SUNY Korea
The State University of New York
한국뉴욕주립대학교

Stony Brook University

Instructor: Sael Lee

CS549 Spring – Computational Biology

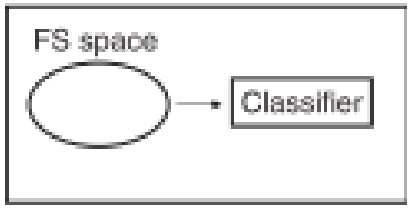
LECTURE 12-13: FEATURE SELECTION

Ref.

1. C. M. Bishop “Pattern Recognition and Machine Learning” 2nd ed. & provided sides

TYPES OF FEATURE SELECTION METHOD

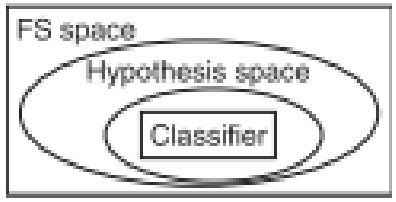
Filtering Methods



relevance of features is evaluated by looking only at the intrinsic properties of the data

* Often **feature relevance score** is used to evaluate each feature (gene)

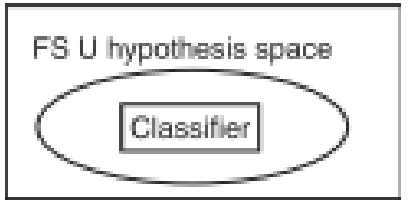
Wrapper Methods



model hypothesis search is embed within the feature subset search

-> **various subsets of features** are generated and evaluated

Embedded Method



optimal feature subset search is built into the classifier construction

-> a search in the **combined space of feature subsets and hypotheses**

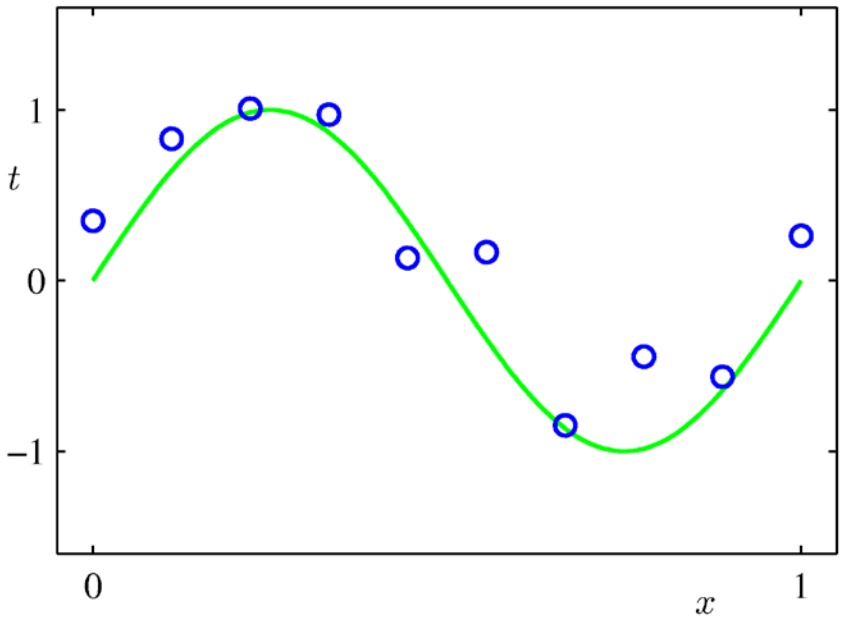
Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, *23*(19), 2507–17.

Chapter 3 of PRML

FEATURE SELECTION WITH LASSO REGRESSION MODEL

LINEAR BASIS FUNCTION MODELS (1)

- Example: Polynomial Curve Fitting



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

LINEAR BASIS FUNCTION MODELS (2)

- Generally

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

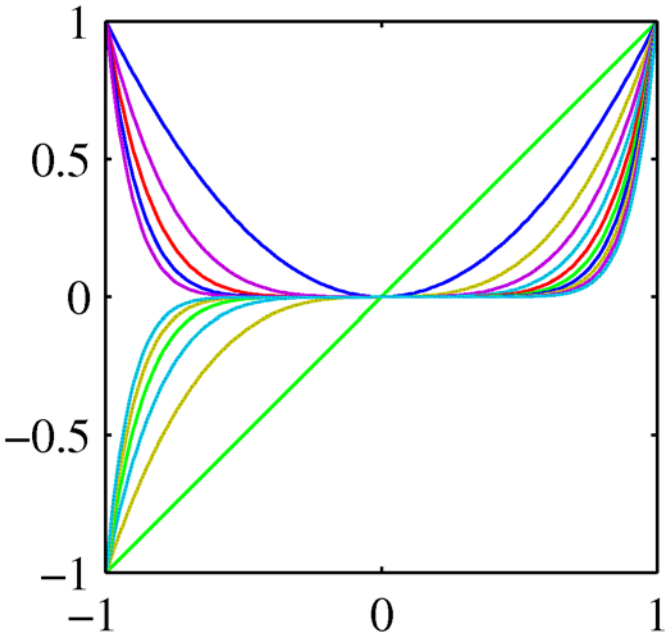
- where $\phi_j(\mathrm{x})$ are known as *basis functions*.
- Typically, $\phi_0(\mathrm{x}) = 1$, so that w_0 acts as a bias.
- In the simplest case, we use linear basis functions : $\phi_d(x) = x_d$.

LINEAR BASIS FUNCTION MODELS (3)

- Polynomial basis functions:

$$\phi_j(x) = x^j.$$

- These are global; a small change in x affect all basis functions.

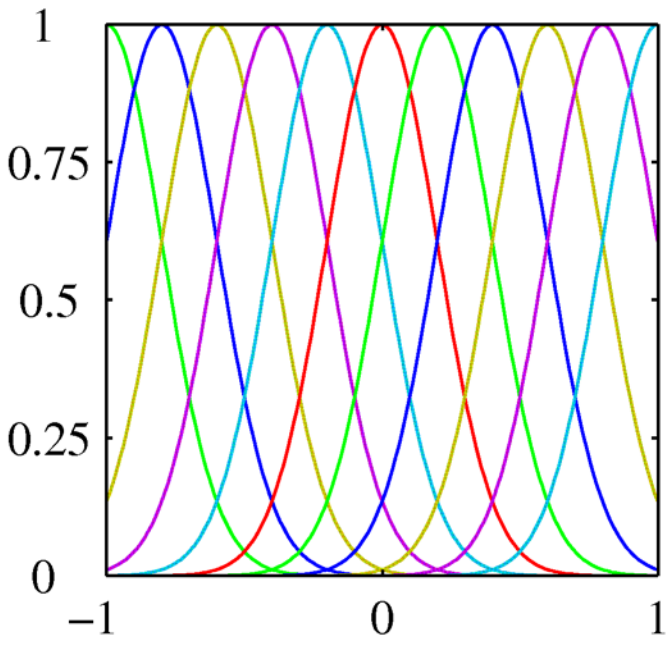


LINEAR BASIS FUNCTION MODELS (4)

- Gaussian basis functions:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

- These are local;
  - a small change in x only affect nearby basis functions.
  - μ_j and s control location and sc ale (width).

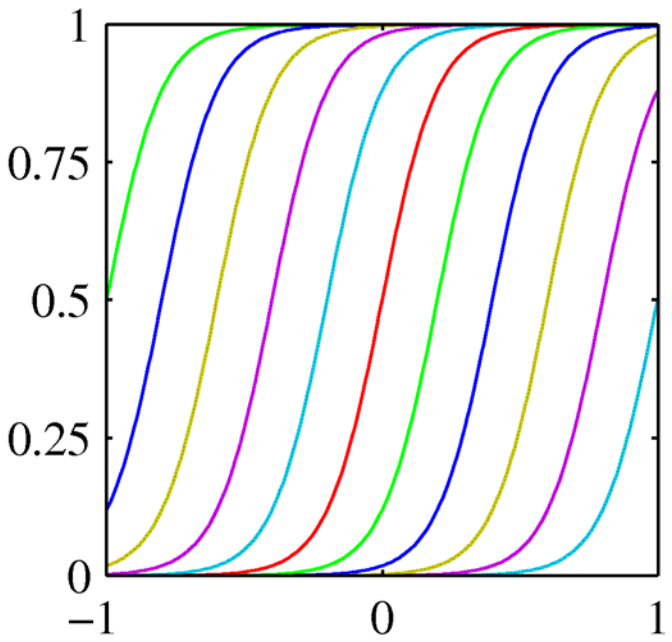


LINEAR BASIS FUNCTION MODELS (5)

- Sigmoidal basis functions:

where $$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$



- Also these are local;
  - a small change in x only affect nearby basis functions.
  - μ_j and s control location and scale (width).

MAXIMUM LIKELIHOOD AND LEAST SQUARES (1)

- Assume observations from a deterministic function with added Gaussian noise:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \qquad \text{where} \qquad p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1})$$

- which is the same as saying,

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

- Given observed inputs, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and targets, $\mathbf{t} = [t_1, \ldots, t_N]^{\mathrm{T}}$, we obtain the likelihood function

likelihood function

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}).$$

MAXIMUM LIKELIHOOD AND LEAST SQUARES (2)

$$N\left(t_n \middle| \boldsymbol{w}^T \boldsymbol{\phi}(\mathbf{x_n}), \beta^{-1}\right) = \left(\frac{\beta}{2\pi}\right)^{\frac{1}{2}} \exp\left(-\frac{\beta}{2}\left(t_n - \boldsymbol{w}^T \boldsymbol{\phi}(\mathbf{x_n})\right)^2\right)$$

- Log likelihood:

$$\ln p(\mathbf{t}|\mathbf{w}, \beta) = \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1})$$

$$= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w})$$

where

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

is the sum-of-squares error.

Relationship of log likelihood and sum-of-squares error in univariate Gaussian noise model.

MAXIMUM LIKELIHOOD AND LEAST SQUARES (3)

- Computing the gradient and setting it to zero yields

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} = \mathbf{0}.$$

- Solving for w, we get

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

The Moore-Penrose pseudo-inverse, $\boldsymbol{\Phi}^{\dagger}$.

- where

Design matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}.$$

MAXIMUM LIKELIHOOD AND LEAST SQUARES (4)

- Maximizing with respect to the bias, w_0, alone, we see that

$$w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \overline{\phi_j}$$

$$= \frac{1}{N}\sum_{n=1}^{N} t_n - \sum_{j=1}^{M-1} w_j \frac{1}{N}\sum_{n=1}^{N} \phi_j(\mathbf{x}_n).$$

- We can also maximize with respect to β, giving

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

Residual variance of the target value around the regression function

REGULARIZED LEAST SQUARES (1)

- Consider the error function:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization term

- With the sum-of-squares error (SSE) function and a quadratic regularizer, we get

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

- which is minimized by

$$\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}.$$

λ is called the regularization coefficient.

REGULARIZED LEAST SQUARES (2)

- With a more general regularizer, we have

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

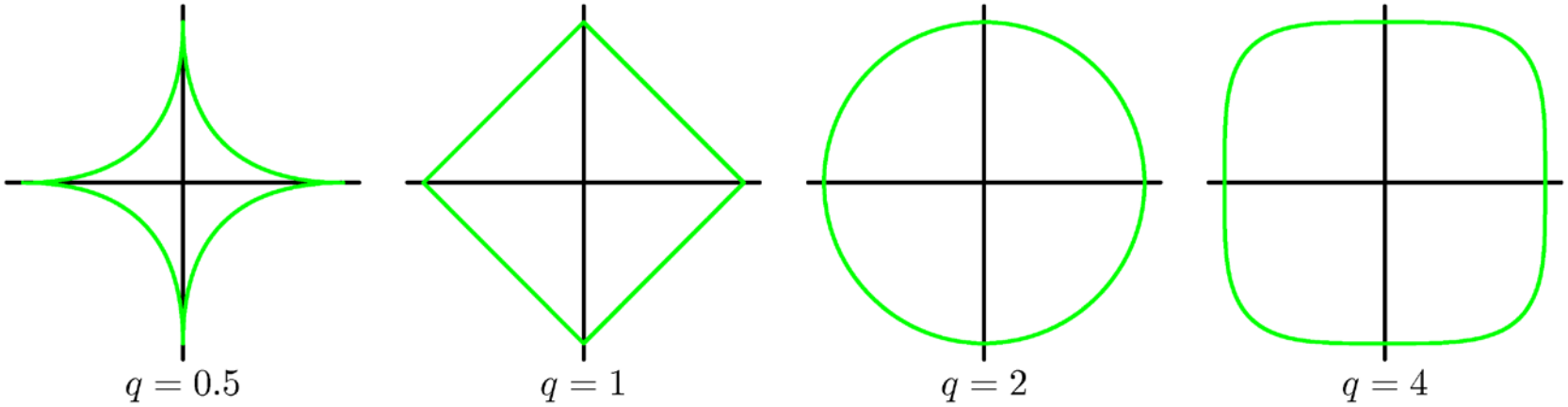


Fig: Contours of the regularization terms

USING LASSO FOR FEATURE SELECTION

Lasso tends to generate sparser solutions

- If λ is sufficiently large, some of the coefficients w_j are driven to zero, leading to a sparse model in which the corresponding basis function pays no role.

Minimizing

general regularizer

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

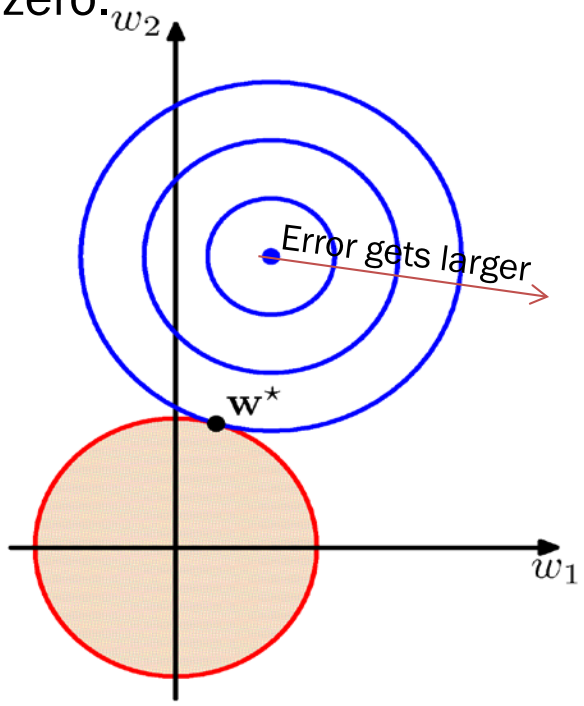
is equivalent to minimizing the unregularized SSE subjected to constraint

Lagrangian Multiplier

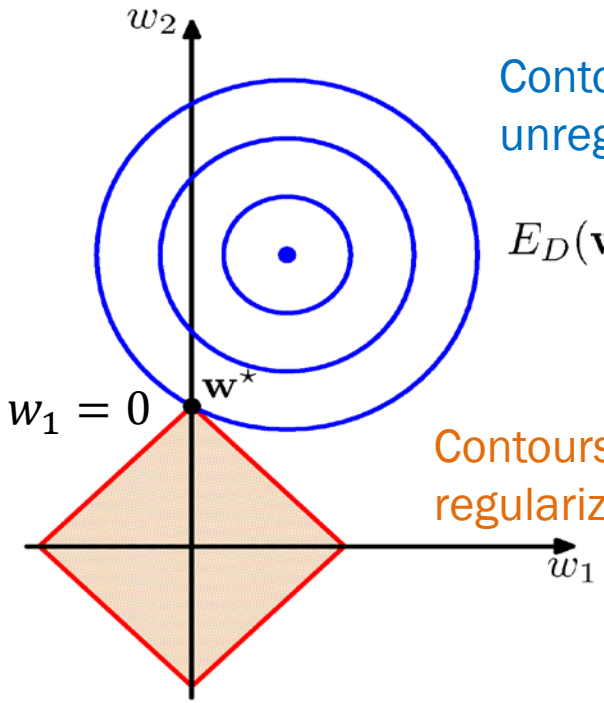
$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 \quad \text{Subjected to} \quad \sum_{j=1}^{M}|w_j|^q \leq \eta$$

REGULARIZED LEAST SQUARES (3)

Figure shows the minimum of the error function, subjected to constraint. As λ is increased, so an increasing number of parameters are driven to zero.



Quadratic



Lasso

Contours of unregularized SSE

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

Contours of the regularization terms $\sum_{j=1}^{M}|w_j|^q \leq \eta$

Lasso give sparse solution in which $w^* = 0$.

Q: So, how do we find the right λ?

LIMITATIONS OF LASSO BASED FEATURE SELECTION

- Linear feature space : inadequate to capture non-linear dependencies from features to output

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

SUPPORT VECTOR MACHINES

REVIEW

linear regression

$$lh_{\mathrm{w}}(\mathrm{x}) = (\mathrm{w}^T\mathrm{x})$$

Logistic regression

Estimated prob. that y=1 on input x

$$h_{\mathrm{w}}(\mathrm{x}) = Logistic(\mathrm{w}^T\mathrm{x}) = \frac{1}{1+e^{-(\mathrm{w}^T\mathrm{x})}}$$

LEARNING THE WEIGHTS

$$w^* = argmin_w Loss(h_w)$$

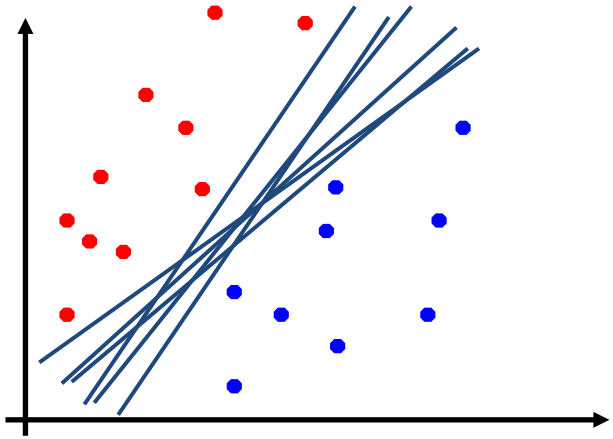
linear regression:

$$Loss(h_w) = \sum_{j=1}^{N} (y_i - (\mathrm{w}^T \mathrm{x}))^2$$

logistic regression:

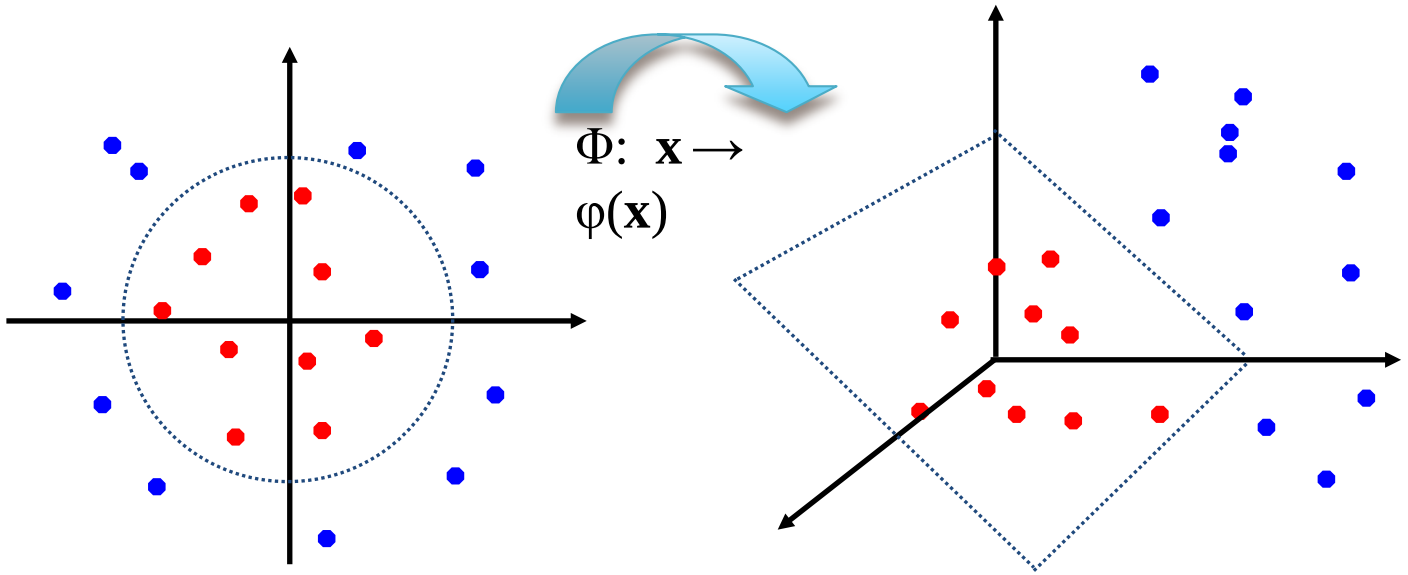
$$Loss(h_w) = \sum_{j=1}^{N} -y_i(\log(h_w(x))) - (y_i - 1)(\log(1 - h_w(x)))$$

- y is classification label in logistics regression (0 or 1)
- y is scalar values in linear regression



KERNELS

- The original feature space can always be mapped to some higher-dimensional feature space (even infinite) where the training set is separable



KERNEL

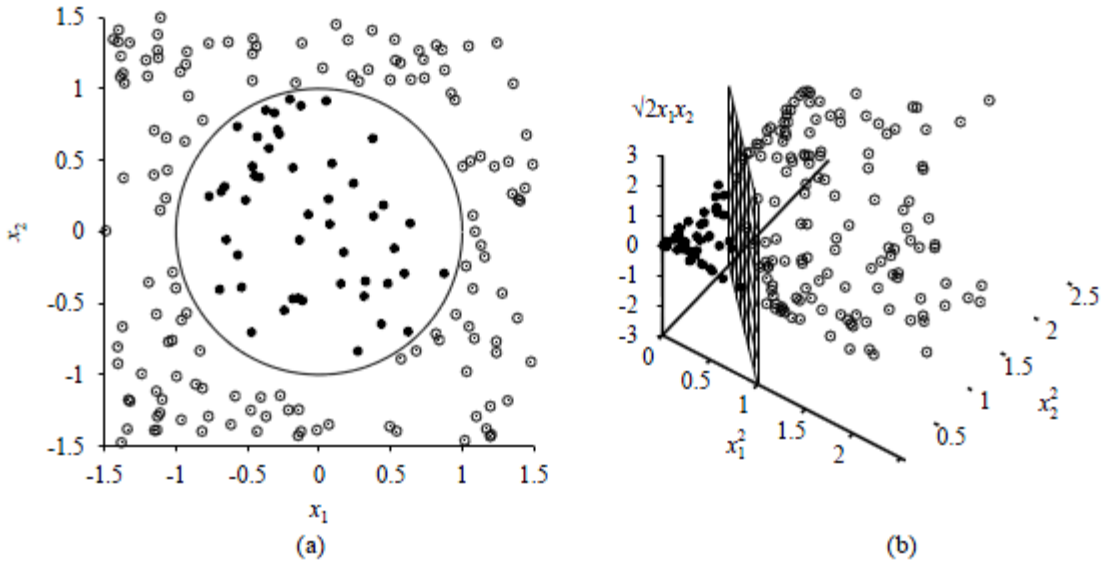


Figure 18.31 **FILES:** . (a) A two-dimensional training set with positive examples as black circles and negative examples as white circles. The true decision boundary, $x_1^2 + x_2^2 \leq 1$, is also shown. (b) The same data after mapping into a three-dimensional input space $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$. The circular decision boundary in (a) becomes a linear decision boundary in three dimensions. Figure 18.29(b) gives a closeup of the separator in (b).

KERNELS

- The linear classifier relies on an inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

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

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

- Kernel function should measure some similarity between data
- kernel must be positive semi-definite

- You should scale the features to have same scale!!

- Most widely used is **linear kernels** and **Gaussian kernels**

GAUSSIAN KERNELS

$$k(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}{2\sigma^2}\right)$$

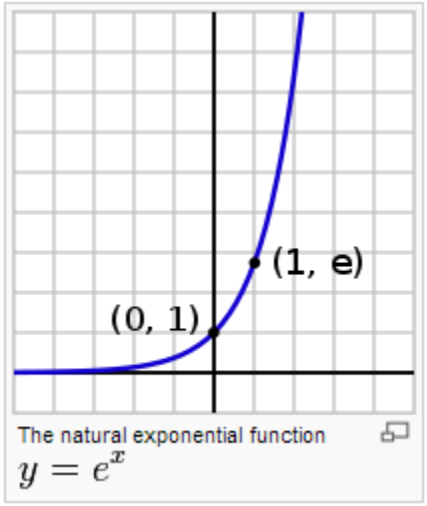
If x_i *and* x_j is similar:

$$k(x_i, x_j) \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1$$

If x_i *and* x_j is different:

$$k(x_i, x_j) \approx \exp\left(-\frac{(large\ number)^2}{2\sigma^2}\right) \approx 0$$

If you use Gaussian kernel,
You will need to pick σ



SUPPORT VECTOR MACHINES

- SVMs constructs a maximum margin separator
- SVMs create a linear separating hyperplane
  - But have ability to embed that in to higher-dimensional space (via **Kernel trick**)
- SVM are a nonparametric method
  - Retain training examples an potentially need to store all or part of the data
  - Some example are more important then others (support vectors)

SVM TERMS

- Distance from example $\mathbf{x}_i$ to the separator is

$$r = {(\mathrm{w}^T\mathrm{x}+\mathrm{b})}/{||\mathrm{w}||}$$

- Examples closest to the hyperplane are *support vectors*.
- *Margin* ρ of the separator is the distance between support vectors

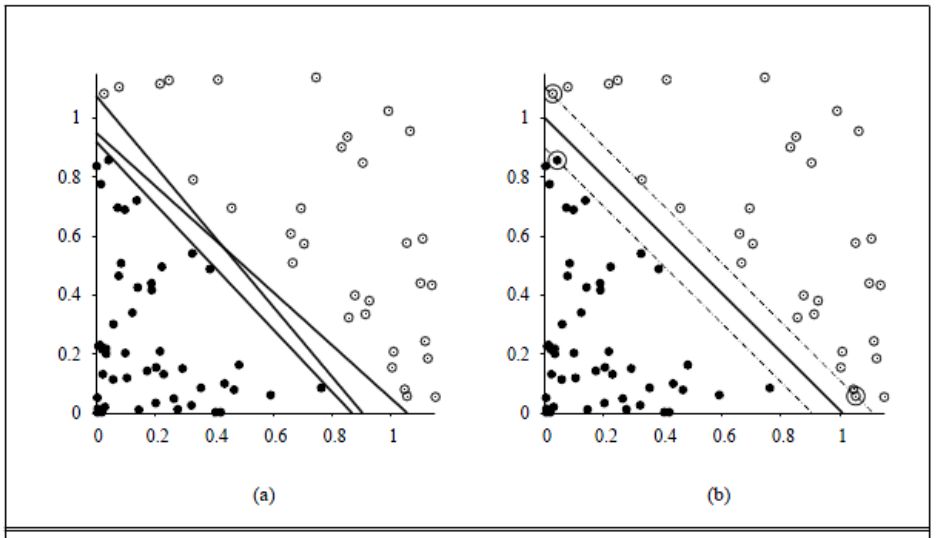
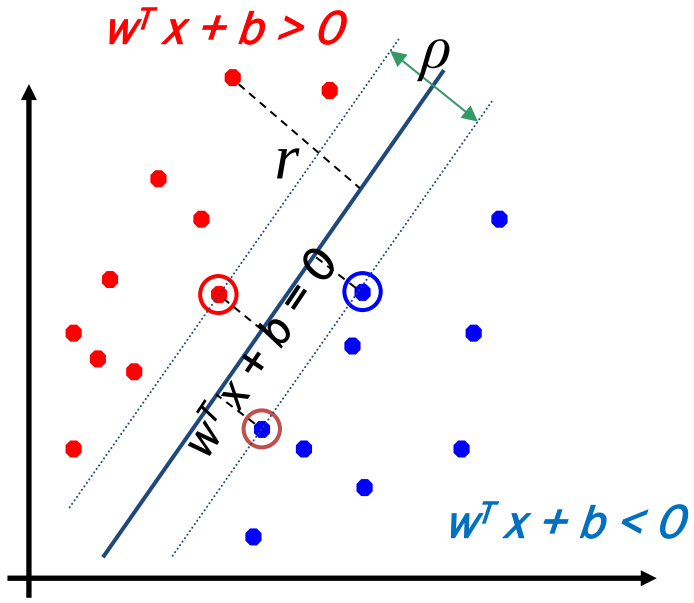


Figure 18.30 **FILES:** . Support vector machine classification: (a) Two classes of points (black and white circles) and three candidate linear separators. (b) The maximum margin separator (heavy line), is at the midpoint of the **margin** (area between dashed lines). The **support vectors** (points with large circles) are the examples closest to the separator.



MARGINS

Instead of minimizing expected empirical loss in the training data,
SVM attempts to minimize expected generalization loss.

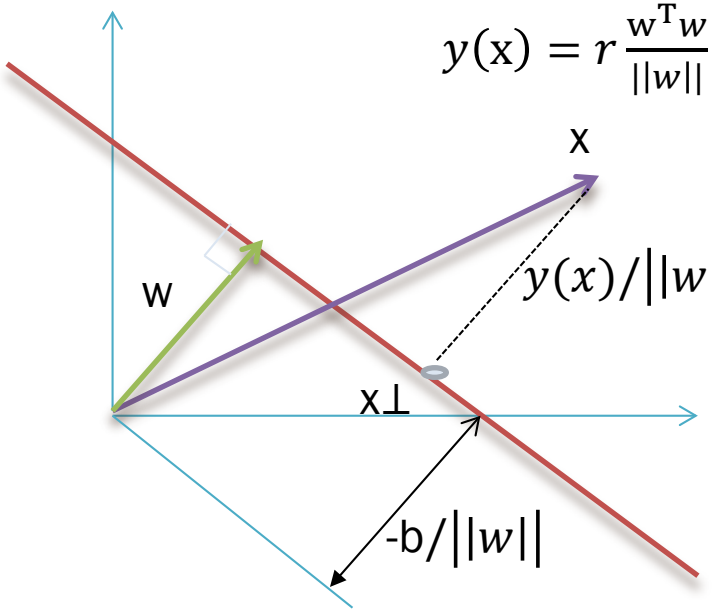
$y(\mathrm{x}) = \mathrm{w}^{\mathrm{T}}\mathrm{x} + b$ where w is weight vector and b is bias

$\mathrm{x} = \mathrm{x}_{\perp} + r\frac{w}{||w||}$ (multiply w^{T} and add b)

$\mathrm{w}^{\mathrm{T}}\mathrm{x} + b = \mathrm{w}^{\mathrm{T}}(\mathrm{x}_{\perp} + r\frac{w}{||w||}) + b \qquad (y(\mathrm{x}) = \mathrm{w}^{\mathrm{T}}\mathrm{x} + b)$

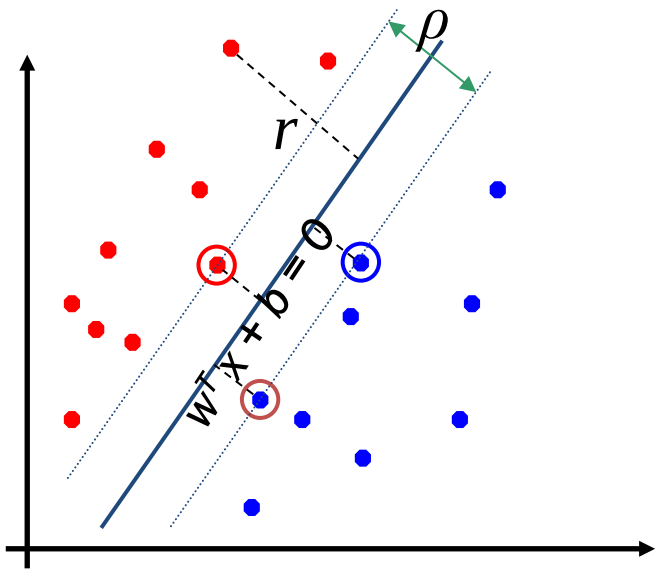
$y(\mathrm{x}) = \mathrm{w}^{\mathrm{T}}\mathrm{x}_{\perp} + r\frac{\mathrm{w}^{\mathrm{T}}w}{||w||} + b \quad (y(\mathrm{x}_{\perp}) = \mathrm{w}^{\mathrm{T}}\mathrm{x}_{\perp} + b = 0)$

$y(\mathrm{x}) = r\frac{\mathrm{w}^{\mathrm{T}}w}{||w||} = \mathrm{I}$



or $r = {}^{(\mathrm{w}^{T}\mathrm{x}+\mathrm{b})}/_{||\mathrm{w}||}$

MAXIMUM MARGINS



Solving this is non-trivial and will not be discussed in class

$\phi(\mathrm{x_n})$ in the feature space

$$r = {(\mathrm{w}^T \mathrm{x} + \mathrm{b})}/{||\mathrm{w}||}$$

$$argmax_{w,b}\ \{\frac{1}{||w||}\ min_n[t_n(\mathrm{w}^{\mathrm{T}}\mathrm{x_n} + b)]\}$$

$$argmin_{w,b}\ \frac{1}{2}||w||^2$$

$$\mathrm{w} = \sum_{n=1}^{N} a_n t_n\, \phi(\mathrm{x_n})$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

SOFT MARGINS

Idea: Allow data point to be in the wrong side of the margin boundary, but with a penalty that increases with the distance from that boundary.

Penalty for each data point : **slack variable ξ**
$\xi_n = 0$ if point is on the right side
$\xi_n = |t_n - y(\mathrm{x}_n)|$ if point is on the wrong side
Such that
$t_n y(\mathrm{x}_n) \geq 1 - \xi_n$ for n = 1, …, N and $\xi_n \geq 0$

- $0 < \xi_n \leq 1$ for points inside the margin
- $\xi_n = 1$ for points on the margin
- $\xi_n > 1$ for points that are on the wrong side

Goal now is to maximize the margin while softly penalizing points that lie on the wrong side of the margin boundary

$$argmin_{w,b}\ C \sum_{n}^{N} \xi_n \quad + \frac{1}{2}||w||^2$$

OPTIMIZATION ON SOFT MARGINS

$$argmin_{w,b}\ C\sum_n^N \xi_n \quad + \frac{1}{2}||w||^2$$

subjected to $t_n y(\mathrm{x}_n) \geq 1 - \xi_n$ for n = 1, …, N and $\xi_n \geq 0$

ξ_n: slack variable for training data x_n

Complex calculations
Lagrangian
Etc.

$$\mathrm{w} = \sum_{n=1}^{N} a_n t_n\, \phi(\mathrm{x_n})$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

a_n is Lagrangian multiplier related to w_n

a_n = C - μ_n

μ_n is Lagrangian multiplier related to ξ_n

$$\mathrm{b} = \frac{1}{N_M}\sum_{n\in M}(t_n - \sum_{n\in S}(a_m t_m\, k(x_n x_m))$$

PREDICTION USING KERNELS

$$y(x) = \mathrm{w}^T \phi(\mathrm{x_n}) + b$$

$$\mathrm{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathrm{x_n})$$

a_n is a Lagrangian multiplier

$$y(x) = \sum_{n=1}^{N} a_n t_n k(\mathrm{x}, \mathrm{x_n}) + b$$

Any data point $a_n = 0$ will not appear in the sum

New Data

Training data

Training data target (-1,1)

METHODS OF USING SVM OF FEATURE SELECTION

1. Linear SVM:
   Evaluation of learned **weights** in Linear SVM

$$y(x) = \mathrm{w}^T \phi(\mathrm{x_n}) + b \qquad \mathrm{w} = \sum_{n=1}^{N} a_n t_n \, \phi(\mathrm{x_n})$$

2. Recursive Feature Elimination:
   Iterative evaluation of significance of features in SVM classification:

3. Feature Vector Machines
   Variation of Lasso like SVM that generate kernels on features not samples.