EECS 545: Machine Learning

Lecture 7. Kernel methods

Honglak Lee

1/31/2011

Outline

- Recap: Kernel methods
- Support Vector Machines
- Next lecture: Gaussian Processes

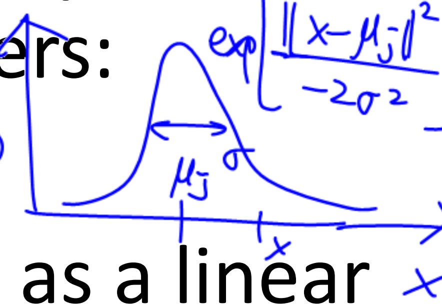
Kernel regression

Radial Basis Functions

- Basis functions can be chosen that depend only on distance from selected centers:

$$\phi_j(\mathbf{x}) = h(\underbrace{||\mathbf{x} - \mu_j||}_{\text{distance}})$$

$\phi_j(x)$ $\exp\left[\frac{||x-\mu_j||^2}{-2\sigma^2}\right]$



- A function *f*(x) can be approximated as a linear combination of the basis functions

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n h(||\mathbf{x} - \mu_n||)$$

- With a basis function at each training data point, the approximation is exact on the training data.

Kernel Regression

- Using radial basis functions around the training data points, predict a value y(x) as the average of target values t_n, weighted by similarities k(x,x$_n$):

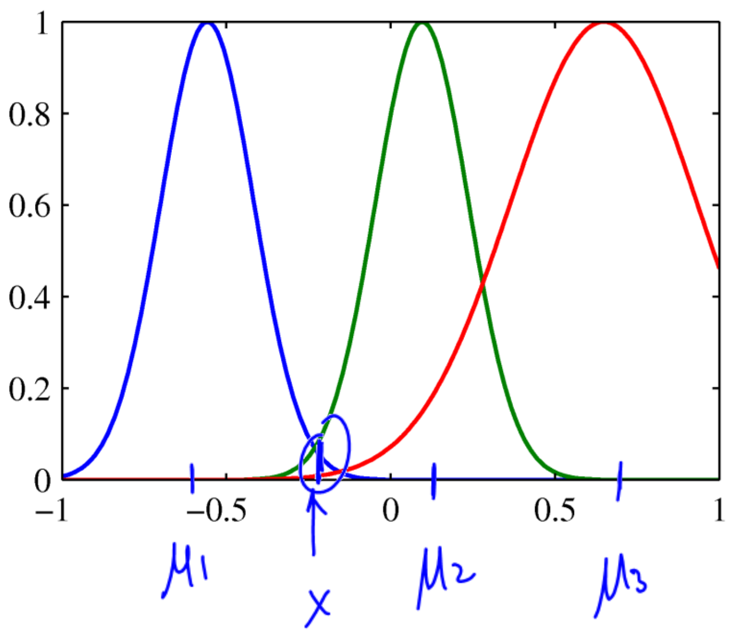
$$y(\mathbf{x}) = \sum_{n=1}^{N} \underline{k(\mathbf{x}, \mathbf{x}_n)}\, t_n$$

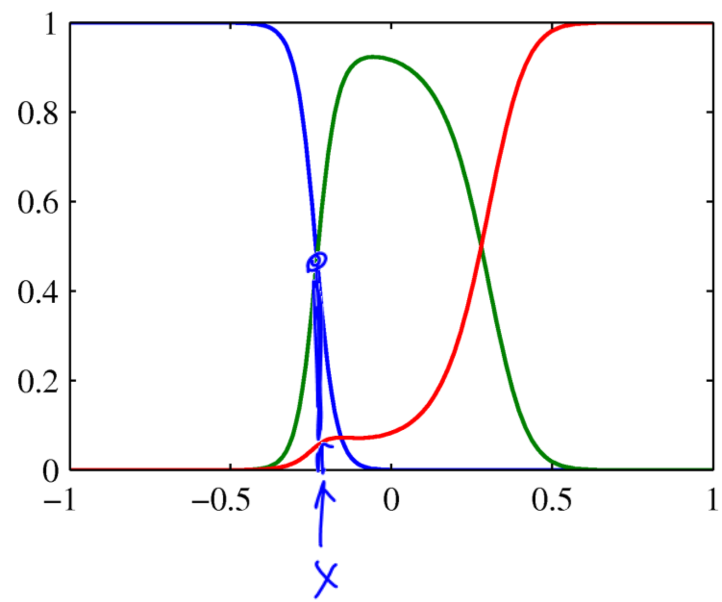
$$\propto \exp\left[-\frac{\|x - x_n\|^2}{2\sigma^2} \right]$$

Kernel Normalization

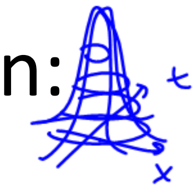
- The weighted average approach assumes

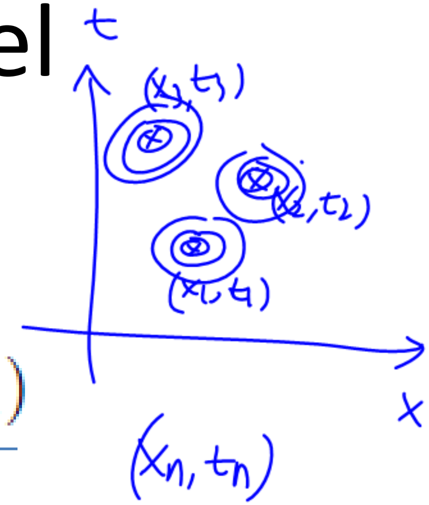
$$\sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) = 1 \quad , \ \forall x$$





Narayada-Watson model





- From kernel density estimation:

$$p(\mathbf{x}, t) = \frac{1}{N}\sum_{n=1}^{N} f(\mathbf{x}-\mathbf{x}_n, t-t_n)$$

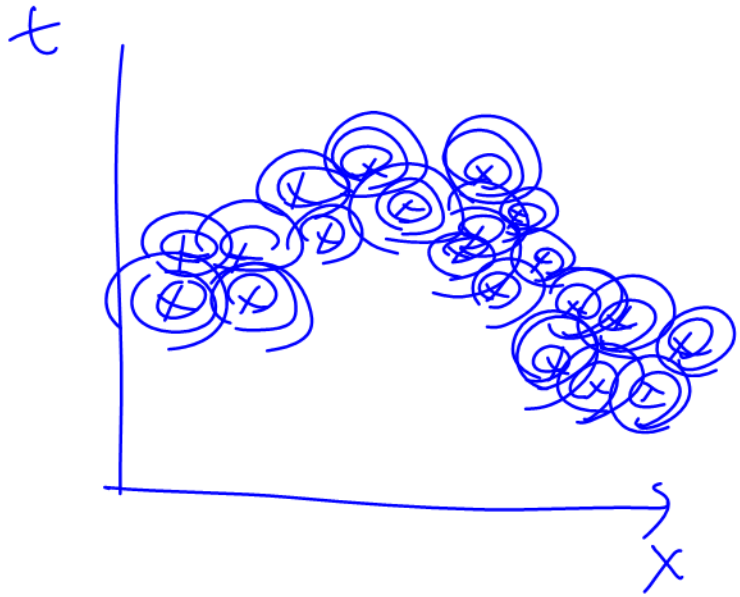
data feature

output

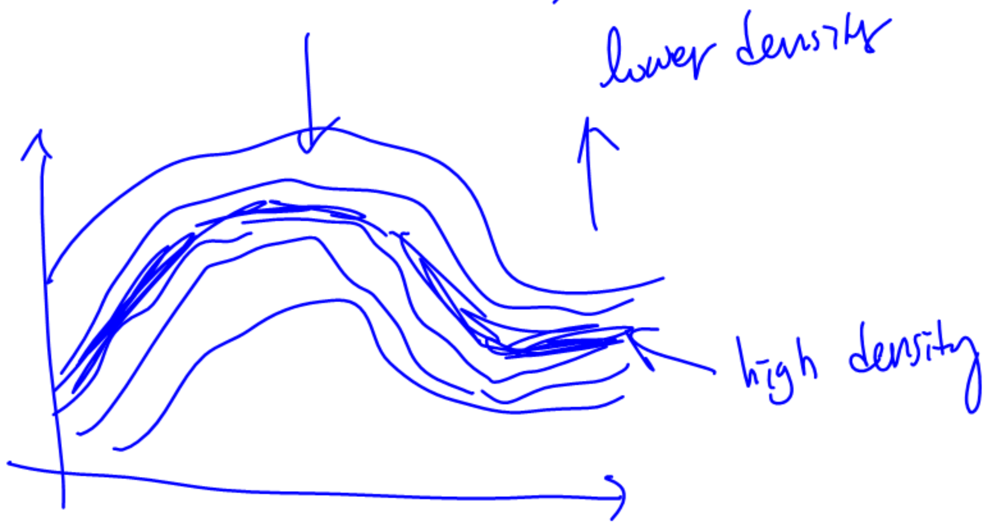
Joint density function centered at x_n, t_n

  - where f(x,t) is the component density function and there is one such component centred on each data point

- We now find an expression for the regression function y(x), corresponding to the conditional average of the target variable conditioned on the input variable



$$f(x) = \sum_n f(x - x_n,\ t - t_n)$$



Narayada-Watson model

$$p(x,t)$$

$$\mathbb{E}[t|x]$$

$$p(t|x) = \frac{p(t,x)}{\int p(t,x)\,dt}$$

$$\begin{aligned} y(\mathbf{x}) &= \mathbb{E}[t|\mathbf{x}] = \int_{-\infty}^{\infty} tp(t|\mathbf{x})\,\mathrm{d}t \\ &= \frac{\int tp(\mathbf{x},t)\,\mathrm{d}t}{\int p(\mathbf{x},t)\,\mathrm{d}t} \\ &= \frac{\sum_n \int tf(\mathbf{x}-\mathbf{x}_n, t-t_n)\,\mathrm{d}t}{\sum_m \int f(\mathbf{x}-\mathbf{x}_m, t-t_m)\,\mathrm{d}t}. \end{aligned} \tag{6.43}$$

$$t = (t - t_n) + t_n$$

$$= g(x - x_m)$$

We now assume for simplicity that the component density functions have zero mean so that

$$\int_{-\infty}^{\infty} f(\mathbf{x},t)t\,\mathrm{d}t = 0 \tag{6.44}$$

for all values of $\mathbf{x}$. Using a simple change of variable, we then obtain

$$\sum_n k(x, x_n) = 1$$

$$\begin{aligned} y(\mathbf{x}) &= \frac{\sum_n g(\mathbf{x}-\mathbf{x}_n)t_n}{\sum_m g(\mathbf{x}-\mathbf{x}_m)} = k(x, x_n) \\ &= \sum_n k(\mathbf{x},\mathbf{x}_n)t_n \end{aligned} \tag{6.45}$$

Narayada-Watson model

- Prediction function:

$$y(\mathbf{x}) = \frac{\sum_{n} g(\mathbf{x}-\mathbf{x}_n)t_n}{\sum_{m} g(\mathbf{x}-\mathbf{x}_m)} = \sum_{n} k(\mathbf{x},\mathbf{x}_n)t_n$$

  – where

$$k(\mathbf{x},\mathbf{x}_n) = \frac{g(\mathbf{x}-\mathbf{x}_n)}{\sum_{m} g(\mathbf{x}-\mathbf{x}_m)}$$

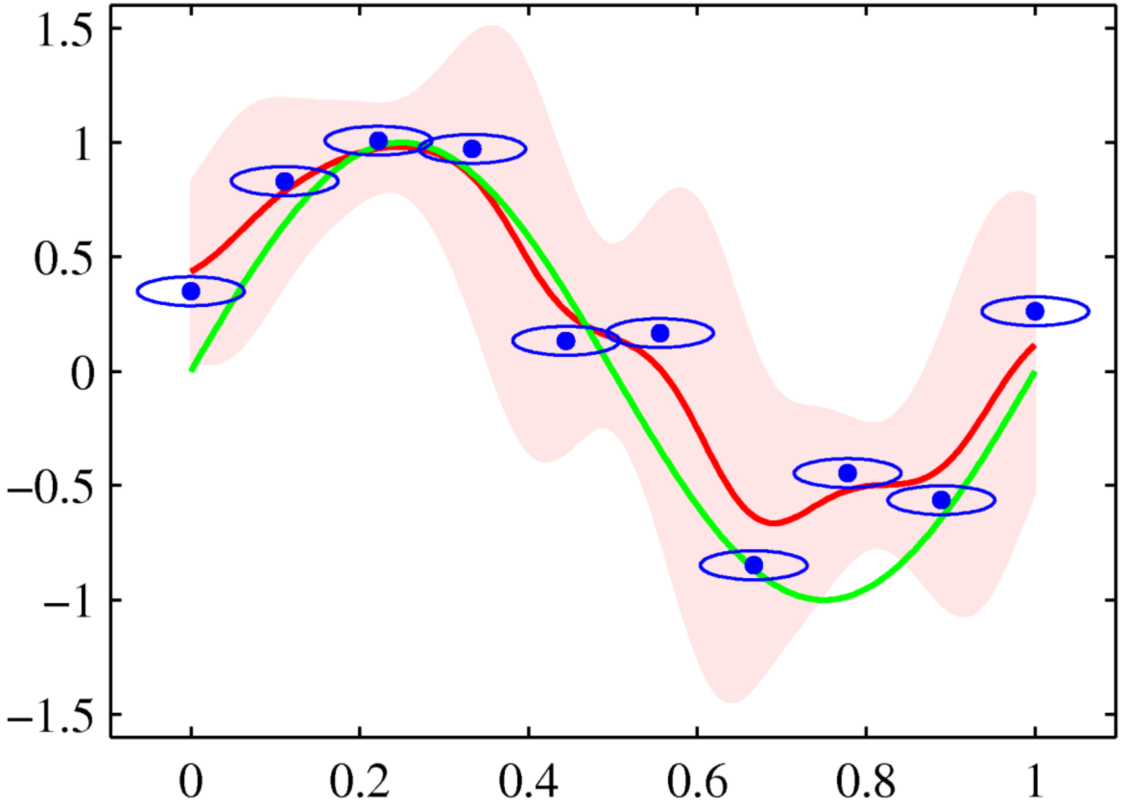
$$g(\mathbf{x}) = \int_{-\infty}^{\infty} f(\mathbf{x},t)\,\mathrm{d}t.$$

Narayada-Watson model

- This model is also known as kernel regression.
- For a localized kernel function, it has the property of giving more weight to data points that a close to x

Kernel Regression Example

- On the familiar sinusoidal data set:



Support Vector Machines

Classification

- Consider a two-class classification problem:
  - Positive: $t = +1$
  - Negative: $t = -1$
- Train a linear model over the feature vector:

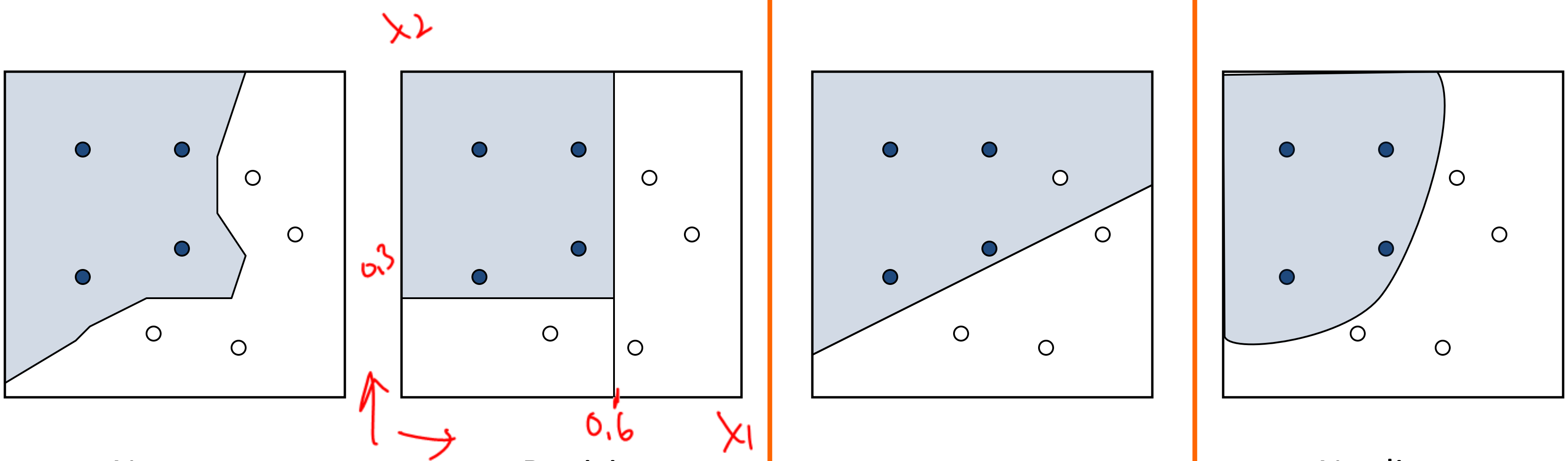
$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

- Train with input vectors x = $\{x_1, \ldots x_N\}$
  - and corresponding target values $\mathbf{t} = \{t_1, \ldots t_N\}$.
  - $y(\mathbf{x})>0 \Rightarrow t = +1$ and $y(\mathbf{x})<0 \Rightarrow t = -1$
  - That is: $t_n \boxed{y(\mathbf{x}_n)} > 0$.

label ← prediction for x_n

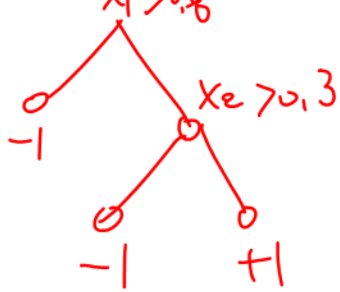
Discriminant Function

- It can be arbitrary functions of *x*, such as:



Nearest Neighbor

Decision Tree



Linear Functions

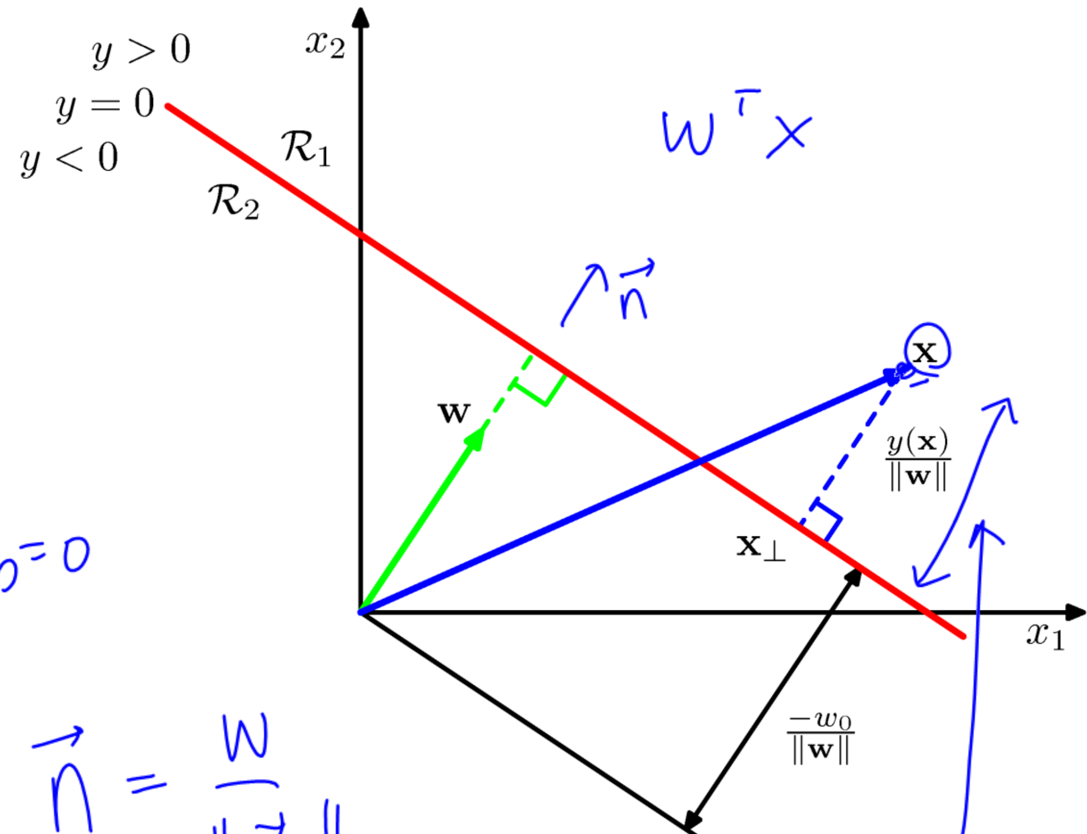
$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

Nonlinear Functions

Distance from Decision Surface

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

- w determines direction.
- *b* determines offset.



$b = 0$

$$\vec{n} = \frac{w}{\|\vec{w}\|}$$

$$\frac{w^T x}{\|w\|} + \frac{b}{\|w\|} = \frac{y(x)}{\|w\|}$$

Linear Discriminant Function

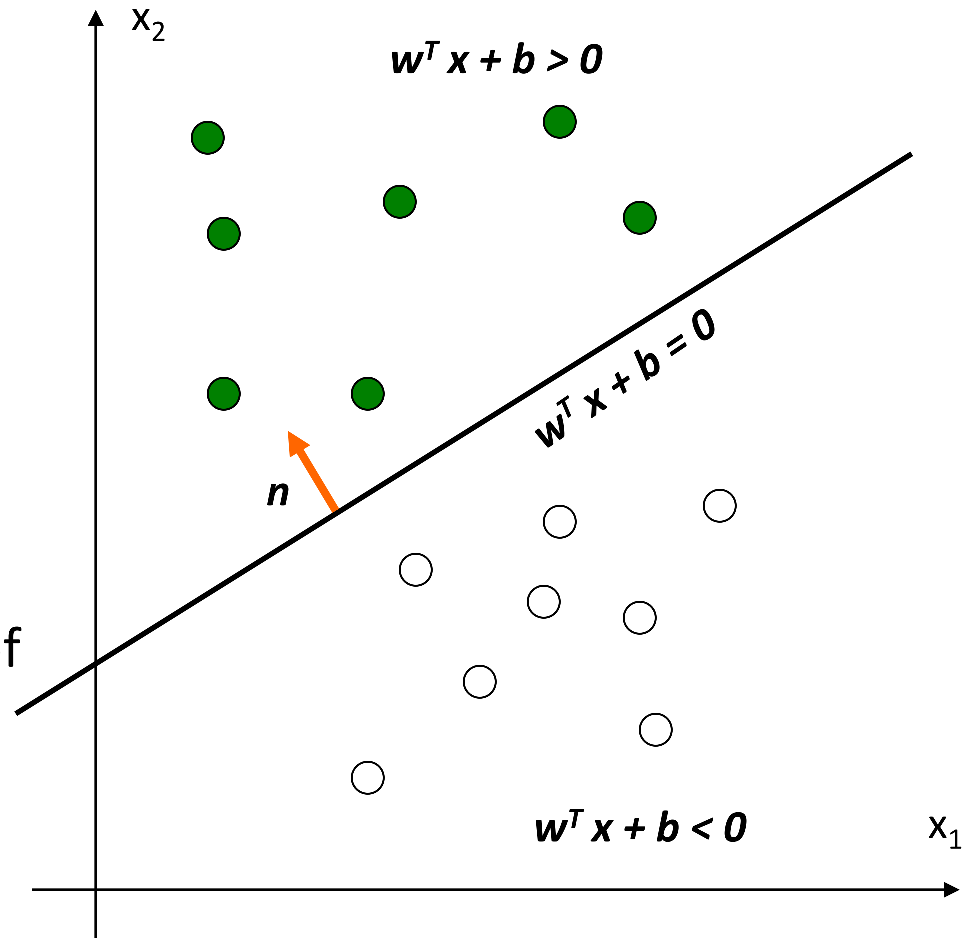
- g(x) is a linear function:

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

- A hyper-plane in the feature space

- (Unit-length) normal vector of the hyper-plane:

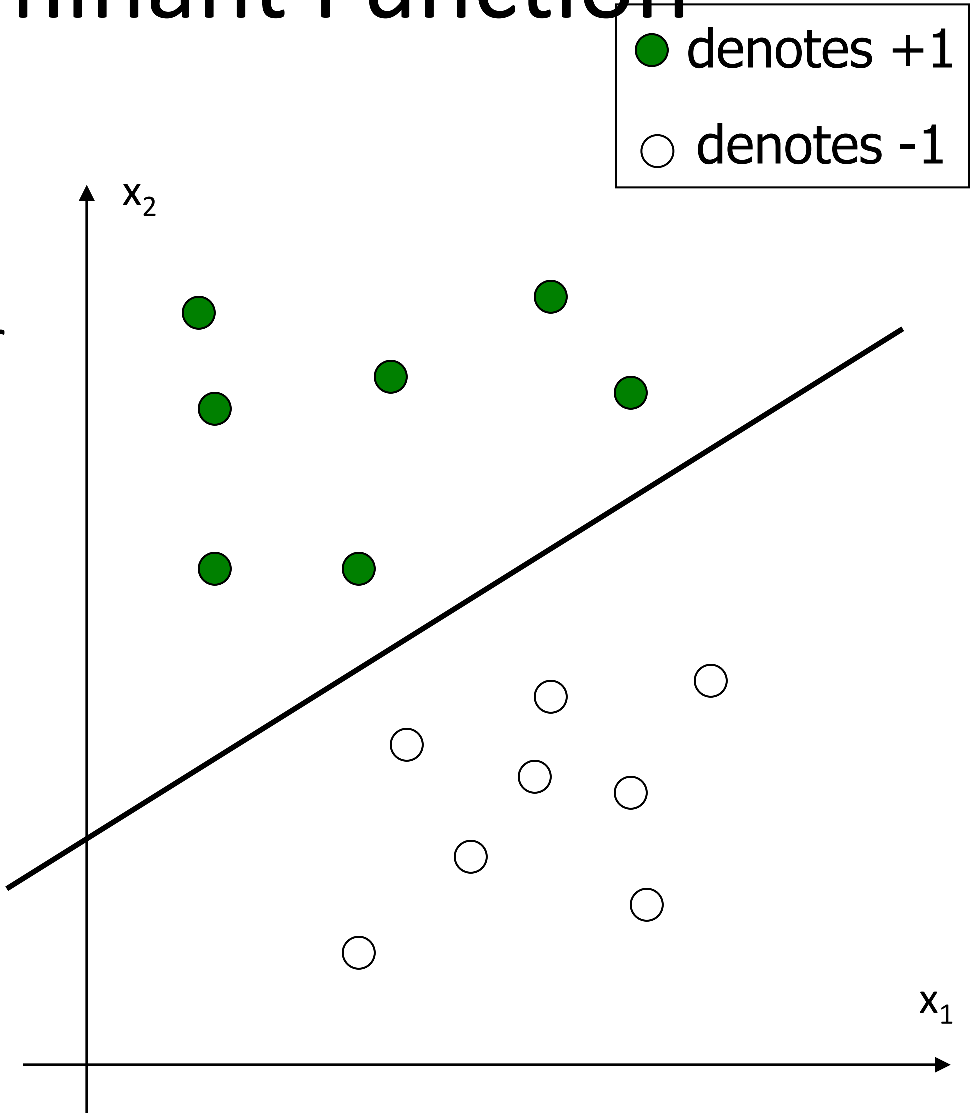
$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

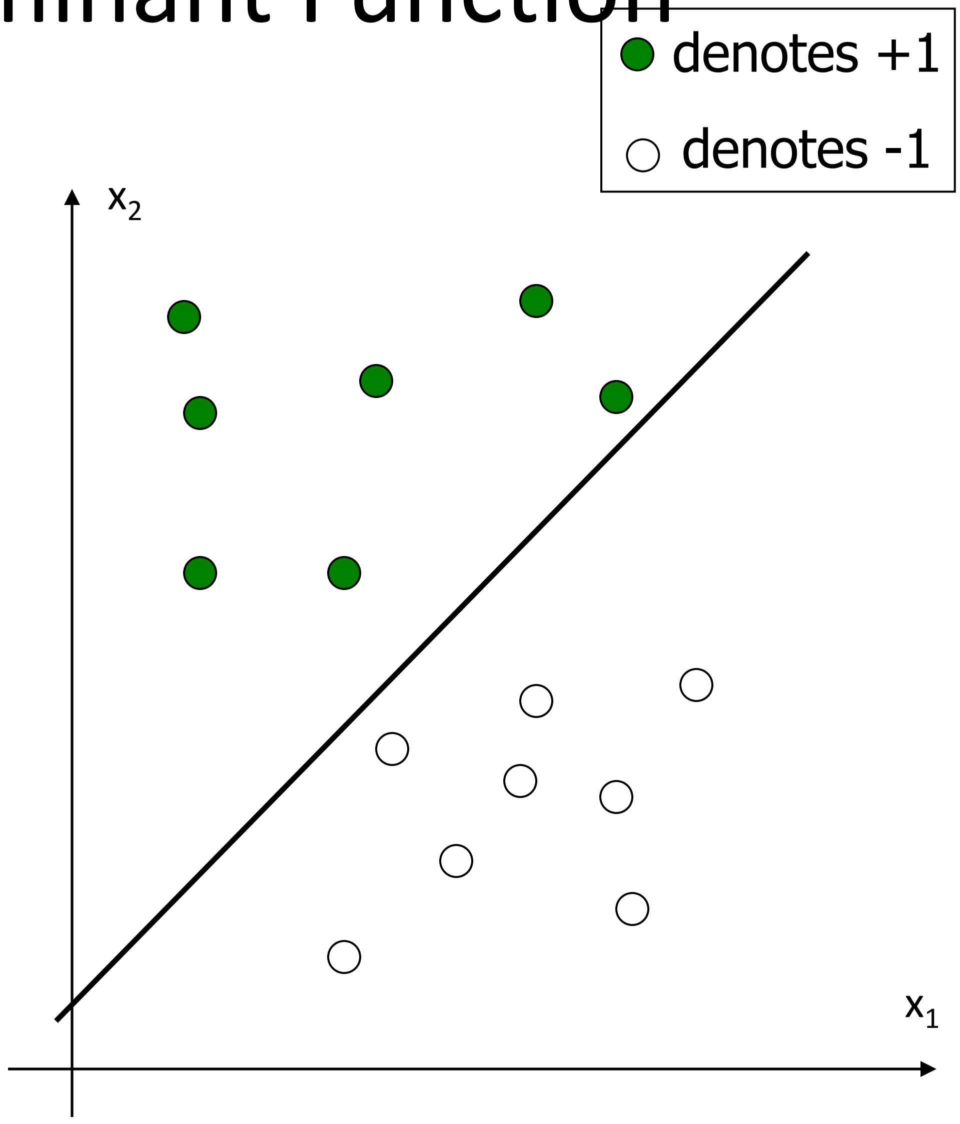
- Infinite number of answers!



Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

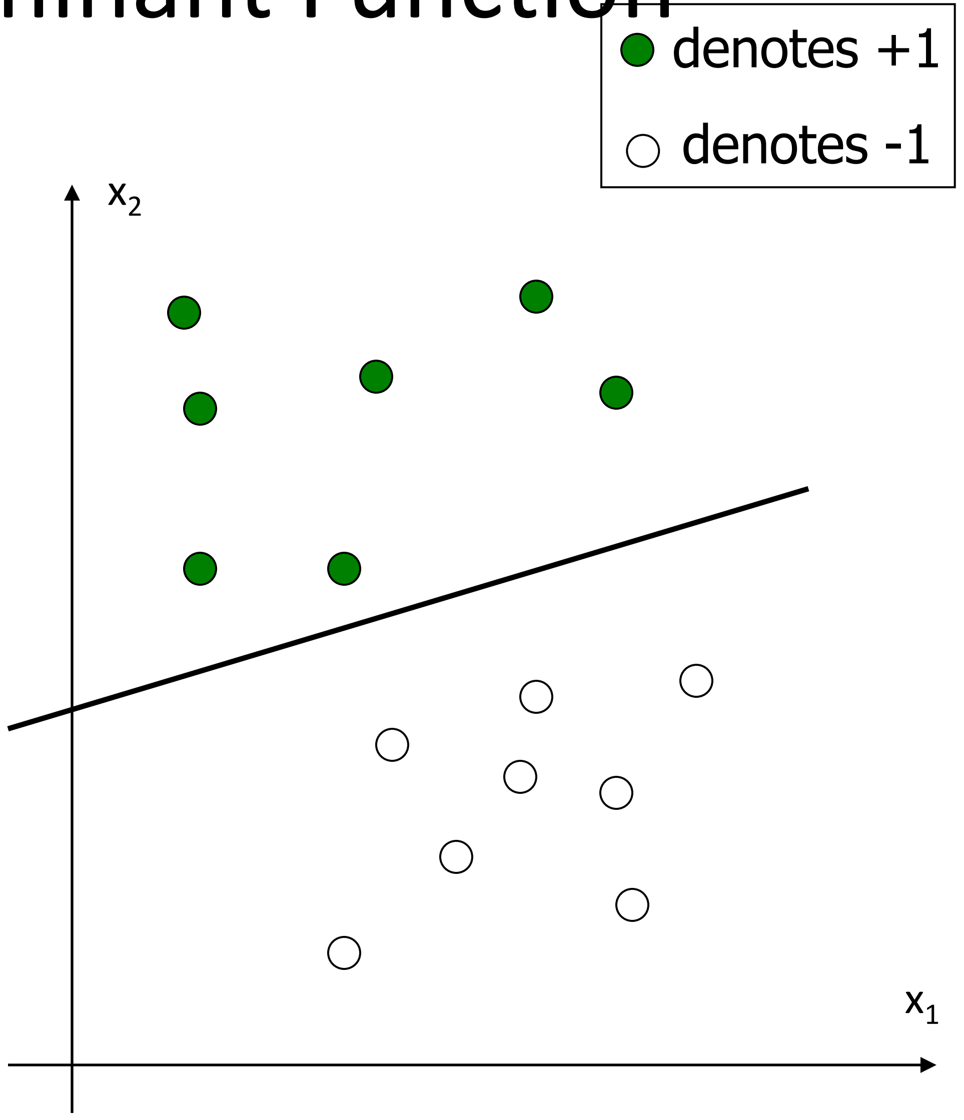
- Infinite number of answers!



Linear Discriminant Function

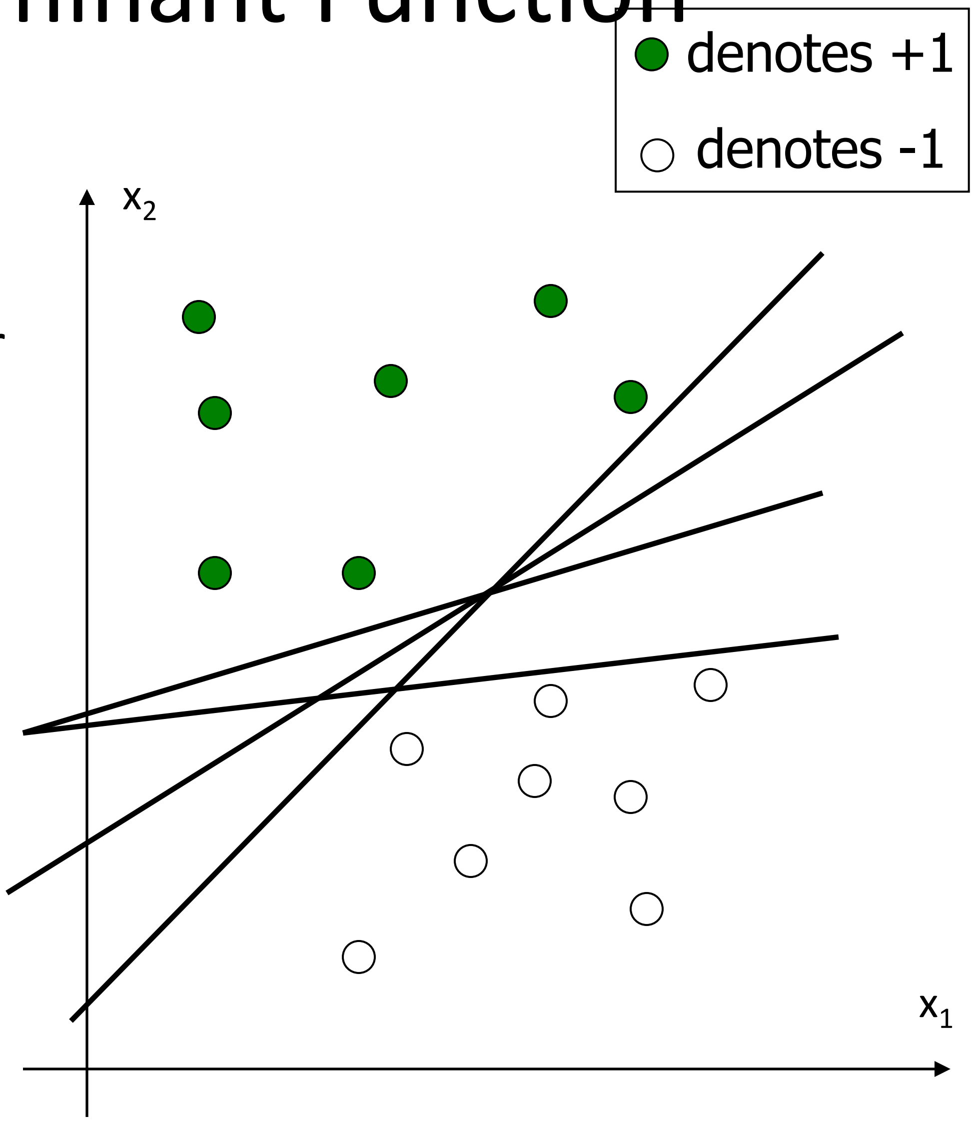
- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- Infinite number of answers!



Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?



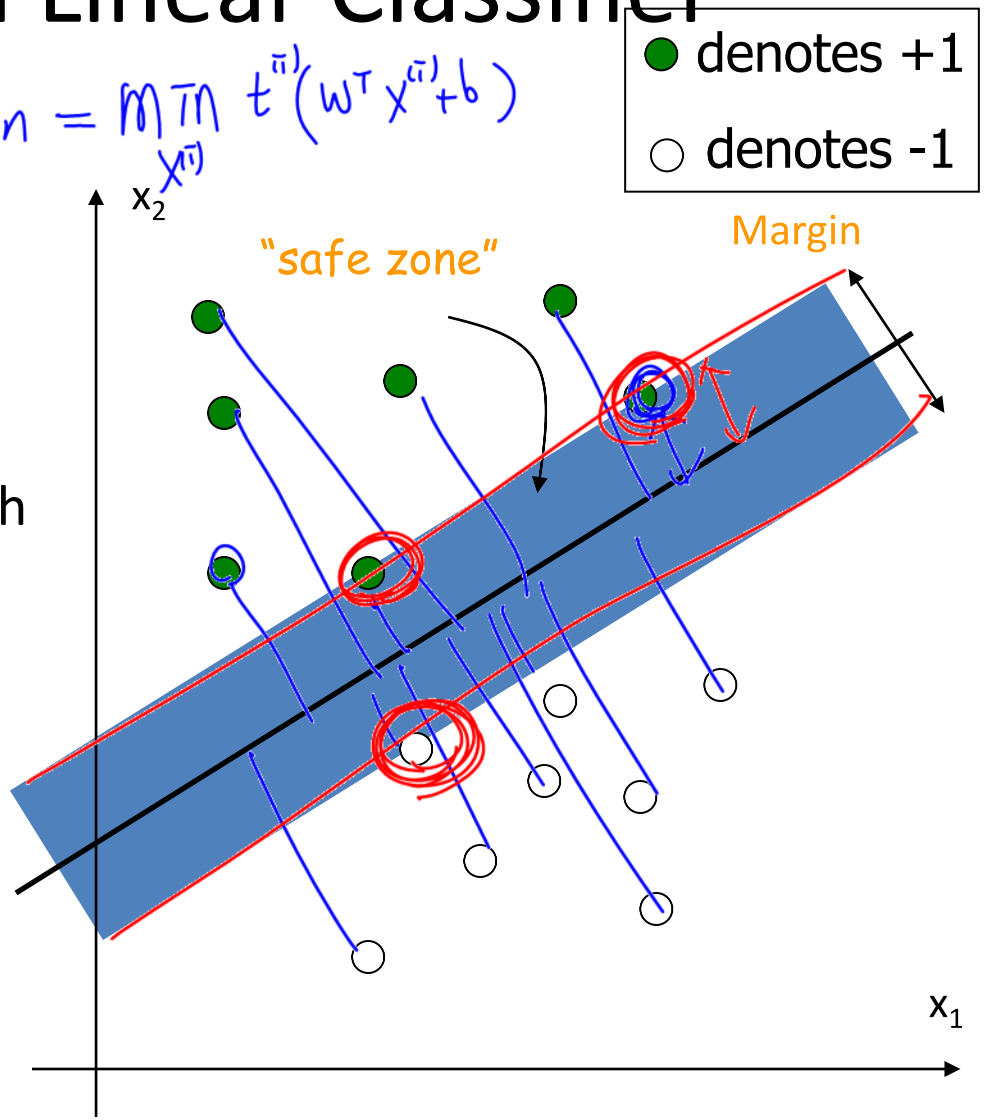
- Infinite number of answers!

- Which one is the best?

Large Margin Linear Classifier

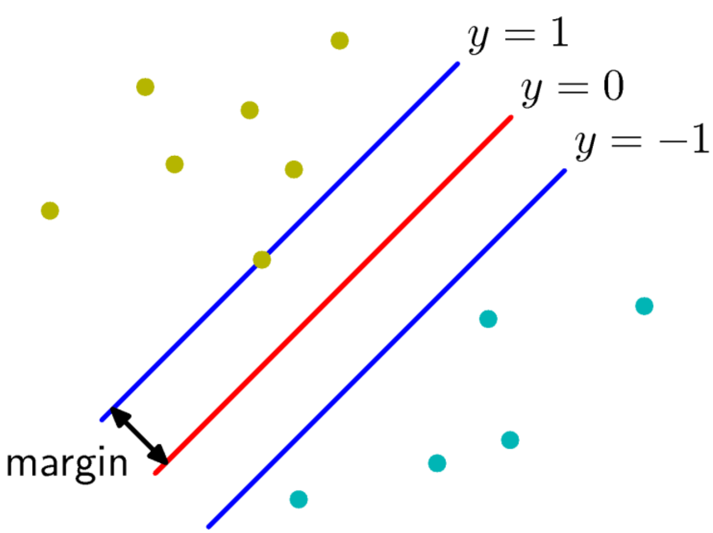
margin $= \min_{x^{(i)}} t^{(i)}(w^T x^{(i)} + b)$

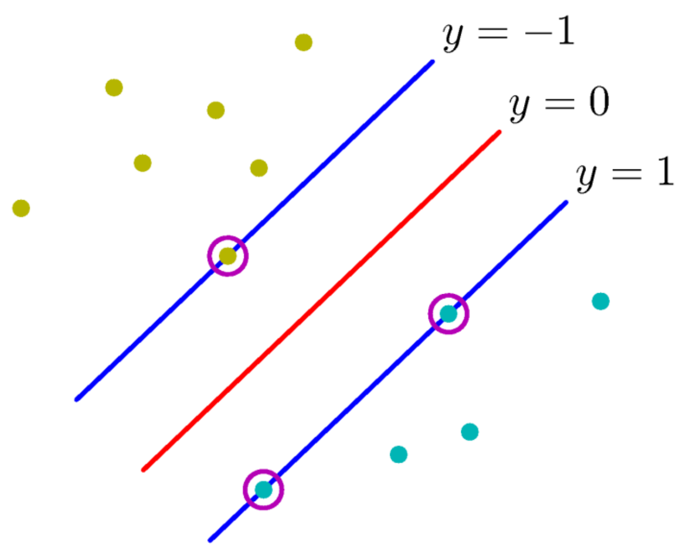
- The linear discriminant function (classifier) with the maximum margin is the best
- Margin is defined as the width that the boundary could be increased by before hitting a data point
- Why it is the best?
  - Robust to outliners and thus strong generalization ability



Maximum Margin

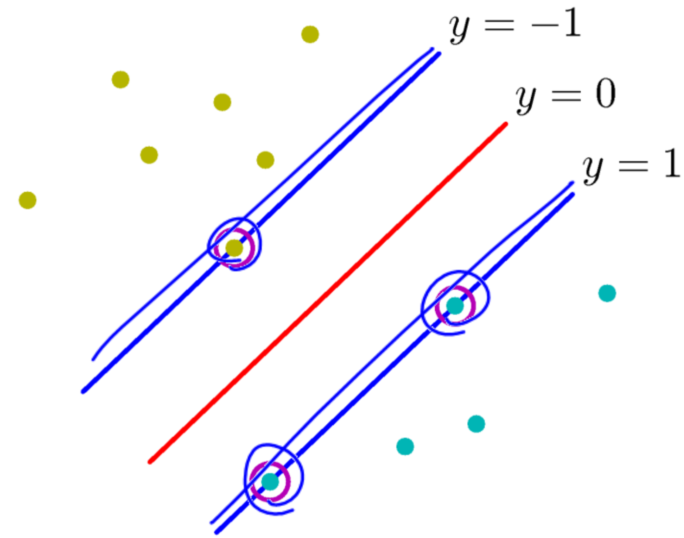
- The margin is the minimum distance of an example from the decision surface.
- Determine w and *b* to maximize the margin.





Support Vectors

- The *support vectors* are the points closest to the decision surface $y(\mathrm{x}) = 0$.
- Set w so that $t_n y(\mathrm{x}_n) = 1$ for support vectors.
- Only the support vectors determine the decision surface.



$t_n\ y(x_n)$ = margin.
$\geq 1\ ,\forall n$

Constraints for Optimization

- Set w and *b* so that, for support vectors:

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) = 1$$

- Then *every* data point must satisfy:

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) \geq 1 \text{ for } n = 1, \ldots, N$$

- It will turn out that only support vectors are active constraints.

Large Margin Linear Classifier

- Given a set of data points:

$$\{(\mathbf{x}_i, y_i)\},\ i = 1, 2, \cdots, n, \text{ where}$$

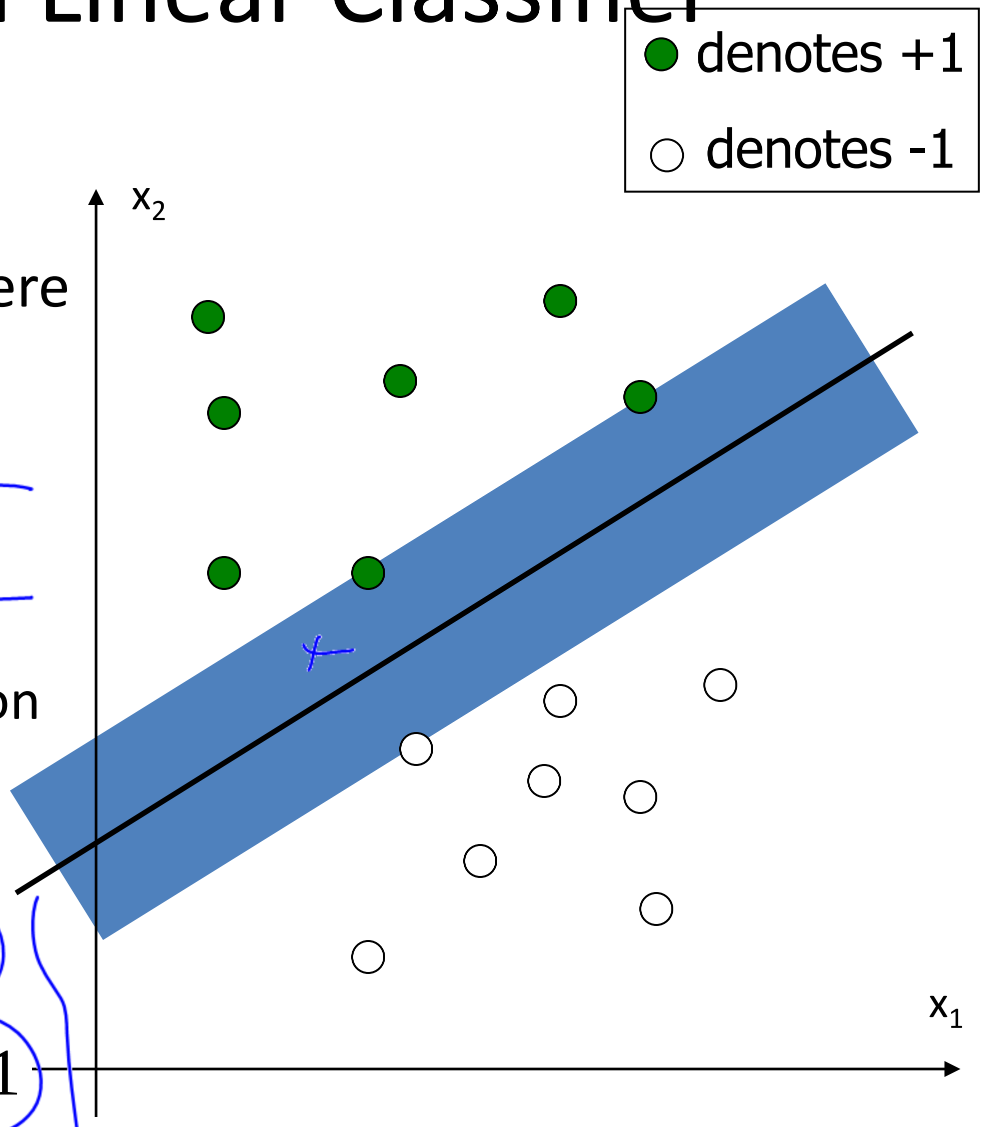
$$\text{For } y_i = +1,\quad \mathbf{w}^T\mathbf{x}_i + b > 0$$

$$\text{For } y_i = -1,\quad \mathbf{w}^T\mathbf{x}_i + b < 0$$

- With a scale transformation on both *w* and *b*, the above is equivalent to

$$\text{For } y_i = +1,\quad \mathbf{w}^T\mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1,\quad \mathbf{w}^T\mathbf{x}_i + b \leq -1$$



Large Margin Linear Classifier

- We know that

$$\mathbf{w}^T\mathbf{x}^+ + b = 1$$

$$\mathbf{w}^T\mathbf{x}^- + b = -1$$

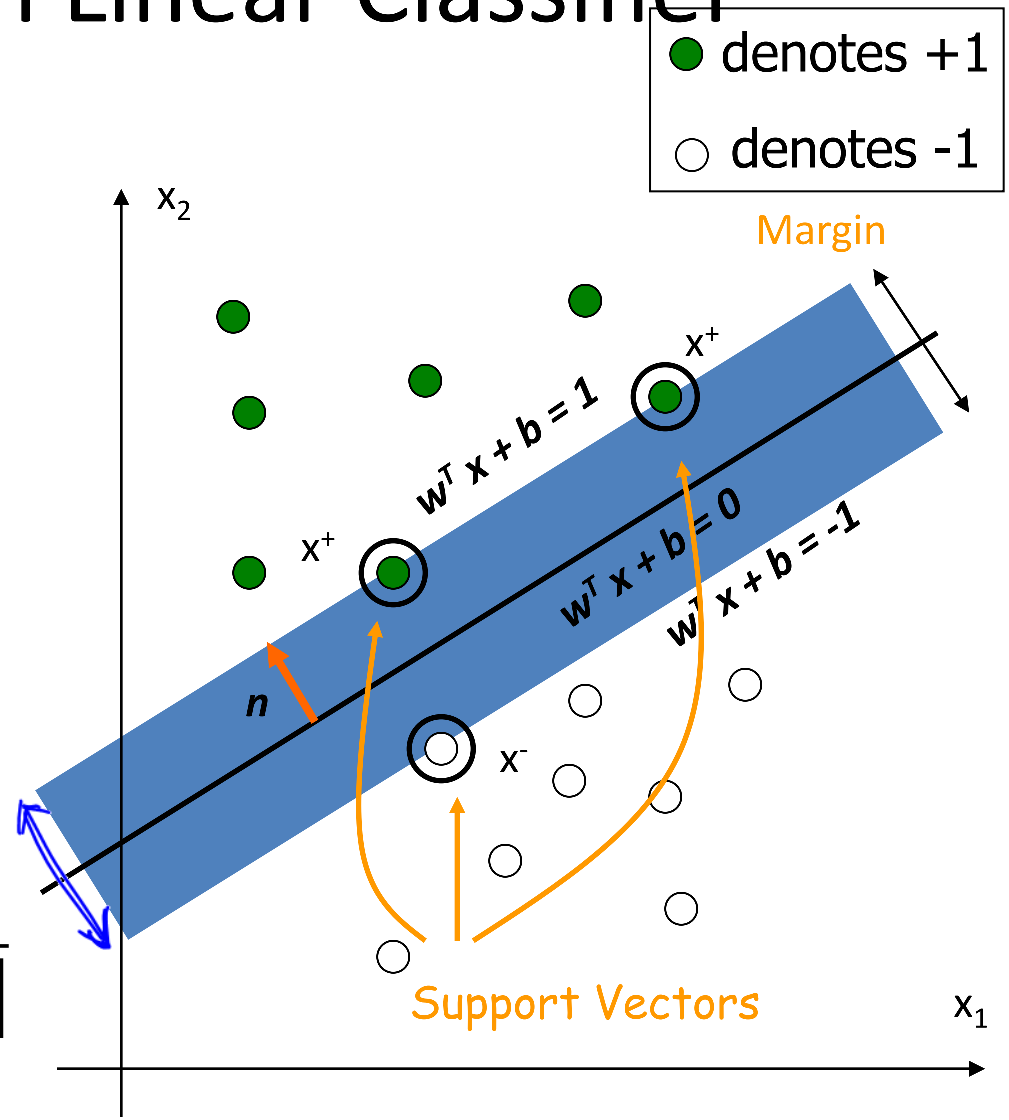
$w^T(x^+ - x^-) = 2$

- The margin width is:

$$M = (\mathbf{x}^+ - \mathbf{x}^-)\cdot\mathbf{n}$$

$$= (\mathbf{x}^+ - \mathbf{x}^-)\cdot\frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

$\vec{n}^T = \frac{w^T}{\|w\|}(x^+ - x^-) = \frac{2}{\|w\|}$



Large Margin Linear Classifier

// minimize $\|w\|^2$

- Formulation:

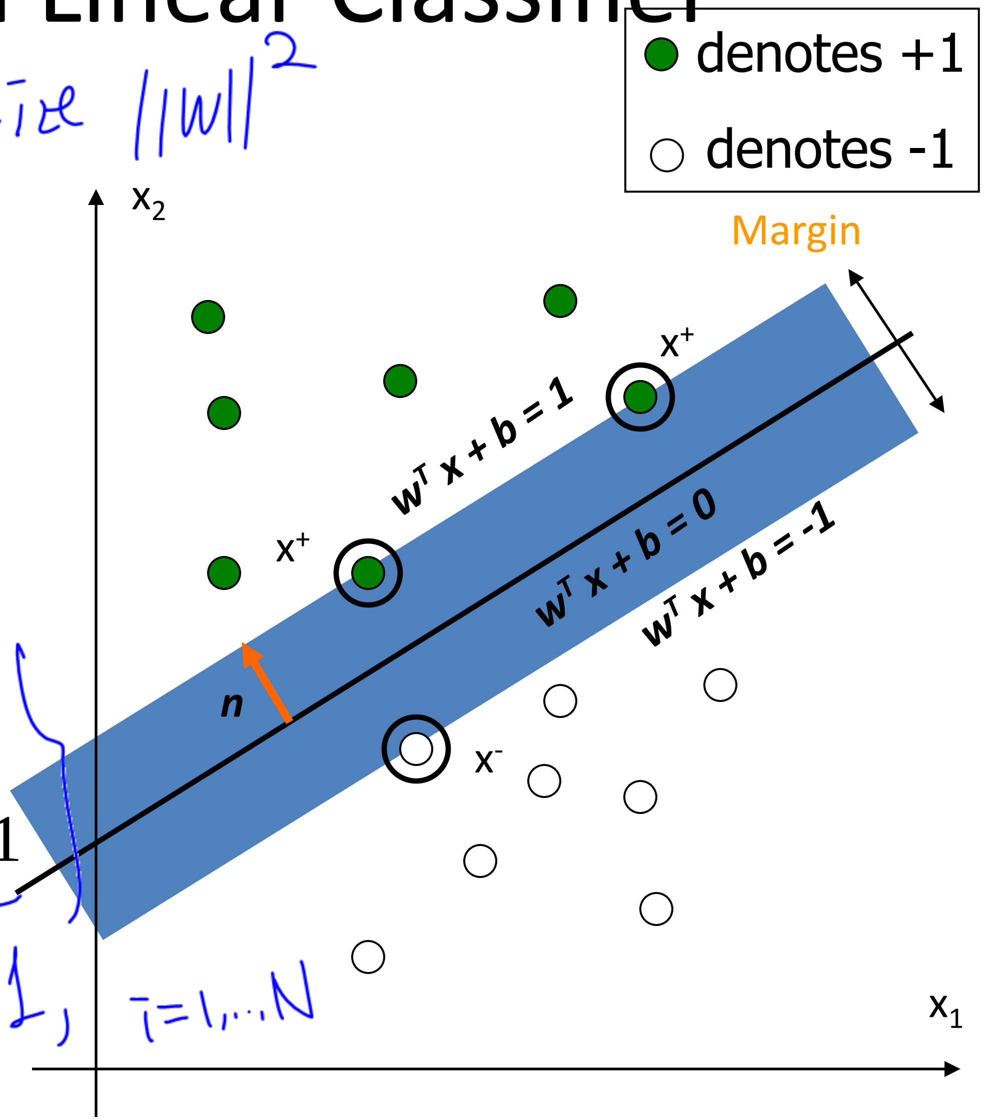
$$\boxed{\text{maximize} \quad \frac{2}{\|\mathbf{w}\|}}$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$$

$t_i(W^T x_i + b) \geq 1, \quad i = 1, \dots, N$



Maximize the Margin

- Distance to decision surface is $y(x_n)/||w||$
- To maximize the margin, maximize $||w||^{-1}$
- This is the same as minimizing $||w||^2$

- Use Lagrange multipliers to enforce constraints while optimizing

$a_n \geq 0$

$\sum_{n=1}^{N} \underbrace{a_n}_{\geq 0} \underbrace{\left[t_n (w^T \phi(x_n) + b) - 1 \right]}_{\geq 0}$

minimize

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}||^2 - \underbrace{\sum_{n=1}^{N} a_n \{t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) - 1\}}_{\geq 0}$$

$\leq \frac{1}{2}||w||^2$

$$\min_{x} \ f(x)$$

s.t. $g_i(x) \leq 0 \quad i=1,\dots,N$

$h_j(x) = 0 \quad j=1,\dots,M$

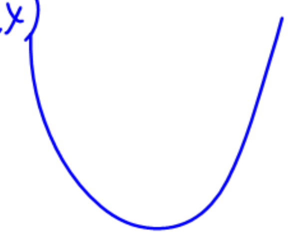
f: convex

g: convex

h: affine

1. Lagrangian

$$\mathcal{L}(x,\mu,\lambda) = f(x) + \sum_{i=1}^{N} \lambda_i g_i(x) + \sum_{j=1}^{M} \mu_j h_j(x)$$



convex

2. Solve Lagrange dual.

$$\max_{\mu,\lambda} \ \underbrace{\min_{x} \mathcal{L}(x,\mu,\lambda)}_{\mathcal{D}(\mu,\lambda)} \quad \text{s.t.} \quad \lambda_i \geq 0,\ i=1,\cdots,N$$

affin.

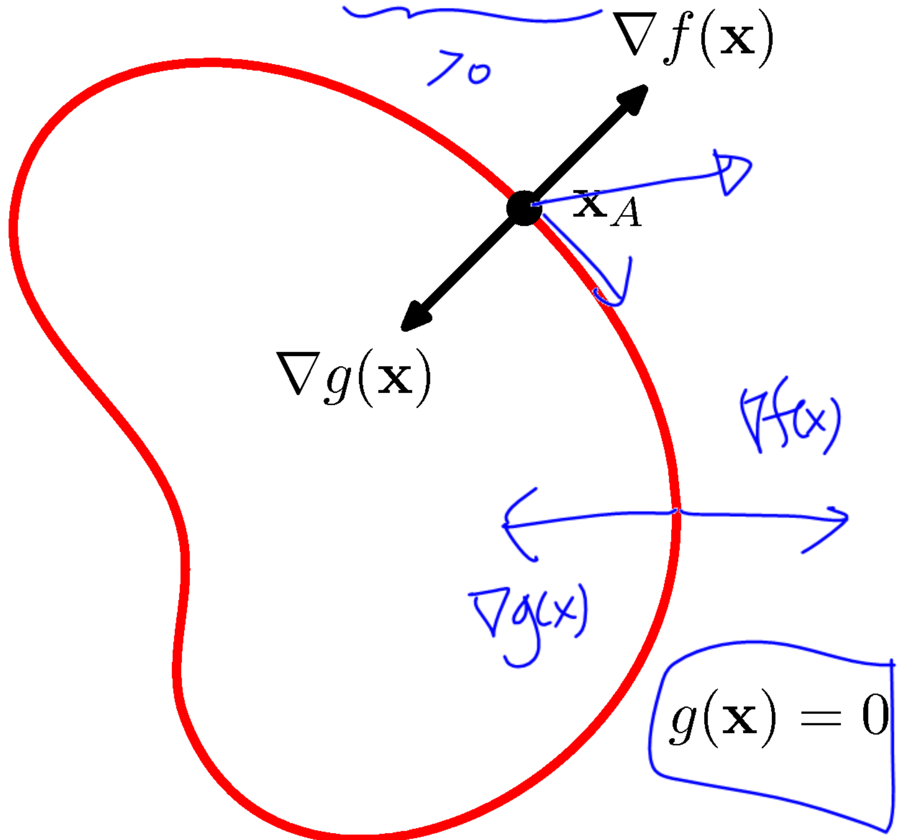
ax+b

Lagrange Multipliers

$x - x_0 \approx \nabla f(x)$

$$f(x) = f(x_0) + \underbrace{\nabla f(x)^T (x - x_0)}_{>0}$$

- Suppose we want to maximize *f*(x), subject to the constraint *g*(x)=0.
- At *every* point on the surface *g*(x)=0 the gradient of *g* is normal to the surface.
- At surface points that maximize *f*(x), the gradient of *f* is normal to the surface.



$\nabla g(x) \propto \nabla f(x)$

Lagrange Multipliers

- Since the gradients are parallel, there must exist a parameter (the Lagrange multiplier)

$$\nabla f + \lambda \nabla g = 0$$

$g(x)=0$

- Then we define the Lagrangian function

$$\underline{L(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) + \lambda g(\mathbf{x})}$$

$\nabla_x L(x, \lambda)$

- to optimize:

$$\nabla_{\mathbf{x}} L = 0 \text{ implies } \nabla f + \lambda \nabla g = 0$$

$$\underline{\partial L / \partial \lambda} = 0 \text{ implies } \boxed{g(\mathbf{x}) = 0}$$

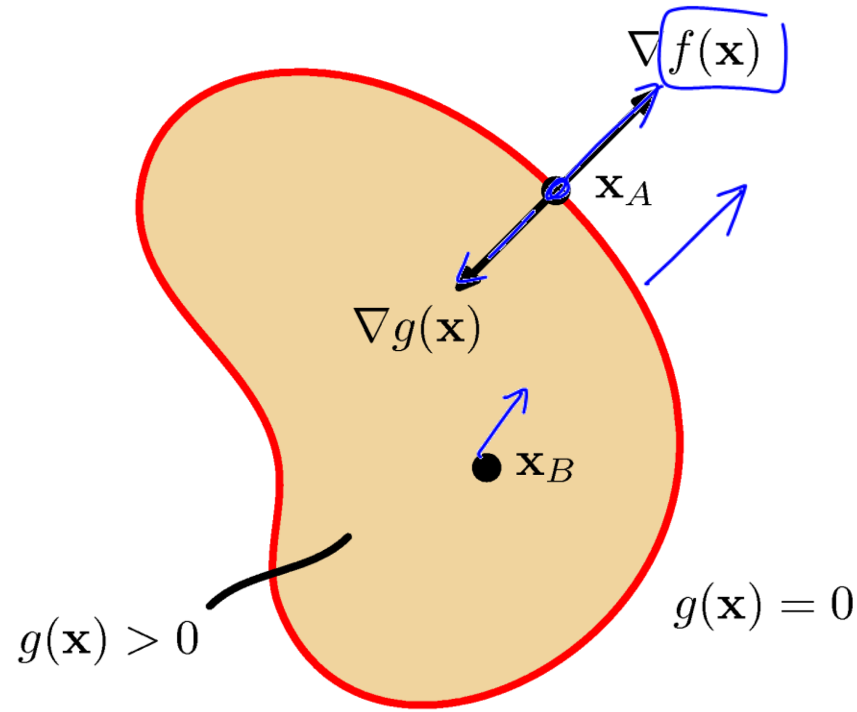
Lagrange Multipliers

- Suppose we have an inequality constraint

$$g(\mathbf{x}) \geq 0$$

- If boundary optimum x_A then gradient of *f* is outward, and $\lambda > 0$

- If internal optimum x_B then $\lambda = 0$



$$\begin{cases} \text{if } x^* \text{ on surface } g(x^*)=0, & \lambda \nabla g(x) + \nabla f(x) = 0 \\ \text{if inside}, & \nabla f(x) = 0 \end{cases}$$

Lagrange Multipliers

- Combining these cases gives us the *Karush-Kuhn-Tucker* (*KKT*) *conditions* when maximizing f(x) subject to an inequality constraint.

$$g(\mathbf{x}) \geq 0$$

$$\lambda \geq 0$$

$$\boxed{\lambda g(\mathbf{x}) = 0}$$

$\lambda = 0$

or $\lambda > 0$, then $g(x) = 0$

Maximize the Margin

- Distance to decision surface is $y(x_n)/||w||$
- To maximize the margin, maximize $||w||^{-1}$
- This is the same as minimizing $||w||^2$

- Use Lagrange multipliers to enforce constraints while optimizing

$\sum a_n t_n \phi(x_n)$

$a_n \geq 0$

Lagrangian

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} a_n \{\underbrace{t_n(\boxed{\mathbf{w}^T}\phi(\mathbf{x}_n) + b) - 1}_{\geq 0}\}$$

$$\nabla_w L = w - \sum_{n=1}^{N} a_n t_n \phi(x_n) = 0$$

Maximize the Margin

- Set the derivatives of *L*(w,*b*,a) to zero, to get

$\nabla_w L(w,b,a)$ $\nabla_b L(w,b,a)$

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n) \qquad 0 = \sum_{n=1}^{N} a_n t_n$$

- Substitute in, to eliminate w and *b*,

Lagrange dual $a_n \geq 0, \ \forall n$

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \underbrace{\boxed{\phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)}}_{= k(x_n, x_m)}$$

Dual Representation (with kernel)

- Define a kernel $k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$
- This gives, to maximize

s.t. $a_n \geqslant 0$

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

- Once we have a, we don't need w. Predict new values using

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \underline{\sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b}$$

$\mathbf{w} = \sum_n a_n t_n \phi(x_n)$

Recovering b

- For any support vector x_n: $t_n \boxed{y(x_n)} = 1$
- Replacing with $y(\mathbf{x}) = \underline{\mathbf{w}^T \phi(\mathbf{x}) + b} = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$

$$t_n \left[t_n \left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1 \right]$$

(index) set of support vectors

$t_n^2 = 1$

- Multiply t_n, and sum over n:

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$$

Support Vectors

- The KKT conditions are:

$$a_n \geq 0$$
$$t_n y(\mathbf{x}_n) - 1 \geq 0$$
$$a_n \{t_n y(\mathbf{x}_n) - 1\} = 0$$

- Which means, either a_n=0 or $t_n y(\mathrm{x}_n)$=1.

- That is, only the support vectors matter!
  – To predict $y(\mathbf{x})$, sum only over support vectors

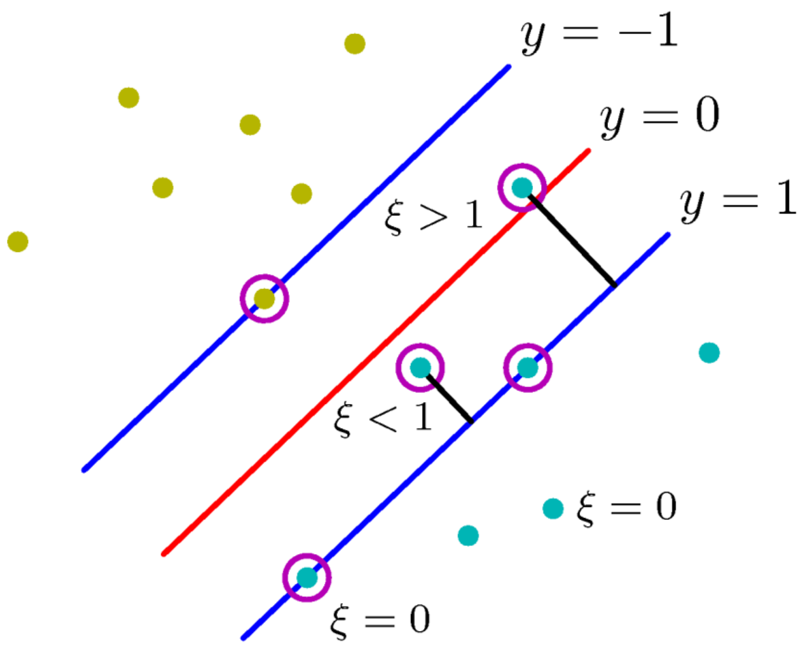
Support Vector Machines

- Hard SVM requires separable sets

$$t_n y(\mathbf{x}_n) - 1 \geq 0$$

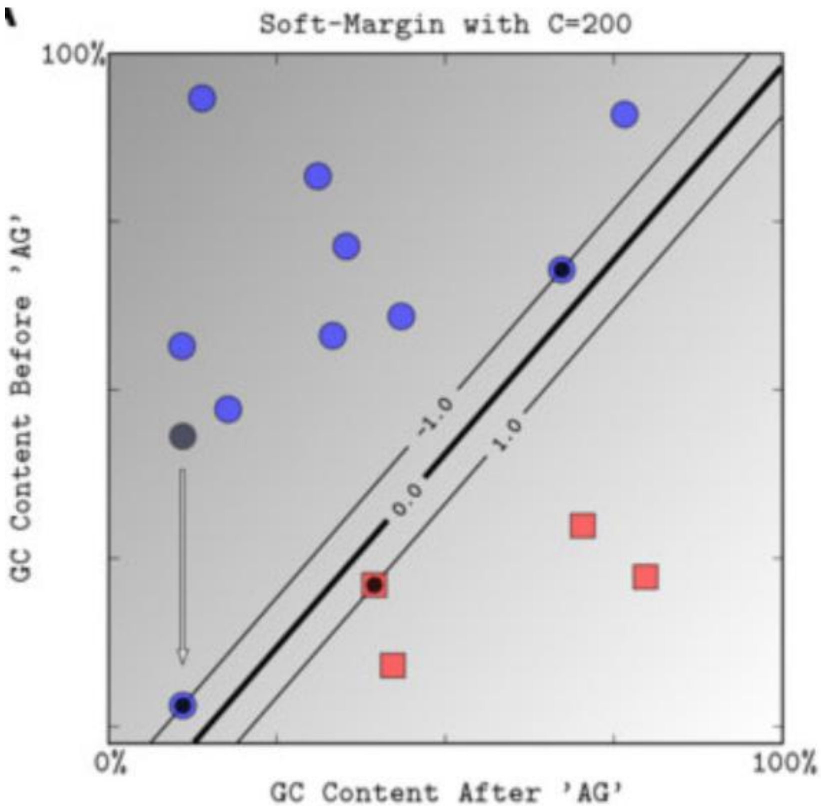
- Soft SVM introduces *slack variables* for each data point

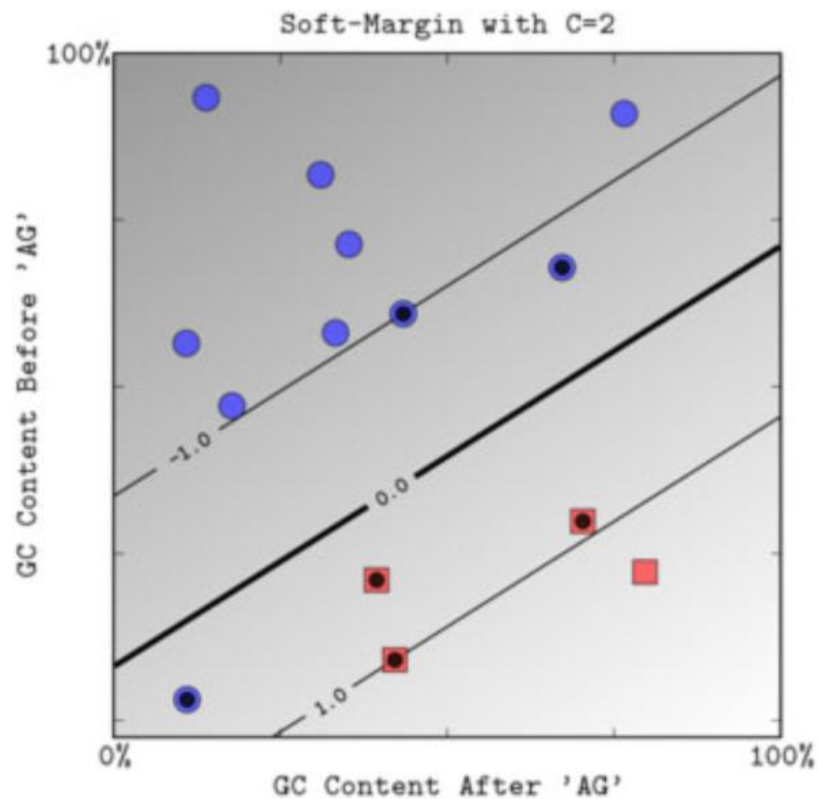
$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$



Soft SVM

- A little slack can give much better margin.





Soft SVM

- Maximize the margin, and also penalize for the slack variables

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2}||\mathbf{w}||^2$$

- The support vectors are now those with

$$t_n y(\mathbf{x}_n) = 1 - \xi_n$$

Formulation of soft-margin SVM

- Primal form
- Minimize (w.r.t. w and ξ_n's)

$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\mathbf{w}\|^2$$

Subject to $t_n y(\mathbf{x}_n) \geq 1 - \xi_n$, $\forall n$

$$\xi_n \geq 0, \forall n$$

Dual formulation of soft-margin SVM

- Lagrangian

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n \xi_n$$

  – Where $a_n \geq 0,\ \mu_n \geq 0,\ \xi_n \geq 0, \forall n$

- KKT conditions for the constraints

$$\begin{aligned} a_n &\geqslant 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n &\geqslant 0 \\ a_n \left(t_n y(\mathbf{x}_n) - 1 + \xi_n\right) &= 0 \\ \mu_n &\geqslant 0 \\ \xi_n &\geqslant 0 \\ \mu_n \xi_n &= 0 \end{aligned}$$

Dual formulation of soft-margin SVM

- Taking derivatives

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N} a_n t_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \quad \Rightarrow \quad a_n = C - \mu_n.$$

Dual formulation of soft-margin SVM

- Lagrange dual

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to

$$0 \leqslant a_n \leqslant C$$

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

- Solve quadratic problem (convex optimization)

Support Vector Machine: Algorithm

- 1. Choose a kernel function
- 2. Choose a value for *C*
- 3. Solve the quadratic programming problem (many software packages available)
- 4. Construct the discriminant function from the support vectors

Some Issues

- Choice of kernel
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures

- Choice of kernel parameters
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

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

Summary: Support Vector Machine

- 1. Large Margin Classifier
  - Better generalization ability & less over-fitting
- 2. The Kernel Trick
  - Map data points to higher dimensional space in order to make them linearly separable.
  - Since only dot product is used, we do not need to represent the mapping explicitly.

Additional Resource

- http://www.kernel-machines.org/

SVM Implementation

- LIBSVM
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - One of the most popular generic SVM solver (supports nonlinear kernels)
- Liblinear
  - http://www.csie.ntu.edu.tw/~cjlin/liblinear/
  - One of the fastest linear SVM solver
- SVMlight
  - http://www.cs.cornell.edu/people/tj/svm_light/
  - Structured outputs, various objective measure (e.g., F1, ROC area), Ranking, etc.

SVM demo code

- http://www.mathworks.com/matlabcentral/fileexchange/28302-svm-demo

- http://www.alivelearn.net/?p=912