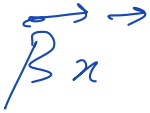
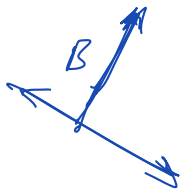
Support Vector Machines

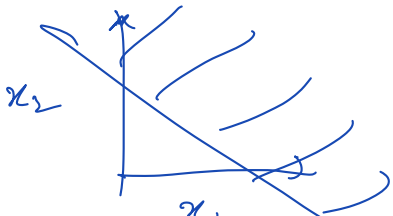
- A leading edge classifier
  - Uses "optimal" hyperplane in a suitable feature space to classify

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.

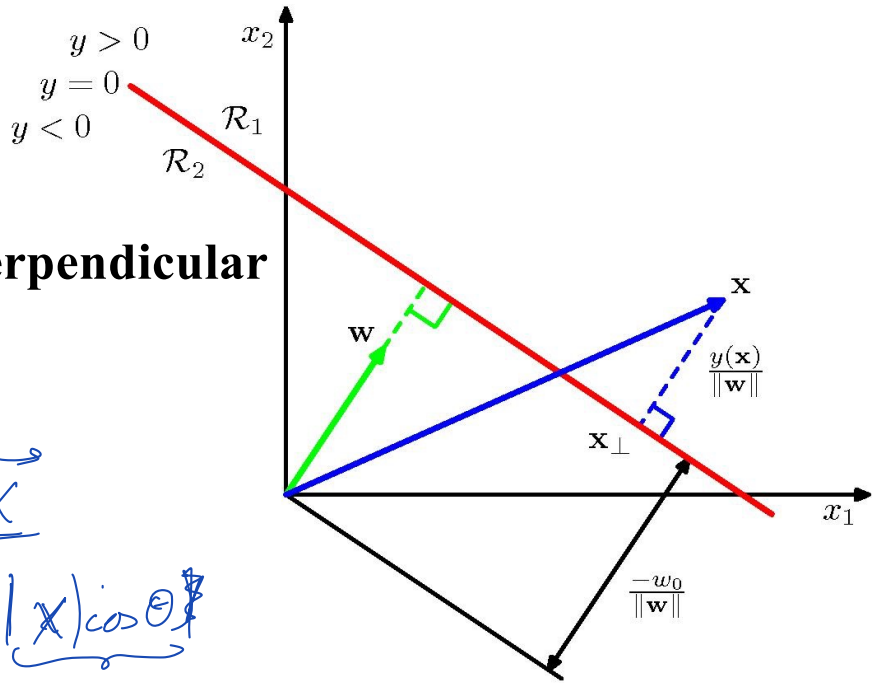






Geometry of Linear Classifiers

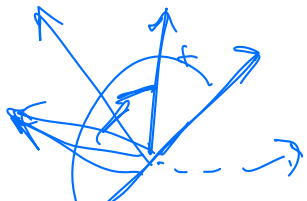
- Reference: Bishop 2006 (B06): 4.1 – 4.3.4.
- You get linear boundaries if discriminant functions (or some monotonic transform thereof) are linear
  - Geometry for 2 classes:
  - $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$



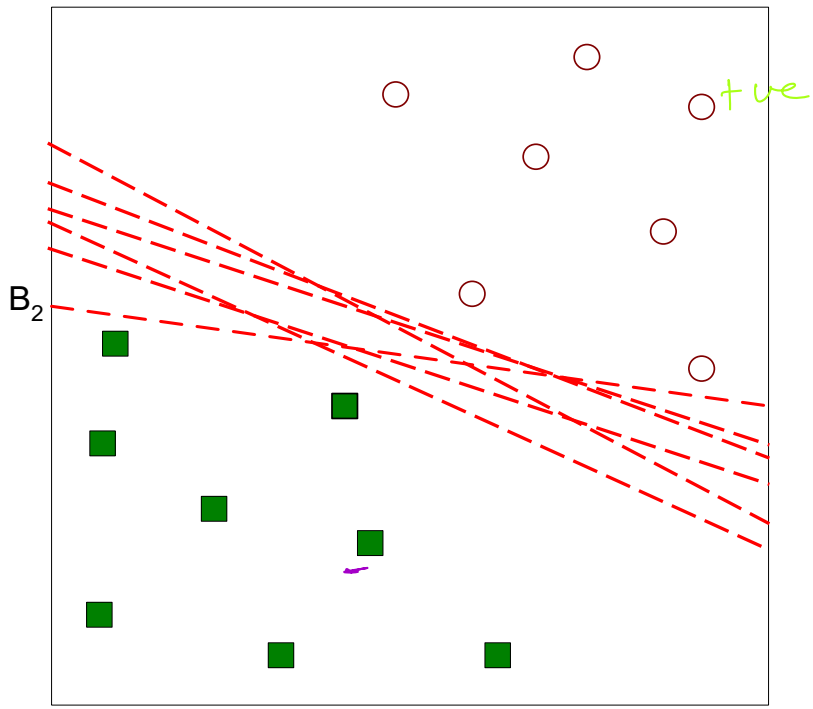
So class boundary is perpendicular to the weight vector.

$\vec{w} \cdot \vec{x}$

$|w|\,|x|\cos\theta$

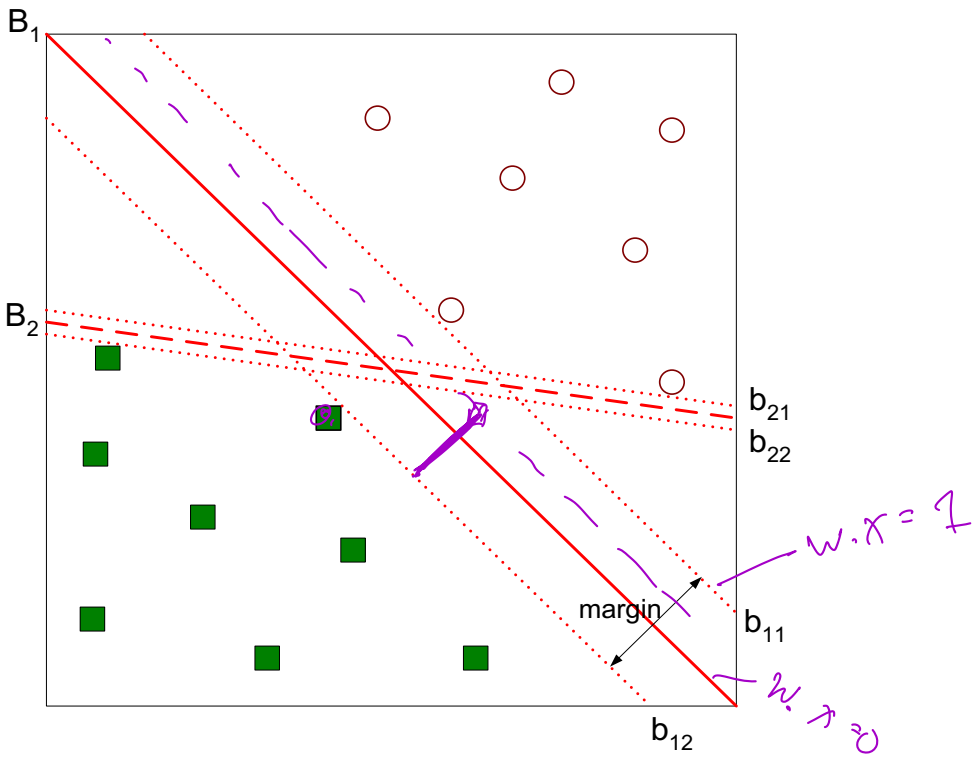


How to Choose a Linear Classifier?



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:

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

Slack Variables*

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

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

    - 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):

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

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

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

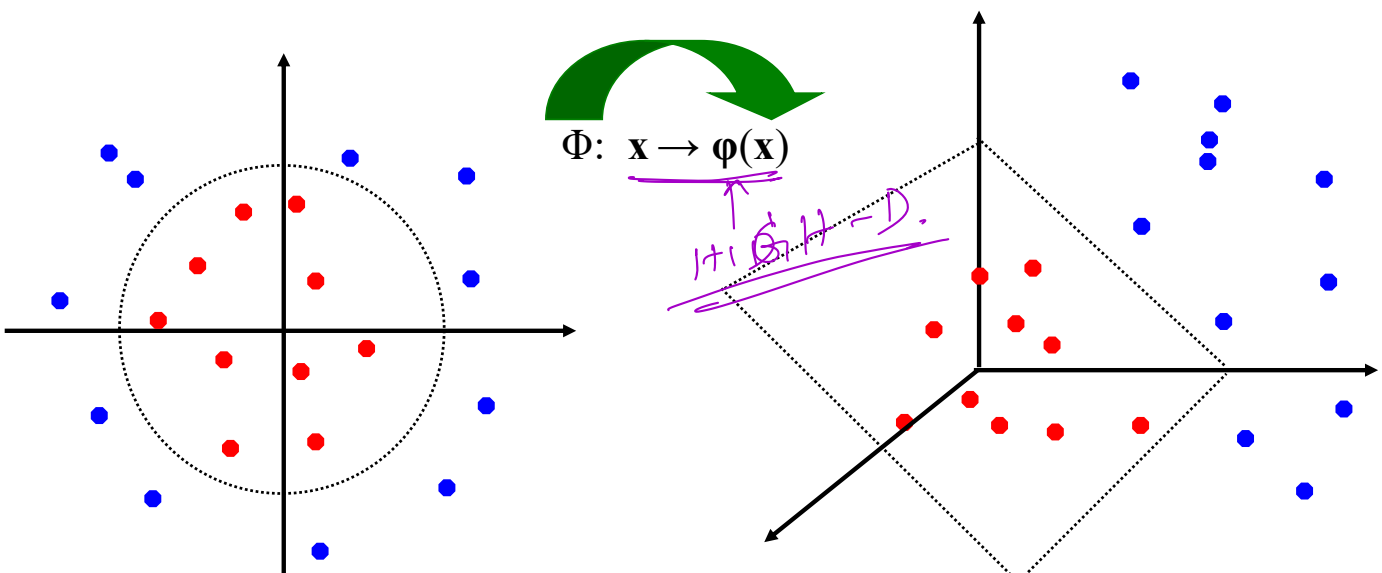
so output for test "**z**" is $$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

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:

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

"SIMILARITY"

- 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 $K(\mathbf{x},\mathbf{y})$ as

$$K(\mathbf{x},\mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- Consider the following transformation

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

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

$$\langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle = (1 + x_1y_1 + x_2y_2)^2$$
$$= K(\mathbf{x},\mathbf{y})$$

$e^{-\|x_i - x_j\|_2^2}$

.. 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

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

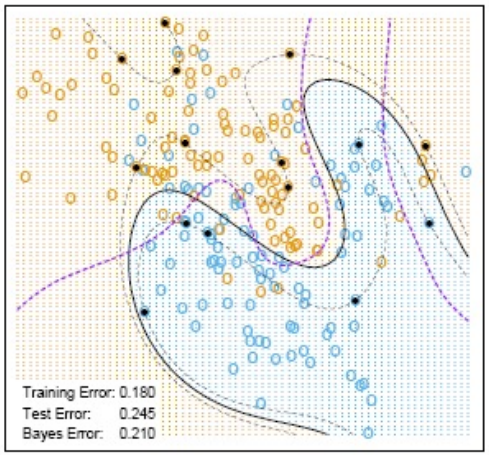
$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

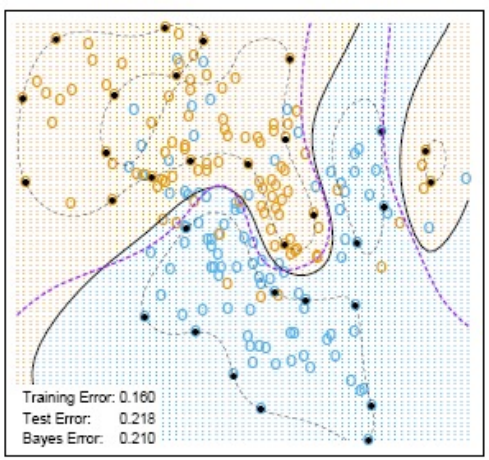
$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

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

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



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

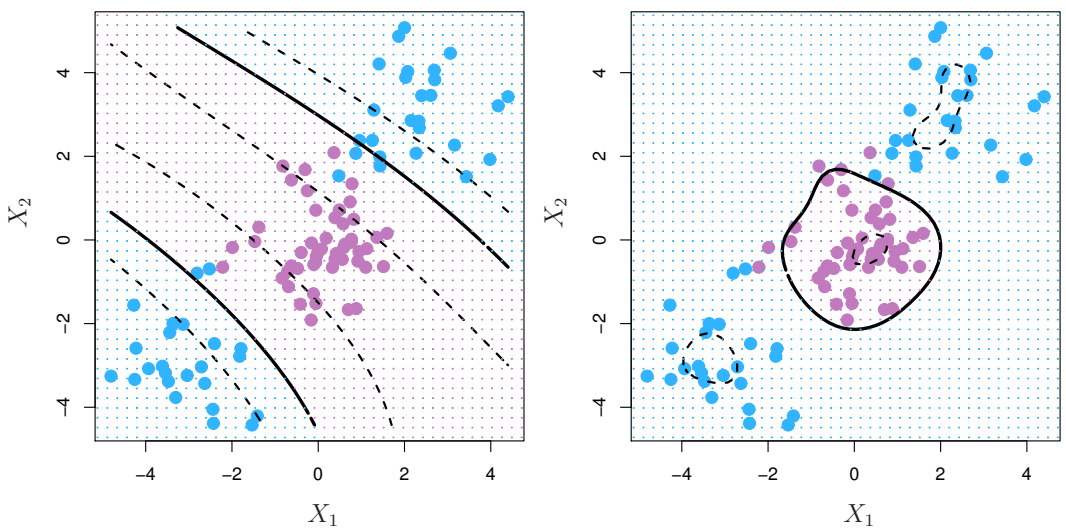
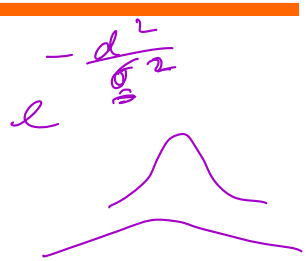


Fig 9.9 of ESLR: Left: polynomial kernel of degree 3 Right: RBF kernel

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

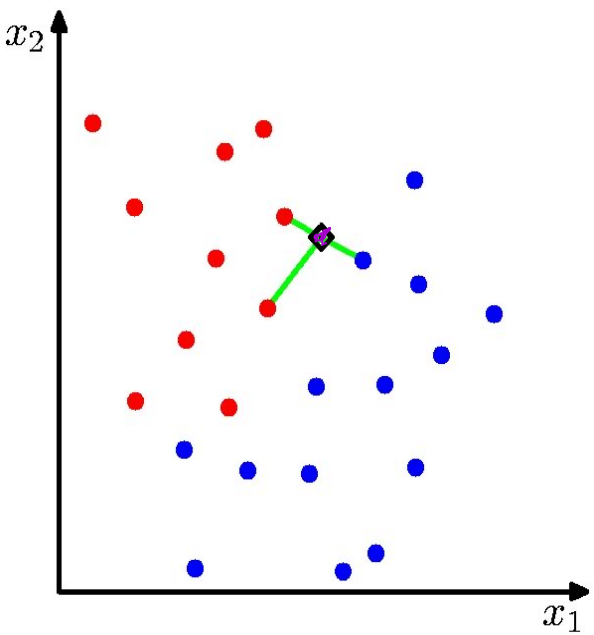
- Discriminative Classifier: gives a class label
  - Hacky procedure to get posterior probabilities
  - Naturally suited to 2-class problems, but multiclass versions exist
- Design choices:
  - Kernel function: determines notion of "similarity"
  - Slack variable, C: bias – variance tradeoff
    - If 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

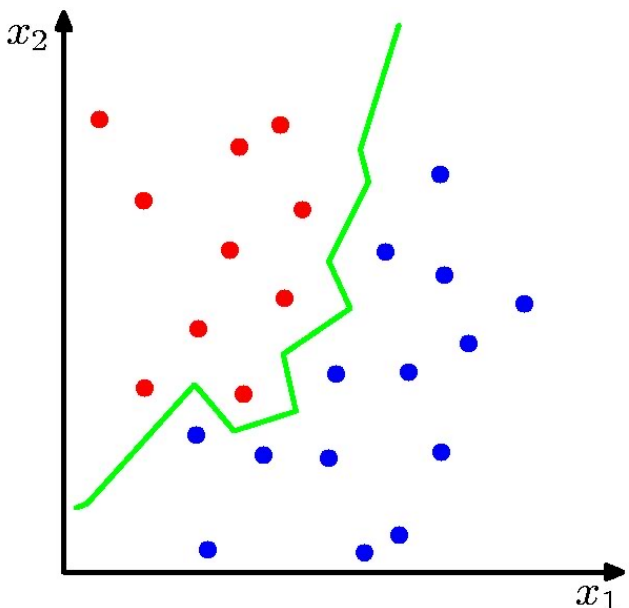
- Strengths
  - Currently among the best performers for a number of classification tasks ranging from text to genomic data.
    - Outside of some big data settings where deep learning is clearly better.
  - **Can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational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

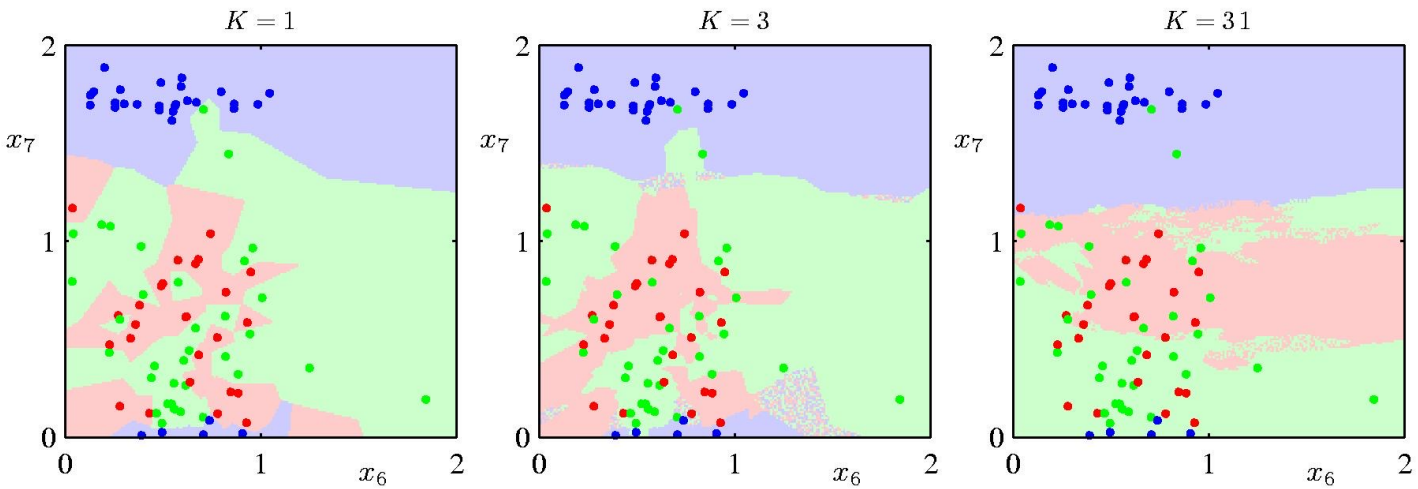


K = 3



K = 1

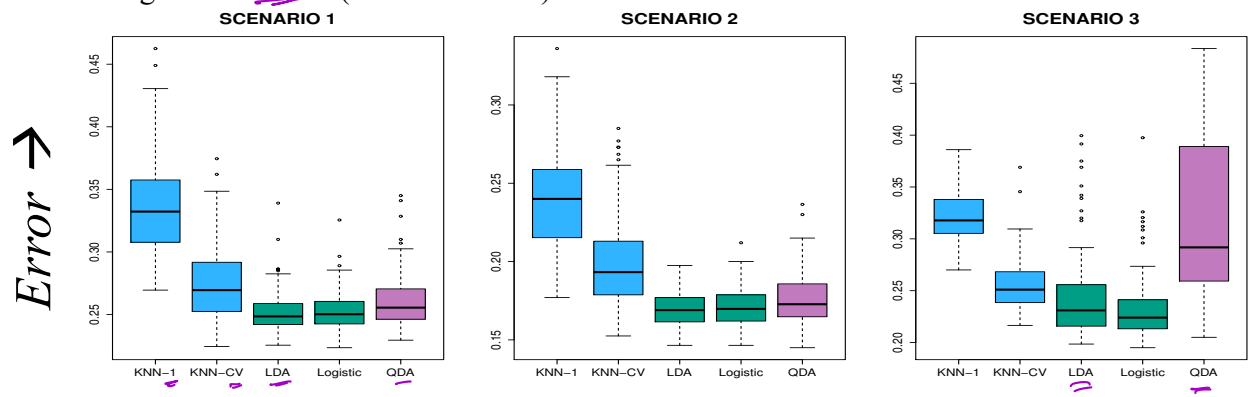
K-Nearest-Neighbours for Classification (3)



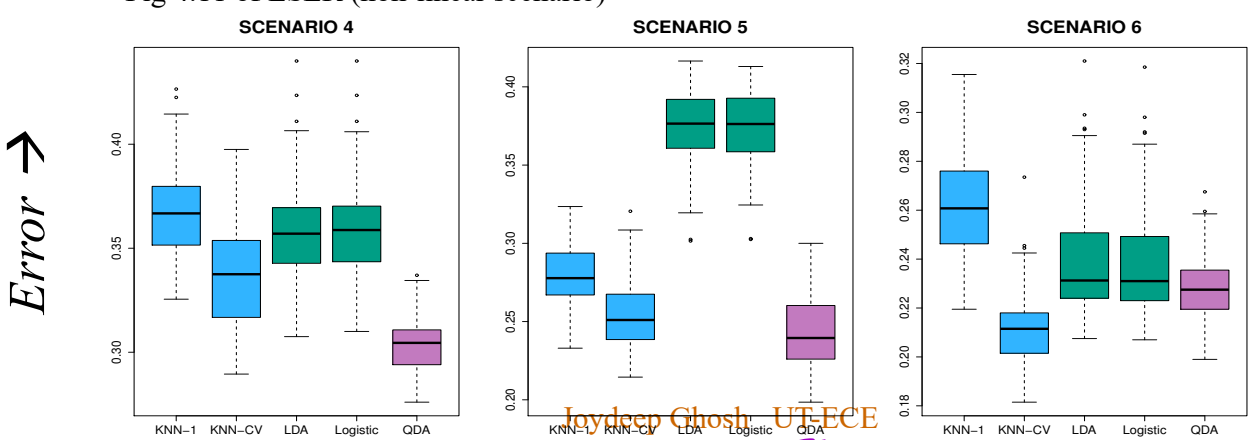
- K acts as a smother
- For $N \to \infty$, the error rate of the 1-nearest-neighbour classifier is never more than twice the optimal error (obtained from the true conditional class distributions).

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)
    - Deep Learning if lots of training data (typically > 100,000); feature representation needs to be determined,…
    - SVMs fairly robust in (reasonably) high-D
      - Also suitable if good kernel is known
- Still lacking? try ensemble approaches


SUPPORT VECTOR MACHINES
END DUALITY GAP
Map Reduce
Map Reuse
Map Recycle
BAYESIANS AGAINST DISCRIMINATION
FREE VARIABLES!
BAN GENETIC ALGORITHMS


BRING
THE
CRAZY

Extras

Linear Regression of an Indicator Matrix

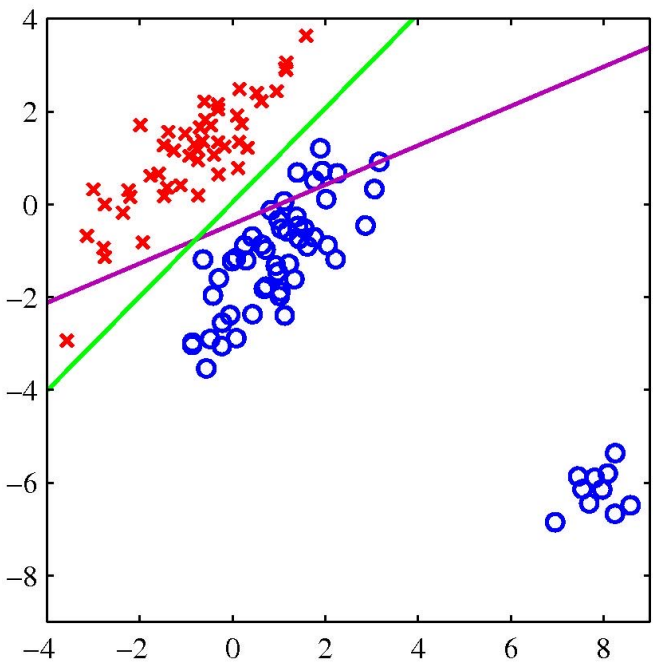
- 0/1 output encoding (aka 1 of C).
  - Decision: Pick class corresp to highest output

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

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

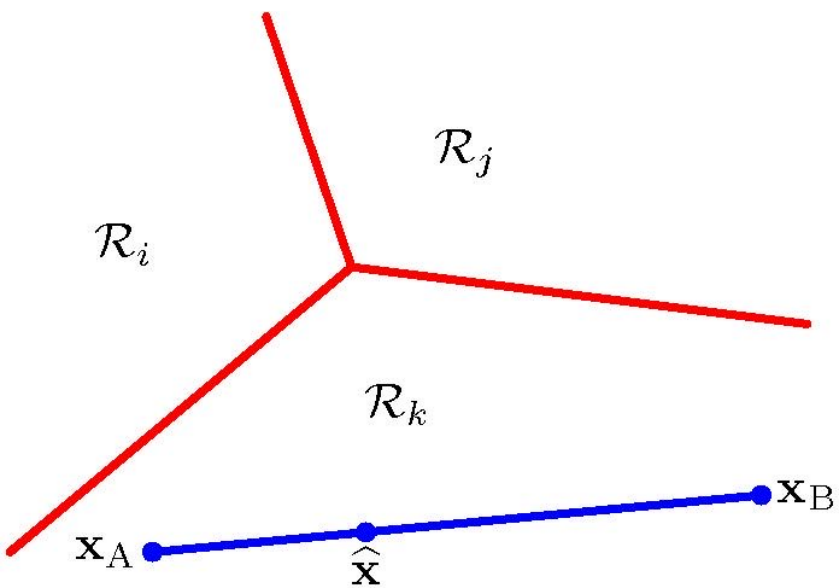
-ve: sensitive to outliers

(green: logistic vs. magenta, linear)



Geometry: Multi-class

- $y_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} + w_{k0}$
  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; $t(.) = +/- 1$

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

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

So

$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\eta\,\mathbf{x}(k)$ if labels don't match

else no weight update.

Perceptron Convergence Theorem

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

Also, Cycling theorem.

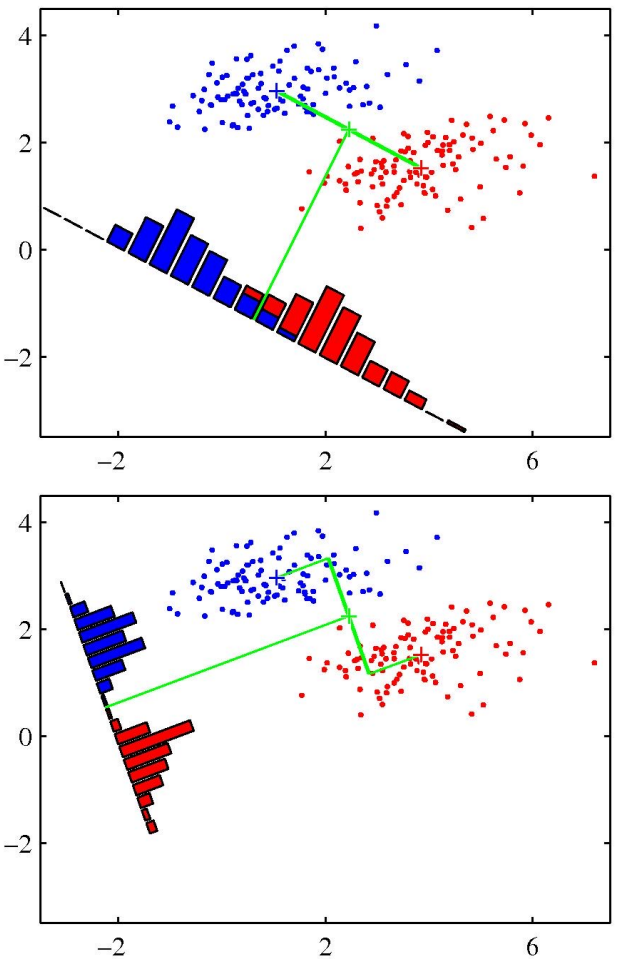
- Problems with Perceptron:
  - Can solve only linearly separable problems.
  - May need large number of steps to converge.
- But good for online learning, in non-stationary environments!
  - Improvements such as adaptive learning rate.

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

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

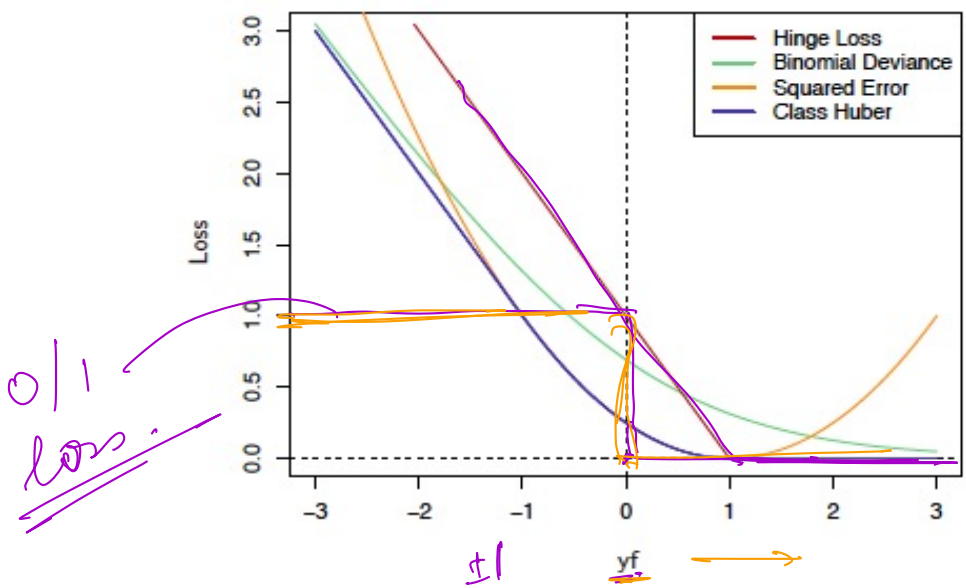


FIGURE 12.4. *The support vector loss function (hinge loss), compared to the negative log-likelihood loss (binomial deviance) for logistic regression, squared-error loss, and a "Huberized" version of the squared hinge loss. All are shown as a function of yf rather than f, because of the symmetry between 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

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

- Since $$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V},$$ Bayes' theorem gives

$$p(\mathcal{C}_k) = N_k/N$$

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