Classification Methods

- Approximating Bayes Decision Rule: (model the likelihood)
  - Linear Discriminant Analysis/ QDA
  - Fisher's linear discriminant
  - **Naïve Bayes**
  - **Bayesian Belief Networks**
  - K-Nearest Neighbor
- Approximating Bayes Decision Rule: (model the posterior)
  - **Logistic Regression**
  - **Feedforward neural networks, including deep learning**
- Focus just on Class Boundaries:
  - Decision trees
  - **Support Vector Machines**

Quiz (Bayes Rule)

- Suppose 0.01% of Austin's population have cancer. A new test for cancer shows positive 90% of the time when a person actually has cancer, and correctly indicates "negative" 95% of the time when run on someone who do not have cancer.

This test is conducted on an Austinite and the results come out positive.

- What is the probability that this person actually has cancer?

Revisiting Bayes Decision Rule

- Let input **x** have d attributes $(x_1, x_2,\ldots,x_d)$;
  C is a random variable over class labels.

- **Bayes Decision rule:** Choose value of C that
  maximizes $P(x_1, x_2, \ldots, x_d|C)\ P(C)$

- Problem: how to estimate $P(x_1, x_2,\ldots,x_d\,|C)$ for each class?
  - Especially in high dimensions, interacting variables?

Naïve Bayes Approach

- Conditional Independence:
  - X is cond. Indep of Y given Z if P (X|Y,Z) = P(X|Z)
    - X, Y, Z could be sets of variables too

Naïve Bayes:

1. Assume independence among attributes x_i when class is given: ("independence of attributes conditioned on class variable").

- $P(x_1, x_2, \ldots, x_d | C_j) = P(x_1| C_j)\, P(x_2| C_j) \ldots P(x_d| C_j) = \Pi_i\, P(x_i| C_j)$

- Note: Conditional independence not equal to attribute independence

2. Estimate probabilities directly from data.

Estimating Probabilities from Data (Discrete Attributes)

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(Example from TSK)

- Class: $P(C) = N_c/N$
  - e.g., P(No) = 7/10,
    P(Yes) = 3/10
- For discrete attributes:

  $P(x_i = v \mid C_k)$ = fraction of examples of class k for which attribute x_i takes value v.
  - Examples:

    P(Status=Married|No) = 4/7
    P(Refund=Yes|Yes)=0

Estimating Probabilities (continuous attributes)

- Discretize the range into bins
  - one ordinal attribute per bin
  - violates independence assumption
- Binarize: (may lose substantial info)
- Probability density estimation:
  - (usually assuming Normal distribution)

How to Estimate Probabilities from Data?

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Evade**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(x_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (x_i,c_i) pair
- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No)
  $\times$ P(Married| Class=No)
  $\times$ P(Income=120K| Class=No)
  = 4/7 $\times$ 4/7 $\times$ 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  $\times$ P(Married| Class=Yes)
  $\times$ P(Income=120K| Class=Yes)
  = 1 $\times$ 0 $\times$ 1.2 $\times$ 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)
=> Class = No

Smoothing Naïve Bayes

- Avoids zero probablity due to one attribute-value/class combo being absent in training data.
  - Zeroes entire product term
- Probability estimation:

$$\text{Original}: P(x_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{m-estimate}: P(x_i \mid C) = \frac{N_{ic} + mp_i}{N_c + m}$$

c: number of classes

p: prior probability

m: weight of prior (i.e. # of virtual samples)

e.g. in text analysis, add a "virtual document that has one instance of every word in the vocabulary (Laplace smoothing): $(N_{ic} + 1) / (Nc + |vocab|)$

Naïve Bayes (Comments)

- Independence assumption often does not hold
  - Poorer estimate of P(C|x), often unrealistically close to 0 or 1
    - but still may pick the right "max"!!
  - If too restrictive, use other techniques such as Bayesian Belief Networks (BBN)
- Somewhat robust to isolated noise points, and irrelevant attributes
- Tries to finesse "curse of dimensionality"
- Most popular with binary or small cardinality categorical attributes
- **Requires only single scan of data; also streaming version is trivial.**

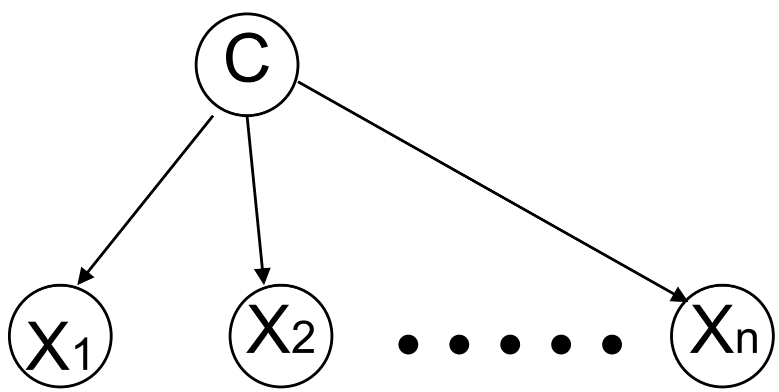
- Notable "Success": Text (bag-of-words representation + multinomial model per class).

Graphical Models

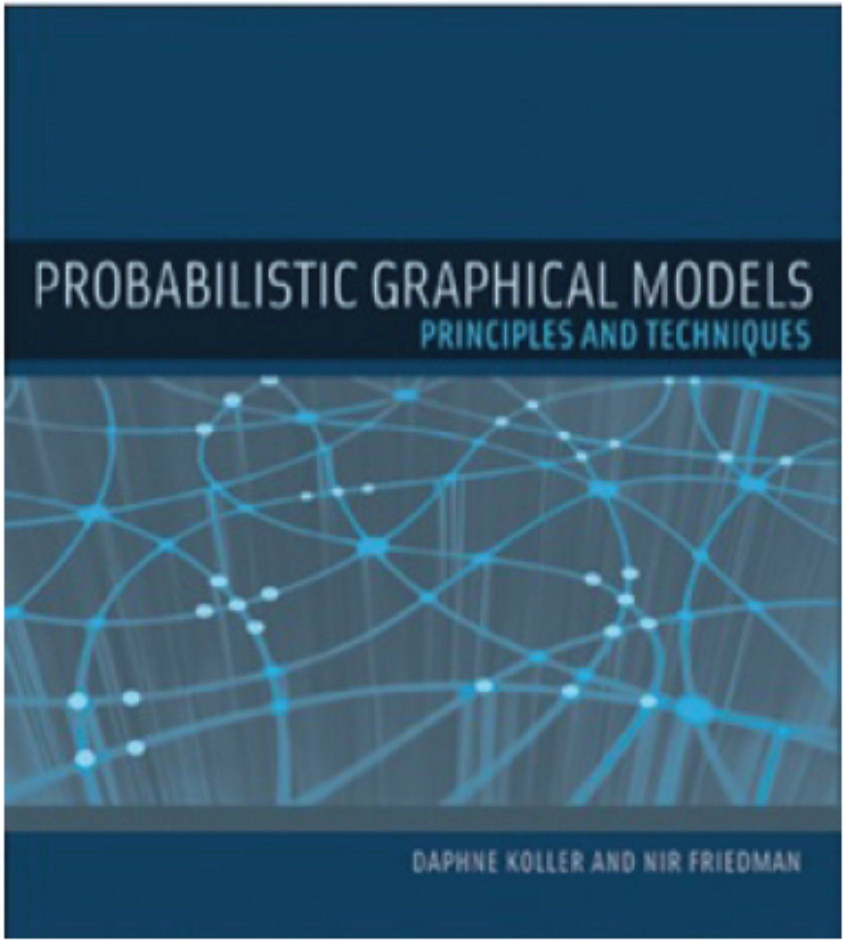
- Graphical models (see Kevin Murphy's survey, 2001) are (directed or undirected) graphs in which nodes represent random variables, and the lack of arcs represent (**assumed**) conditional independences: two sets of nodes are **conditionally independent** given a third set D, if all paths between nodes in A and B are separated by D.
  – Graphical models include mixture models, hidden Markov models, Kalman filters, etc

  Several R packages: http://cran.at.r-project.org/web/views/gR.html

- Naïve Bayes is a simple directed graphical model



(Way Beyond Classification)



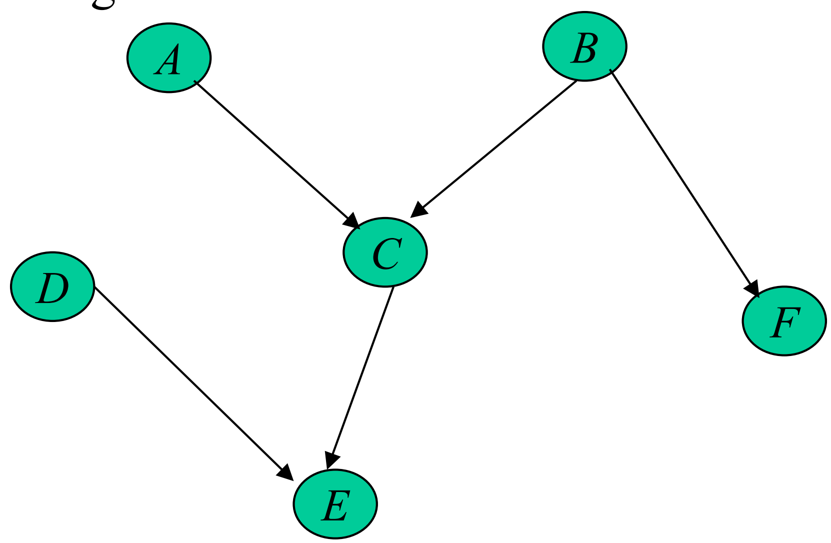
Daphne Koller
Computer Science Dept.
Stanford University

Nir Friedman
School of Computer Science & Engineering
Hebrew University

MIT Press, 2009, 1231 pages

Bayesian (Belief) Networks

- Directed Acyclic Graph on **all** variables.
- Allows combining prior knowledge about (in)dependences among variables with observed training data



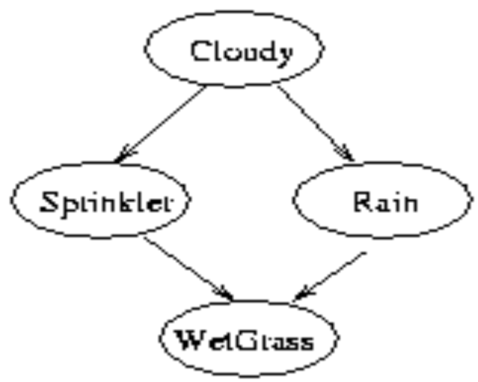
1. Any variable is conditionally independent of all non-descendent variables given its parents.
2. Graph also imposes partial ordering, e.g. A,B,C,D,E,F

   From (1) and (2), get P(A,B,C,D,E,F) = P(A)P(B)P(C|A,B)P(D)P(E|C,D)P(F|B) = Π_i P(node i | parents of node i)

Example – Wet Grass (Murphy 01)

- See http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html
- From data, get Conditional Probability Distribution/Table (CPD or CPT) for each variable.

P(C=F)	P(C=T)
0.5	0.5



C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

Takeaways

- Network structure is modeling assumption
  - Exploit domain knowledge
  - Few edges means more independence among variables , so smaller CPTs
- Very flexible: can infer in any direction and involving any subset of variables.
  - Also suggest explanations
- Training data used to fill up the CPTs

Network Properties and Usage

- Network structure is a modeling assumption, not necessarily unique
  - Some are better than others (good data fit + low complexity)
- If causality is known, make network from root causes …. to end effects.
- Usage: Inferencing
  - Infer the (probabilities of) values of one or more variables given observed values of some others.

Software: OpenBUGS http://mathstat.helsinki.fi/openbugs/

Infer.nehttp://research.microsoft.com/en-us/um/cambridge/projects/infernet/default.aspx

R packages such as bnlearn

Inference: Effect to Cause (Bottom Up)

We observe the grass is wet. Is this because of sprinkler or because of rain? (T=1, F=0).

$P(S=1|W=1) =$

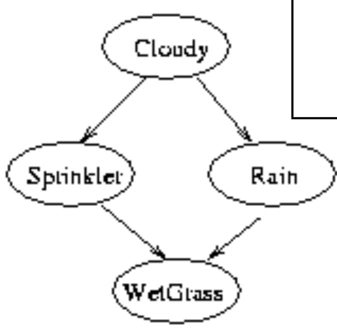
$\Sigma_{c,r}$ $P(C=c,S=1, R=r, W=1)$ / **P(W=1) = 0.2781/ .6471 = .43**

$P(R=1|W=1) =$

$\Sigma c,s$ $P(C=c,S=s, R=1, W=1) / P(W=1)$ **= 0.4581/ .6471 = .708**

So more likely it is because of rain!

P(C=F)	P(C=T)
0.5	0.5



C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

Special Types of inference

Diagnostic - B is evidence of A (bottom-up)

previous example

Predictive - A can cause B (top-down)

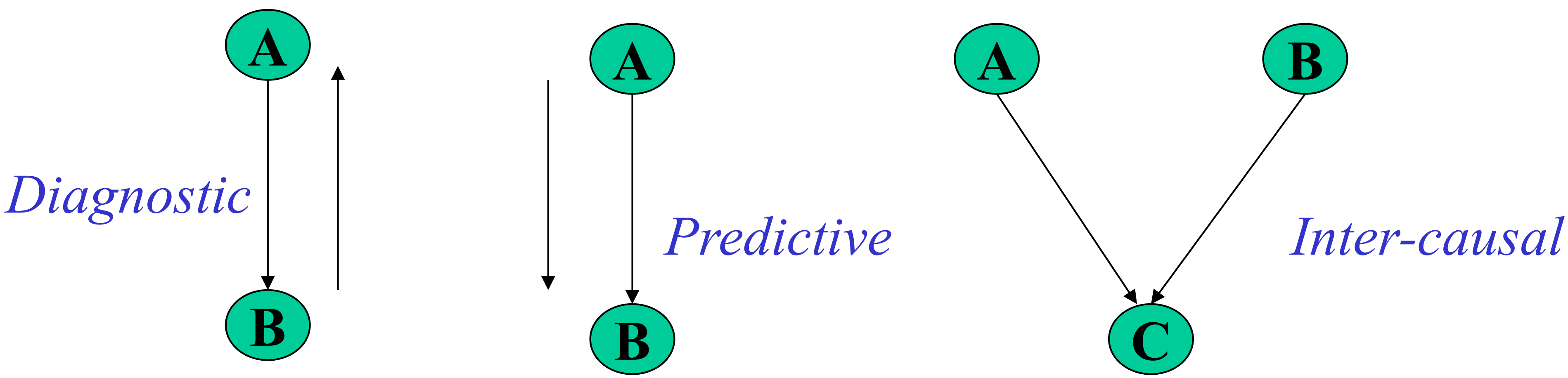
- e.g. P(grass wet | cloudy)

Inter-causal - suppose both A and B can cause C

if A "explains" C, it is evidence against B

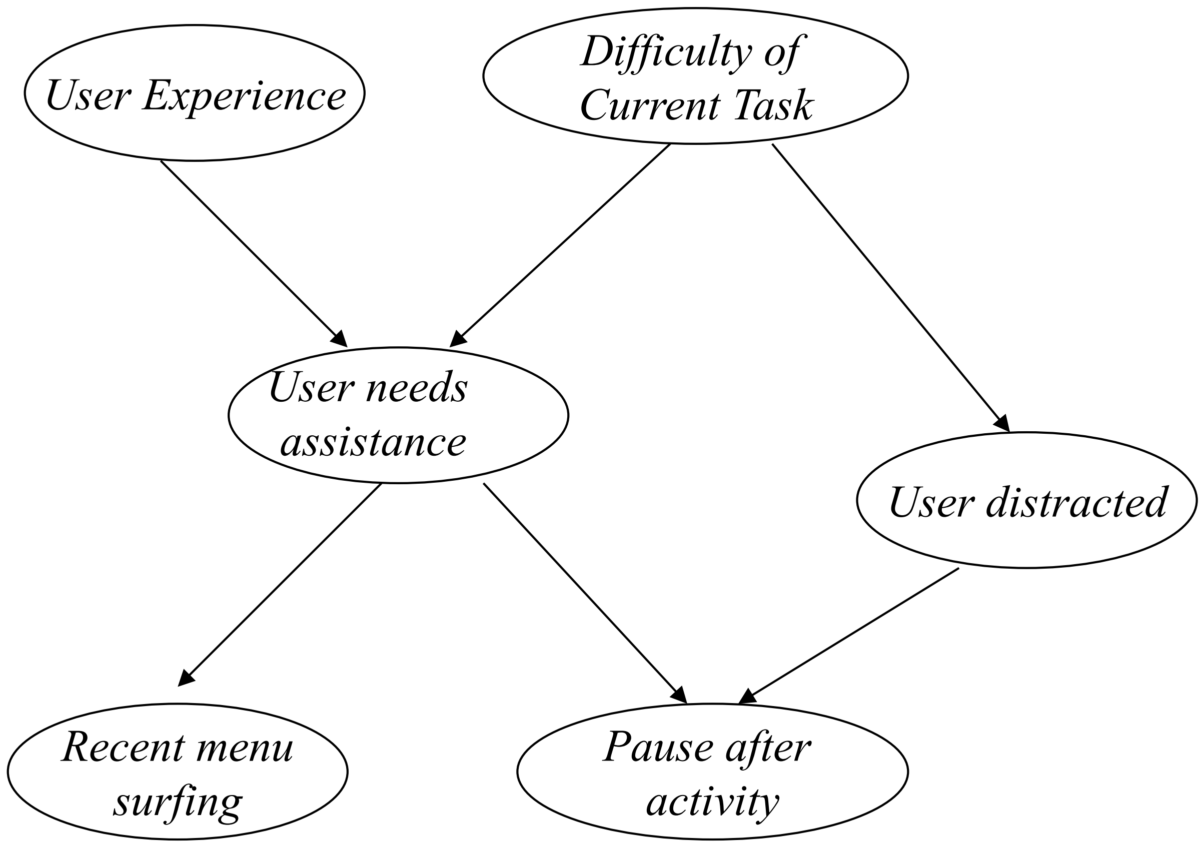
e.g. $P(S=1 \mid W=1, R=1) = 0.1945$, i.e. lower chance of sprinkler being ON if one also knows that it rained!

("explaining away","Berkson's paradox", or "selection bias")



Microsoft Office Assistant

- Only part of Bayesian network shown (Horvitz et al, Lumiere Project)



More Classification Methods

Directly getting to the Posterior: Logistic Regression, Neural Networks

Misc: SVMs

Logistic Regression

- Models a categorical variable (e.g. class label) as a function of predictors
- Studied extensively and have well-developed theory (variable selection methods, model diagnostic checks, extensions for dealing with correlated data)

Let $y(\mathbf{x})$ = indicator Target variable**;** $\mu = E(y \mid \mathbf{x}) = P(C_1|\mathbf{x})$

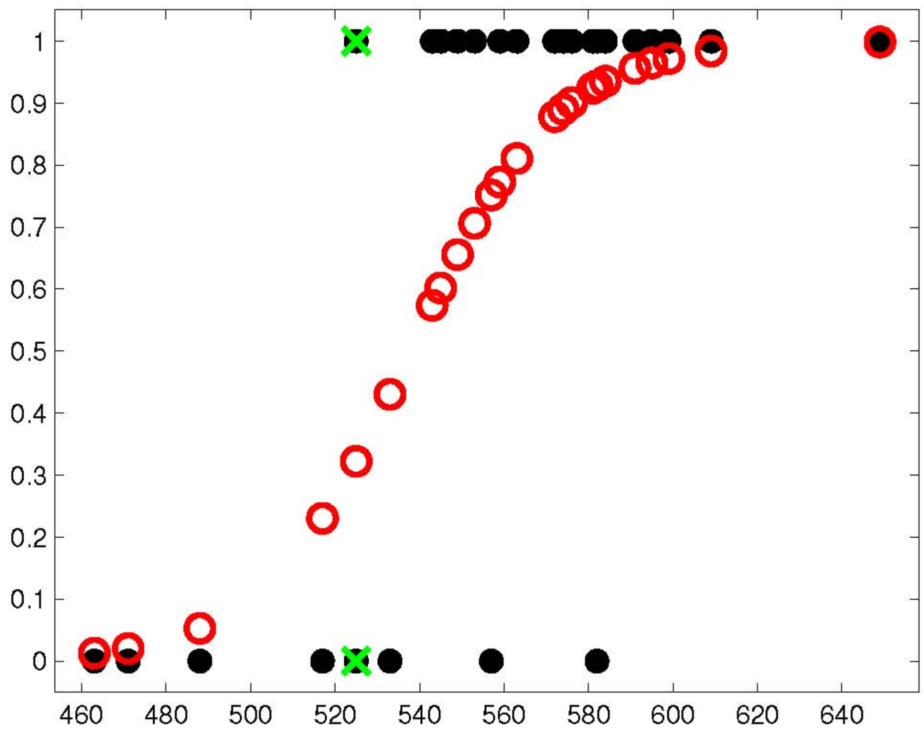
Model: $\ln [\mu / (1 - \mu)] = \boldsymbol{\beta}_0 + \beta_1 x_1 + \ldots + \beta_k x_k$

$$= \boldsymbol{\beta} . \mathbf{x} \qquad \text{(I)}$$

- i.e. model "log-odds ratio" (aka **logit**) as a linear function of predictors

Formulation

- Equivalently: model $\mu = 1/(1+\exp(-\boldsymbol{\beta}.\,\mathbf{x})) = \boldsymbol{\sigma}\,(\boldsymbol{\beta}.\,\mathbf{x})$
  - σ is called **the logistic function**, which is the inverse of logit
  - Rewrite model as
  - $\mu = \exp(\boldsymbol{\beta}.\,\mathbf{x})/(\exp(\boldsymbol{\beta}.\,\mathbf{x}) + \mathbf{1})$ and note that $1 = \exp(0)$ to get a hint of how to generalize for more than 2 classes.



SAT data
From KM pg 259.

Visit to SAS


Y
H
C
B
$\Pr(Y=1)=\dfrac{1}{1+\exp\{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\beta_4 x_4)\}}$
G
O
A
X
K
P
L
-Y
c
B
x

Training*

Minimize Negative Log-Likelihood (NLL) of a suitable probability model

- Implied Stat Model: y' s are i.i.d. Bernoulli
  - $p(y|\mathbf{x}, \beta) = \text{Bernoulli}(y \mid \boldsymbol{\sigma}(\boldsymbol{\beta} . \mathbf{x}))$
  - So $\text{NLL}(\beta) = - \Sigma_i [\, y_i \log \mu_i + (1-y_i) \log(1- \mu_i) \,]$

  (cross-entropy error function)

  If we use $\tilde{y}_i \in \{-1,1\}$ *then*

  $\text{NLL}(\beta) = \Sigma_i \log(1+ \exp(- \tilde{y}_i \, \beta . \mathbf{x}_i))$

  Takeaway: a non-linear max-likelihood problem needs to be solved iteratively.

  The unknown parameters (β) are estimated by maximum likelihood. (gradient descent or iterative solutions, e.g. Newton-Raphson, or iterative reweighted least squares see KM 8.3)

-

Properties

- Logistic regression is an example of a generalized linear model (GLM), with canonical link function = logit, corresponding to Bernoulli (see glmnet in R)

- Disadvantages:
  - Parametric – but form works for entire exponential family of distributions
  - Solution not closed form, but still reasonably fast
- Advantages:
  - Have parameters with useful interpretations
    - the effect of a unit change in x_i is to increase the odds of a response multiplicatively by the factor $\exp(\beta_i)$
  - Quite robust, well developed

Multiclass Logistic Regression

- Extension to K classes: use K-1 models
  - one each for ln $[P(C_i|x)/P(C_k|x)]$
  - Set all coefficients for class K to 0 (to make the system **identifiable;** this choice is arbitrary)

- Put them together to get posteriors.
  - $P(C_i|\mathbf{x}) = \exp(\beta_{i0} + \beta_{i1}x_1 ..) / (1 + \Sigma_j \exp(\beta_{j0} + \beta_{j1}x_1 ..)$, $i \neq k$
  - $P(C_K|\mathbf{x}) = \ldots.$

Multilayered Feedforward Networks for Classification

- choose sufficiently large network (no. of hidden units)
- trained by "1 of M" desired output values
  - ("one-hot coding")
- use validation set to determine when to stop training
- try several initializations, training regimens
- + powerful, nonlinear, flexible
- - interpretation? (Needs extra effort); slow ?

ANNs as Approximate Bayes Classifiers

- Output of "universal" Feedforward neural nets (MLP, RBF) trained by "1 of M" desired output values, estimate Bayesian *a posteriori* probabilities if the cost function is

  mean square error OR cross-entropy

Nowadays some packages use softmax, but often not needed.

- Significance:
  - interpretation of outputs; quality of results
  - setting of thresholds for acceptance/rejection
  - can combine outputs of multiple networks

Support Vector Machines

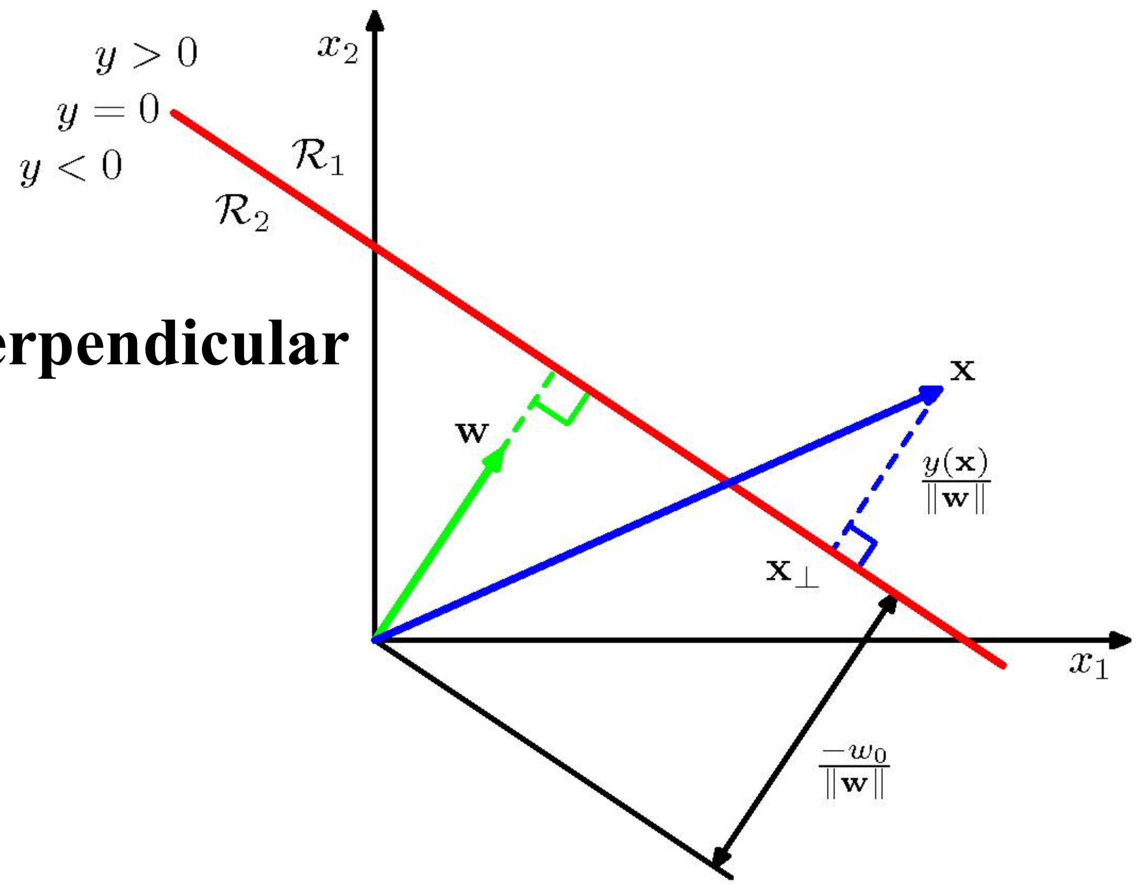
- A leading edge classifier
  - Uses "optimal" hyperplane in a suitable feature space to classify

Good software available:

- A list of SVM implementations can be found at http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- **Latest:** Stochastic gradient descent techniques (e.g. Leon Bottou's work) – much faster for very large datasets.

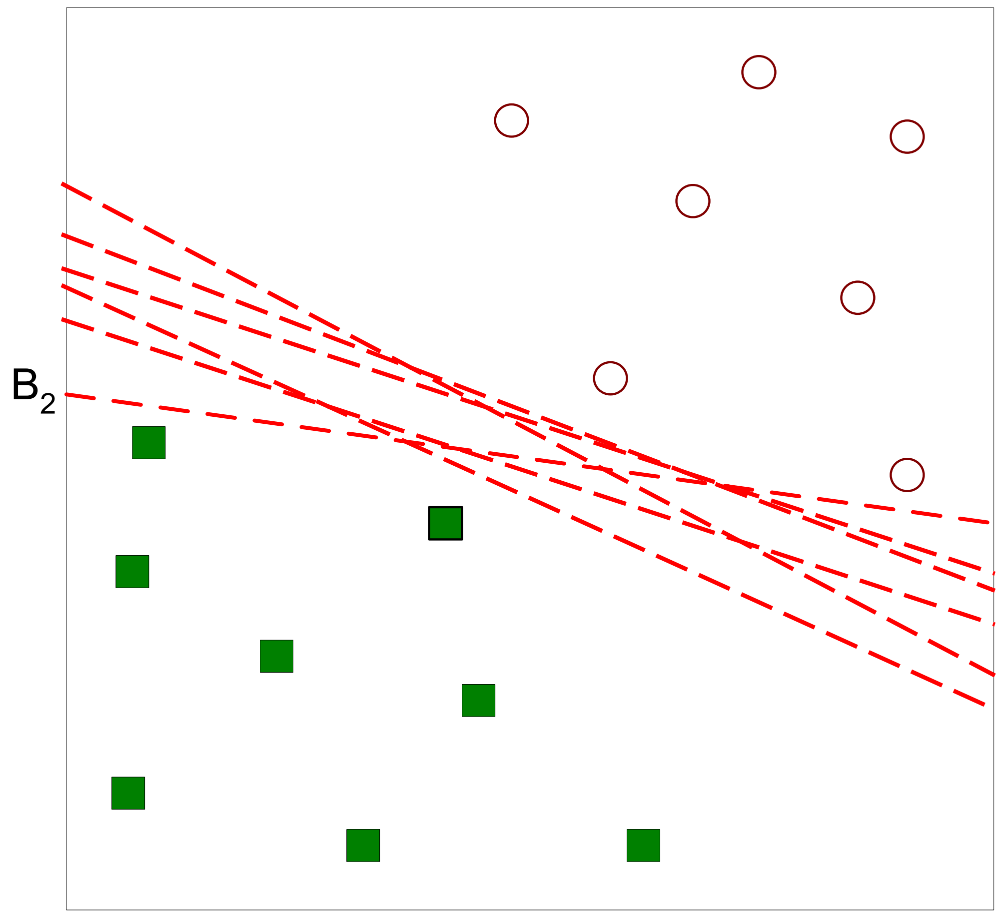
Geometry of Linear Classifiers

- Reference: Bishop 2006 (B06): 4.1 – 4.3.4.
- You get linear boundaries if discriminant functions (or some monotonic transform thereof) are linear
  - Geometry for 2 classes:
  - $y(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0$



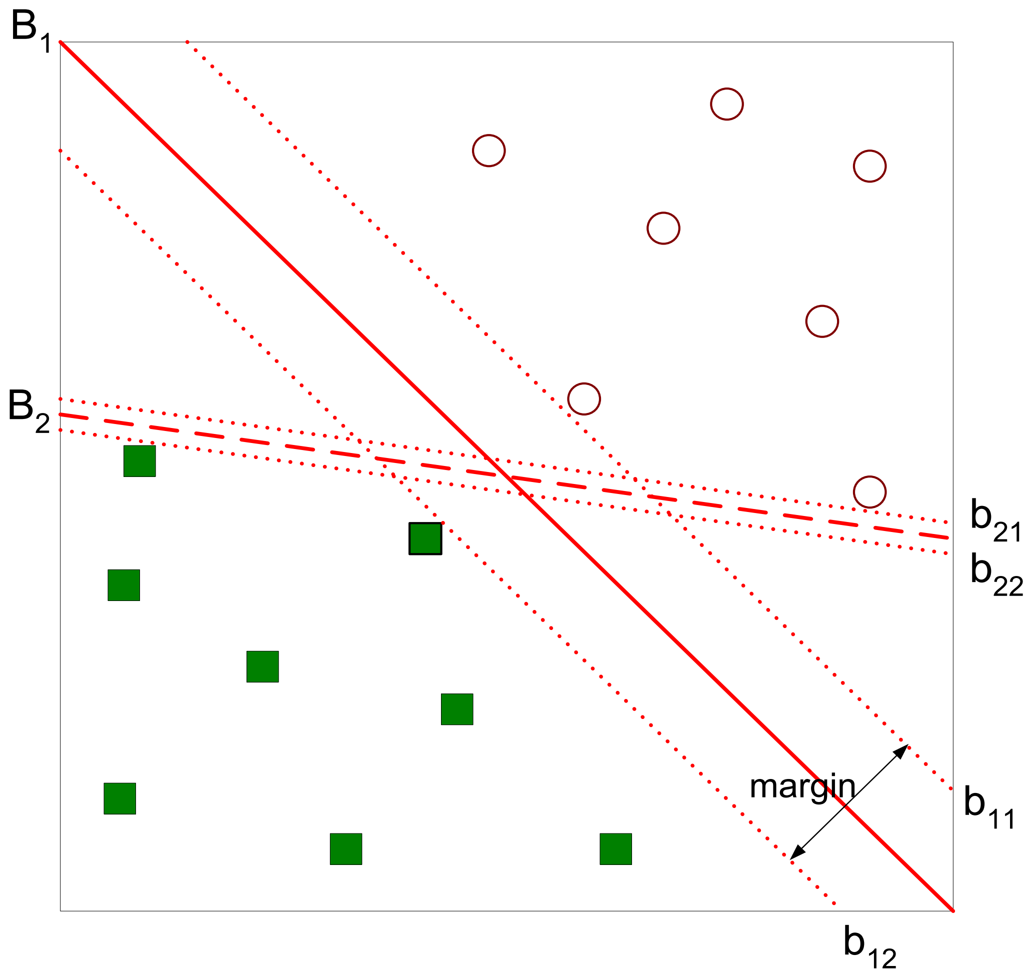
So class boundary is perpendicular to the weight vector.

How to Choose a Linear Classifier?



Which solution to choose?

Maximum Margin Classifier



- Find hyperplane that maximizes the margin => B1 is better than B2

The (Primal) Optimization Problem

- Rescale the data so that the points closest to separating hyperplane satisfy: wx + b = 1 (class 1) or wx + b = -1 (class 0)
  - These points form the Support Vectors

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$

- Constrained optimization problem: Maximize margin (actually the squared norm is minimized for convenience using Quadratic Programming), subject to:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \leq -1 \end{cases}$$

Slack Variables*

- What if the problem is not linearly separable?
  - Introduce slack variables (ξs)
    - Need to minimize:

$$L(w) = \frac{\| \vec{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k \right)$$

Takeaway: a slack penalty C is needed to specify the cost of any violations – points on wrong side of margin.
C governs a bias-variance tradeoff

    - Subject to:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 - \xi_i \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 + \xi_i \end{cases}$$

  - Can rewrite as ("Hinge loss" form, with $\lambda = 1/nC$; targets y = +/- 1):

$$\underset{w}{\mathit{minimize}} \; \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y(w \cdot x))$$

Working in Dual Space*

- Solve by Lagrangian multipler method to get

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Note: The dual variables, α's are non-zero only for support vectors

Then
$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

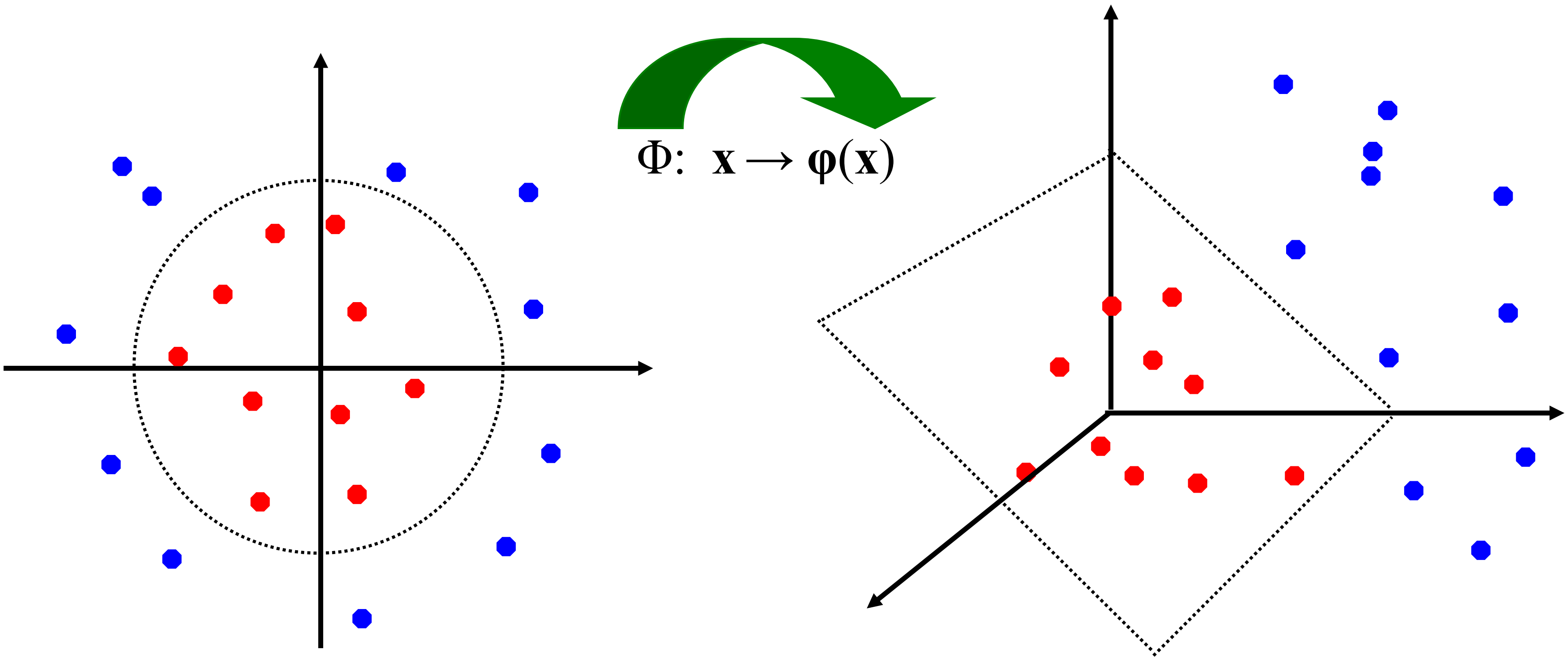
so output for test "**z**" is $f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$

Finally threshold to get class label: f > 0 => class 1.

– Note: vectors **x, z** appear only as dot products.

Nonlinear Support Vector Machines

- What if decision boundary is not linear?



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

The "Kernel Trick"

- If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$$

- A *kernel function* is a function that is equivalent to an inner product in some feature space.

- Thus, a kernel function *implicitly* maps data to a high-dimensional space and does inner product
  – without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly!!

Example Transformation

- Define the kernel function K (**x**,**y**) as

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1 y_1 + x_2 y_2)^2$$

- Consider the following transformation

$$\phi(\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right]) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$

$$\phi(\left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right]) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1 y_2)$$

- The inner product can be computed by K without going through the map $\phi(.)$

$$\begin{aligned} \langle \phi(\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right]), \phi(\left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right]) \rangle &= (1 + x_1 y_1 + x_2 y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned}$$

.. And Work in Dual Space*

- Change all inner products to kernel functions
- For training (to get the dual variables, α' s),
  - α' s are non-zero only for support vectors:

Original

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

With kernel function

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Modification Due to Kernel Function*

- For testing, the new data **z** is classified as class 1 if $f \geq 0$, and as class 0 if $f < 0$

Original

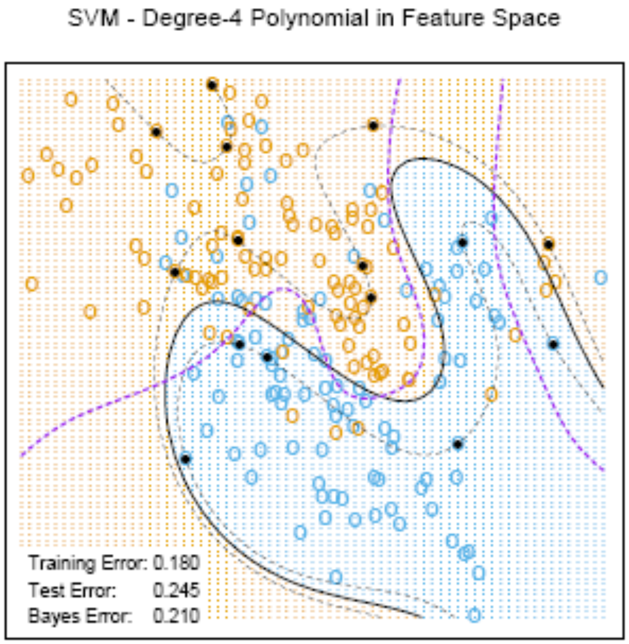
$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

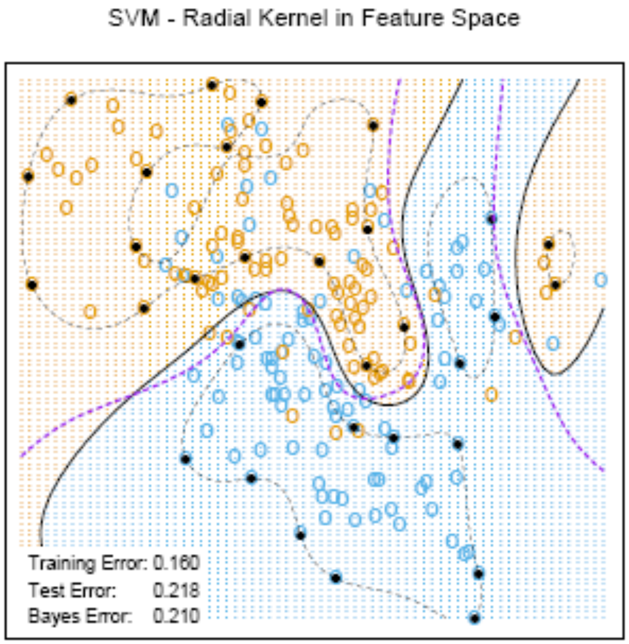
$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$





Effect of Kernel Choice

FIGURE 12.3. *Two nonlinear SVMs for the mixture data. The upper plot uses a 4th degree polynomial kernel, the lower a radial basis kernel (with $\gamma = 1$). In each case C was tuned to approximately achieve the best test error performance, and $C = 1$ worked well in both cases. The radial basis kernel performs the best (close to Bayes optimal), as might be expected given the data arise from mixtures of Gaussians. The broken purple curve in the background is the Bayes decision boundary.*

(From HTF Ch 12)

More on Kernel Choices

- There is a rich variety of kernels (e.g. string kernels, graph kernels etc) for different applications

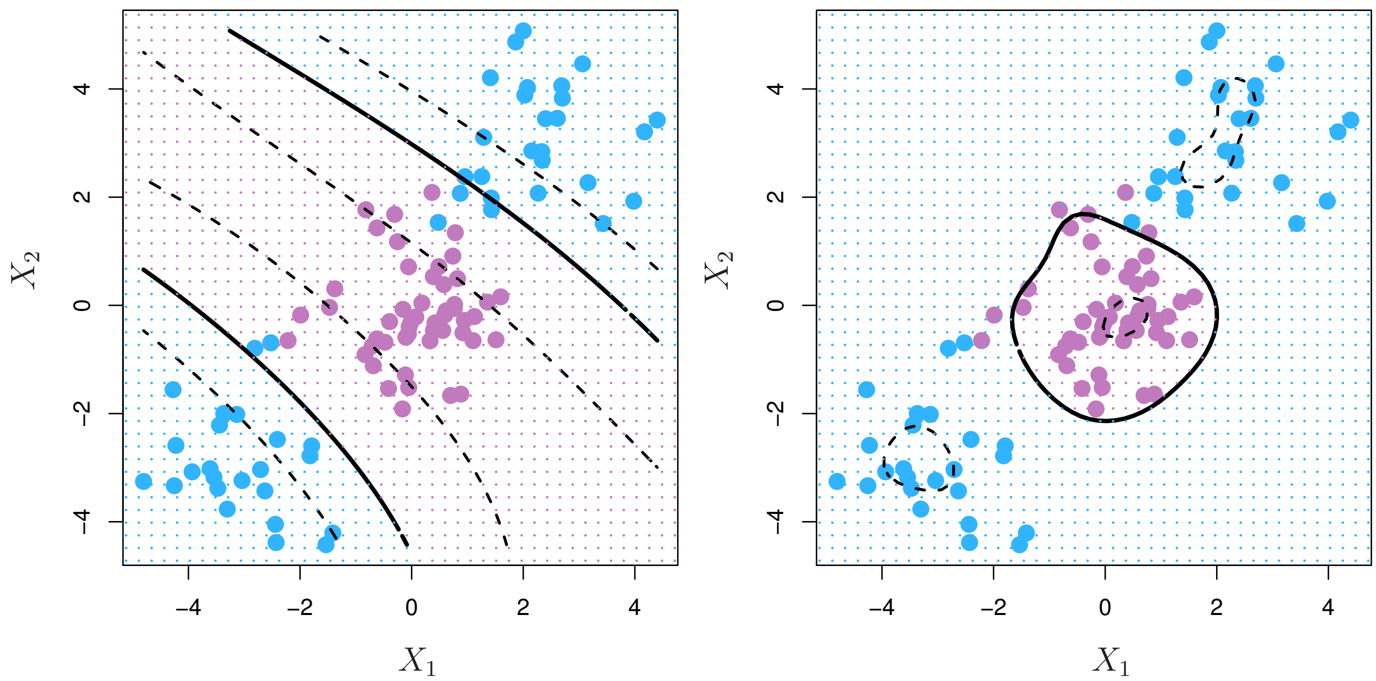


Fig 9.9 of ESLR: Left: polynomial kernel of degree 3
Right: RBF kernel

Summary: Steps for Classification

- Select the kernel function to use
  - Generic choices: linear, gaussian, polynomial
- Select the parameter of the kernel function (if any) and the value of slack variable *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 parameters
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

SVMs: Key Takeaways

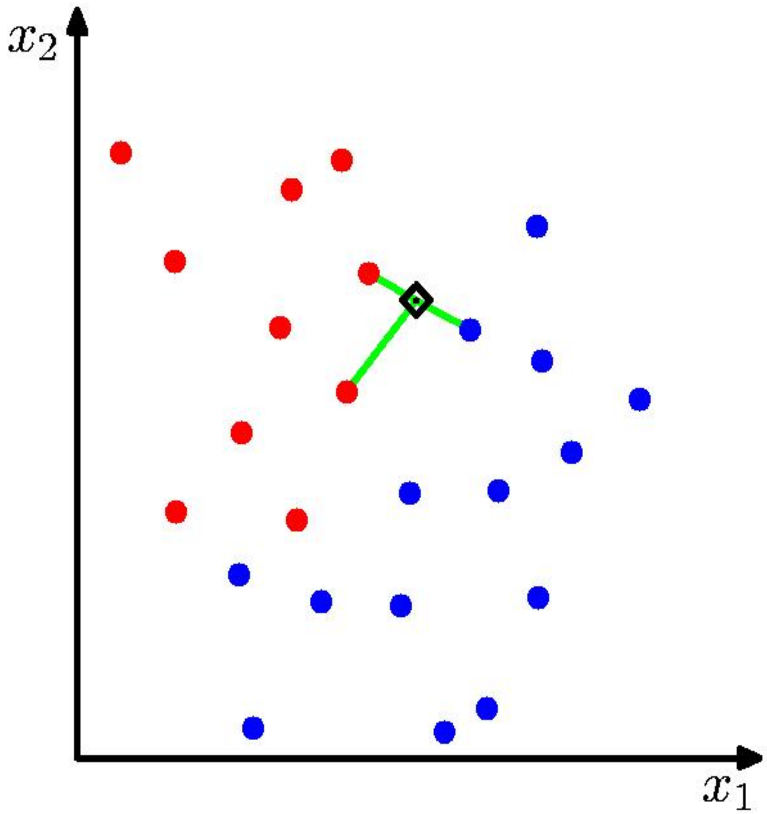
- Discriminative Classifier: gives a class label
  - Hacky procedure to get posterior probabilities
  - Naturally suited to 2-class problems, but multiclass versions exist
- Design choices:
  - Kernel function: determines notion of "similarity"
  - Slack variable, C: bias – variance tradeoff
    - If kernel function involves a tunable parameter σ (e.g. width of Gaussian kernel), then grid search is needed in 2-D parameter space (σ, C) to find best setting
- Properties:
  - SVMs fairly robust in (reasonably) high-D
  - Suitable kernels exist for certain complex data types

Strengths and Weaknesses of SVM

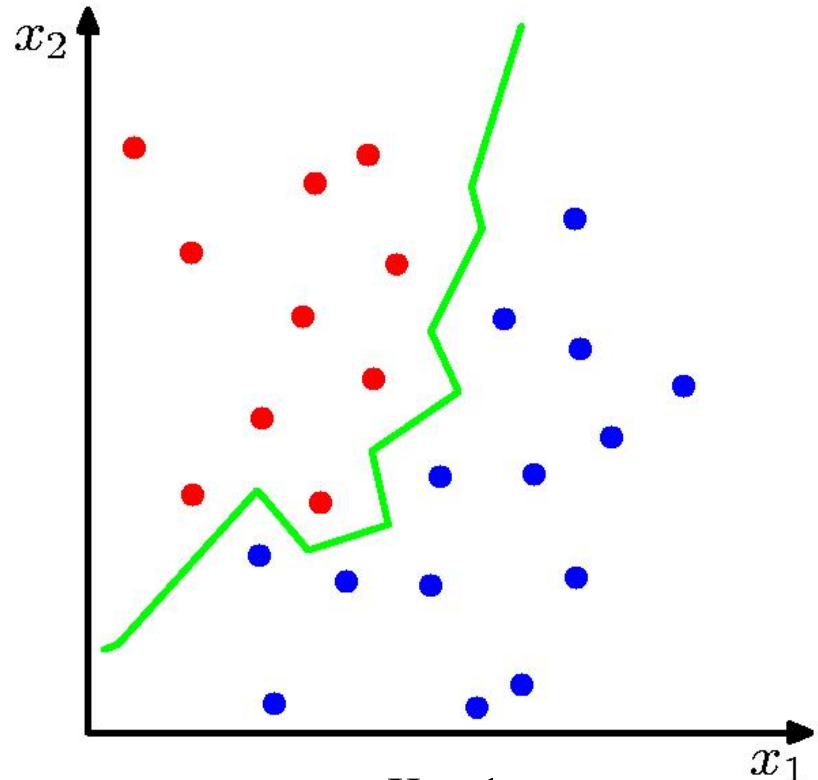
- Strengths
  - Currently among the best performers for a number of classification tasks ranging from text to genomic data.
    - Outside of some big data settings where deep learning is clearly better.
  - **Can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.**
  - It scales relatively well to high dimensional data

- Weaknesses
  - Need a “good” kernel function
  - Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner.
  - Slow! But new SGD methods are able to scale to large data

K-Nearest-Neighbours for Classification (2)

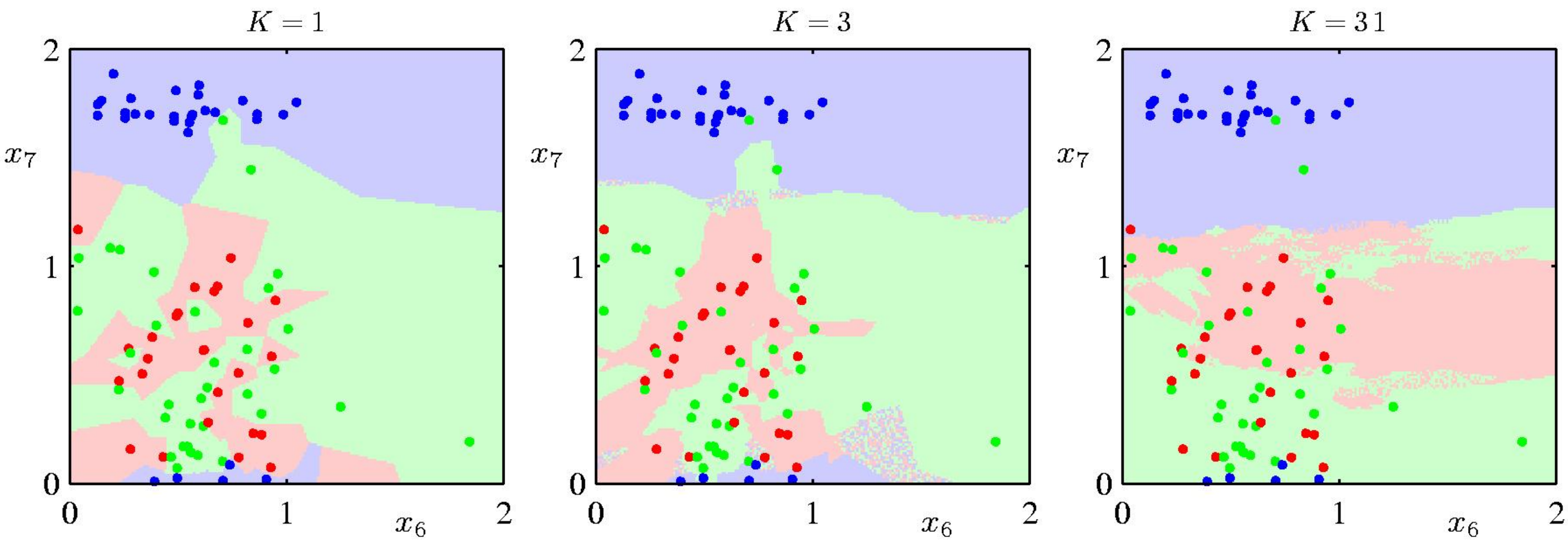


K = 3



K = 1

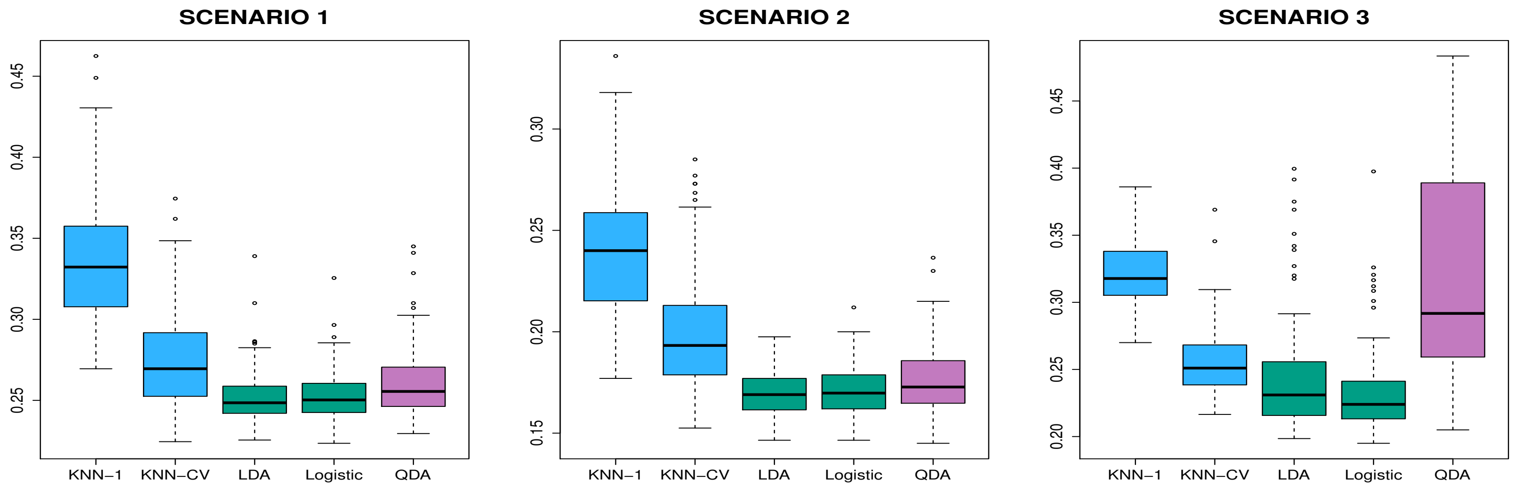
K-Nearest-Neighbours for Classification (3)



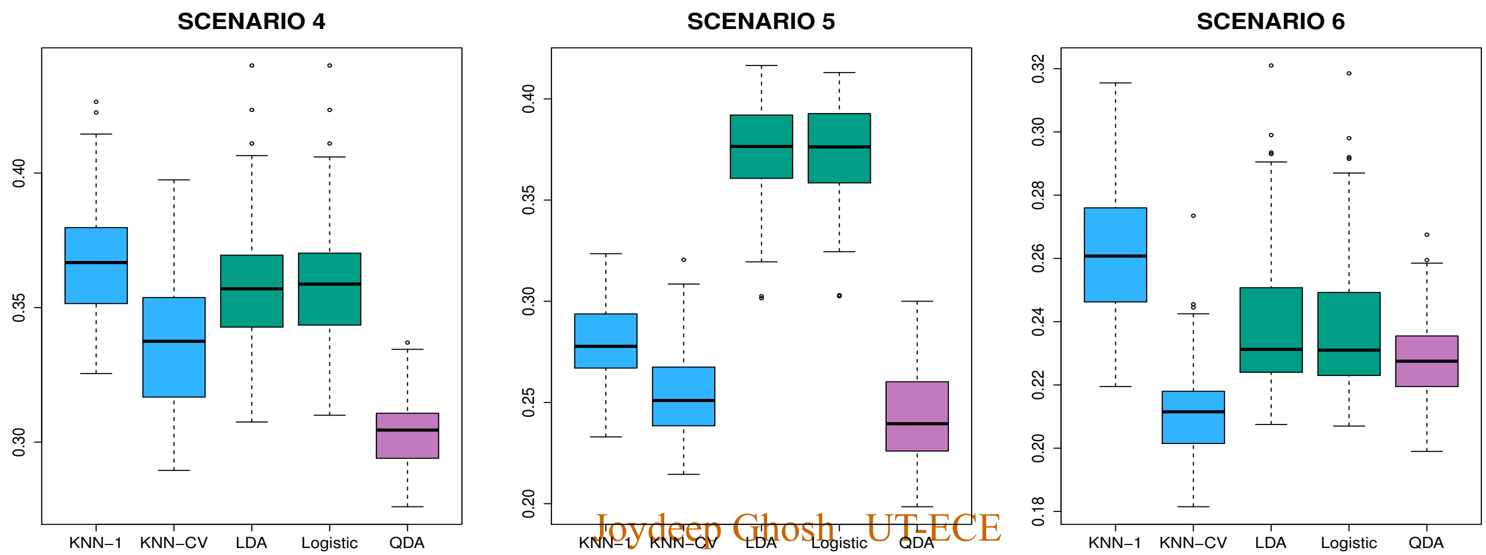
- K acts as a smother
- For $N \to \infty$, the error rate of the 1-nearest-neighbour classifier is never more than twice the optimal error (obtained from the true conditional class distributions).

No Silver Bullet

- Fig 4.10 of ESLR (linear scenarios)



- Fig 4.11 of ESLR (non-linear scenario)



Revisiting The Classification Methods

- Approximating Bayes Decision Rule: (model the likelihood)
  - Linear Discriminant Analysis/ QDA
  - **Fisher's linear discriminant**
  - **Naïve Bayes**
  - **Bayesian Belief Networks**
  - K-Nearest Neighbor
- Approximating Bayes Decision Rule: (directly model the posterior)
  - **Logistic Regression**
  - **Feedforward neural networks, including deep learning**

Question: What are the assumptions made in each approach?

So which method should I choose?

- Depends on type, complexity of problem; data size
  - Binary/few categories in each variable? Try DT
  - continuous variables: first try linear (Fisher) discriminant
    - also try 1 or 3-nearest neighbor if memory is not a problem
  - reasonably linear (in transformed space): logistic regression
    - Generally quite robust, may add ridge regression penalty to regularize
  - performance? try MLP
    (estimate complexity of fit using a few trial runs)
    - *RBFs good for low-dimensional space
    - SVMs fairly robust in (reasonably) high-D
      - Also suitable if good kernel is known
- Still lacking? try ensemble approaches


SUPPORT VECTOR MACHINES
REPEA POWE LAWS
END DUALITy GAP
Map Reduce
Map Reuse
Map Recycle
BAYSIANS AGAINST DISCRIMINATION
FREE VARIABLES!
BAN GENETIC ALGORITHMS


BRING
THE
CRAZY
G20
GEN
ALGOR

Extras

Linear Regression of an Indicator Matrix

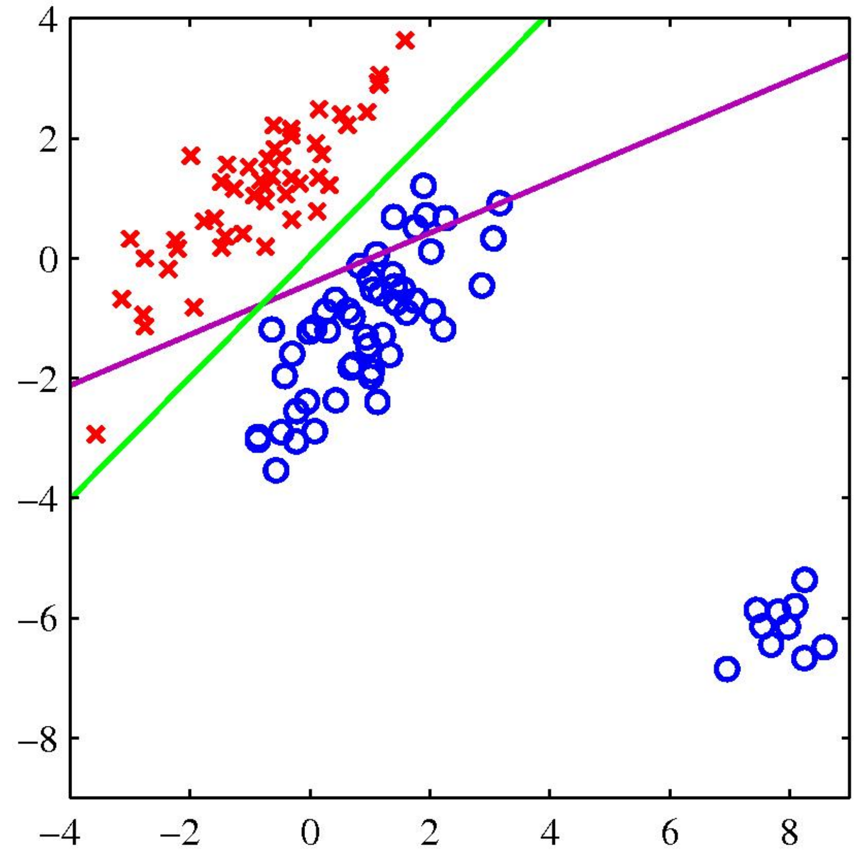
- 0/1 output encoding (aka 1 of C).
  - Decision: Pick class corresp to highest output

+ve: Sum $y_k(\mathbf{x}) = 1$,

-ve:y(x) may be outside [0,1].

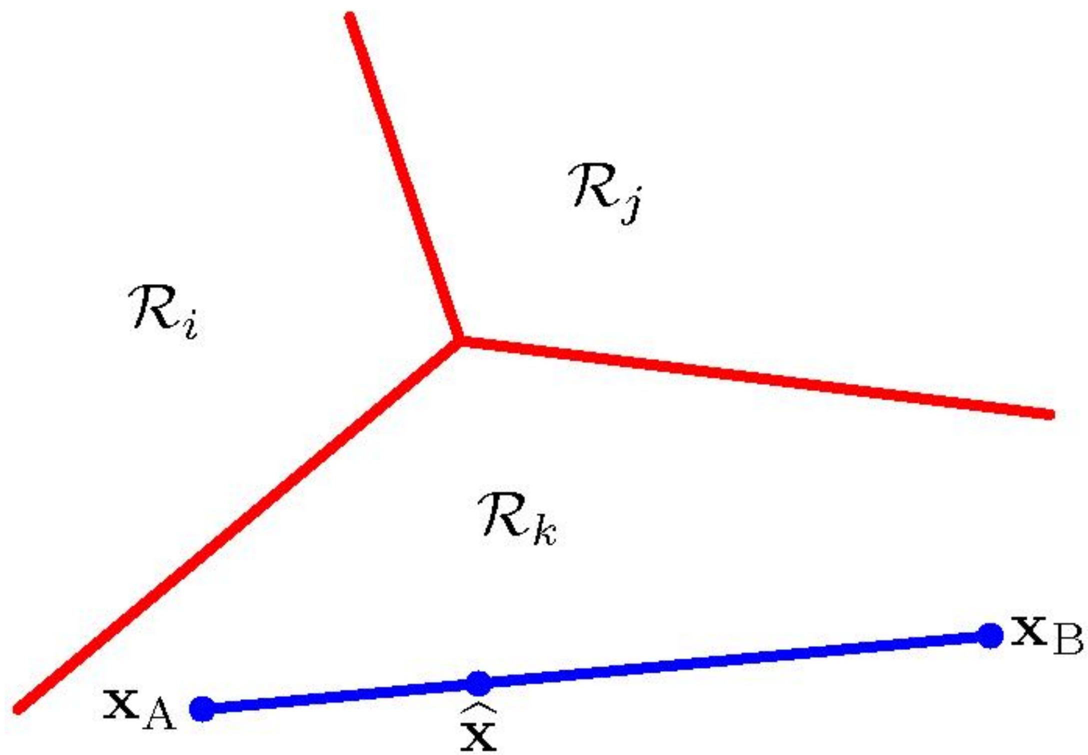
-ve: sensitive to outliers

(green: logistic vs. magenta, linear)



Geometry: Multi-class

– $y_k(\mathbf{x}) = \mathbf{w}_k^{\mathbf{T}}\mathbf{x} + w_{k0}$
  pick class k with highest $y_k(\mathbf{x})$
– Decision boundaries are singly connected and convex.



Perceptron (1960)

Online learning for separating two classes

$y(\mathbf{x}) = 1$ if $\mathbf{w}^T\mathbf{x} + w_0 > 0$ (class 1)

$y(\mathbf{x}) = -1$ if $\mathbf{w}^T\mathbf{x} + w_0 < 0$ (class 0)

($\mathbf{x}(k)$, $t(k)$) is k^{th} training example; $t(.) = +/- 1$

- <u>Perceptron learning rule</u>: ($\eta > 0$: Learning rate)

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta\,(t(k) - y(k))\,\mathbf{x}(k)$$

If $t(k)$ not equal to $y(k)$; else no weight update.

Perceptron Convergence Theorem

<u>Convergence Theorem</u> – If (<u>x</u>(k), t(k)) is <u>linearly separable</u>, then <u>W</u>* can be found in <u>finite number</u> of steps using the perceptron learning algorithm.

Also, Cycling theorem.

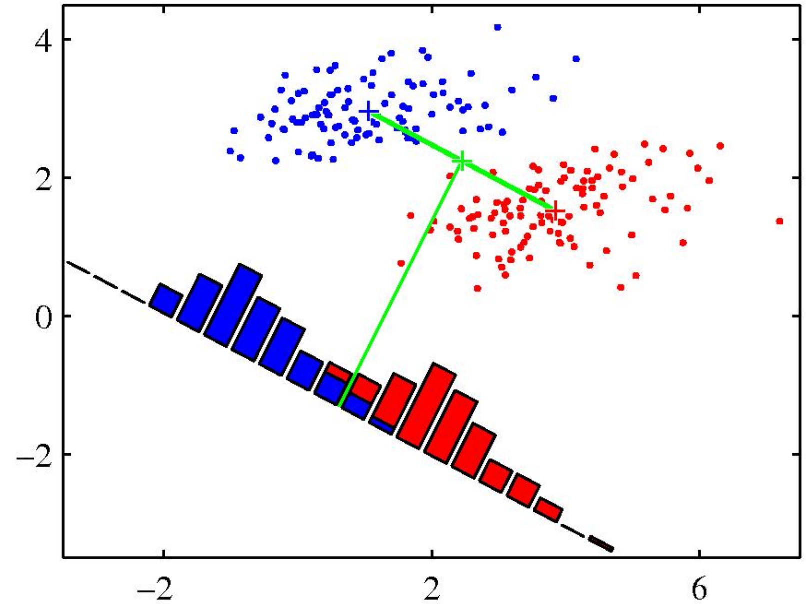
- Problems with Perceptron:
  - Can solve only linearly separable problems.
  - May need large number of steps to converge.
- But good for online learning, in non-stationary environments!
  - Improvements such as adaptive learning rate.

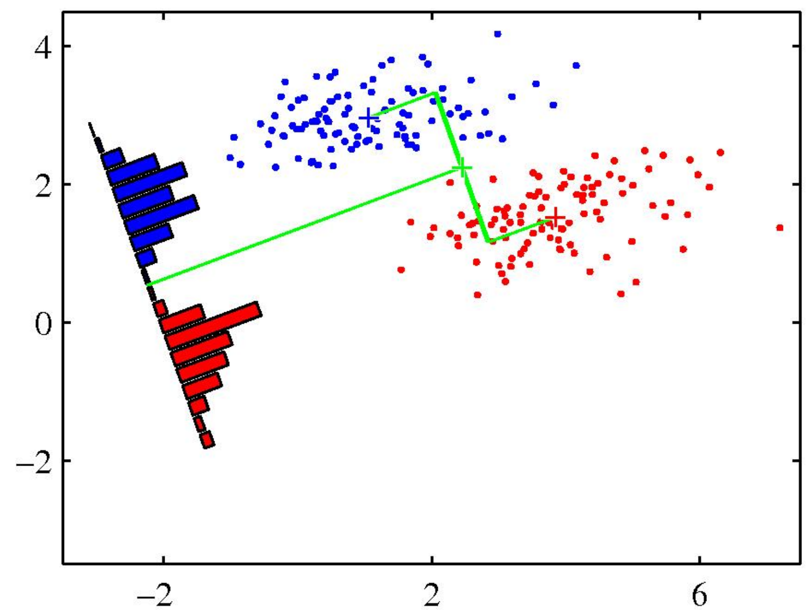
Fisher's Linear "Discriminant"

- Project data in direction **w** that maximizes ratio of between-class variance to within-class variance (of projected data).
- Model projected data by gaussian/class (why more reasonable than LDA?), and apply Bayes decision rule.

- **Projection Direction:** $\mathbf{S}_w^{-1}\,(\mathbf{m}_2 - \mathbf{m}_1)$, where $\mathbf{S}_w$ is the total within-class covariance matrix, $\mathbf{m}_i$ is the mean of class i.

- Generalize: can get C-1 projections for C class problem.





SVM as a Penalization Method (HTF)

- "Hinge Loss" form $$\underset{w}{minimize}\ \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y(w \cdot x))$$

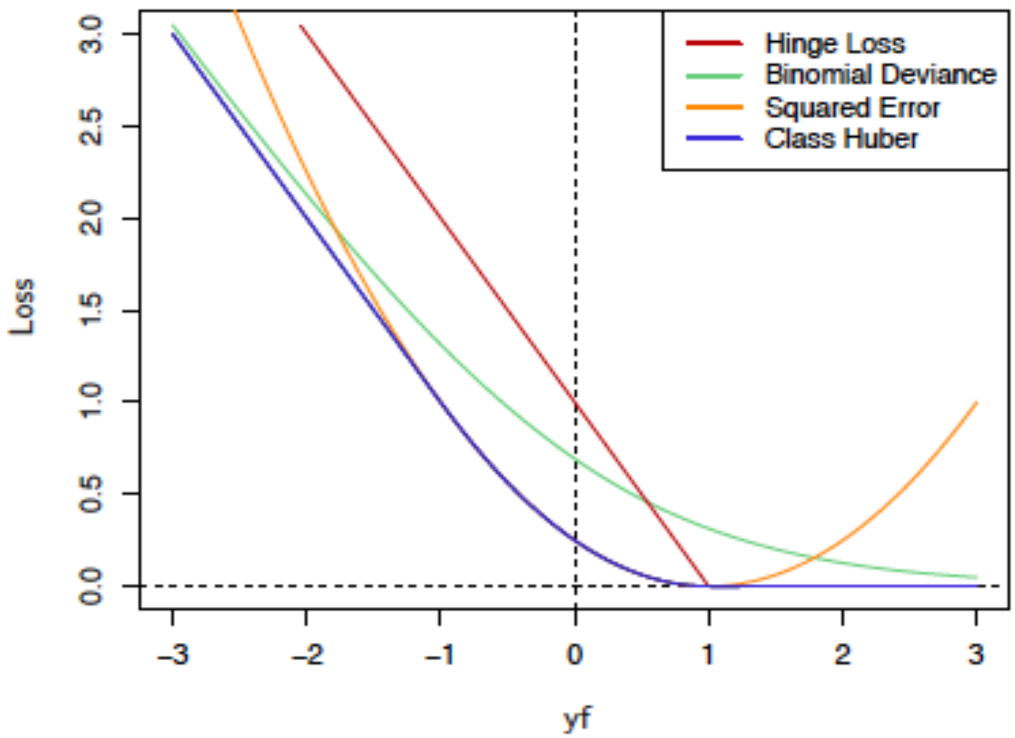


FIGURE 12.4. *The support vector loss function (hinge loss), compared to the negative log-likelihood loss (binomial deviance) for logistic regression, squared-error loss, and a "Huberized" version of the squared hinge loss. All are shown as a function of yf rather than f, because of the symmetry between the $y = +1$ and $y = -1$ case. The deviance and Huber have the same asymptotes as the SVM loss, but are rounded in the interior All are scaled to have the limiting left-tail slope of -1.*

K-Nearest-Neighbours for Classification (B06: 2.5.2)

- Assume uniform density for each class in the neighborhood of test point. Then, Given a data set with N_k data points from class C_k and $\sum_k N_k = N$, we have

$$p(\mathbf{x}) = \frac{K}{NV}$$

- and correspondingly

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V}.$$

- Since $p(\mathcal{C}_k) = N_k/N$, Bayes' theorem gives

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{K_k}{K}.$$