EECS 545: Machine Learning

Lecture 8. Kernel methods

Honglak Lee

2/2/2011

Outline

- Support Vector Machines
- Convex optimization overview

Support Vector Machines

Classification

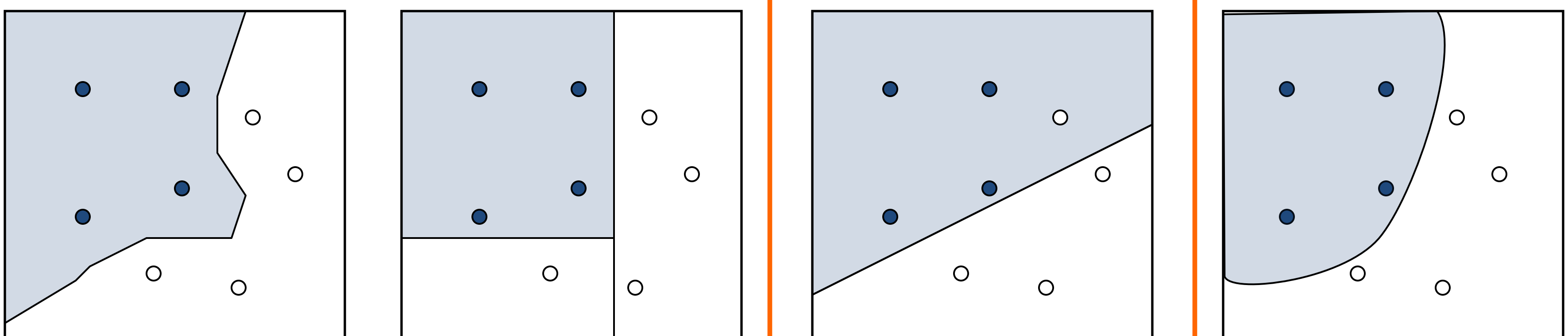
- Consider a two-class classification problem:
  - Positive: $t = +1$
  - Negative: $t = -1$
- Train a linear model over the feature vector:

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

- 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: $t_n \, y(\mathbf{x}_n) > 0$.

Discriminant Function

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



Nearest Neighbor

Decision Tree

Linear Functions

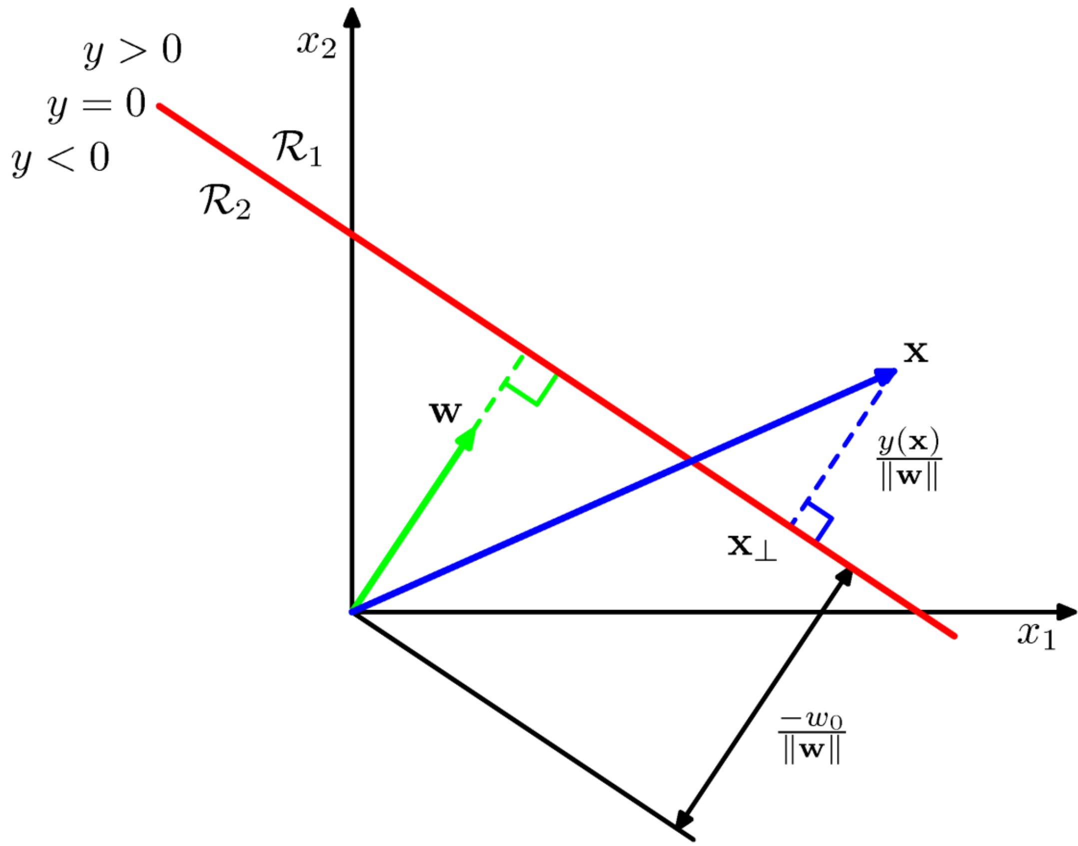
$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

Nonlinear Functions

Distance from Decision Surface

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

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



Linear Discriminant Function

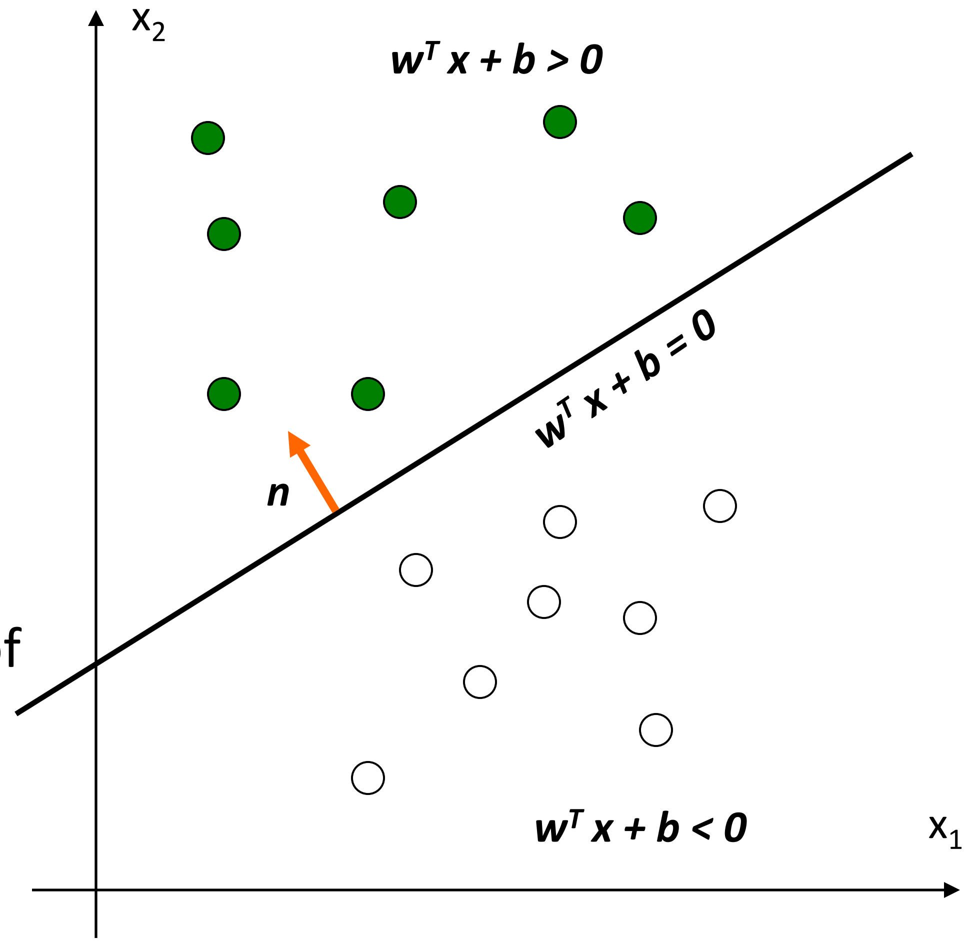
- g(x) is a linear function:

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

- A hyper-plane in the feature space

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

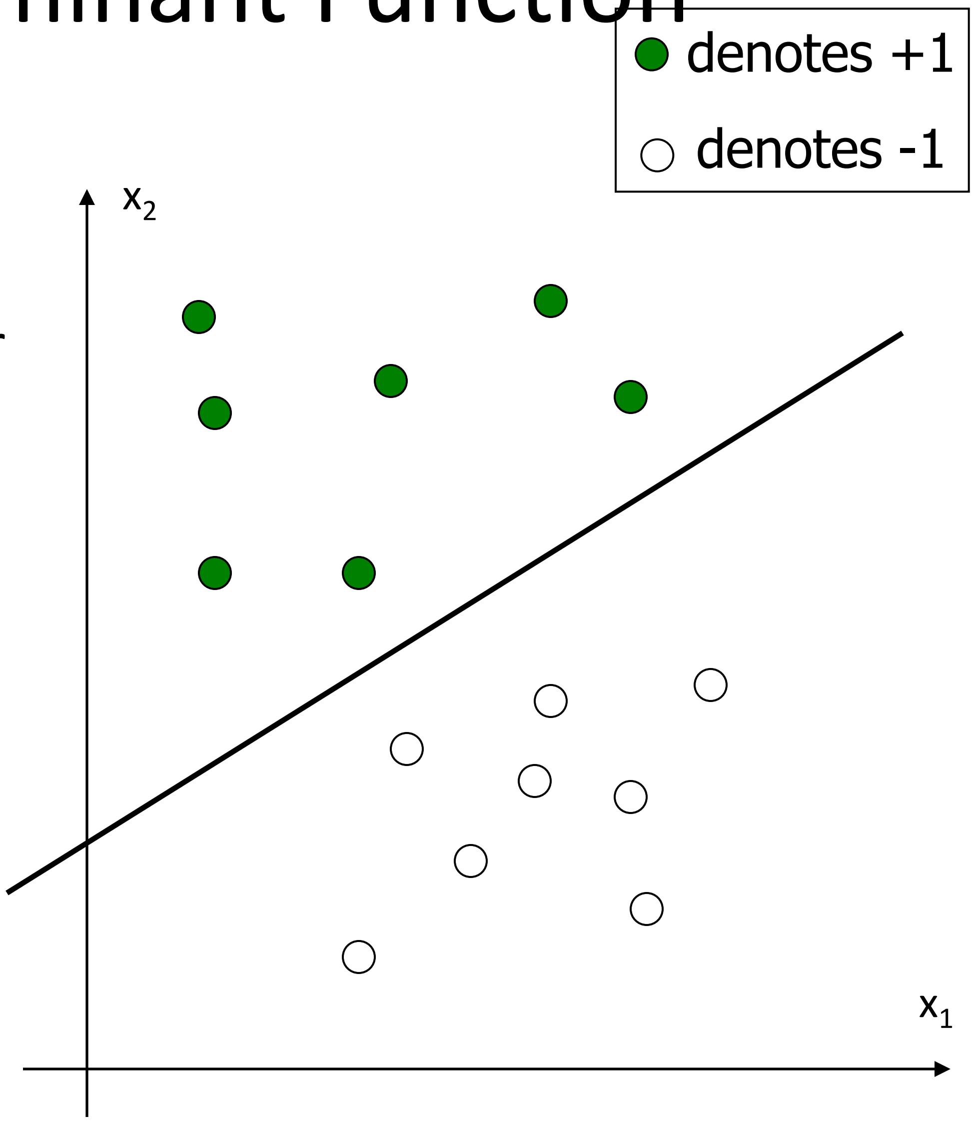
$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



Linear Discriminant Function

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

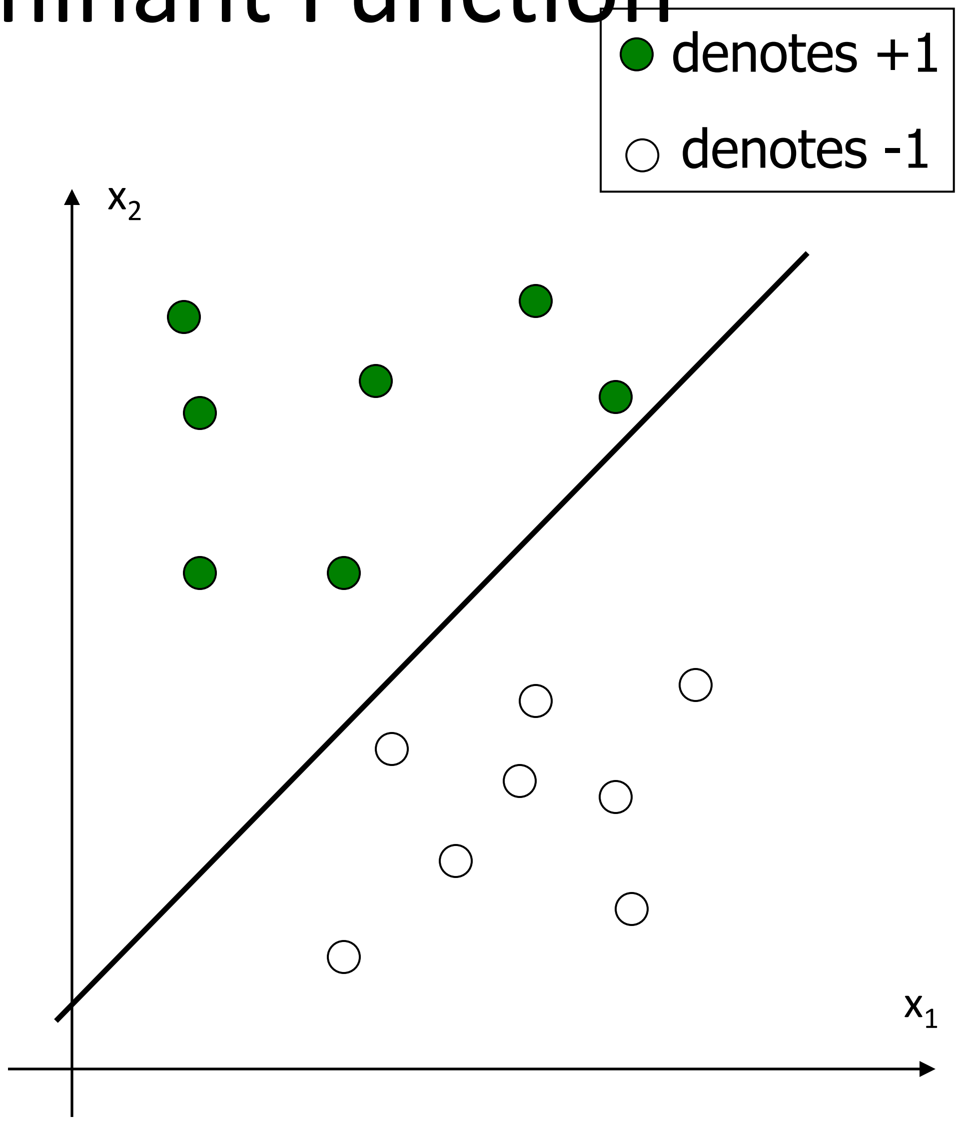
- Infinite number of answers!



Linear Discriminant Function

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

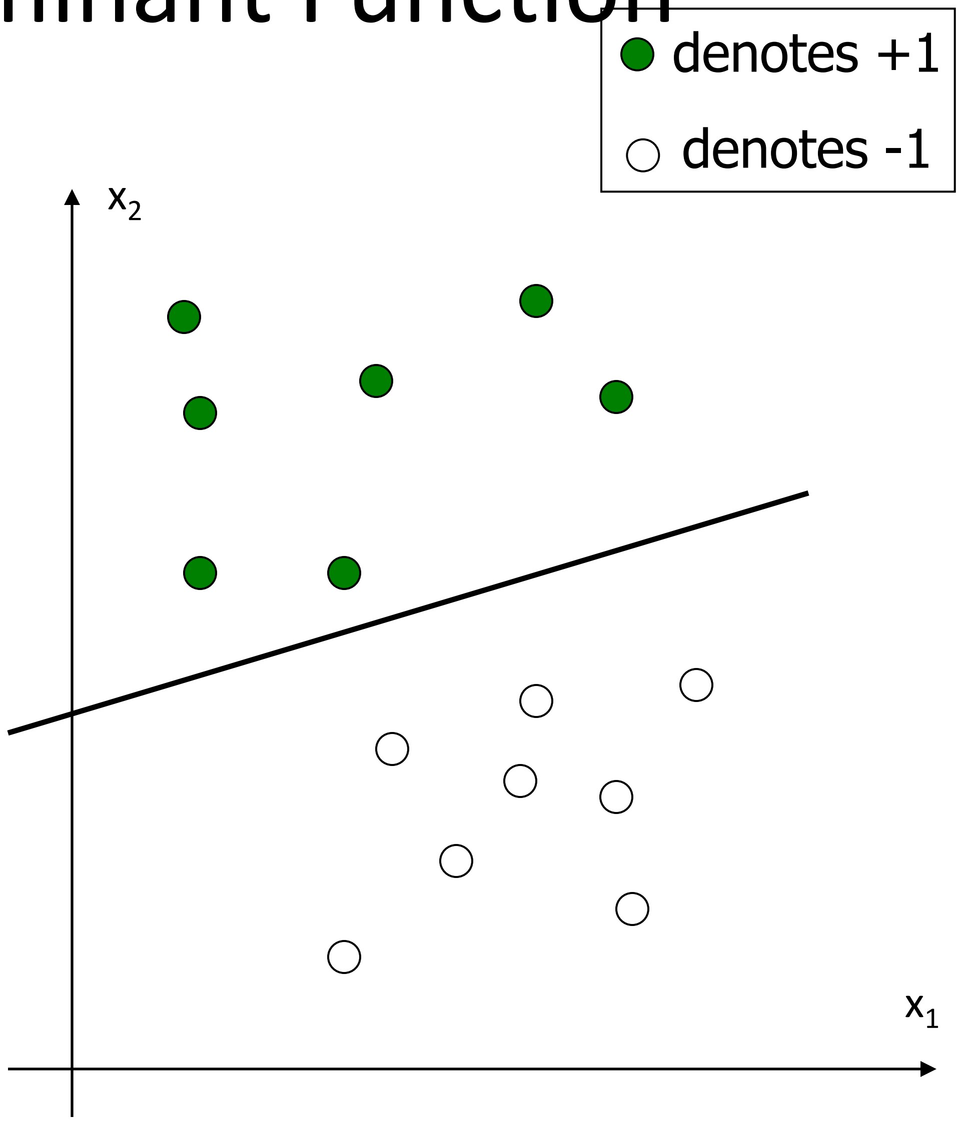
- Infinite number of answers!



Linear Discriminant Function

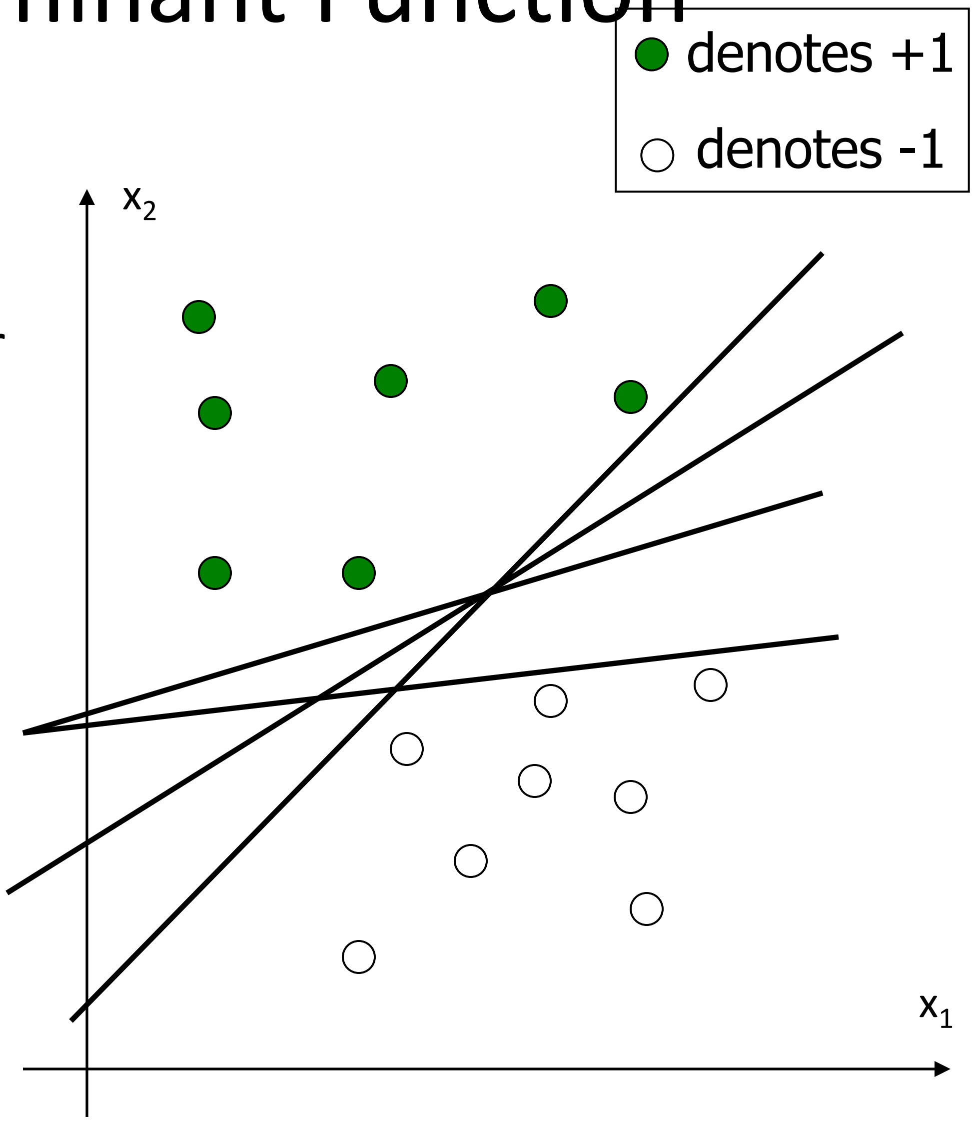
- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- Infinite number of answers!



Linear Discriminant Function

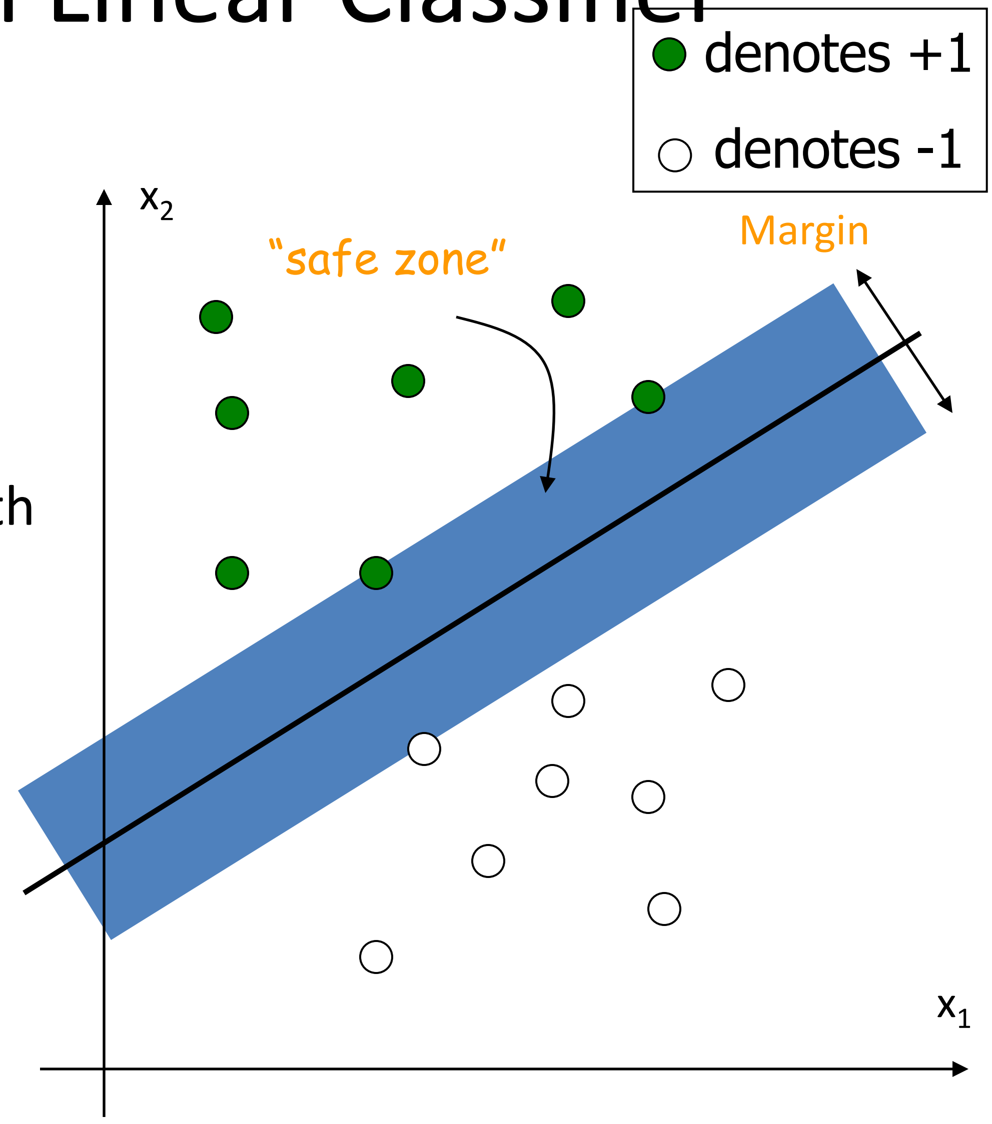
- How would you classify these points using a linear discriminant function in order to minimize the error rate?



- Infinite number of answers!
- Which one is the best?

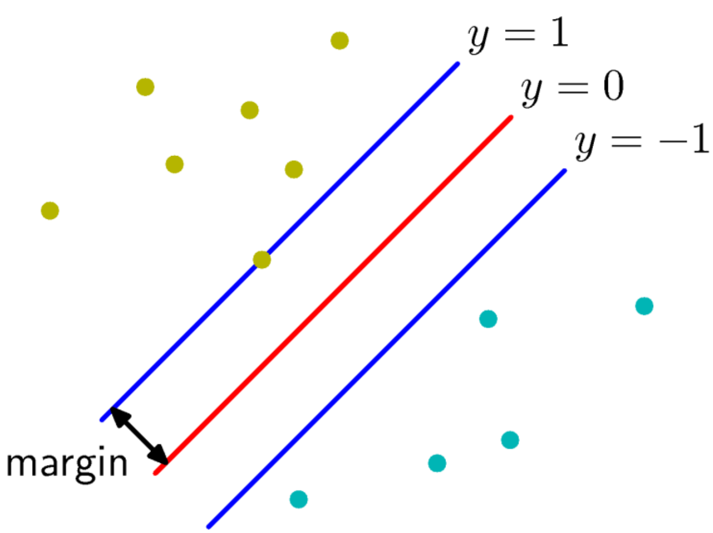
Large Margin Linear Classifier

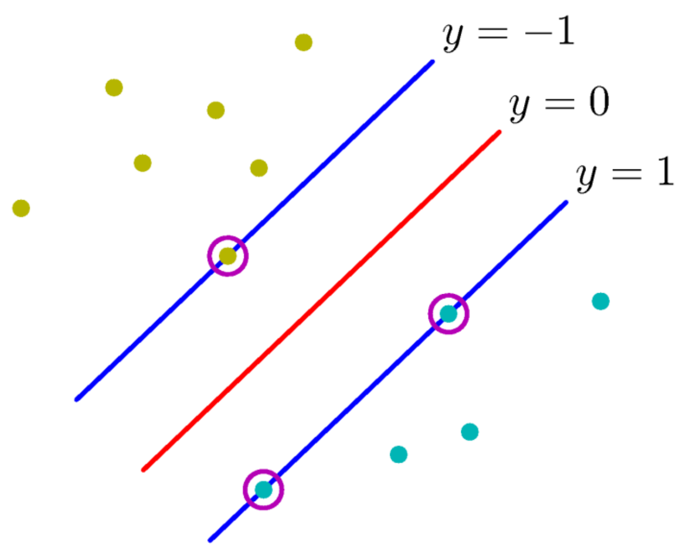
- The linear discriminant function (classifier) with the maximum margin is the best
- Margin is defined as the width that the boundary could be increased by before hitting a data point
- Why it is the best?
  - Robust to outliners and thus strong generalization ability



Maximum Margin

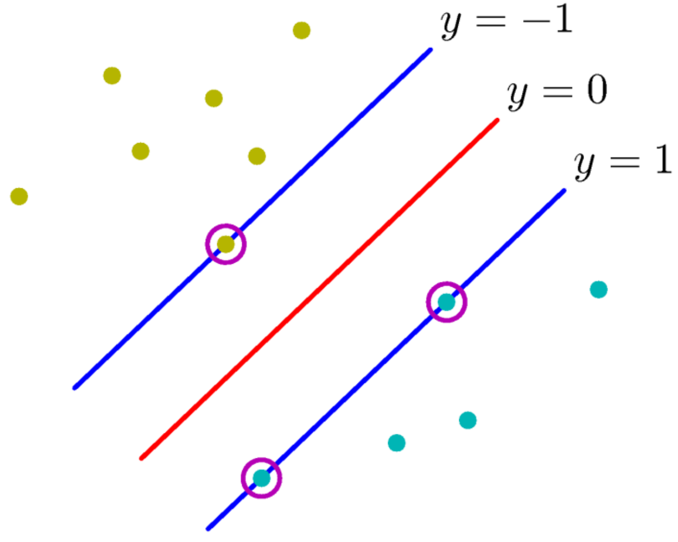
- The margin is the minimum distance of an example from the decision surface.
- Determine w and *b* to maximize the margin.





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.



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:

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

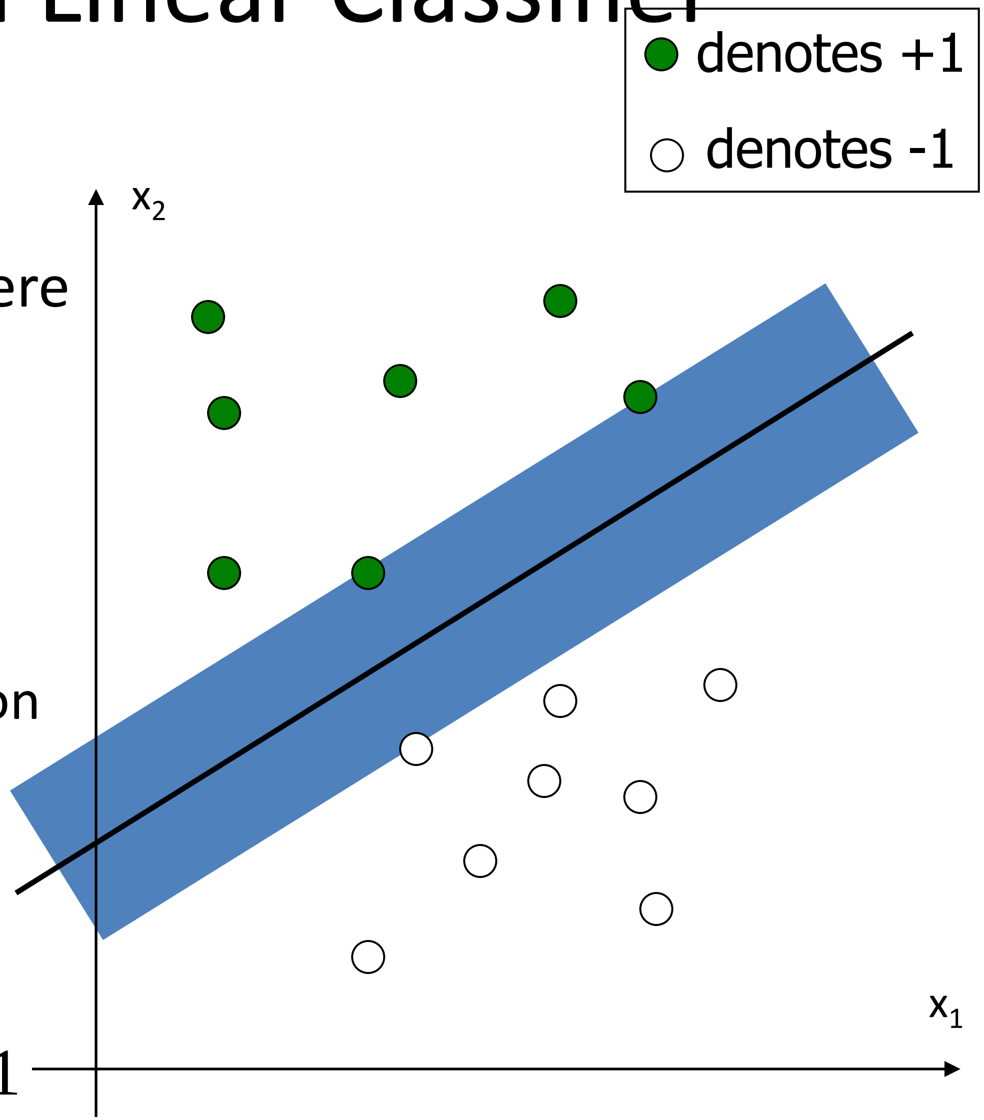
$$\text{For } y_i = +1,\quad \mathbf{w}^T \mathbf{x}_i + b > 0$$

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

- 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

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

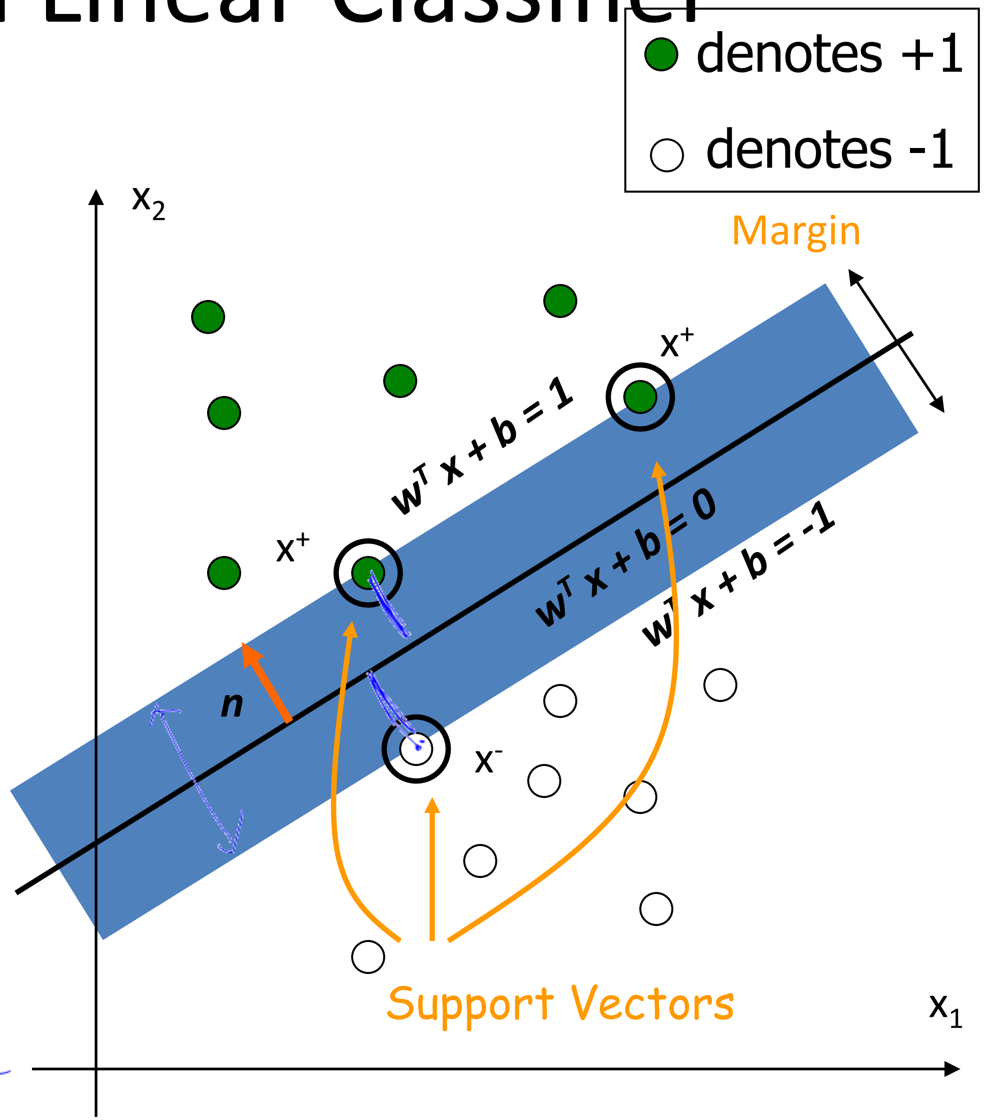
subtract, divide by $||w||$

- The margin width is:

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

normal vector

2x distance



Large Margin Linear Classifier

minimize $\|w\|^2$

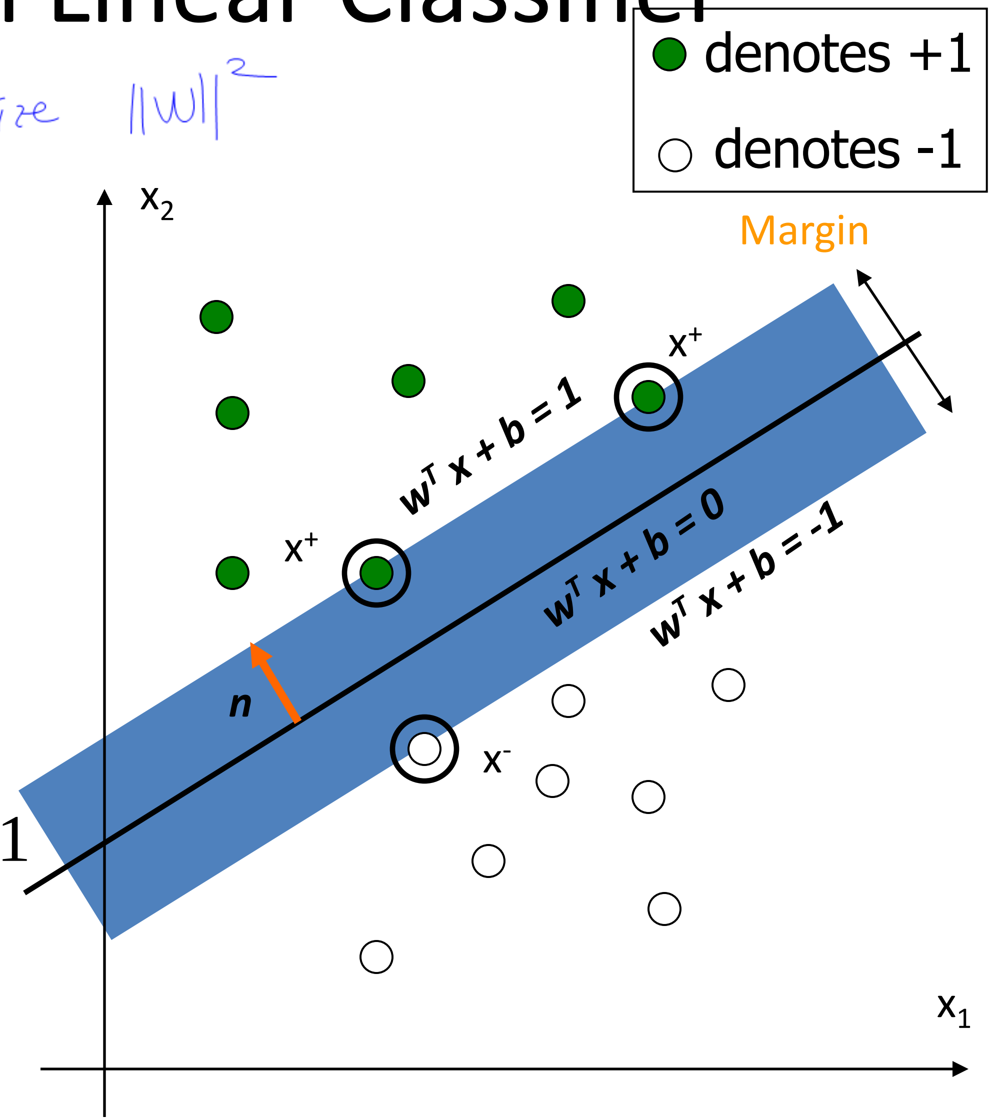
- Formulation:

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

such that

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

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



Maximize the Margin

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

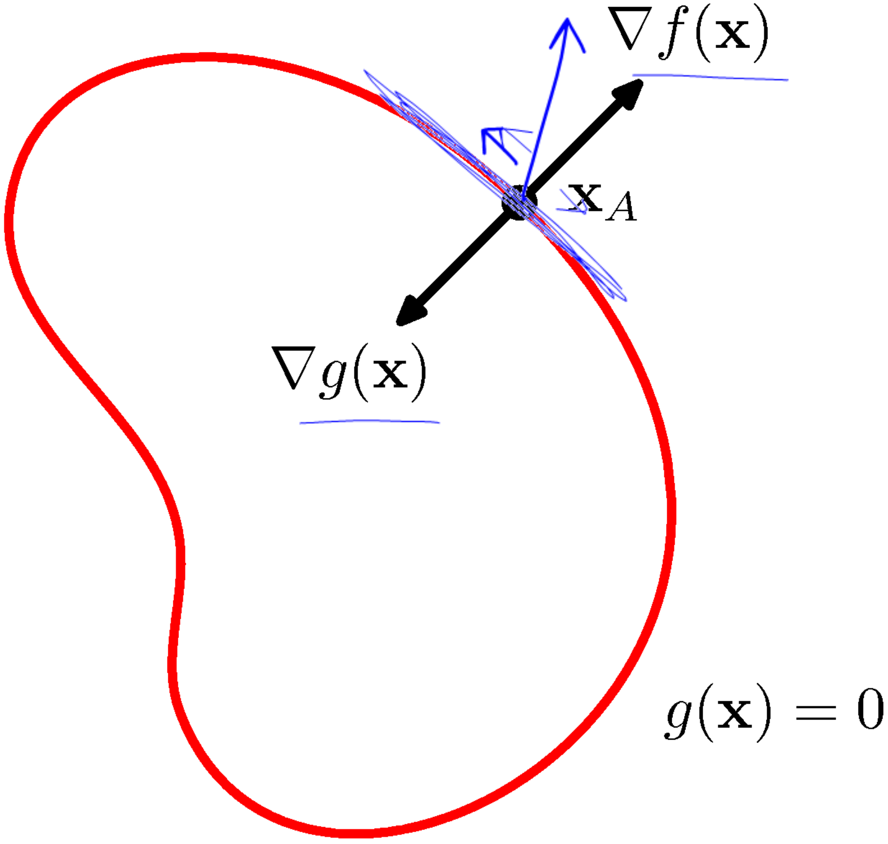
- Use Lagrange multipliers to enforce constraints while optimizing

Lagrange multipliers $\quad a_n \geq 0$

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

Lagrange Multipliers

- 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.



Lagrange Multipliers

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

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

- Then we define the Lagrangian function

$$L(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) + \lambda \underbrace{\boxed{g(\mathbf{x})}}_{=0}$$

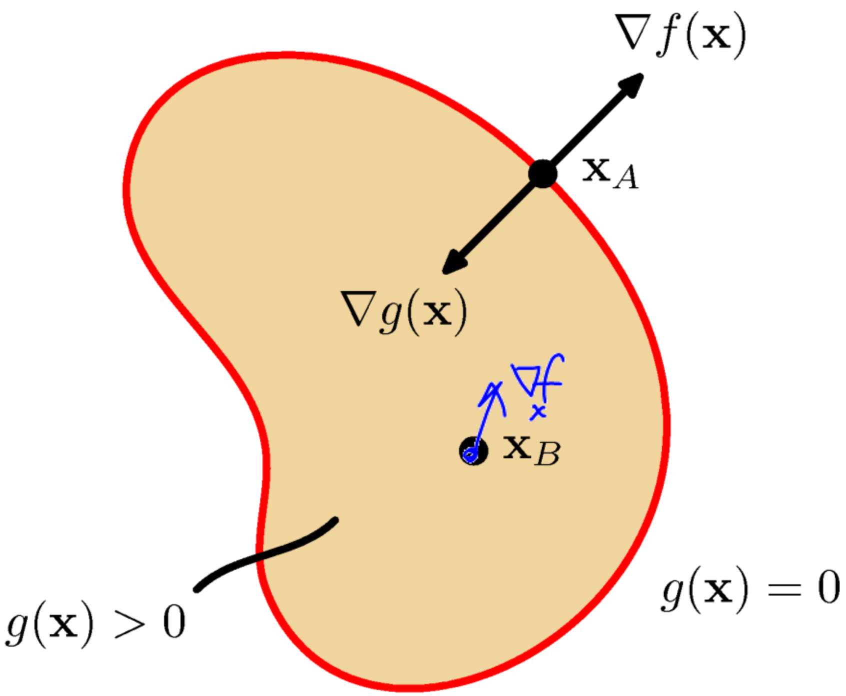
- to optimize:

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

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

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 $\lambda = 0$



Lagrange Multipliers

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

$$\left.\begin{aligned} g(\mathbf{x}) &\geq 0 \\ \lambda &\geq 0 \end{aligned}\right\} \text{original constraint}$$

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

$$\begin{cases} \lambda > 0, & \text{then } g(x) = 0 \\ \lambda = 0 \end{cases}$$

$$\begin{cases} \underline{g(x) > 0}, & \text{then } \lambda = 0 \\ g(x) = 0 \end{cases}$$

Maximize the Margin

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

- Use Lagrange multipliers to enforce constraints while optimizing

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

Maximize the Margin

- Set the derivatives of $L(\mathrm{w},b,\mathrm{a})$ to zero, to get

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

- Substitute in, to eliminate w and b,

Lagrange dual.

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

max.

s.t. $a_i \geq 0$

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

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

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

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

Recovering b

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

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

(index) set of support vectors

- 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:

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

original

- 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

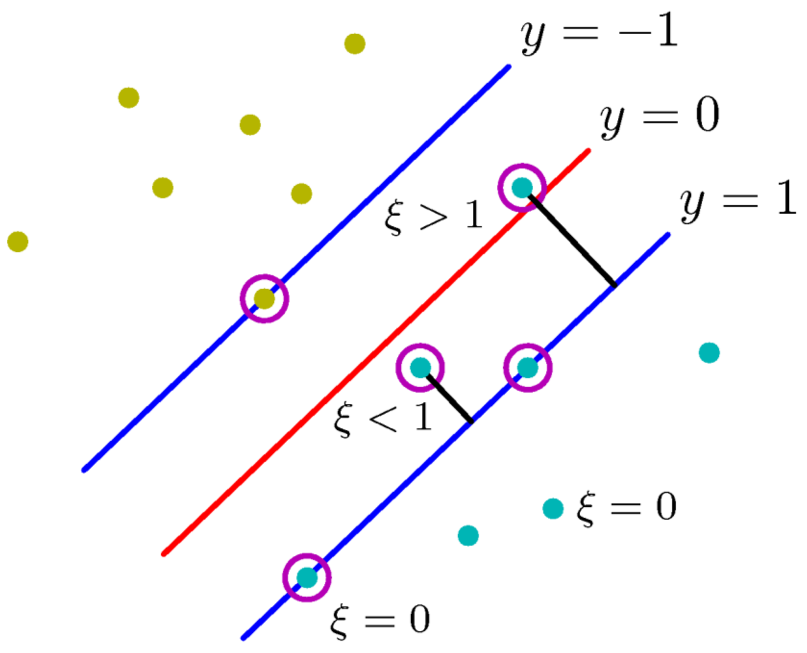
- Hard SVM requires separable sets

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

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

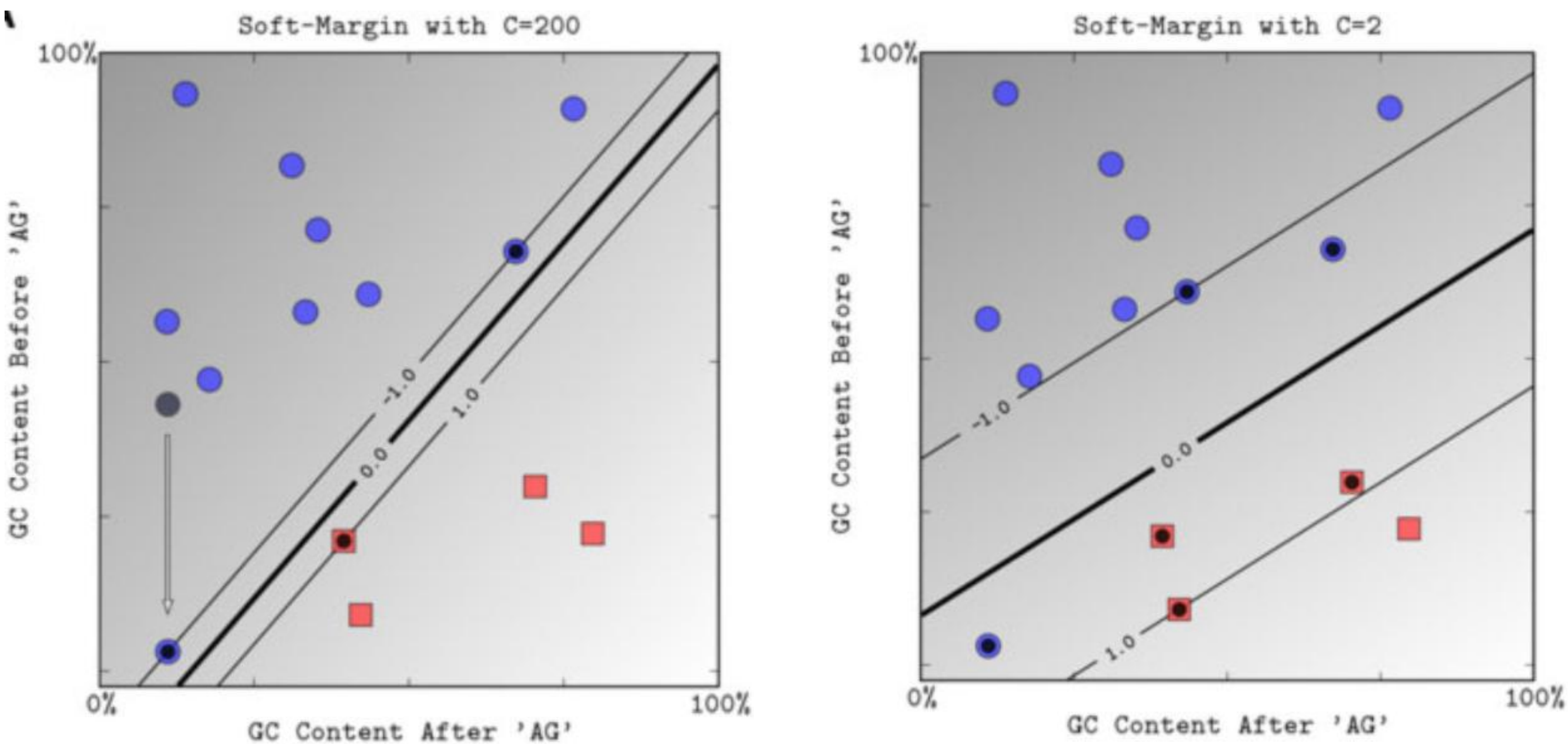
$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

"violation of margin constraint"



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

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

$$\min \quad ||w||^2 + C\sum_{n=1}^{N} \xi_i$$

$$\text{s.t.} \quad t_n\left(w^T\phi(x_n) + b\right) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

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

$\xi_n \geqslant 0$

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

$\mu_n \geqslant 0$

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

- KKT conditions for the constraints

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

original primal const.

KKT. complementary slackness

Dual formulation of soft-margin SVM

- Taking derivatives

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

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

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

$$\mu_n \geqslant 0$$

$$\Rightarrow \boxed{0 \leq a_n \leq C} \qquad \text{c.f. } \boxed{a_n \geqslant 0.}$$

Dual formulation of soft-margin SVM

- Lagrange dual

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

a_n

subject to

$$\boxed{0 \leqslant a_n \leqslant C}$$

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

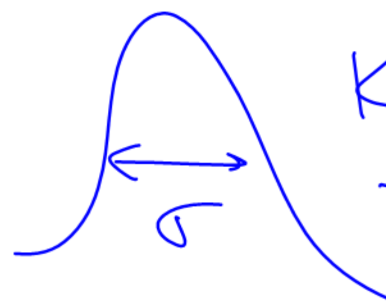
- Solve quadratic problem (convex optimization)

Support Vector Machine: Algorithm

- 1. Choose a kernel function { linear, gaussian, poly...
- 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



$$K(x_n, x_m) = \exp\left[-\frac{\|x_n - x_m\|^2}{2\sigma^2}\right]$$

- 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.

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