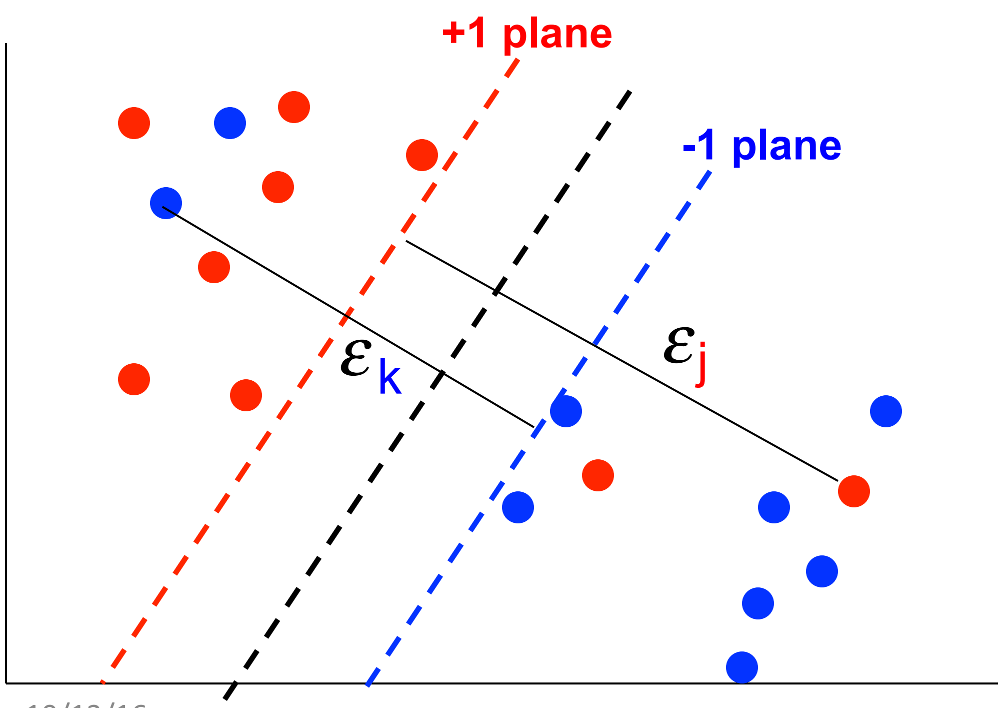
Hard to encode in a QP problem

Linearly Non separable case

- Instead of minimizing the number of misclassified points we can minimize the ***distance*** between these points and their correct plane

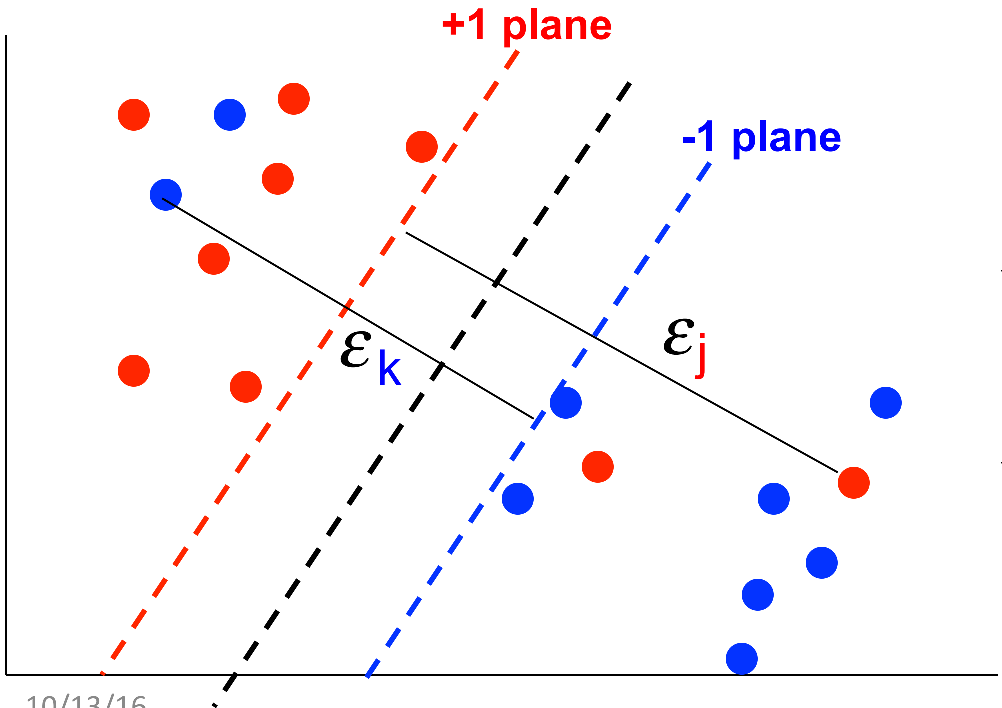
The new optimization problem is:

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$



Linearly Non separable case

- Instead of minimizing the number of misclassified points we can minimize the ***distance*** between these points and their correct plane



The new optimization problem is:

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

subject to the following inequality constraints:

For all x_i in class + 1

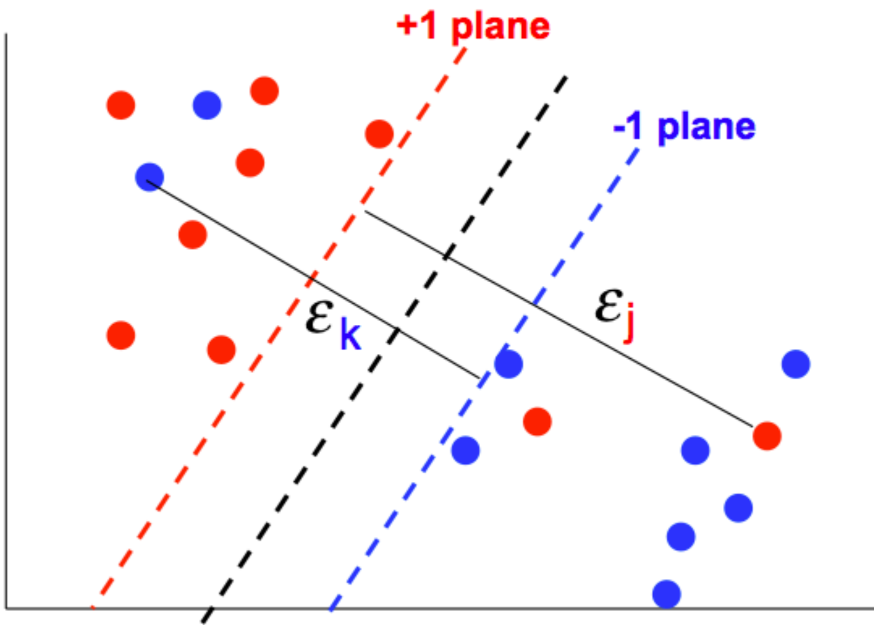
$w^T x_i + b >= 1 - \varepsilon_i$

For all x_i in class - 1

$w^T x_i + b <= -1 + \varepsilon_i$

Wait. Are we missing something?

Final optimization for linearly non-separable case



The new optimization problem is:

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

subject to the following inequality constraints:

For all x_i in class + 1

$w^T x_i + b >= 1 - \varepsilon_i$

For all x_i in class - 1

$w^T x_i + b <= -1 + \varepsilon_i$

A total of n constraints

n constraints

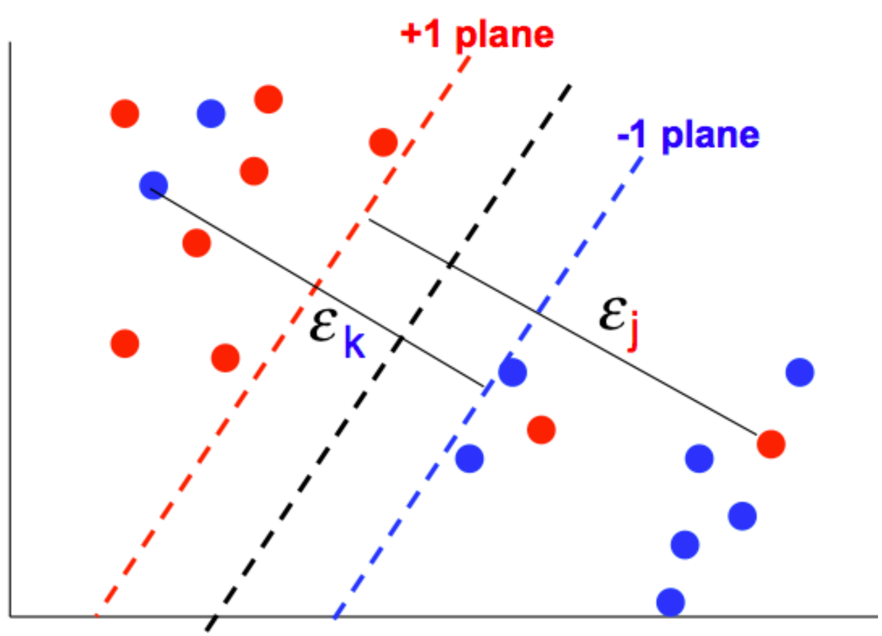
For all i

$$\varepsilon_i \geq 0$$

Another n constraints

n constraints

Final optimization for linearly non-separable case



The new optimization problem is:

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

hyperparam

subject to the following inequality constraints:

For all x_i in class + 1

$w^T x_i + b >= 1 - \varepsilon_i$

For all x_i in class - 1

$w^T x_i + b <= -1 + \varepsilon_i$

A total of n constraints

n

For all i

$$\varepsilon_i \geq 0$$

Another n constraints

n

Where are we ?

Two optimization problems: For the separable and non separable cases

$$\min_w \frac{w^T w}{2}$$

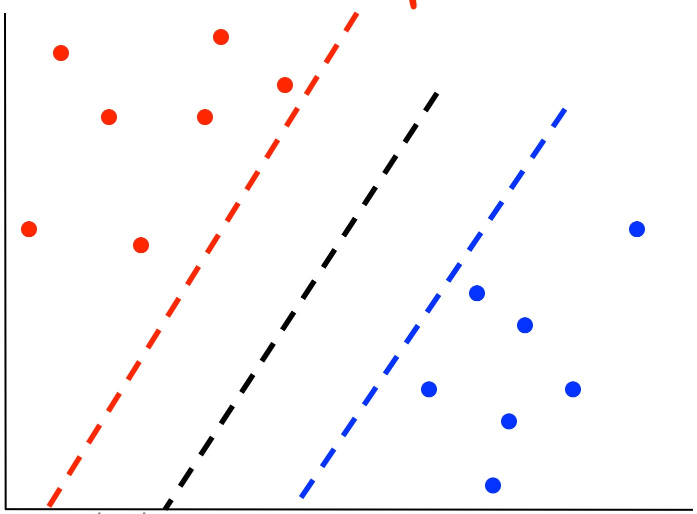
For all x in class + 1

$w^T x + b >= 1$

For all x in class - 1

$w^T x + b <= -1$

separable



$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

$w^T x_i + b >= 1 - \varepsilon_i$

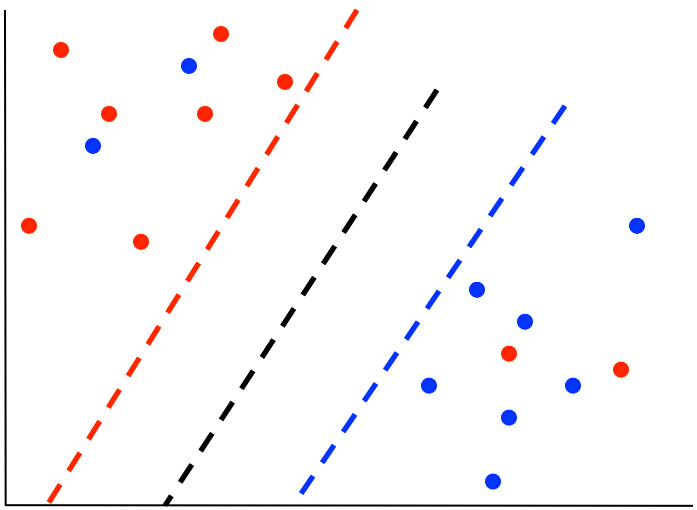
For all x_i in class - 1

$w^T x_i + b <= -1 + \varepsilon_i$

For all i

$$\varepsilon_i \geq 0$$

non separable

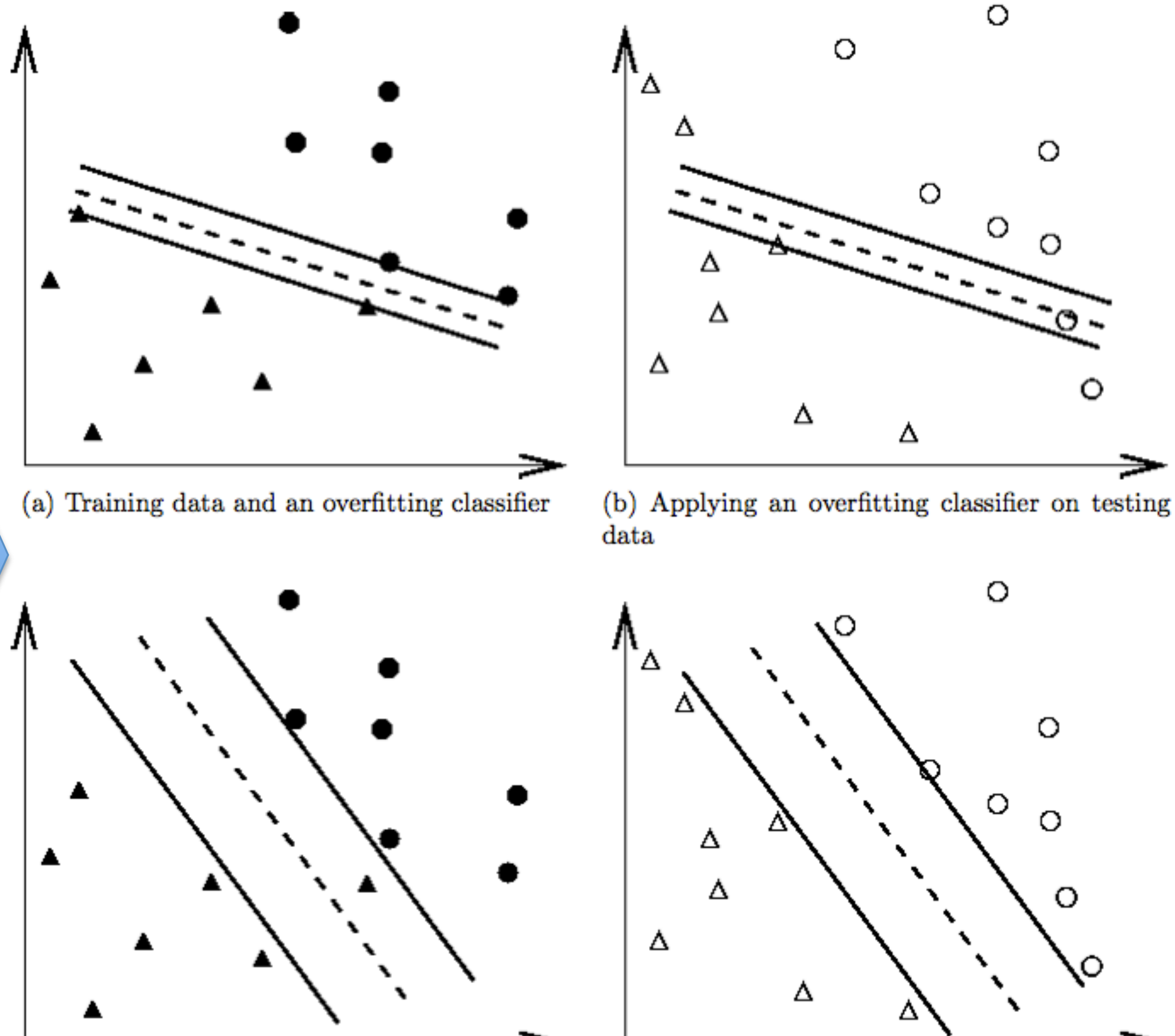


Model Selection, find right C

large C

small C

Select the right penalty parameter C



(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier

(d) Applying a better classifier on testing data

Today

- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Linearly non-separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide

Where are we ?

Two optimization problems: For the separable and non separable cases

Min $(w^T w)/2$

For all x in class + 1

$w^T x+b >= 1$

For all x in class - 1

$w^T x+b <= -1$

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

$w^T x_i+b >= 1-\varepsilon_i$

For all x_i in class - 1

$w^T x_i+b <= -1+\varepsilon_i$

For all i

$$\varepsilon_i \geq 0$$

- Instead of solving these QPs directly we will solve a dual formulation of the SVM optimization problem

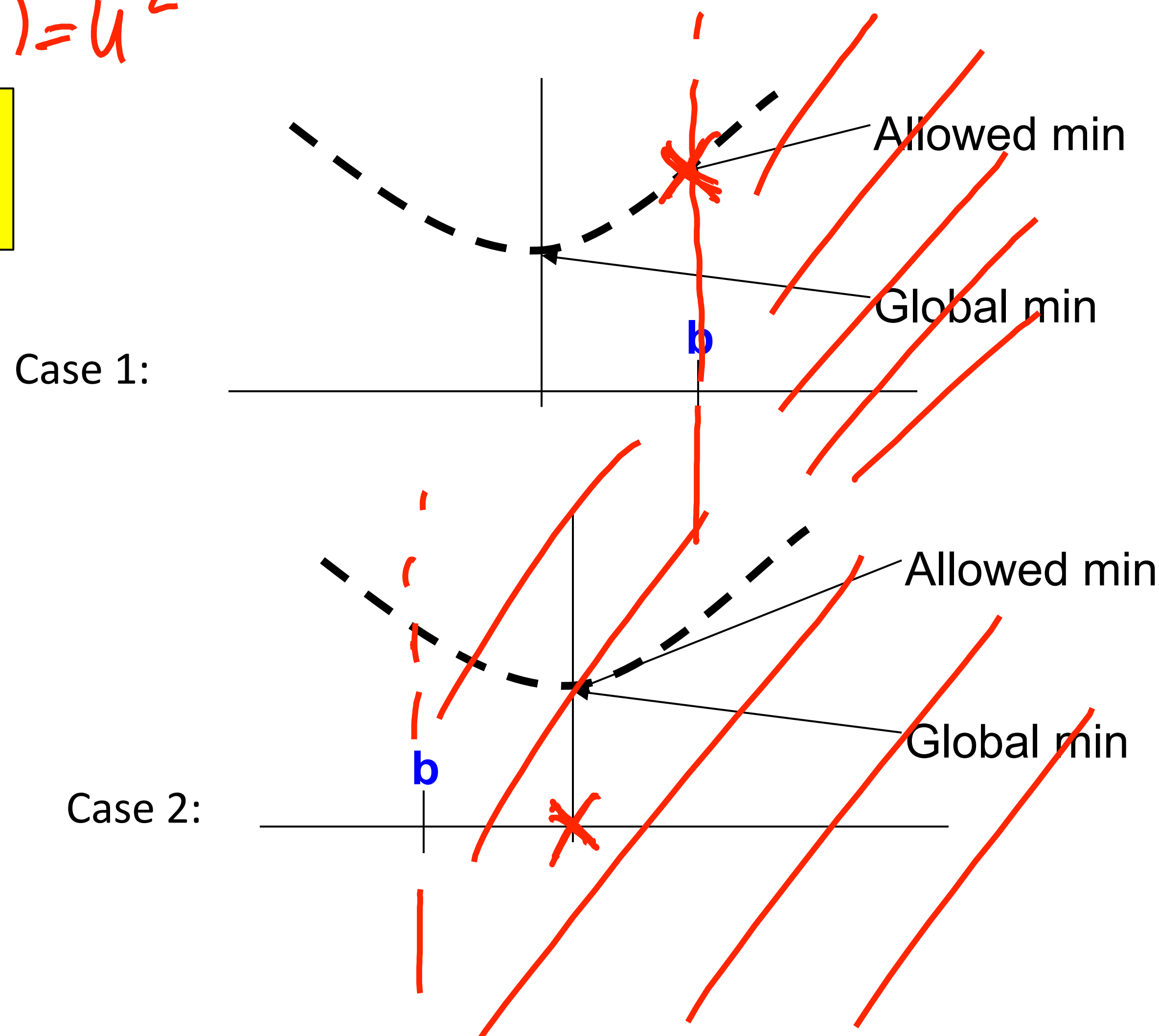
- The main reason for switching to this type of representation is that it would allow us to use a neat trick that will make our lives easier (and the run time faster)

Optimization Review:
Constrained Optimization

$f(u) = u^2$

$\min_u u^2$

s.t. u >= b



Optimization Review:
Constrained Optimization

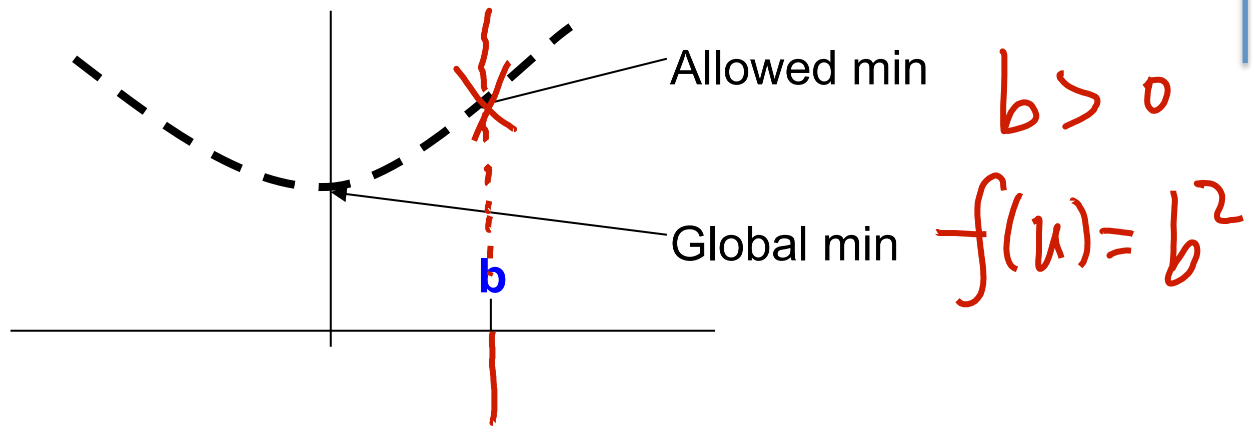
$f(u)$

$\min_u u^2$

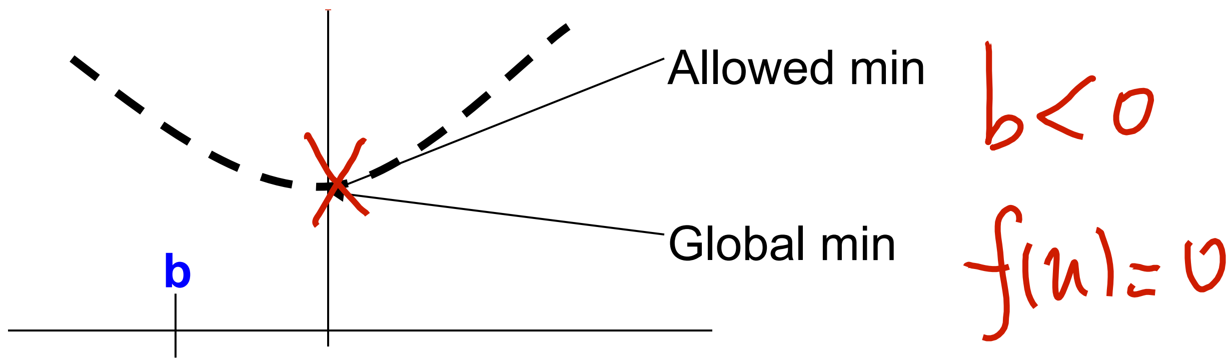
s.t. $u >= b$

[Subject to]

Case 1:



Case 2:



Optimization Review:
Constrained Optimization with Lagrange

- When with equal constraints
- ➔ optimize *f(x)*, subject to $g_i(x) <= 0$

$i = 1, \ldots, k$

- We can solve the above using the "Method of Lagrange multipliers"
  - convert to a higher-dimensional problem
  - i.e., to Minimize

$$f(x) + \sum_{i=1}^{k} \lambda_i g_i(x)$$

w.r.t. $(x_1 \ldots x_n ; \lambda_1 \ldots \lambda_k)$

Introducing a Lagrange multiplier for each constraint
Construct the Lagrangian for the original optimization problem

Optimization Review:
Constrained Optimization with Lagrange

- When with equal constraints
- ➔ optimize *f(x)*, subject to $g_i(x) <= 0$

- We can solve the above using the "Method of Lagrange multipliers"
  - convert to a higher-dimensional problem
  - i.e., to Minimize

(w, b)

$(x_1, x_2, \ldots, x_n)$

$\alpha_1, \alpha_2, \ldots n \ldots \alpha_n$

$$f(x) + \sum \lambda_i g_i(x) \qquad \alpha_i$$

w.r.t. $(x_1 \ldots x_n ; \lambda_1 \ldots \lambda_k)$

Introducing a Lagrange multiplier for each constraint

Construct the Lagrangian for the original optimization problem

$\min_u u^2$

s.t. $u >= b$

$\min_u u^2$

s.t. $u >= b$

$$\Rightarrow \begin{cases} \min_{u} f_0(u) = u^2 \\ \text{s.t.} \quad b - u \le 0 \end{cases}$$

primal problem

$\min_u u^2$

s.t. u >= b

① $\begin{cases} \min\limits_{u} f_0(u) = u^2 \\ \text{s.t.} \quad b - u \le 0 \end{cases}$

$\Rightarrow$ multiplier variable

② $L(\underset{1\times 1}{u}, \underset{1\times 1}{\alpha}) = u^2 + \underbrace{\alpha}_{\geq 0}\underbrace{(b-u)}_{\leq 0}$

$\min_u u^2$

s.t. $u >= b$

① $\begin{cases} \min_u f_0(u) = u^2 \\ \text{s.t.} \quad b - u \le 0 \end{cases}$

② $L(\underset{1\times 1}{u}, \underset{1\times 1}{\alpha}) = u^2 + \underbrace{\alpha}_{\geq 0} \underbrace{(b-u)}_{\leq 0}$

③ $\dfrac{\partial L(u,\alpha)}{\partial u} = 2u - \alpha = 0$

$u = \dfrac{\alpha}{2}$

$\rightarrow \operatorname{argmin}_u L(u, \alpha)$

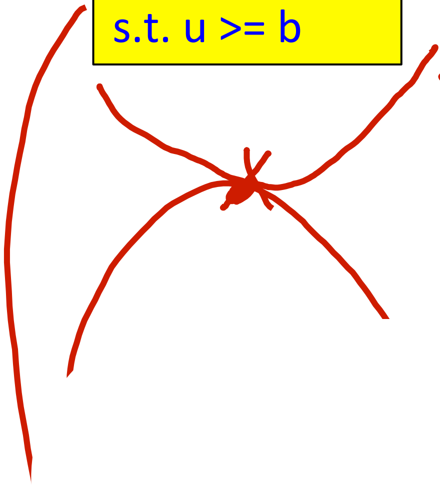
$\min_u u^2$

s.t. $u >= b$

$$g(\alpha) = L(u,\alpha) = \frac{\alpha^2}{4} + \alpha\left(b - \frac{\alpha}{2}\right)$$

$\min_u u^2$

s.t. u >= b



$f(u)$

$$g(\alpha) = L(u, \alpha) = \frac{\alpha^2}{4} + \alpha\left(b - \frac{\alpha}{2}\right)$$

$$u = \alpha/2$$

$$g(\alpha) = -\frac{\alpha^2}{4} + b\alpha$$

$\min_u u^2$

s.t. $u >= b$

$$g(\alpha) = L(u,\alpha) = \frac{\alpha^2}{4} + \alpha\left(b - \frac{\alpha}{2}\right)$$

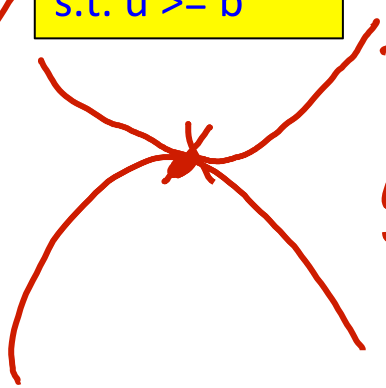
$$u = \alpha/2$$

$$g(\alpha) = -\frac{\alpha^2}{4} + b\alpha$$

$$\frac{\partial g(\alpha)}{\partial \alpha} = -\frac{\alpha}{2} + b = 0, \quad \alpha \geq 0$$

$\min_u u^2$

s.t. $u >= b$



$$g(\alpha) = L(u,\alpha) = \frac{\alpha^2}{4} + \alpha\left(b - \frac{\alpha}{2}\right)$$

$$u = \alpha/2$$

$$g(\alpha) = -\frac{\alpha^2}{4} + b\alpha$$

$$\frac{\partial g(\alpha)}{\partial \alpha} = -\frac{\alpha}{2} + b = 0, \quad \alpha \geq 0$$

$\Rightarrow$ Dual $\begin{cases} b > 0, & \alpha = 2b, \quad g(\alpha) = b^2 \\ b < 0, & \alpha = 0, \quad g(\alpha) = 0 \end{cases}$

$\Rightarrow$ Primal $\begin{cases} b > 0, & f(u) = b^2 \quad u = b \\ b < 0, & f(u) = 0 \quad u = 0 \end{cases}$

Optimization Review: Lagrangian Duality

- The Primal Problem

Primal:

$$\begin{aligned} \min_w \quad & f_0(w) \\ \text{s.t.} \quad & f_i(w) \le 0, \quad i = 1, \ldots, k \end{aligned}$$

The generalized Lagrangian:

$$\mathcal{L}(w,\alpha) = f_0(w) + \sum_{i=1}^{k} \alpha_i f_i(w)$$

the *a*'s ($a_i \ge 0$) are called the Lagarangian multipliers

Lemma:

$$\max_{\alpha, \alpha_i \ge 0} \mathcal{L}(w,\alpha) = \begin{cases} f_0(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{o/w} \end{cases}$$

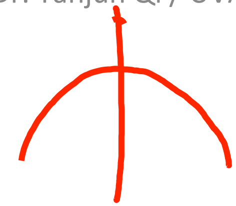
A re-written Primal:

$$\min_w \max_{\alpha, \alpha_i \ge 0} \mathcal{L}(w,\alpha)$$

primal : $\min_{w} \max_{\alpha} L(w, \alpha)$

Dual : $\underbrace{\max_{\alpha} \min_{w} L(w, \alpha)}$

$\Rightarrow \max_{\alpha} g(\alpha)$



$$f(x): \begin{cases} \min x^2 \\ \text{s.t. } x \geq b \end{cases}$$

$$g(\alpha): \begin{cases} \max -\dfrac{\alpha^2}{4} + b\alpha = \max\left\{ -\underbrace{\left(\cancel{\tfrac{\alpha}{2}} - b\right)^2} + \underbrace{b^2} \right\} \\ \text{s.t. } \alpha \geq 0 \end{cases}$$

$$\begin{cases} \text{if}_{\,b \geq 0} \; b = \alpha/2, \quad x = b, \quad g = b^2 \\ \text{if}_{\,b < 0} \; b \neq \alpha/2, \quad \alpha = 0, \quad g = 0 \end{cases}$$

$$\Rightarrow \alpha(b - x) = 0$$

KKT condition

$\Rightarrow$ Dual $\begin{cases} b > 0, & \alpha = 2b, & g(\alpha) = b^2 \\ b < 0, & \alpha = 0, & g(\alpha) = 0 \end{cases}$

$\Rightarrow$ Primal $\begin{cases} b > 0, & f(u) = b^2 & u = b \\ b < 0, & f(u) = 0 & u = 0 \end{cases}$

strong duality

$\Rightarrow$ Optim-Dual = Primal_Optim

when [KKT] conditions hold $\begin{cases} \alpha(b - u) = 0 \\ \end{cases}$

Optimization Review: Lagrangian Duality, cont.

- Recall the Primal Problem:

$$\min_w \max_{\alpha,\alpha_i \geq 0} L(w,\alpha)$$

- The Dual Problem:

$$\max_{\alpha,\alpha_i \geq 0} \min_w L(w,\alpha)$$

- **Theorem (weak duality):**

$$d^* = \max_{\alpha,\alpha_i \geq 0} \min_w L(w,\alpha) \quad \leq \quad \min_w \max_{\alpha,\alpha_i \geq 0} L(w,\alpha) = p^*$$

- **Theorem (strong duality):**

Iff there exist a saddle point of $L(w,\alpha)$

we have $d^* = p^*$

An alternative representation of the SVM QP

- We will start with the linearly separable case
- Instead of encoding the correct classification rule and constraint we will use Lagrange multiplies to encode it as part of the our minimization problem

Min $(w^Tw)/2$

s.t.

$(w^Tx_i+b)y_i >= 1$

Recall that Lagrange multipliers can be applied to turn the following problem:

w^Tw

$$L_{primal} = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 \right)$$

An alternative representation of the SVM QP

- We will start with the linearly separable case
- Instead of encoding the correct classification rule and constraint we will use Lagrange multipliers to encode it as part of the our minimization problem

$f(w)$ $f(u)$

Min $(w^Tw)/2$ $\Leftarrow u^2$

s.t. s.t.

$(w^Tx_i+b)y_i >= 1$ $\Leftarrow u \geq b$

n constraints

$\Downarrow$ $b - u \leq 0$

$+\ \alpha_i\left(1-(w^Tx_i+b)y_i\right) \leq 0$

Recall that Lagrange multipliers can be applied to turn the following problem:

$$L_{primal} = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 \right)$$

$$\min_{w,b} \max_{\alpha} \frac{\mathrm{w}^{\mathrm{T}}\mathrm{w}}{2} - \sum_i \alpha_i [(\mathrm{w}^{\mathrm{T}} x_i + b) y_i - 1]$$

$$\alpha_i \geq 0 \qquad \forall i$$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w - \sum_i^{\text{train}} \alpha_i x_i y_i = 0$$

$$\min_{w,b} \max_{\alpha} \left\{ \frac{w^T w}{2} - \sum_i \alpha_i [(w^T x_i + b) y_i - 1] \right\} \Rightarrow \max_{\alpha} \min_{w,b} L(w,b,\alpha)$$

$$\alpha_i \geq 0 \qquad \forall i$$

train

$$\begin{cases} \frac{\partial L}{\partial w} = 0 \Rightarrow w - \sum_i \alpha_i x_i y_i = 0 \\ \frac{\partial L}{\partial b} = 0 \Rightarrow \sum \alpha_i y_i = 0 \end{cases}$$

The Dual Problem

$$\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha)$$ Dual formulation

- We minimize L with respect to w and b first:

$$\nabla_w L(w,b,\alpha) = w - \sum_{i=1}^{train} \alpha_i y_i x_i = 0, \quad (*)$$

$$\nabla_b L(w,b,\alpha) = \sum_{i=1}^{train} \alpha_i y_i = 0, \quad (**)$$

Note that (*) implies: $w = \sum_{i=1}^{train} \alpha_i y_i x_i$ $\quad (***)$

- Plus (***) back to L , and using (**), we have:

$$L(w,b,\alpha) = \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j=1} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$L_{primal} = \frac{1}{2}\left\|\mathbf{w}\right\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 \right)$$

$$L_{dual} = \frac{1}{2}\left(\sum_i \alpha_i x_i y_i\right)^T \left(\sum_j \alpha_j x_j y_j\right) - \sum_i \alpha_i y_i \left(\sum_j \alpha_j x_j y_j\right)^T x_i$$

$$- \underbrace{\sum \alpha_i y_i}_{0} b + \underbrace{\sum \alpha_i}$$

$$= \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \left(x_i^T x_j \right)$$

Summary: Dual for SVM

Solving for **w** that gives maximum margin:

1. Combine objective function and constraints into new objective function, using Lagrange multipliers \alpha$_i$

$$L_{primal} = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right)$$

2. To minimize this Lagrangian, we take derivatives of **w** and *b* and set them to 0:

Summary: Dual for SVM

3. Substituting and rearranging gives the dual of the Lagrangian:

$$L_{dual} = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

which we try to maximize (not minimize).

4. Once we have the *alpha*$_i$, we can substitute into previous equations to get **w** and *b*.
5. This defines **w** and *b* as linear combinations of the training data.

$$w = \sum_{i=1}^{train} \alpha_i y_i x_i$$

Summary: Dual SVM for linearly separable case

Substituting w into our target function and using the additional constraint we get:

Dual formulation

n α_i

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j \mathbf{x}_\mathbf{i}^T \mathbf{x}_\mathbf{j}$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Min $(w^Tw)/2$

subject to the following inequality constraints:

For all x in class + 1

$w^Tx+b >= 1$

For all x in class - 1

$w^Tx+b <= -1$

A total of n constraints if we have n input samples

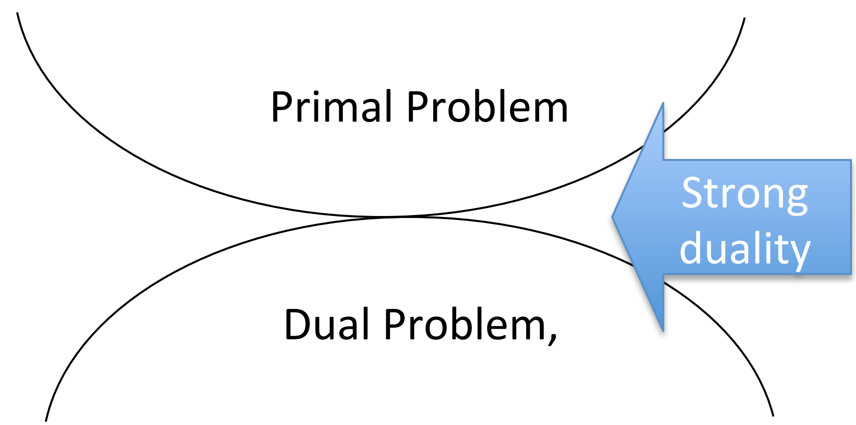
Easier than original QP, more efficient algorithms exist to find a_i

Optimization Review: Dual Problem

- Solving dual problem if the dual form is easier than primal form

- Need to change primal minimization to dual maximization (OR → Need to change primal maximization to dual minimization)

- Only valid when the original optimization problem is convex/ concave (strong duality)



EXTRA

Optimization Review: Lagrangian (even more general standard form)

standard form problem (not necessarily convex)

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array}$$

variable $x \in \mathbf{R}^n$, domain $\mathcal{D}$, optimal value $p^\star$

Lagrangian: $L : \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}$, with $\mathbf{dom}\, L = \mathcal{D} \times \mathbf{R}^m \times \mathbf{R}^p$,

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

- weighted sum of objective and constraint functions
- λ_i is Lagrange multiplier associated with $f_i(x) \leq 0$
- ν_i is Lagrange multiplier associated with $h_i(x) = 0$

From Stanford "Convex Optimization — Boyd & Vandenberghe

Optimization Review: Lagrange dual function

Lagrange dual function: $g : \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}$,

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu)$$

$$= \inf_{x \in \mathcal{D}} \underbrace{\left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right)}_{L(x,\lambda,\nu)}$$

≥ 0 ≤ 0 0

g is concave, can be $-\infty$ for some λ, ν

lower bound property: if $\lambda \succeq 0$, then $g(\lambda, \nu) \le p^\star$

Inf(.): greatest lower bound

proof: if $\tilde{x}$ is feasible and $\lambda \succeq 0$, then

$$f_0(\tilde{x}) \ge L(\tilde{x}, \lambda, \nu) \ge \inf_{x \in \mathcal{D}} L(x, \lambda, \nu) = g(\lambda, \nu)$$

minimizing over all feasible $\tilde{x}$ gives $p^\star \ge g(\lambda, \nu)$

Optimization Review:

Complementary slackness

assume strong duality holds, $x^\star$ is primal optimal, $(\lambda^\star, \nu^\star)$ is dual optimal

inf (.): greatest lower bound

$$
\begin{aligned}
f_0(x^\star) = g(\lambda^\star, \nu^\star) &= \inf_x \left(f_0(x) + \sum_{i=1}^m \lambda_i^\star f_i(x) + \sum_{i=1}^p \nu_i^\star h_i(x) \right) \\
&\leq f_0(x^\star) + \sum_{i=1}^m \lambda_i^\star f_i(x^\star) + \sum_{i=1}^p \nu_i^\star h_i(x^\star) \\
&\leq f_0(x^\star)
\end{aligned}
$$

obj $\Rightarrow f(u^*)$ $\quad g(\alpha^*)$

$u^* \begin{cases} b>0 & u^*=b \\ b<0 & u^*=0 \end{cases}$ $\quad \alpha^*=2b$ $\quad \alpha^*=0$

$=0$

$\geq 0 \quad \leq 0$

$\Rightarrow \alpha_i(1-(w^T x_i+b)y_i)$

① $\alpha_i = 0$

hence, the two inequalities hold with equality

- $x^\star$ minimizes $L(x, \lambda^\star, \nu^\star)$
- $\lambda_i^\star f_i(x^\star) = 0$ for $i = 1, \ldots, m$ (known as complementary slackness):

② $\alpha_i > 0$

$$\lambda_i^\star > 0 \Longrightarrow f_i(x^\star) = 0, \qquad f_i(x^\star) < 0 \Longrightarrow \lambda_i^\star = 0$$

Optimization Review:

Karush-Kuhn-Tucker (KKT) conditions

the following four conditions are called KKT conditions (for a problem with differentiable f_i, h_i):

1. primal constraints: $f_i(x) \leq 0$, $i = 1, \ldots, m$, $h_i(x) = 0$, $i = 1, \ldots, p$
2. dual constraints: $\lambda \succeq 0$
3. complementary slackness: $\lambda_i f_i(x) = 0$, $i = 1, \ldots, m$
4. gradient of Lagrangian with respect to x vanishes:

$$\nabla f_0(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + \sum_{i=1}^{p} \nu_i \nabla h_i(x) = 0$$

Key for SVM Dual

from page 5-17: if strong duality holds and x, λ, ν are optimal, then they must satisfy the KKT conditions

NOT EXTRA

KKT Condition for Strong Duality

$\min_w \max_\alpha L(w,\alpha)$

Primal Problem

Strong duality

Dual Problem, $\max_\alpha \min_w L(w,\alpha)$

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0, & i = 1, \ldots, m \\ & h_i(x) = 0, & i = 1, \ldots, p \end{array}$$

Lagrangian: $L : \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}$, with $\mathbf{dom}\, L = \mathcal{D} \times \mathbf{R}^m \times \mathbf{R}^p$,

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

complementary slackness: $\lambda_i f_i(x) = 0,\ i = 1, \ldots, m$

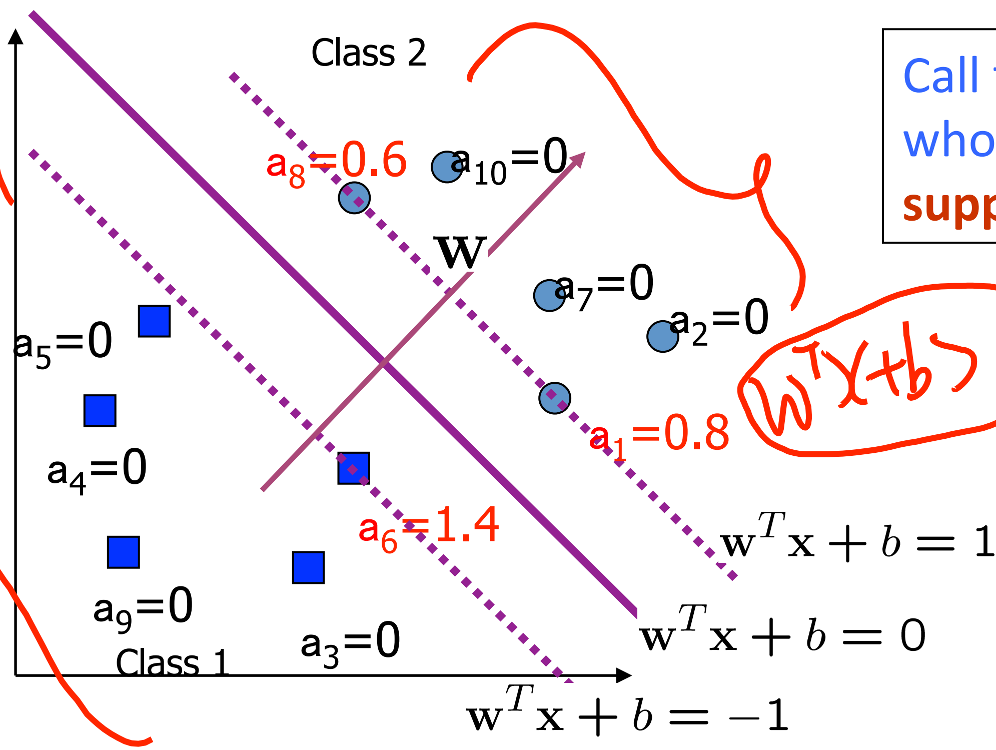
Key for SVM Dual

KKT => Support vectors

- Note the KKT condition --- only a few a_i's can be nonzero!!

$$\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i+b)-1\right)=0,\quad i=1,\dots,n$$

$w^Tx+b<-1$



Call the training data points whose a_i's are nonzero the **support vectors** (SV)

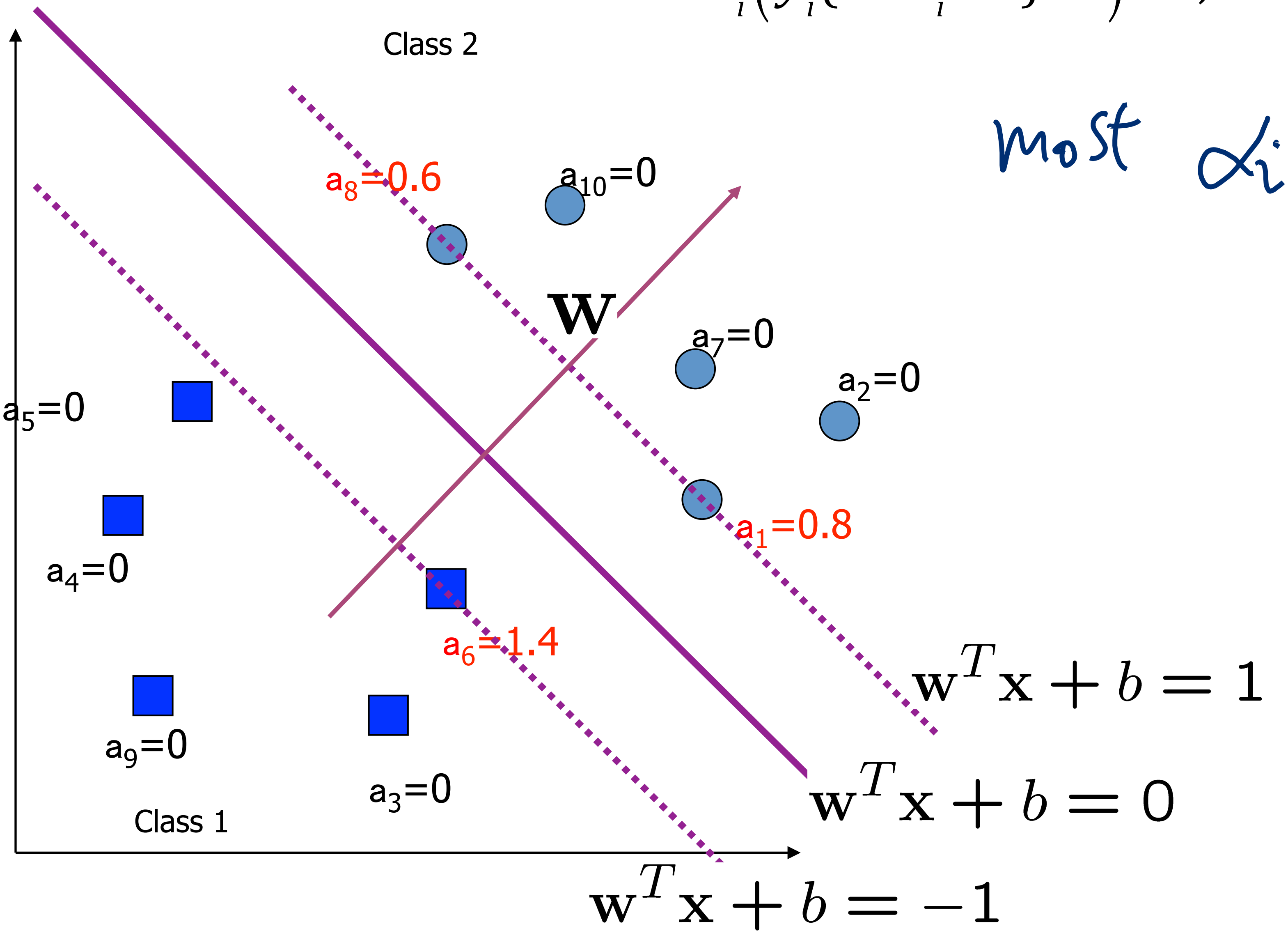
$w^Tx+b>1$

$\boxed{y_i(w^Tx_i+b)-1} > 1$

for most $\alpha_i \Rightarrow \alpha_i = 0$ >0

$$\alpha_i\left(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1\right)=0,\quad i=1,\ldots,n$$

most $\alpha_i = 0$



Dual SVM for linearly separable case – Training

$$\sum_{i=1}^{n}\sum_{j=1}^{n}$$

Our dual target function: $\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j \mathbf{x_i}^T \mathbf{x_j}$

Dot product for all training samples

$$\sum_i \alpha_i \mathrm{y_i} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

$$\begin{array}{c|ccccc} & 1 & 2 & \cdots & j & \cdots & n \\ \hline 1 & & & & \vdots & & \\ 2 & & & & \vdots & & \\ 3 & & & & \vdots & & \\ i & \cdots & \cdots & \cdots & x_i^T x_j & \cdots & \cdots \\ \vdots & & & & \vdots & & \\ n & & & & \vdots & & \end{array} \quad n \times n$$

matrix

Dual SVM for linearly separable case – Testing

$\{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n\}$

Dot product with ("all" ??) training samples

To evaluate a new sample x_{ts} we need to compute:

$$w^T x_{ts} + b = \sum_{i=1}^{n} \alpha_i y_i \underbrace{\mathbf{x_i}^T \mathbf{x}_{ts}} + b$$

$O(n \times \text{Inner})$

$y_{ts} = \text{sign}(W^T x + b)$

$$\widehat{y_{ts}} = \text{sign}\left(\underbrace{\sum_{i \in SupportVectors}} \alpha_i y_i \left(\mathbf{x}_i^T \mathbf{x}_{ts} \right) + b \right)$$

$\forall \alpha_i > 0$

$O(\#SV \times \text{Inner})$

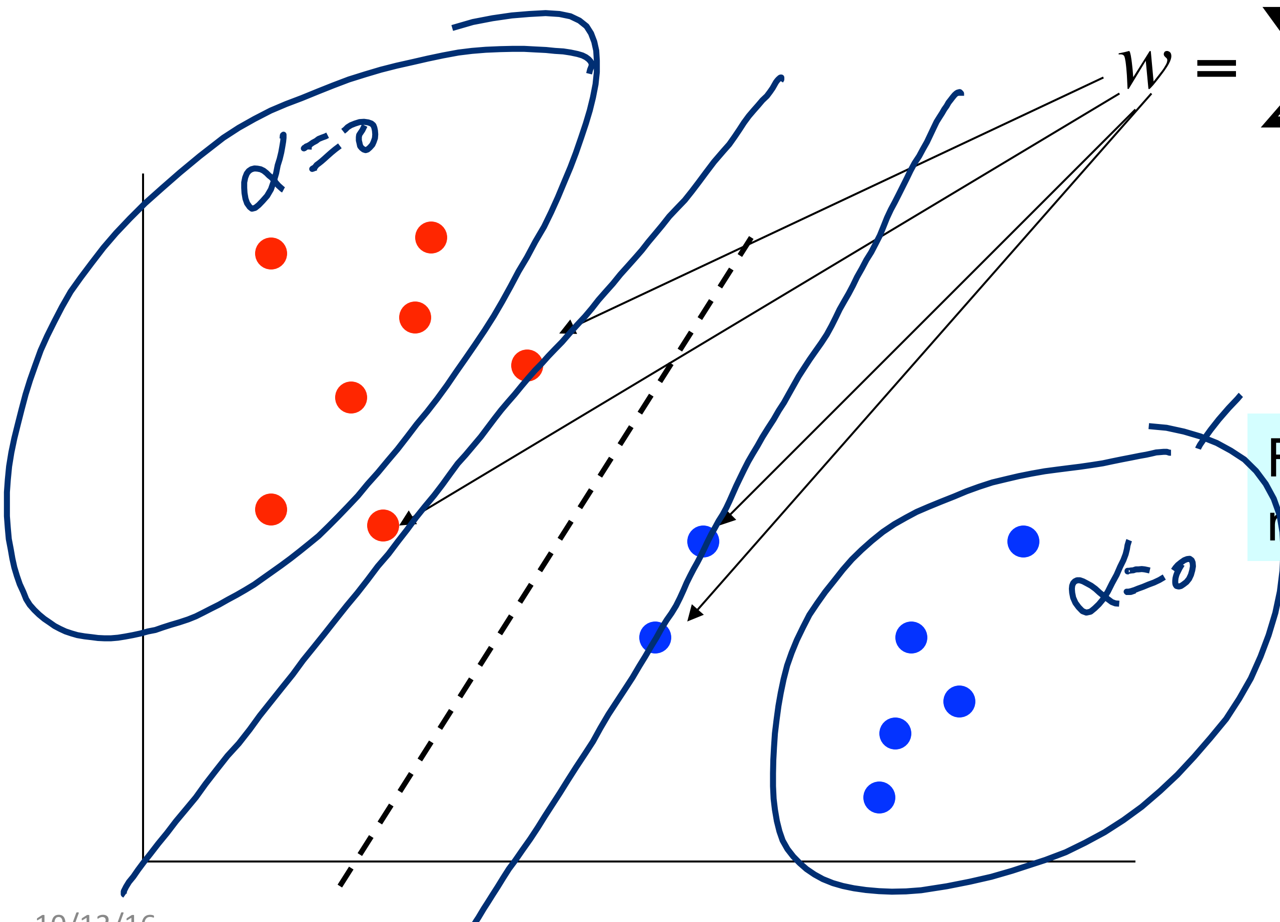
Dual SVM - interpretation

training points

$$w = \sum_i \alpha_i x_i y_i$$

$\Leftarrow \alpha_i > 0$

$\cancel{\alpha_i \geq 0}$



For α_i that are 0, no influence

$\varepsilon_i \; \{i=1,\dots,n\}$

Dual formulation for linearly non-separable case

Dual target function:

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_i \alpha_i y_i = 0$$

$$C > \alpha_i \geq 0, \forall i$$

The only difference is that the \alpha are now bounded

Hyperparameter C should be tuned through k-folds CV

This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on a_i now

Once again, efficient algorithm exist to find a_i

Dual formulation for linearly non-separable case

Dual target function:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$C > \alpha_i \geq 0, \forall i$$

Hyperparameter C should be tuned through k-folds CV

The only difference is that the \alpha are now bounded

To evaluate a new sample x_{test} we need to compute:

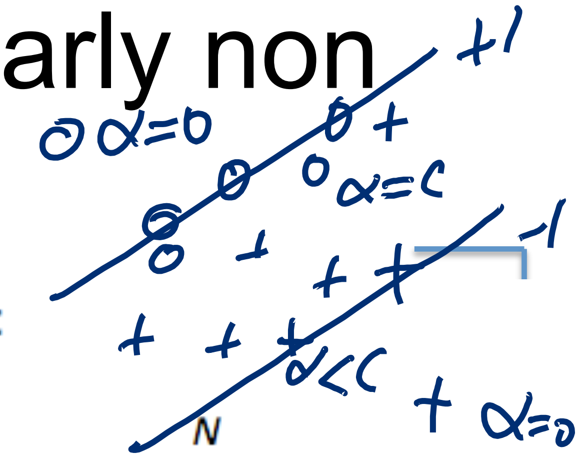
$$w^T x_{test} + b = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_{test} + b$$

This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on a_i now

Once again, efficient algorithm exist to find a_i

EXTRA

Dual formulation for linearly non separable case



Substituting (1), (2), and (3) into the Lagrange, we have:

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k, \text{ with } 0 \le \alpha_i \le C \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0. \quad (4)$$

- $\hat{\alpha}_i > 0$: which implies $y_i(x_i^T \hat{\mathbf{w}} + \hat{b}) - 1 + \hat{\xi}_i = 0$ according to (5). These points are the *support vectors*.
  - $\hat{\xi}_i = 0$: which implies $\hat{\mu}_i > 0$ from (6) and so $\hat{\alpha}_i < C$ from (3). There are the support points which lie on the edge of the margin.
  - $\hat{\xi}_i > 0$: which implies $\hat{\mu}_i = 0$ from (6) and so $\hat{\alpha}_i = C$ from (3). There are the support points which violate the margin.
- $\hat{\alpha}_i = 0$: These points are not support vectors, which play no role in determining the hyperplane.

Fast SVM Implementations

- SMO: Sequential Minimal Optimization
- SVM-Light
- LibSVM
- BSVM
- ……

SMO: Sequential Minimal Optimization

$$\underset{\alpha_1,\alpha_2,\ldots,\alpha_n}{\operatorname{argmax}}\left\{\sum \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j\right\}$$

- Key idea
  - Divide the large QP problem of SVM into a series of smallest possible QP problems, which can be solved analytically and thus avoids using a time-consuming numerical QP in the loop (a kind of SQP method).
  - Space complexity: O(n).
  - Since QP is greatly simplified, most time-consuming part of SMO is the evaluation of decision function, therefore it is very fast for linear SVM and sparse data.

$$\alpha_i\left(y_i(\underbrace{w^T x_i + b}) - 1\right) = 0$$

SMO

- At each step, SMO chooses 2 Lagrange multipliers to jointly optimize, find the optimal values for these multipliers and updates the SVM to reflect the new optimal values.
- Three components
  - An analytic method to solve for the two Lagrange multipliers
  - A heuristic for choosing which (next) two multipliers to optimize
  - A method for computing b at each step, so that the KTT conditions are fulfilled for both the two examples (corresponding to the two multipliers)

Choosing Which Multipliers to Optimize

- First multiplier
  - Iterate over the entire training set, and find an example that violates the KTT condition.
- Second multiplier
  - Maximize the size of step taken during joint optimization.
  - $|E_1-E_2|$, where E_i is the error on the *i*-th example.

NOT EXTRA

Today

- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide

Dual SVM for linearly separable case – Training / Testing

Our dual target function: $\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y_i} y_j \mathbf{x_i}^T \mathbf{x_j}$

$$\sum_i \alpha_i \mathrm{y_i} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dot product for all training samples

Dot product with ("all" ??) training samples

To evaluate a new sample x_{ts} we need to compute:

$$\mathrm{w}^{\mathrm{T}} x_{ts} + b = \sum_{\mathrm{i}} \alpha_i \mathrm{y_i} \mathbf{x_i}^T \mathbf{x}_{ts} + b$$

$$\widehat{y_{ts}} = \mathrm{sign}\left(\sum_{i \in SupportVectors} \alpha_i y_i \left(\mathbf{x}_i^T \mathbf{x}_{ts} \right) + b \right)$$

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j \mathbf{x}_\mathbf{i}^T \mathbf{x}_\mathbf{j}$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

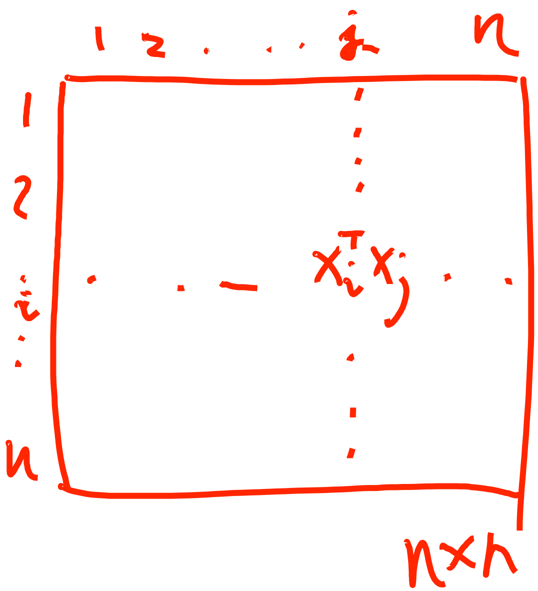
$$C > \alpha_i \geq 0, \forall i$$

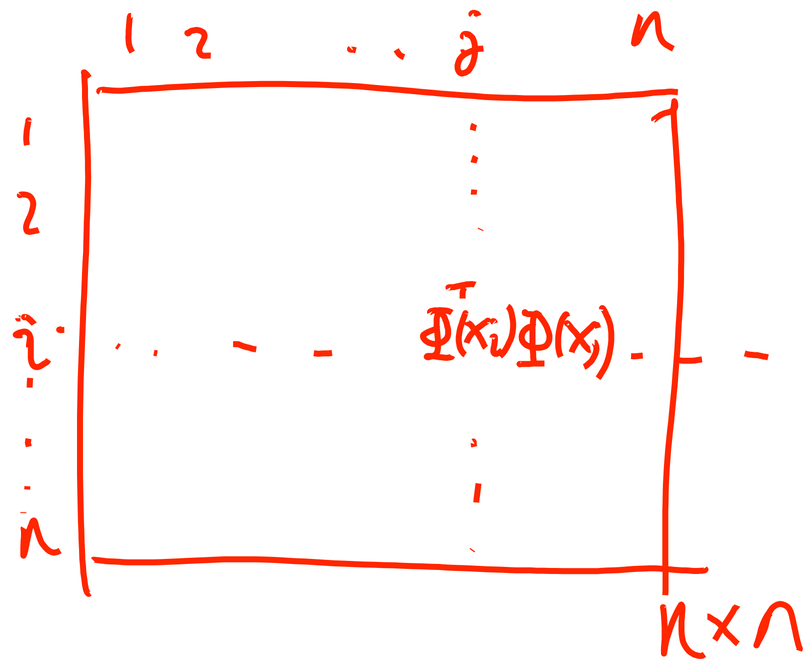
$\Longrightarrow$

$$\max_\alpha \sum_i \alpha_i - \sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j \Phi(\mathbf{x}_\mathbf{i})^T \Phi(\mathbf{x}_\mathbf{j})$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

$$C > \alpha_i \geq 0, \forall i$$

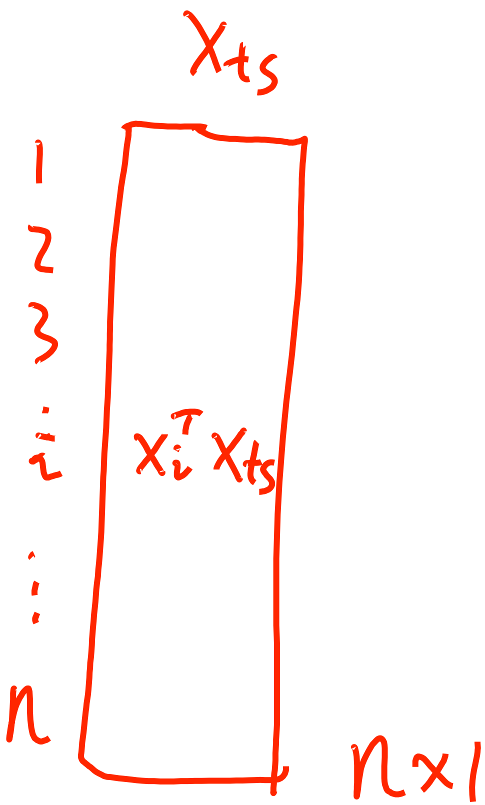




$$\mathrm{w}^{\mathrm{T}} x_{ts} + b = \sum_{\mathrm{i}} \alpha_i \mathrm{y}_{\mathrm{i}} \mathbf{x}_{\mathbf{i}}^{T} \mathbf{x}_{ts} + b$$

$$\widehat{y_{ts}} = \mathrm{sign}\left(\sum_{i \in SupportVectors} \alpha_i y_i \left(\mathbf{x}_i^T \mathbf{x}_{ts} \right) + b \right)$$

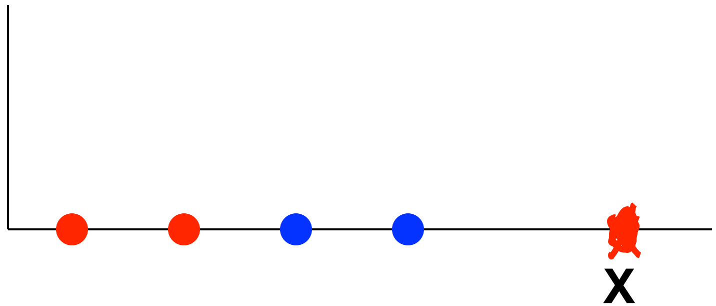
$$\Rightarrow \sum_{SV} \alpha_i y_i \Phi(x_i) \Phi(x_{ts}) + b$$

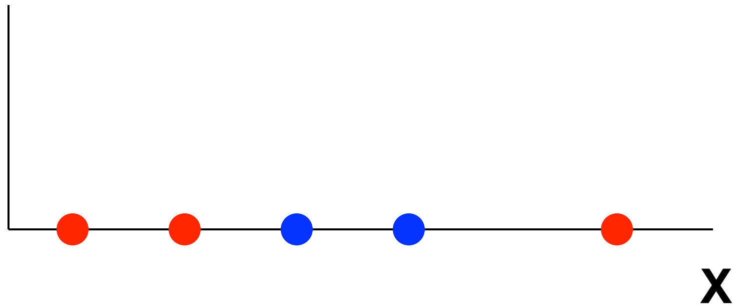


Classifying in 1-d

Can an SVM correctly classify this data?

What about this?

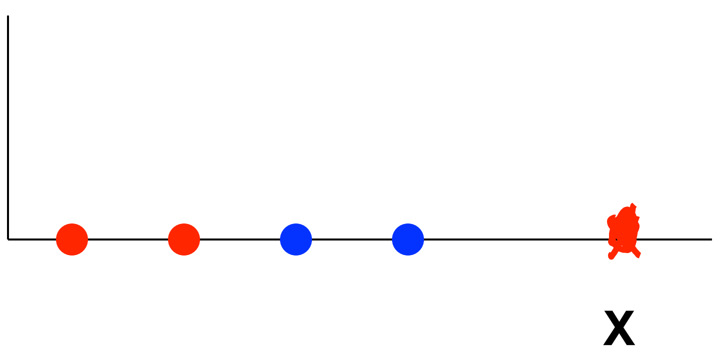




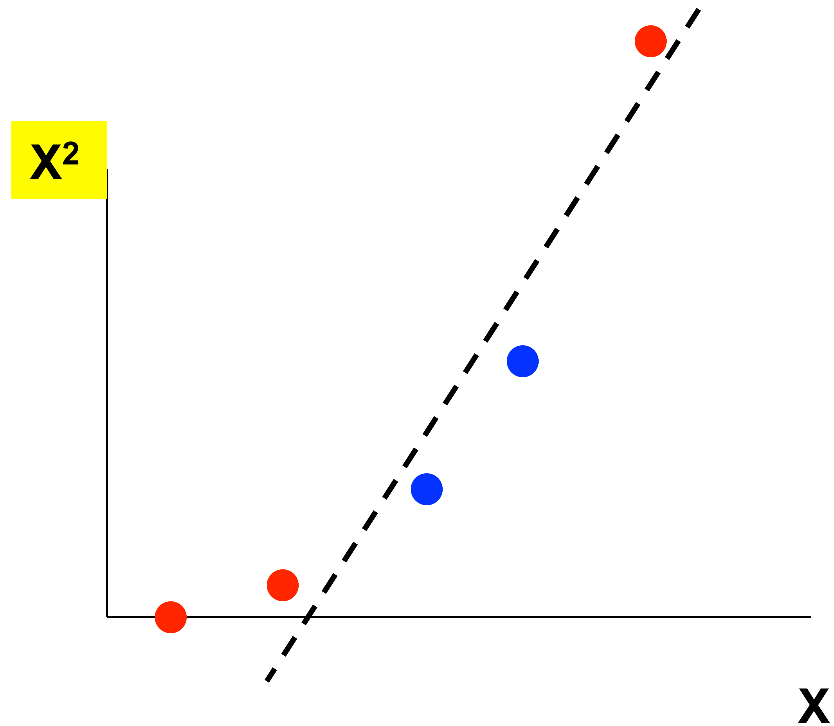
Classifying in 1-d

→ separable
→ nonlinear

Can an SVM correctly classify this data?

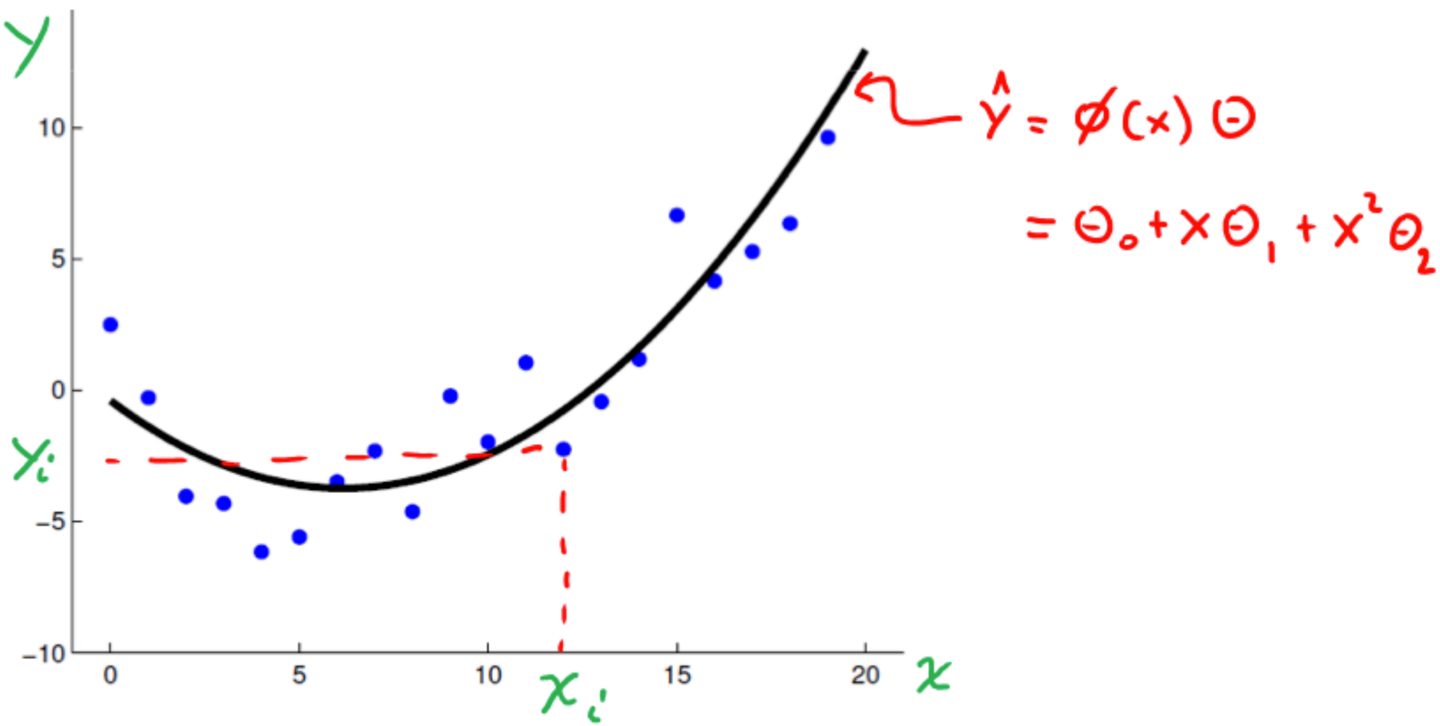


And now? (extend with polynomial basis)



RECAP: **Polynomial regression**

For example, $\phi(x) = [1, x, x^2]$



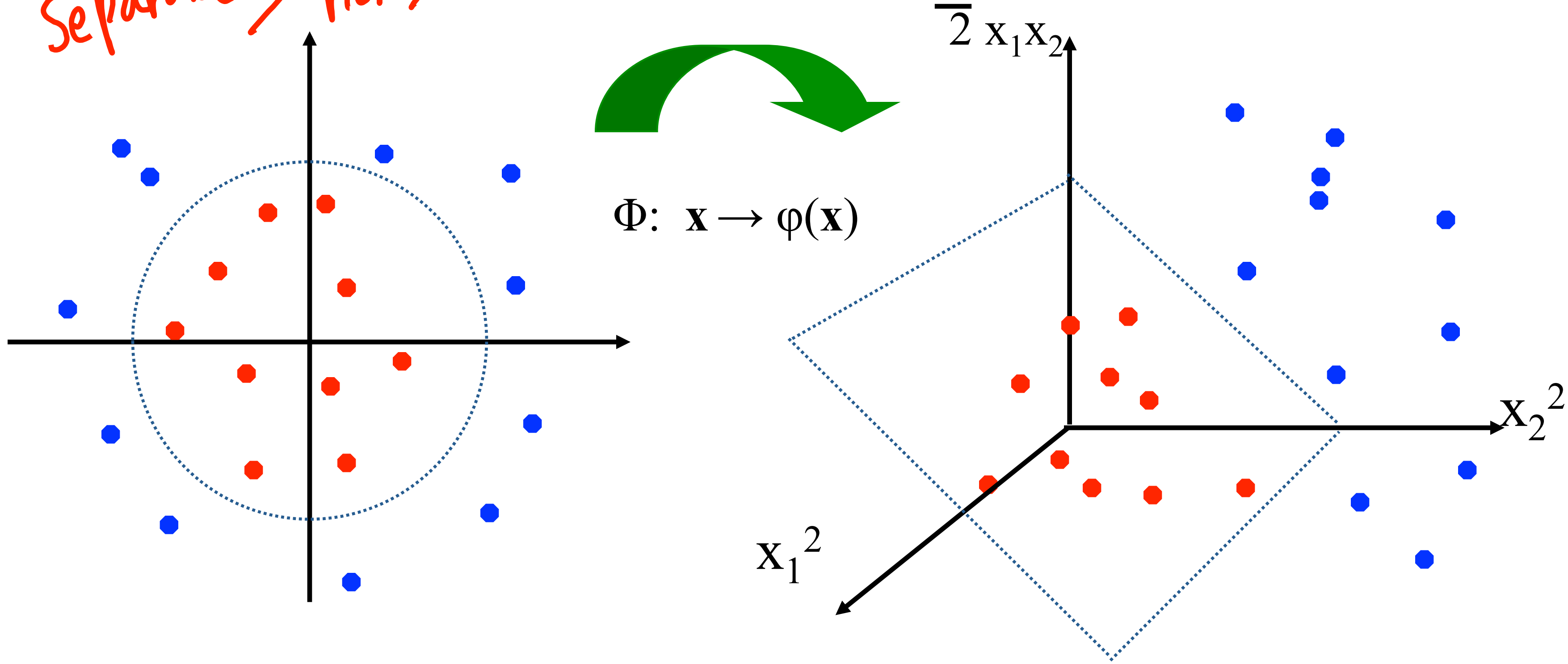
Dr. Nando de Freitas's tutorial slide

Non-linear SVMs: 2D

- The original input space (x) can be mapped to some higher-dimensional feature space ($\phi(\mathbf{x})$)where the training set is separable:

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

Separable / non linear



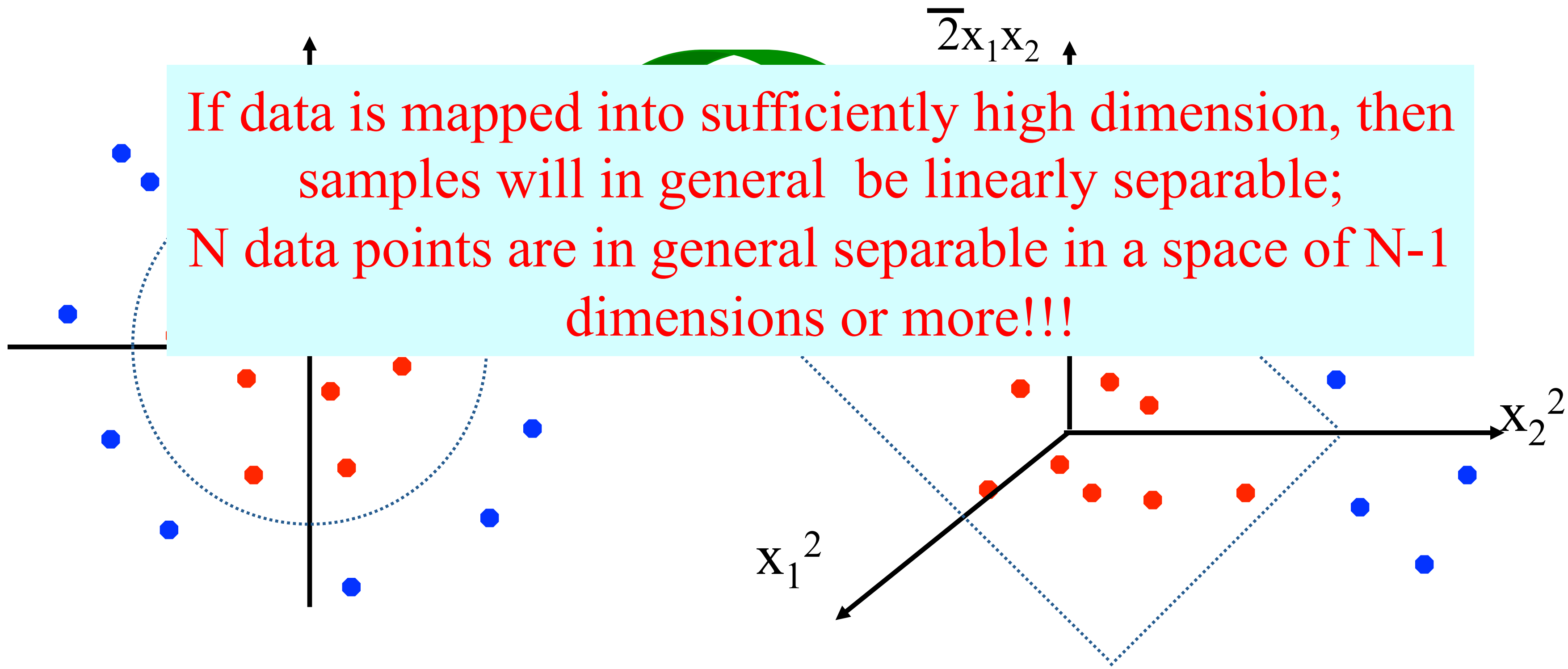
This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm_tutorial.ppt***

Non-linear SVMs: 2D

- The original input space (x) can be mapped to some higher-dimensional feature space ($\phi(\mathbf{x})$)where the training set is separable:

$\mathbf{x}=(x_1,x_2)$

$\varphi(\mathbf{x}) =(x_1^2,x_2^2, \sqrt{2}x_1x_2)$



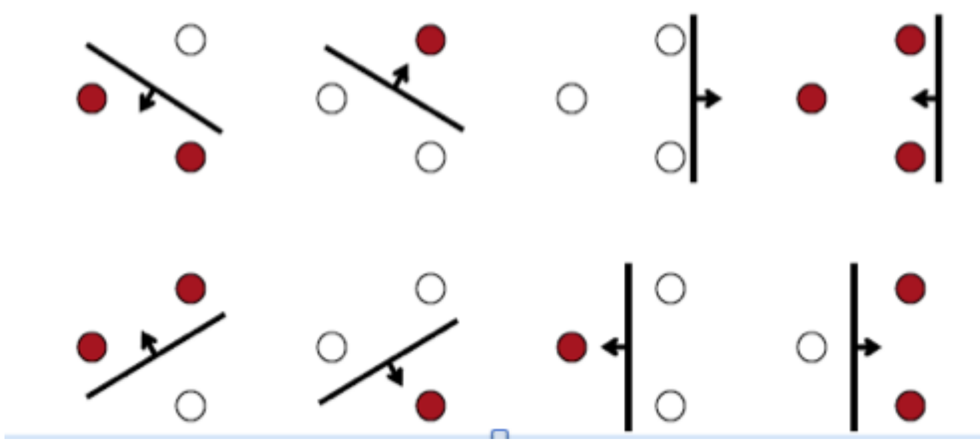
If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

A little bit theory: Vapnik-Chervonenkis (VC) dimension

If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

- **VC dimension of the set of oriented lines in R^2 is 3**
  - It can be shown that the VC dimension of the family of oriented separating hyperplanes in R^N is at least N+1



Transformation of Inputs

- Possible problems

Is this too much computational work?

  - High computation burden due to high-dimensionality ①
  - Many more parameters ②

$X \longrightarrow \Phi(X)$

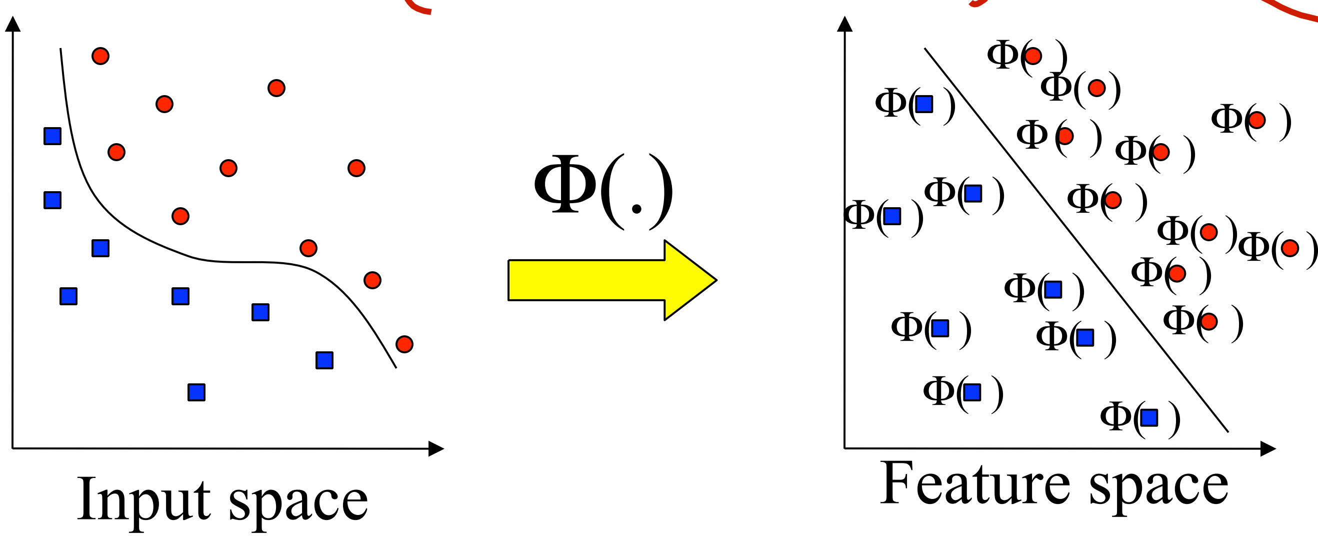
If data is mapped into sufficiently high dimension, then samples will in general be linearly separable;
N data points are in general separable in a space of N-1 dimensions or more!!!

Transformation of Inputs

- Possible problems

  Is this too much computational work?

  - High computation burden due to high-dimensionality (1)
  - Many more parameters (2)
- SVM solves these two issues simultaneously
  - "Kernel tricks" for efficient computation (1)
  - Dual formulation only assigns parameters to samples, not features (2) $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$



- SVM solves these two issues simultaneously
  - "Kernel tricks" for efficient computation
  - Dual formulation only assigns parameters to samples, not features

(1). "Kernel tricks" for efficient computation

$\cancel{\Phi(x)}$

Never represent features explicitly

- ☐ Compute dot products in closed form

Very interesting theory – Reproducing Kernel Hilbert Spaces

- ☐ Not covered in detail here

$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ is called the kernel function.

- Linear kernel (we've seen it) $K(\mathbf{x}, z) = \mathbf{x}^T z$

$\begin{cases} x \in R^p \\ z \in R^p \end{cases}$

- Polynomial kernel (we just saw an example)

$$K(\mathbf{x}, z) = \left(1 + \mathbf{x}^T z\right)^d = \Phi_p(x)^T \underbrace{\Phi_p(z)}_{p_\Phi \to O(p^d)}$$

where p = 2, 3, … To get the feature vectors we concatenate all pth order polynomial terms of the components of x (weighted appropriately)

- Radial basis kernel

$$K(\mathbf{x}, z) = \exp\left(-r \left\|\mathbf{x} - z\right\|^2\right) = \underbrace{\Phi_r(x)^T}_{p_\Phi = \infty} \Phi_r(z)$$

In this case., r is hyperpara. The feature space of the RBF kernel has an infinite number of dimensions

Never represent features explicitly
- ☐ Compute dot products in closed form

Very interesting theory – Reproducing Kernel Hilbert Spaces
- ☐ Not covered in detail here

$(1+x^T z)^2$

Kernel Trick: Quadratic kernels

- While working in higher dimensions is beneficial, it also increases our running time because of the dot product computation
- However, there is a neat trick we can use
- consider all quadratic terms for $x_1, x_2 \ldots x_m$

[for a quadratic kernel]

m

$$\max_\alpha \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x_i})^T \Phi(\mathbf{x_j})$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

normally we use p as #features. Here m

m is the number of features in each vector

weights on quadratic terms will become clear in the next slide

$$\Phi(x) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{bmatrix}$$

m+1 linear terms

m quadratic terms

m(m-1)/2 pairwise terms

$X \to \Phi(X)$

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

a very long vector

Dot product for quadratic kernels

How many operations do we need for the dot product?

$\begin{cases} x \in R^m \\ z \in R^m \end{cases}$

$O(m^2)$ $O(m^2)$ $O(m^2)$

$$\Phi(x)^T \Phi(z) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{bmatrix} \bullet \begin{bmatrix} 1 \\ \sqrt{2}z_1 \\ \vdots \\ \sqrt{2}z_m \\ z_1^2 \\ \vdots \\ z_m^2 \\ \sqrt{2}z_1z_2 \\ \vdots \\ \sqrt{2}z_{m-1}z_m \end{bmatrix} = \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_i \sum_{j=i+1} 2x_i x_j z_i z_j + 1$$

Inner product symbol

m m m(m-1)/2 **=~ m²**

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

$$K(\mathbf{x},z)=\left(1+\mathbf{x}^T z\right)^2$$

$[d=2]$, $[p=2]$

$$\begin{cases} x=(x_1,x_2) \\ z=(z_1,z_2) \end{cases}$$

$$K(x,z)=(1+x_1z_1+x_2z_2)^2 \Rightarrow O(p)$$

$$O(p^2)\begin{cases} =(1,\sqrt{2}x_1,\sqrt{2}x_2,x_1^2,x_2^2,\sqrt{2}x_1x_2)^T \\ (1,\sqrt{2}z_1,\sqrt{2}z_2,z_1^2,z_2^2,\sqrt{2}z_1z_2) \end{cases}$$

$$=\Phi(x)^T\Phi(z)$$

In the previous page, we use m as # features. Normally we use p

The kernel trick

How many operations do we need for the dot product?

$$\underbrace{\Phi(x)^T \Phi(z)}_{O(m^2)} = \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_i \sum_{j=i+1} 2x_i x_j z_i z_j + 1$$

m m m(m-1)/2 **=~ m²**

$(x^T z + 1)^2$

However, we can obtain dramatic savings by noting that

$$\Phi(x)^T \Phi(z) = \underbrace{(x^T z + 1)^2}_{O(m)} = (x.z+1)^2 \quad = \quad (x.z)^2 + 2(x.z) + 1$$

$$= \quad (\sum_i x_i z_i)^2 + \sum_i 2x_i z_i + 1$$

$$= \quad \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_i \sum_{j=i+1} 2x_i x_j z_i z_j + 1$$

We only need m operations!

So, if we define the **kernel function** as follows, there is no need to carry out basis function $\Phi(\ )$ explicitly

$$K(\mathbf{x}, z) = (x^T z + 1)^2$$

Kernel Trick

$x \to \Phi(x)$

Our dual target function:

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \Phi(\mathbf{x_i})^T\Phi(\mathbf{x_j})$$

$\sum_{i=1}^{n}\sum_{j=1}^{n}$

$K(x_i, x_j)$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

$O(pn^2)$

mn[2] operations at each iteration

training

To evaluate a new sample x_k we need to compute:

$$w^T\Phi(\mathbf{x_{ts}}) + b = \sum_i \alpha_i y_i \Phi(\mathbf{x_i})^T\Phi(\mathbf{x_{ts}}) + b$$

$K(x_i, x_{ts})$

$O(pn_{sv})$

mr operations where *r* are the number of support vectors (whose \alpha$_i$>0)

testing on one sample

$$K(\mathbf{x}, z) = (x^T z + 1)^2$$

Summary: Modification Due to Kernel Trick

- Change all inner products to kernel functions
- For training,

Original Linear

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j \mathbf{x}_\mathbf{i}^T \mathbf{x}_\mathbf{j}$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

$$C > \alpha_i \geq 0, \forall i \in train$$

With kernel function - nonlinear

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j K(\mathbf{x}_\mathbf{i}, \mathbf{x}_\mathbf{j})$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

$$C > \alpha_i \geq 0, \forall i \in train$$

Summary: Modification Due to Kernel Function

- For testing, the new data **x_ts**

Original Linear

$$\widehat{y_{ts}} = \text{sign}\left(\sum_{\text{i}\in\text{train}} \alpha_i \text{y}_\text{i} \mathbf{x_i}^T \mathbf{x}_{ts} + b \right)$$

With kernel function - nonlinear

$$\widehat{y_{ts}} = \text{sign}\left(\sum_{\text{i}\in\text{train}} \alpha_i \text{y}_\text{i} K(\mathbf{x_i}, \mathbf{x}_{ts}) + b \right)$$

An example: Support vector machines with polynomial kernel

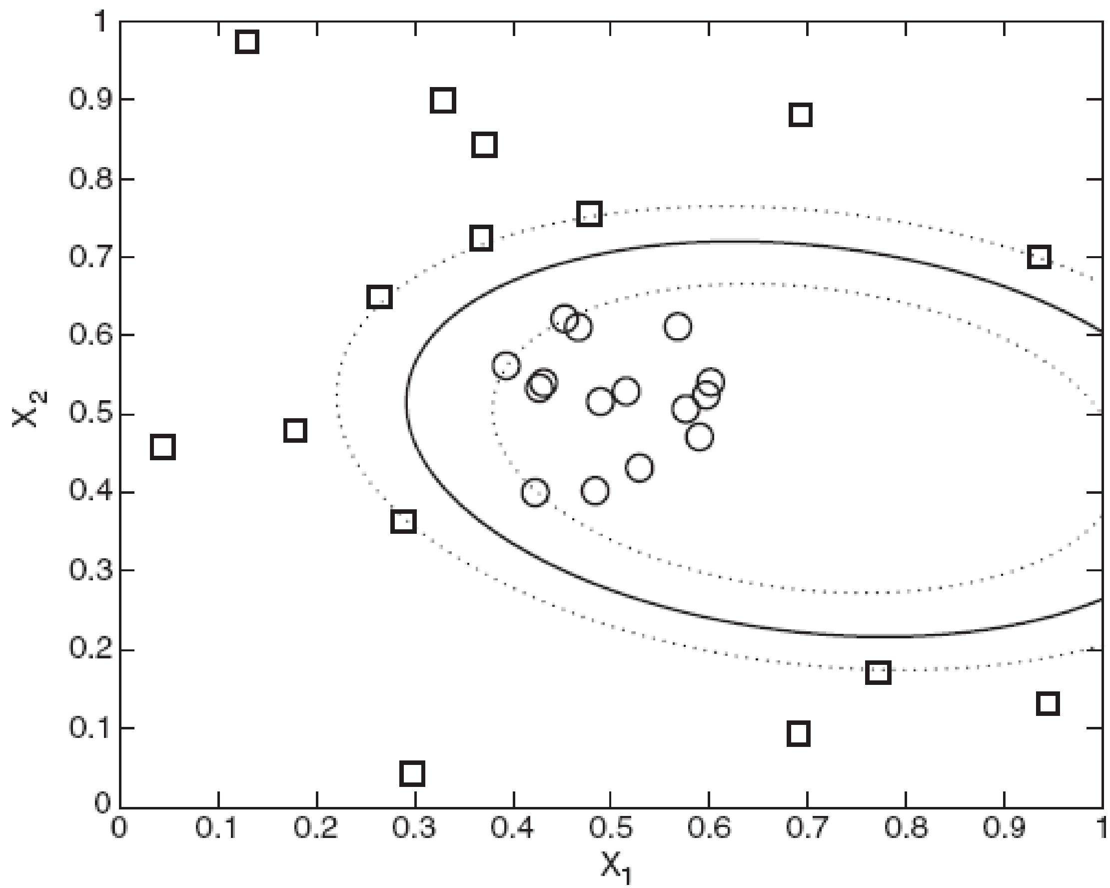


Figure 5.29. Decision boundary produced by a nonlinear SVM with polynomial kernel.

Kernel Trick: Implicit Basis Representation

- For some kernels (e.g. RBF) the implicit transform basis form \phi(**x**) is infinite-dimensional!
  - But calculations with kernel are done in original space, so computational burden and curse of dimensionality aren't a problem.

$$K(\mathbf{x},z) = \exp\left(-r\left\|\mathbf{x}-z\right\|^2\right)$$

➔ Gaussian RBF Kernel corresponds to an infinite-dimensional vector space.

YouTube video of Caltech: Abu-Mostafa explaining this in more detail
https://www.youtube.com/watch?v=XUj5JbQihlU&t=25m53s

Kernel Functions

- In practical use of SVM, only the kernel function (and not $\Phi(x)$) is specified
- Kernel function can be thought of as a similarity measure between the input objects
- Not all similarity measure can be used as kernel function, however Mercer's condition states that any positive semi-definite kernel *K*(*x*, *y*), i.e.

$$\sum_{i,j} K(x_i, x_j) c_i c_j \geq 0$$

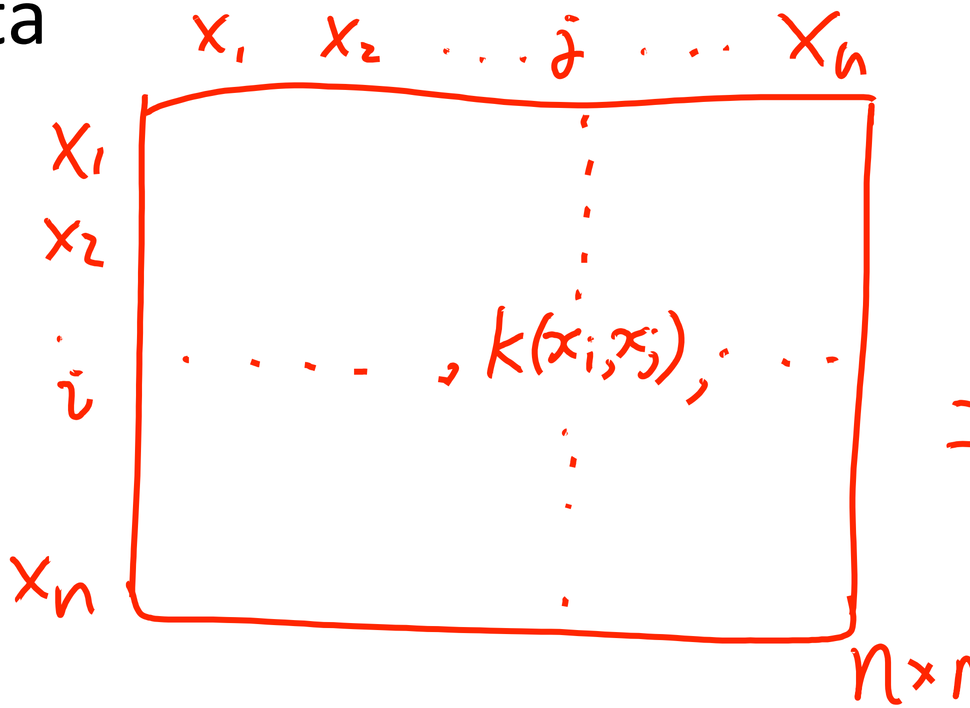
can be expressed as a dot product in a high dimensional space.

Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarize all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, tree kernel, graph kernel, ...)
  - Kernel trick has helped Non-traditional data like strings and trees able to be used as input to SVM, instead of feature vectors
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try for most applications.

Kernel Matrix

- The kernel function is important because it creates the kernel matrix, which summarize all the data



$\Rightarrow K_{n \times n}$

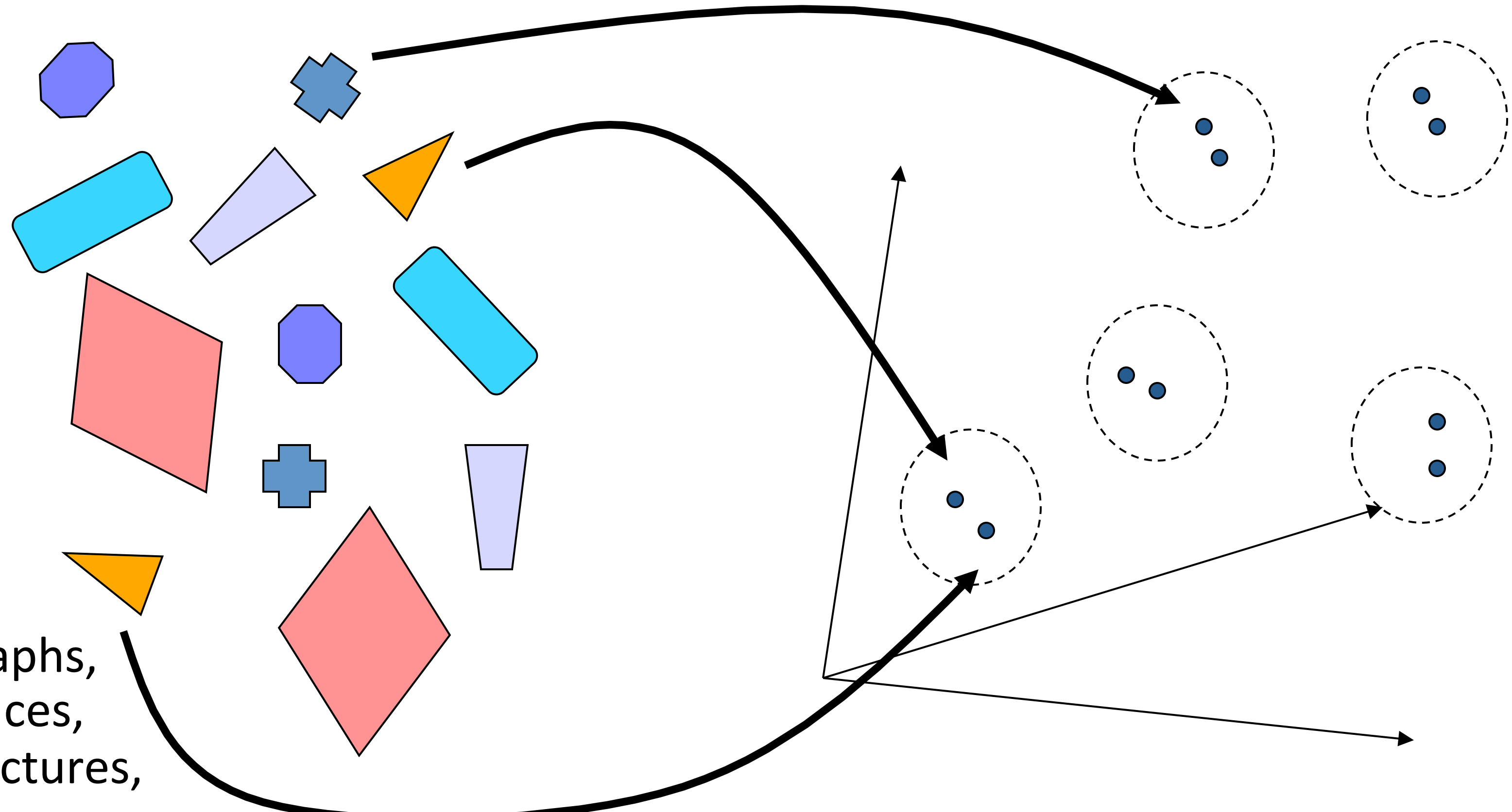
[kernel matrix]

positive-semi-definite

Kernel trick has helped Non-traditional data like strings and trees able to be used as input to SVM, instead of feature vectors

$k(x, z)$

Vector vs. Relational data



e.g. Graphs, Sequences, 3D structures,

Original Space

Feature Space

Mercer Kernel vs. Smoothing Kernel

- The Kernels used in Support Vector Machines are different from the Kernels used in LocalWeighted /Kernel Regression.

- We can think
  - Support Vector Machines' kernels as **Mercer Kernels**
  - Local Weighted / Kernel Regression's kernels as **Smoothing Kernels**

Why do SVMs work?

- ❑ If we are using huge features spaces (e.g., with kernels), how come we are not overfitting the data?
  - ✓ Number of parameters remains the same (and most are set to 0)
  - ✓ While we have a lot of input values, at the end we only care about the support vectors and these are usually a small group of samples
  - ✓ The minimization (or the maximizing of the margin) function acts as a sort of regularization term leading to reduced overfitting

Why SVM Works?

- Vapnik argues that the fundamental problem is not the number of parameters to be estimated. Rather, the problem is about the flexibility of a classifier

- Vapnik argues that the flexibility of a classifier should not be characterized by the number of parameters, but by the capacity of a classifier
  - This is formalized by the "VC-dimension" of a classifier

- The SVM objective can also be justified by structural risk minimization: the empirical risk (training error), plus a term related to the generalization ability of the classifier, is minimized

- Another view: the SVM loss function is analogous to ridge regression. The term $\frac{1}{2}||w||^2$ "shrinks" the parameters towards zero to avoid overfitting

Today

- Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide

Software

- A list of SVM implementation can be found at
  - http://www.kernel-machines.org/software.html

- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

Summary: Steps for Using SVM in HW

- Prepare the feature-data matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of *C*
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

Practical Guide to SVM

- From authors of as LIBSVM:
  - A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, 2003-2010
  - http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

LIBSVM

- http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - ✓Developed by Chih-Jen Lin etc.
  - ✓Tools for Support Vector classification
  - ✓Also support multi-class classification
  - ✓C++/Java/Python/Matlab/Perl wrappers
  - ✓Linux/UNIX/Windows
  - ✓SMO implementation, fast!!!

A Practical Guide to Support Vector Classification

(a) Data file formats for LIBSVM

- Training.dat

+1 1:0.708333 2:1 3:1 4:-0.320755

-1 1:0.583333 2:-1 4:-0.603774 5:1

+1 1:0.166667 2:1 3:-0.333333 4:-0.433962

-1 1:0.458333 2:1 3:1 4:-0.358491 5:0.374429

…

- Testing.dat

(b) Feature Preprocessing

- (1) Categorical Feature
  - Recommend using m numbers to represent an m-category attribute.
  - Only one of the m numbers is one, and others are zero.
  - For example, a three-category attribute such as {red, green, blue} can be represented as (0,0,1), (0,1,0), and (1,0,0)

A Practical Guide to Support Vector Classification

Feature Preprocessing

- (2) Scaling before applying SVM is very important
  - to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges.
  - to avoid numerical difficulties during the calculation
  - Recommend linearly scaling each attribute to the range [1, +1] or [0, 1].

① Normalization → { mean 0, std 1 }

② Scaling → linear ⇒ $[ax+b]$

e.g. $\left[\dfrac{X - X_{min}}{max - X_{min}} \right]$

A Practical Guide to Support Vector Classification

For i-th feature $\Rightarrow$ [Column operation on $X_{n \times p}$]

Centering : $X_i - \bar{X}_i \Rightarrow E(X_i) = 0$

Scaling : $aX_i + b \Rightarrow$ e.g. $\frac{X_i - \min(X_i)}{\max(X_i) - \min(X_i)}$

Normalization : $\Rightarrow \begin{cases} E(X_i) = 0 \\ var(X_i) = 1 \end{cases}$

Of course we have to use the same method to scale both training and testing data. For example, suppose that we scaled the first attribute of training data from $[-10, +10]$ to $[-1, +1]$. If the first attribute of testing data lies in the range $[-11, +8]$, we must scale the testing data to $[-1.1, +0.8]$. See Appendix B for some real examples.

If training and testing sets are separately scaled to $[0, 1]$, the resulting accuracy is lower than 70%.

```
$ ../svm-scale -l 0 svmguide4 > svmguide4.scale
$ ../svm-scale -l 0 svmguide4.t > svmguide4.t.scale
$ python easy.py svmguide4.scale svmguide4.t.scale
Accuracy = 69.2308% (216/312) (classification)
```

Using the same scaling factors for training and testing sets, we obtain much better accuracy.

```
$ ../svm-scale -l 0 -s range4 svmguide4 > svmguide4.scale
$ ../svm-scale -r range4 svmguide4.t > svmguide4.t.scale
$ python easy.py svmguide4.scale svmguide4.t.scale
Accuracy = 89.4231% (279/312) (classification)
```

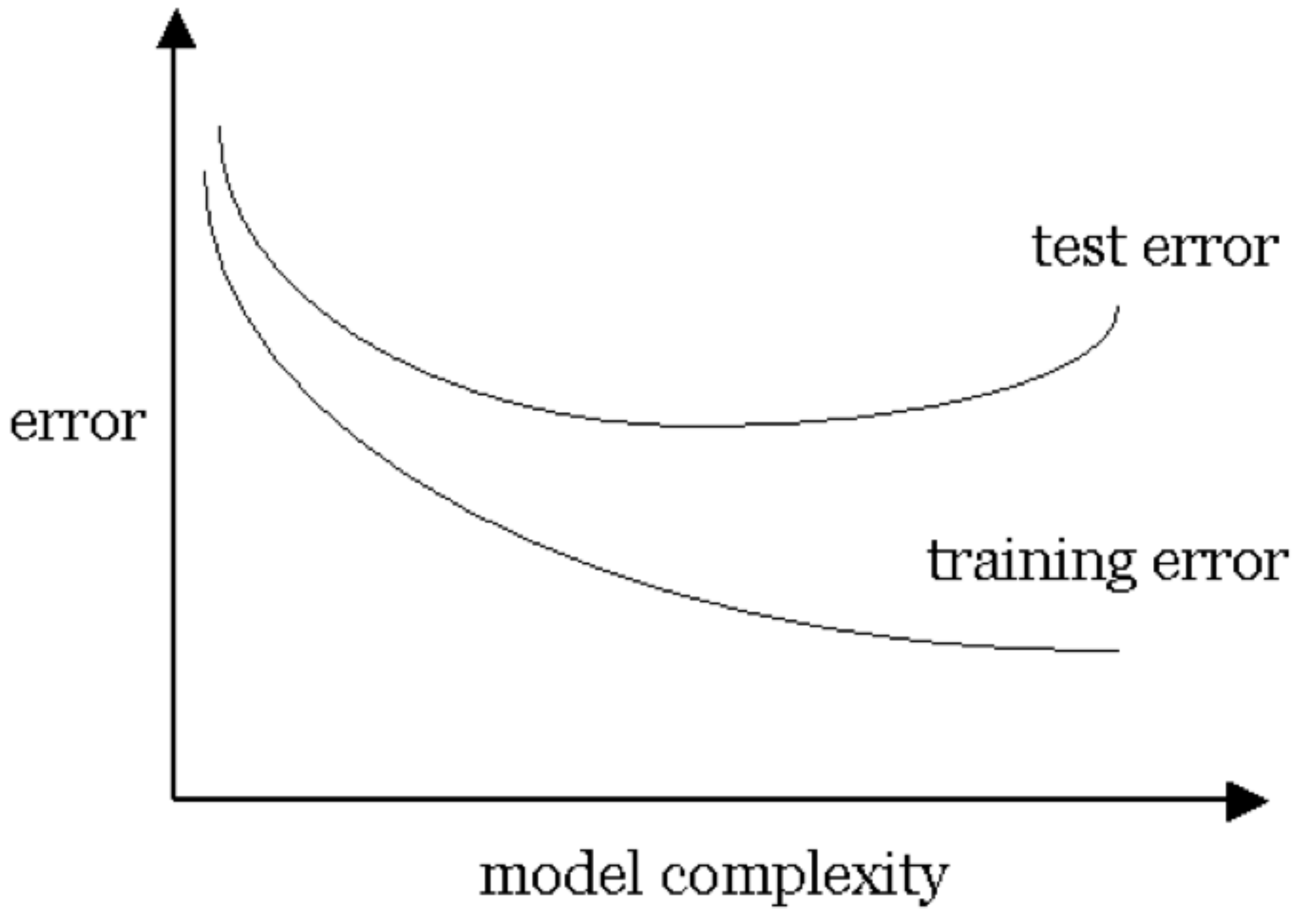
Feature Preprocessing

- (3) missing value
  - Very very tricky !
  - Easy way: to substitute the missing values by the mean value of the variable
  - A little bit harder way: imputation using nearest neighbors
  - Even more complex: e.g. EM based (beyond the scope)

A Practical Guide to Support Vector Classification

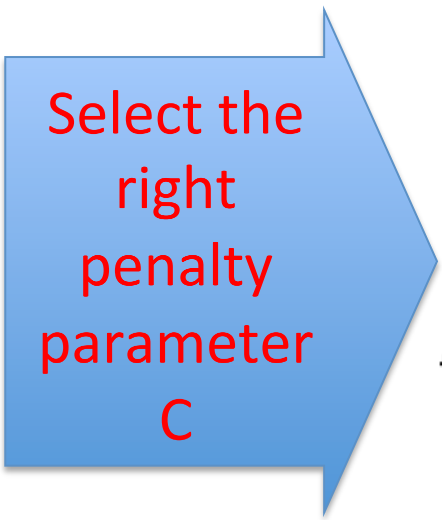
(c) Model Selection

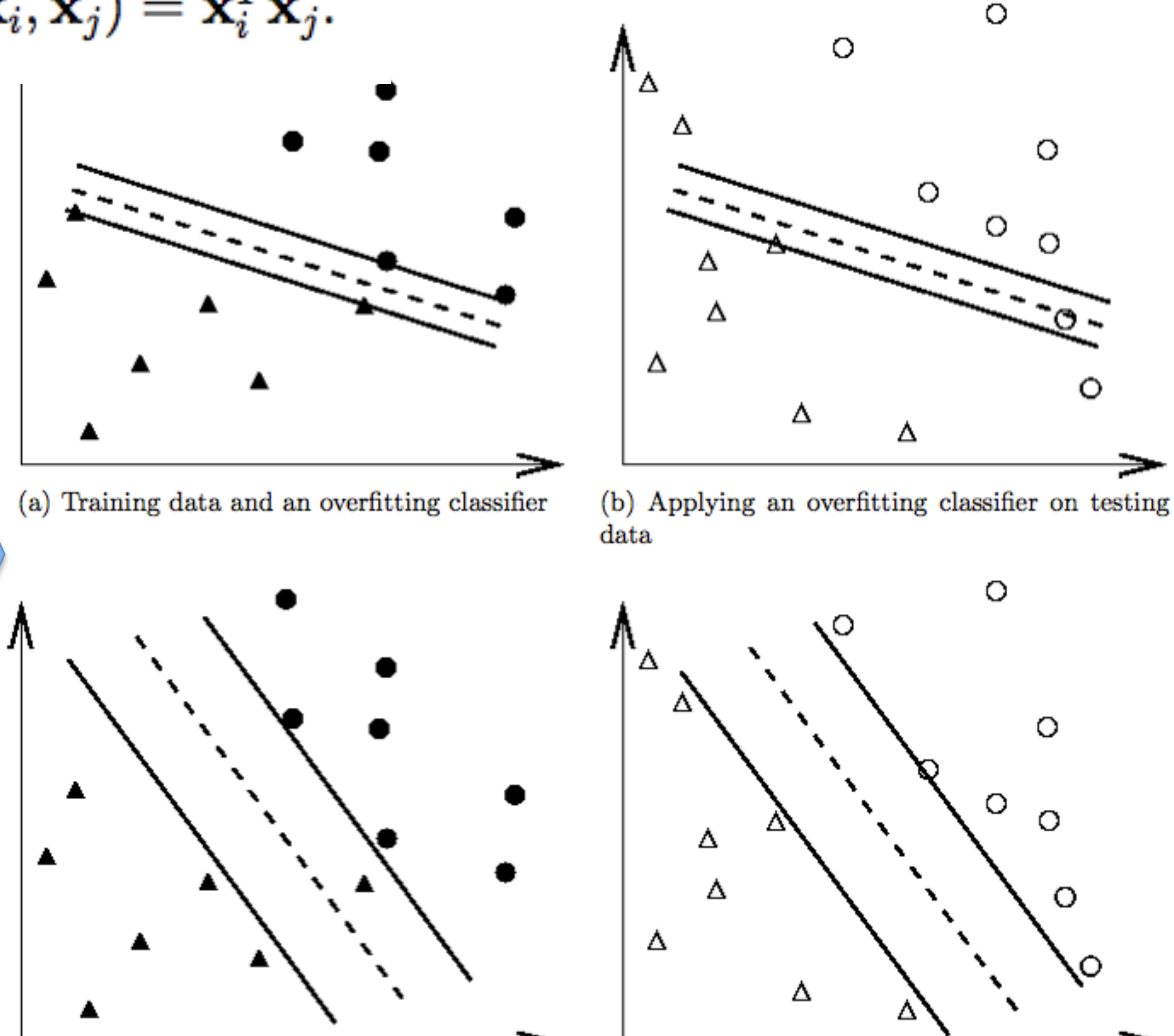
Our goal: find the model M which minimizes the test error:



(c) Model Selection (e.g. for linear kernel)

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.





(a) Training data and an overfitting classifier

(b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier

(d) Applying a better classifier on testing data

(c) Model Selection

- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$, $\gamma > 0$.

  two parameters for an RBF kernel: C and γ

- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma {\mathbf{x}_i}^T \mathbf{x}_j + r)^d$, $\gamma > 0$.

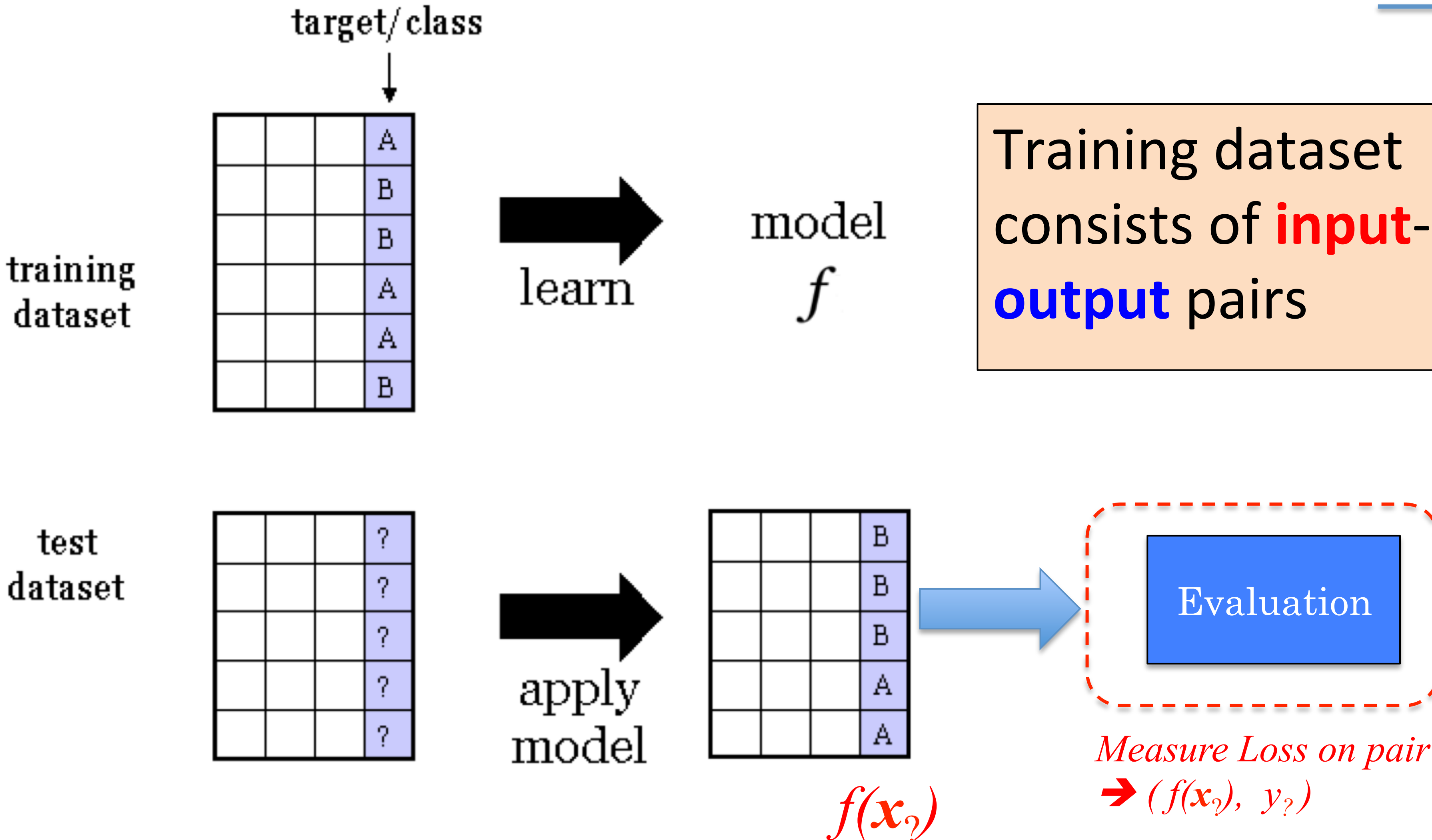
  Three parameters for a polynomial kernel

A Practical Guide to Support Vector Classification

(d) Pipeline Procedures

- (1) train / test
- (2) k-folds cross validation
- (3) k-CV on train to choose hyperparameter / then test

Evaluation Choice-I: Train and Test



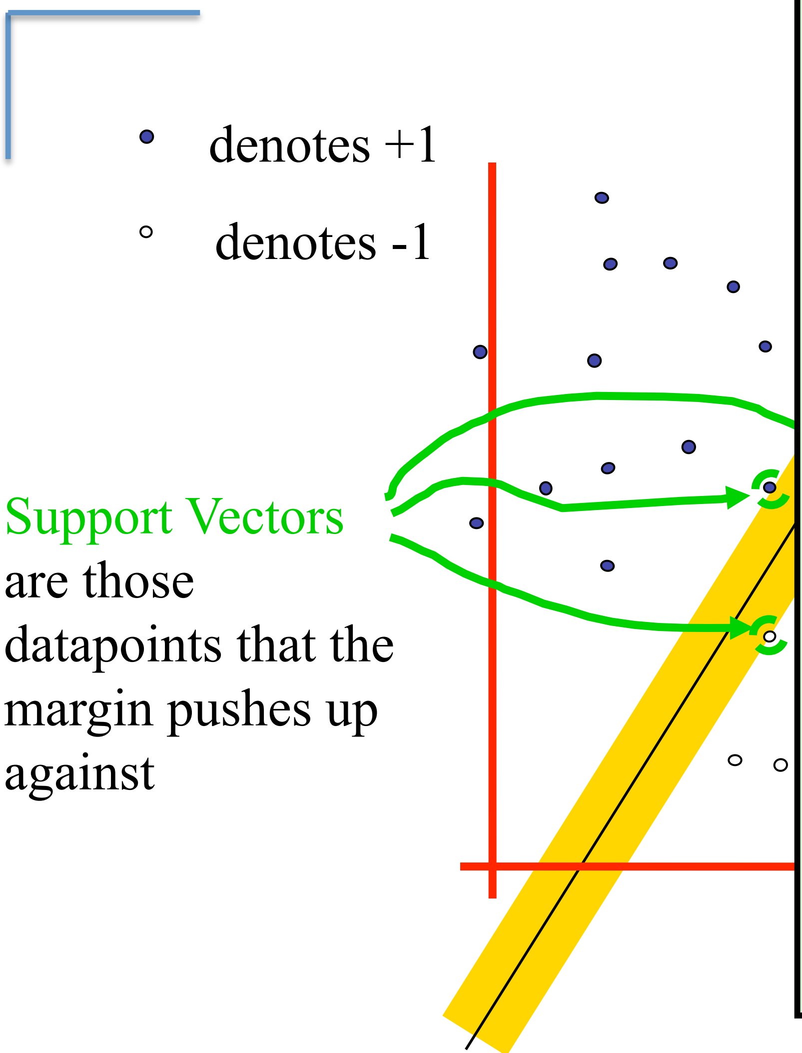
Training dataset consists of **input-output** pairs

Evaluation Choice-II: Cross Validation

- Problem: don't have enough data to set aside a test set
- Solution: Each data point is used both as train and test
- Common types:
  - -K-fold cross-validation (e.g. K=5, K=10)
  - -2-fold cross-validation
  - -Leave-one-out cross-validation (LOOCV)

A good practice is : to random shuffle all training sample before splitting

Why Maximum Margin for SVM ?



1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
3. **LOOCV is easy since the model is immune to removal of any non-support-vector datapoints.**
4. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
5. Empirically it works very very well.

Evaluation Choice-III:

Many beginners use the following procedure now:

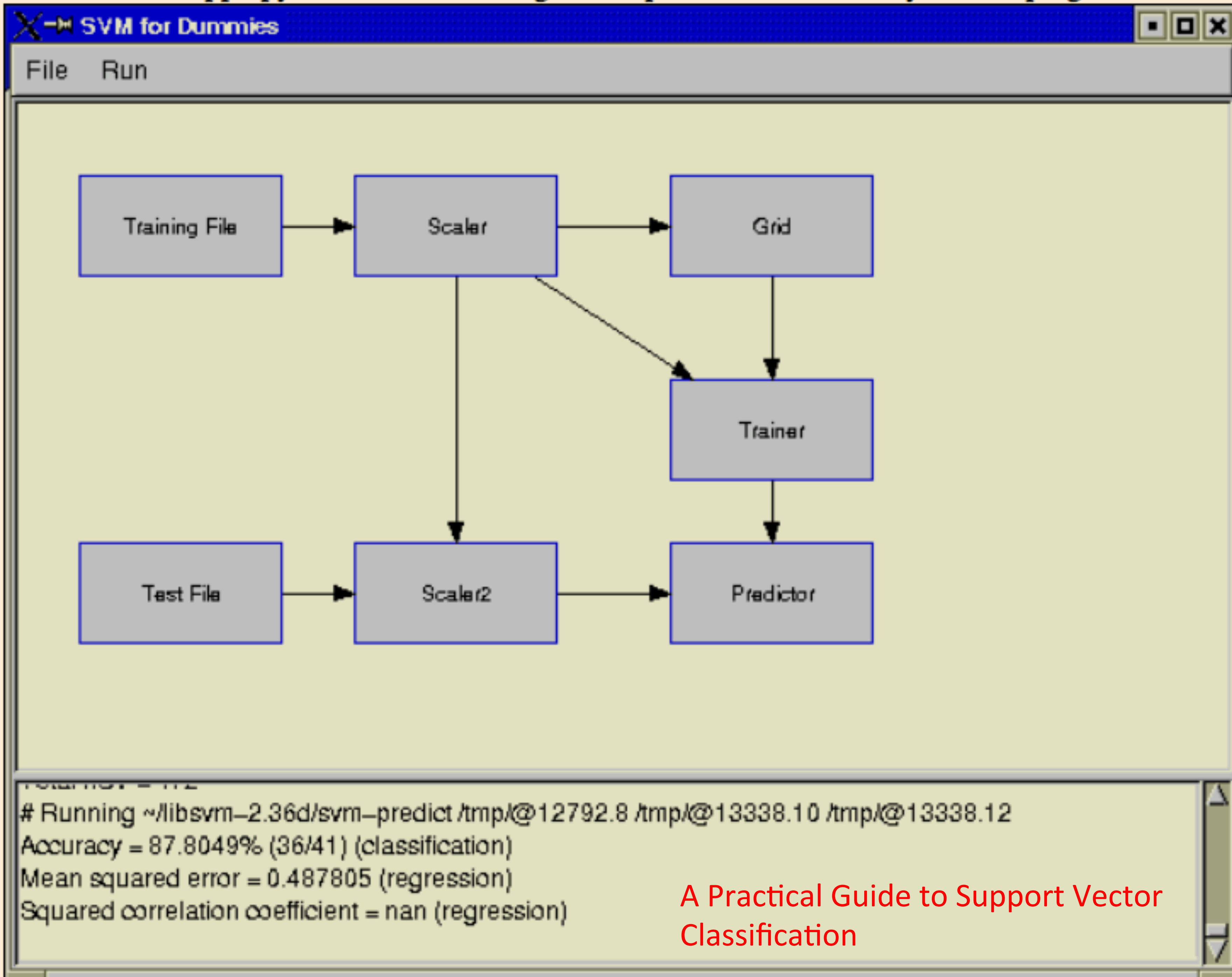
- Transform data to the format of an SVM package
- Randomly try a few kernels and parameters
- Test

We propose that beginners try the following procedure first:

- Transform data to the format of an SVM package
- Conduct simple scaling on the data
- Consider the RBF kernel $K(\mathbf{x}, \mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}$
- Use cross-validation to find the best parameter C and γ
- Use the best parameter C and γ to train the whole training set[5]
- Test

For HW2-Q2

A Practical Guide to Support Vector Classification



A Practical Guide to Support Vector Classification

Today: Review & Practical Guide

- ❑ Support Vector Machine (SVM)
  - ✓ History of SVM
  - ✓ Large Margin Linear Classifier
  - ✓ Define Margin (M) in terms of model parameter
  - ✓ Optimization to learn model parameters (w, b)
  - ✓ Non linearly separable case
  - ✓ Optimization with dual form
  - ✓ Nonlinear decision boundary
  - ✓ Practical Guide
    - ✓ File format / LIBSVM
    - ✓ Feature preprocsssing
    - ✓ Model selection
    - ✓ Pipeline procedure

Support Vector Machine

Task → Representation → Score Function → Search/Optimization → Models, Parameters

classification → Kernel Tric- Func K(xi, xj) → Margin + Hinge Loss (optional) → QP with Dual form → Dual Weights

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

$$w = \sum_i \alpha_i x_i y_i$$

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + C\sum_{i=1}^{n} \varepsilon_i$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1 - \varepsilon_i$

$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T \mathbf{x_j}$$

$$\sum_i \alpha_i y_i = 0, \qquad \alpha_i \geq 0 \qquad \forall i$$

References

- Big thanks to Prof. Ziv Bar-Joseph and Prof. Eric Xing @ CMU for allowing me to reuse some of his slides
- Elements of Statistical Learning, by Hastie, Tibshirani and Friedman
- Prof. Andrew Moore @ CMU's slides
- Tutorial slides from Dr. Tie-Yan Liu, MSR Asia
- A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, 2003-2010
- Tutorial slides from Stanford "Convex Optimization I — Boyd & Vandenberghe