Kernel Methods and Support Vector Machines (SVMs)

CMSC 678

UMBC

Slides courtesy Hamed Pirsiavash

Outline

Recap

Kernel Methods & Feature Mapping

Support Vector Machines (SVMs)

Math: Subgradients

Math: Lagrange Multipliers for Constrained Optimization

SVMs Revisited: Dual Formulation

Recap from last time…

K-Means

Initialize k centers by picking k points randomly among all the points

Repeat till convergence (or max iterations)

Assign each point to the nearest center (assignment step)

Estimate the mean of each group (update step)

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} ||\mathbf{x} - \mu_i||^2$$

k-means++ algorithm for initialization:

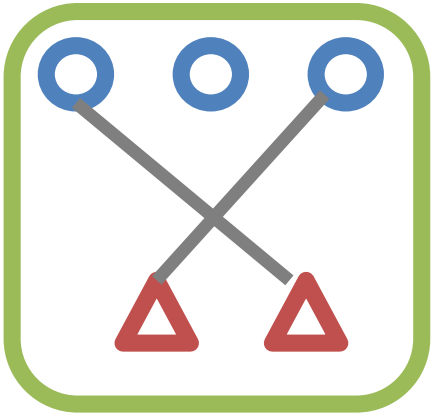
1. Chose one center uniformly at random among all the points
2. For each point **x**, compute D(**x**), the distance between x and the nearest center that has already been chosen
3. Chose one new data point at random as a new center, using a weighted probability distribution where a point **x** is chosen with a probability proportional to D(**x**)^2
4. Repeat Steps 2 and 3 until k centers have been chosen

https://www.csee.umbc.edu/courses/graduate/678/spring18/kmeans/

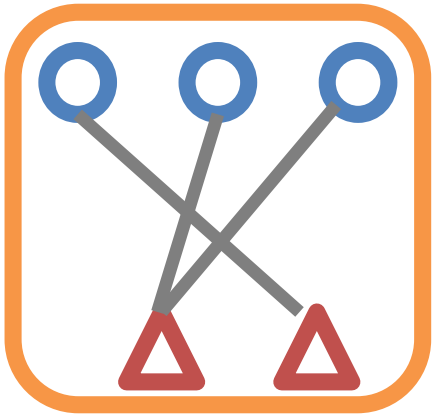
Clustering Evaluation

(Classification: accuracy, recall, precision, F-score)

Greedy mapping: one-to-one



Optimistic mapping: many-to-one



Rigorous/information theoretic: V-measure

homogeneity

$$= \begin{cases} 1, & H(K,C) = 0 \\ 1 - \dfrac{H(C|K)}{H(K)}, & \text{o/w} \end{cases}$$

completeness

$$= \begin{cases} 1, & H(K,C) = 0 \\ 1 - \dfrac{H(K|C)}{H(C)}, & \text{o/w} \end{cases}$$

Hierarchical clustering

Agglomerative: a "bottom up" approach where elements start as individual clusters and clusters are merged as one moves up the hierarchy

Divisive: a "top down" approach where elements start as a single cluster and clusters are split as one moves down the hierarchy

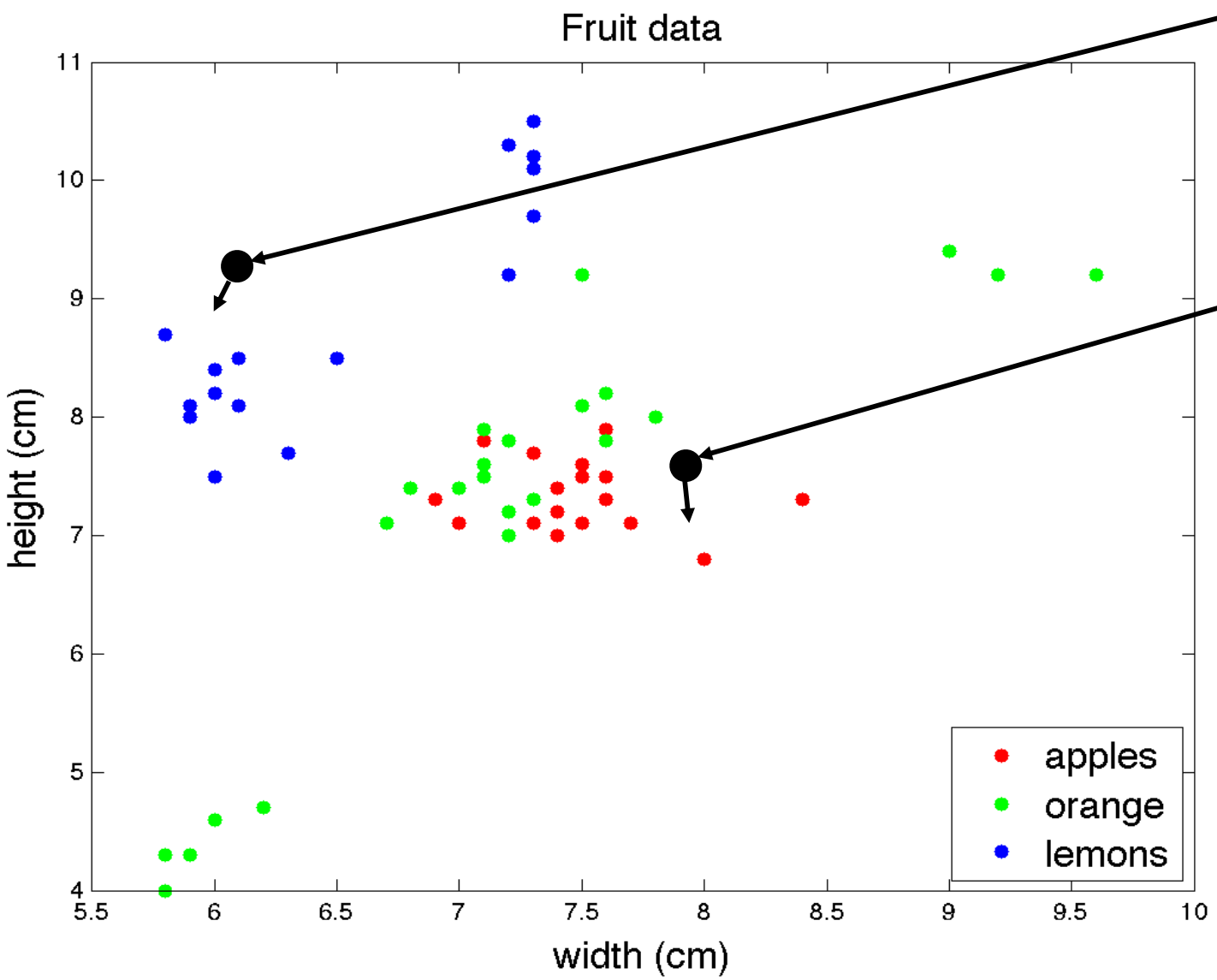
Agglomerative clustering:

- First merge very similar instances
- Incrementally build larger clusters out of smaller clusters

Algorithm:

- Maintain a set of clusters
- Initially, each instance in its own cluster
- Repeat:
  - Pick the two "closest" clusters
  - Merge them into a new cluster
  - Stop when there's only one cluster left

K-Nearest neighbor predictor



(a, b) ?
lemon

(c, d) ?
apple

test data

Take majority vote among the k nearest neighbors

Outline

Recap

Kernel Methods & Feature Mapping

Support Vector Machines (SVMs)

Math: Subgradients

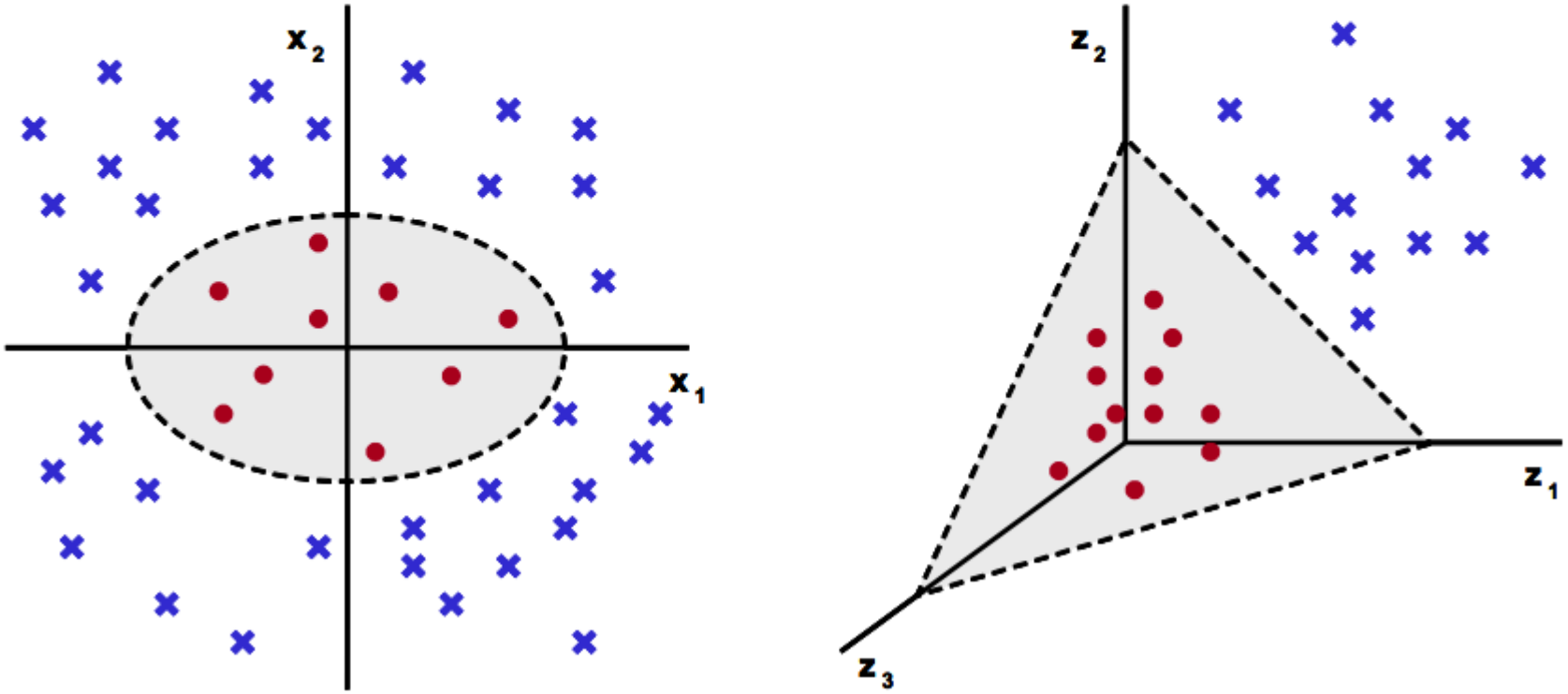
Math: Lagrange Multipliers for Constrained Optimization

SVMs Revisited: Dual Formulation

Feature mapping

$$\varphi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$



Learn non-linear classifiers by mapping features

Quadratic feature map

$$\mathbf{x} = [x_1, x_2, \ldots, x_D]$$

$$\begin{aligned}\phi(\mathbf{x}) = [&1, \sqrt{2}x_1, \sqrt{2}x_2, \ldots, \sqrt{2}x_D, \\ &x_1^2, x_1x_2, x_1x_3 \ldots, x_1x_D, \\ &x_2x_1, x_2^2, x_2x_3 \ldots, x_2x_D, \\ &\ldots, \\ &x_Dx_1, x_Dx_2, x_Dx_3 \ldots, x_D^2]\end{aligned}$$

Contains all single and pairwise terms

Quadratic kernel

The dot product between feature maps of **x** and **z** is:

$$\begin{aligned}
\phi(\mathbf{x})^T\phi(\mathbf{z}) &= 1 + 2x_1z_1 + 2x_2z_2, \ldots, 2x_Dz_D + x_1^2z_1^2 + x_1x_2z_1z_2 + \ldots + x_1x_Dz_1z_D + \ldots \\
&\quad \ldots + x_Dx_1z_Dz_1 + x_Dx_2z_Dz_2 + \ldots + x_D^2z_D^2 \\
&= 1 + 2\left(\sum_i x_iz_i\right) + \sum_{i,j} x_ix_jz_iz_j \\
&= 1 + 2\left(\mathbf{x}^T\mathbf{z}\right) + \left(\mathbf{x}^T\mathbf{z}\right)^2 \\
&= \left(1 + \mathbf{x}^T\mathbf{z}\right)^2 \\
&= K(\mathbf{x}, \mathbf{z}) \qquad \longleftarrow \text{quadratic kernel}
\end{aligned}$$

Compute $\phi(\mathbf{x})^T\phi(\mathbf{z})$ in almost the same time as needed to compute $\mathbf{x}^T\mathbf{z}$ (one extra addition and multiplication)

We will rewrite various algorithms using only dot products (or kernel evaluations), and not explicit features

Feature mapping

$$K(x,z) = (x^T z)^2$$

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_i z_i\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) \end{aligned}$$

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix}$$

Feature mapping

$$K(x,z) = (x^T z + c)^2$$

$$= \sum_i x_i^2 z_i^2 + 2\sum_i \sum_{j}^{i-1} x_i x_j z_i z_j + \sum_i 2c x_i z_i + c^2$$

$$\phi(x) = \begin{pmatrix} x_1 x_1 \\ x_1 x\_2 \\ \dots \\ x_N x_N \\ x_1\sqrt{2c} \\ \dots \\ x_N\sqrt{2c} \\ c \end{pmatrix}$$

Drawbacks of feature mapping

Computational

- Suppose training time is linear in feature dimension, quadratic feature map squares the training time

Memory

- Quadratic feature map squares the memory required to store the training data

Statistical

- Quadratic feature mapping squares the number of parameters
- For now lets assume that regularization will deal with overfitting

Perceptron revisited

Input: training data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$

feature map ϕ

Initialize $\mathbf{w} \leftarrow [0, \ldots, 0]$

for iter = 1,…,T

for i = 1,..,n

predict according to the current model

$$\hat{y}_i = \begin{cases} +1 & \text{if } \mathbf{w}^T \phi(\mathbf{x}_i) > 0 \\ -1 & \text{otherwise} \end{cases}$$

dependence on φ

if $y_i = \hat{y}_i$ no change

else, $\mathbf{w} \leftarrow \mathbf{w} + y_i \phi(\mathbf{x}_i)$

Obtained by replacing **x** by $\phi(\mathbf{x})$

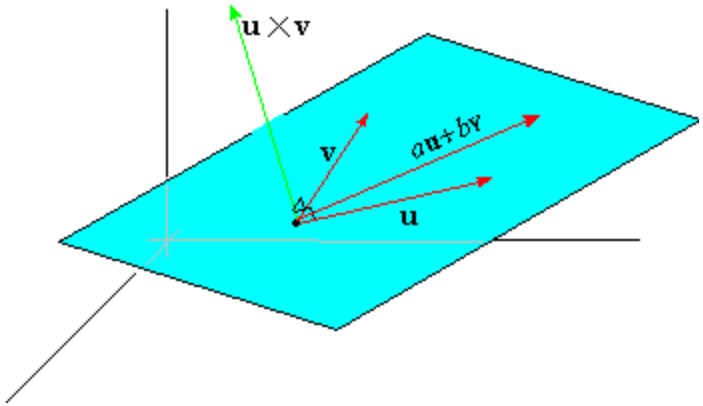
Properties of the weight vector

Linear algebra recap:

Let U be set of vectors in R^D, i.e., U = $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_D\}$ and $\mathbf{u}_i \in R^D$

Span(U) is the set of all vectors that can be represented as $\sum_i a_i \mathbf{u}_i$, such that $a_i \in R$

Null(U) is everything that is left i.e., $R^D \setminus$ Span(U)



Perceptron representer theorem: During the run of the perceptron training algorithm, the weight vector **w** is always in the span of $\phi(\mathbf{x}_1)$, $\phi(\mathbf{x}_1)$, …, $\phi(\mathbf{x}_D)$

$$\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i) \qquad \text{updates} \qquad \alpha_i \leftarrow \alpha_i + y_i$$

$$\mathbf{w}^T \phi(\mathbf{z}) = \left(\sum_i \alpha_i \phi(\mathbf{x}_i)\right)^T \phi(\mathbf{z}) = \sum_i \alpha_i \phi(\mathbf{x}_i)^T \phi(\mathbf{z})$$

Perceptron Representer Theorem

Perceptron representer theorem: During the run of the perceptron training algorithm, the weight vector **w** is always in the span of $\phi(\mathbf{x}_1)$, $\phi(\mathbf{x}_1)$, …, $\phi(\mathbf{x}_D)$

$$\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i) \qquad \text{updates} \qquad \alpha_i \leftarrow \alpha_i + y_i$$

$$\mathbf{w}^T \phi(\mathbf{z}) = \left(\sum_i \alpha_i \phi(\mathbf{x}_i)\right)^T \phi(\mathbf{z}) = \sum_i \alpha_i \phi(\mathbf{x}_i)^T \phi(\mathbf{z})$$

Why? Weights are always computed from the training instance representations

Example 2: Incorrect → $w \ += \ y_2 \phi(x_2)$

Example 3: Correct → $w \ += 0 * y_3 \phi(x_3)$

Example 4: Incorrect → $w \ += \ y_4 \phi(x_4)$

Kernelized perceptron

Input: training data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$

feature map ϕ

Kernelized perceptron training algorithm

Initialize $\alpha \leftarrow [0, 0, \ldots, 0]$

for iter = 1,…,T

for i = 1,..,n

predict according to the current model

$$\hat{y}_i = \begin{cases} +1 & \text{if } \sum_n \alpha_n \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_i) > 0 \\ -1 & \text{otherwise} \end{cases}$$

if $y_i = \hat{y}_i$, no change

else, $\alpha_i = \alpha_i + y_i$

$\phi(\mathbf{x})^T \phi(\mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^p$ polynomial kernel of degree p

Kernelized perceptron

Input: training data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$

feature map ϕ

Kernelized perceptron training algorithm

Initialize $\alpha \leftarrow [0, 0, \ldots, 0]$

for iter = 1,…,T

for i = 1,..,n

predict according to the current model

$$\hat{y}_i = \begin{cases} +1 & \text{if } \sum_n \alpha_n \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_i) > 0 \\ -1 & \text{otherwise} \end{cases}$$

if $y_i = \hat{y}_i$, no change

else, $\alpha_i = \alpha_i + y_i$

Upshot: Feature expansion for "free"

$\phi(\mathbf{x})^T \phi(\mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^p$ polynomial kernel of degree p

Kernel k-means

Repeat till convergence (or max iterations)

1. Assign each point to the nearest center (assignment step)
2. Estimate the mean of each group (update step)

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} ||\phi(\mathbf{x}) - \mu_i||^2$$

Representer theorem is easy in (2)

$$\mu_i \leftarrow \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \phi(\mathbf{x})$$

What about (1)? How to compute using dot products

$$||\phi(\mathbf{x}) - \mu_i||^2$$

Kernel k-means

$$z_n = \arg\min_k \left|\left| \phi(\boldsymbol{x}_n) - \boldsymbol{\mu}^{(k)} \right|\right|^2 \qquad \text{definition of } z_n$$

$$= \arg\min_k \left|\left| \phi(\boldsymbol{x}_n) - \sum_m \alpha_m^{(k)} \phi(\boldsymbol{x}_m) \right|\right|^2 \qquad \text{definition of } \boldsymbol{\mu}^{(k)}$$

$$= \arg\min_k ||\phi(\boldsymbol{x}_n)||^2 + \left|\left| \sum_m \alpha_m^{(k)} \phi(\boldsymbol{x}_m) \right|\right|^2 + \phi(\boldsymbol{x}_n) \cdot \left[\sum_m \alpha_m^{(k)} \phi(\boldsymbol{x}_m) \right] \qquad \text{expand quadratic term}$$

$$= \arg\min_k \sum_m \sum_{m'} \alpha_m^{(k)} \alpha_{m'}^{(k)} \phi(\boldsymbol{x}_m) \cdot \phi(\boldsymbol{x}_{m'}) + \sum_m \alpha_m^{(k)} \phi(\boldsymbol{x}_m) \cdot \phi(\boldsymbol{x}_n) + const \qquad \text{linearity and constant}$$

Kernel Definition

A kernel is a mapping ***K:*** X xX →R

Functions that can be written as

dot products are valid kernels

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

Examples: polynomial kernel

$$K^d_{(\text{poly})}(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^d$$

Kernel Definition

A kernel is a mapping ***K:*** X xX →R

Functions that can be written as

dot products are valid kernels

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

Examples: polynomial kernel

$$K^d_{(\text{poly})}(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^d$$

Alternatively:

Mercer's Conditions: A function ***K:*** X xX →R is a kernel if ***K*** is positive semi-definite (psd)

$$\int f(\mathbf{x})^2 d\mathbf{x} < \infty$$

This means that for all functions f that are squared integrable except the zero function, the following property holds:

$$\int \int f(\mathbf{x}) K(\mathbf{x}, \mathbf{z}) f(\mathbf{z}) d\mathbf{z} d\mathbf{x} > 0$$

Why is this characterization useful?

We can prove some properties about kernels that are otherwise hard to prove

Theorem: If K_1 and K_2 are kernels, then $K_1 + K_2$ is also a kernel

Proof:

$$\begin{aligned}\int\int f(\mathbf{x})K(\mathbf{x},\mathbf{z})f(\mathbf{z})d\mathbf{z}d\mathbf{x} &= \int\int f(\mathbf{x})\left(K_1(\mathbf{x},\mathbf{z})+K_2(\mathbf{x},\mathbf{z})\right)f(\mathbf{z})d\mathbf{z}d\mathbf{x} \\ &= \int\int f(\mathbf{x})K_1(\mathbf{x},\mathbf{z})f(\mathbf{z})d\mathbf{z}d\mathbf{x} + \int\int f(\mathbf{x})K_2(\mathbf{x},\mathbf{z})f(\mathbf{z})d\mathbf{z}d\mathbf{x} \\ &\geq 0+0\end{aligned}$$

More generally if K_1, K_2,…, K_n are kernels then $\sum_i \alpha_i K_i$ with $\alpha_i \geq 0$, is also a kernel

Can build new kernels by linearly combining existing kernels

Kernels in practice

Feature mapping via kernels often improves performance

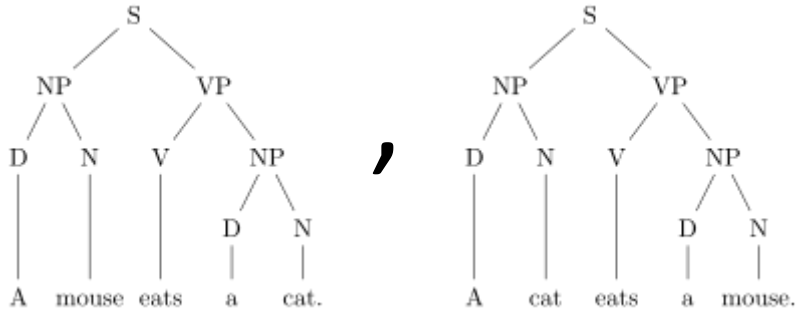
MNIST digits test error:

- **8.4%** SVM linear
- **1.4%** SVM RBF
- **1.1%** SVM polynomial (d=4)

http://yann.lecun.com/exdb/mnist/

Kernels over general structures

Kernels can be defined over any pair of inputs such as strings, trees and graphs!

K (

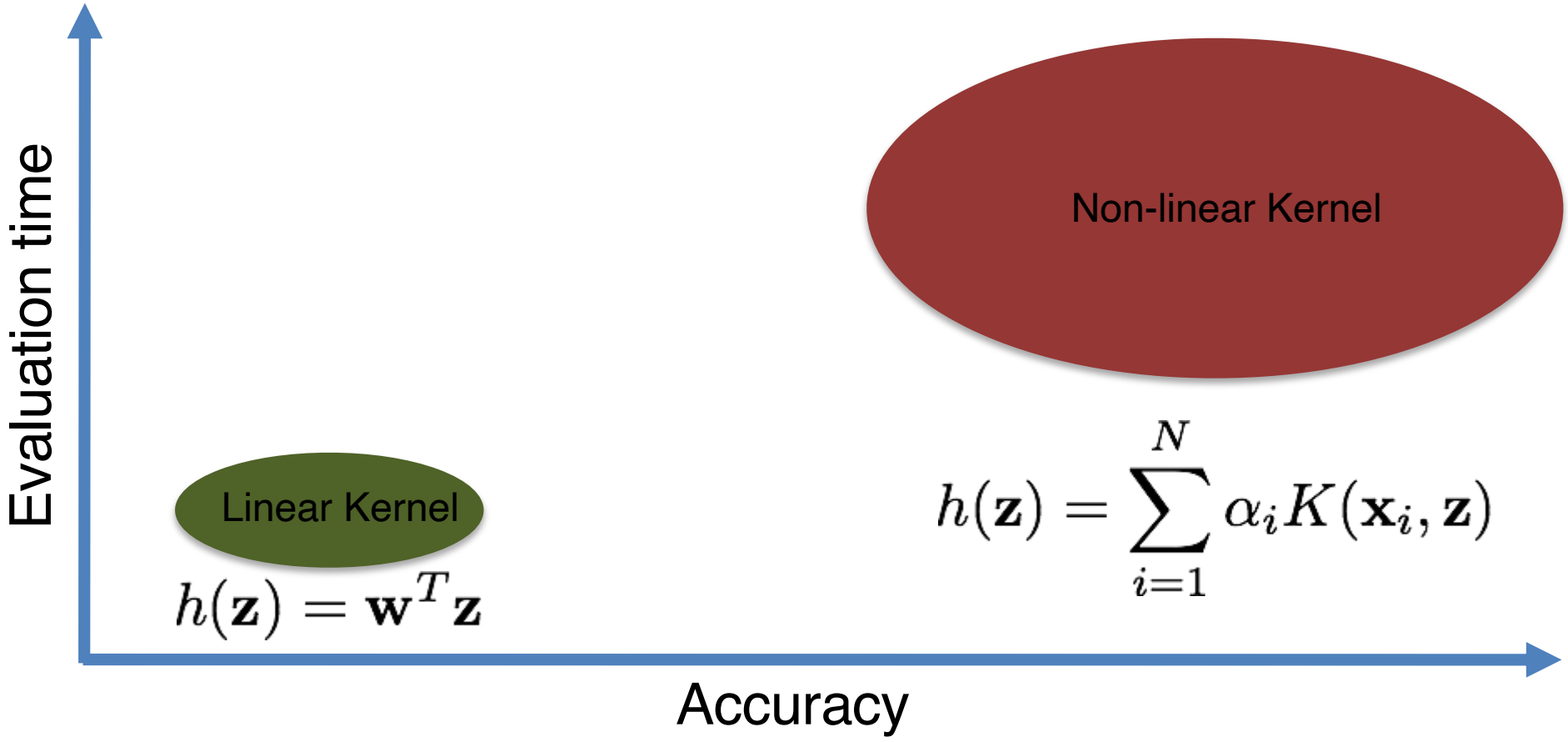
,) = number of common subtrees

http://en.wikipedia.org/wiki/Tree_kernel

For strings number of common substrings is a kernel

Graph kernels that measure graph similarity (e.g. number of common subgraphs) have been used to predict toxicity of chemical structures

Kernel classifiers tradeoffs



Linear: O (feature dimension)
Non Linear: O (**N** X feature dimension)

Outline

Recap

Kernel Methods & Feature Mapping

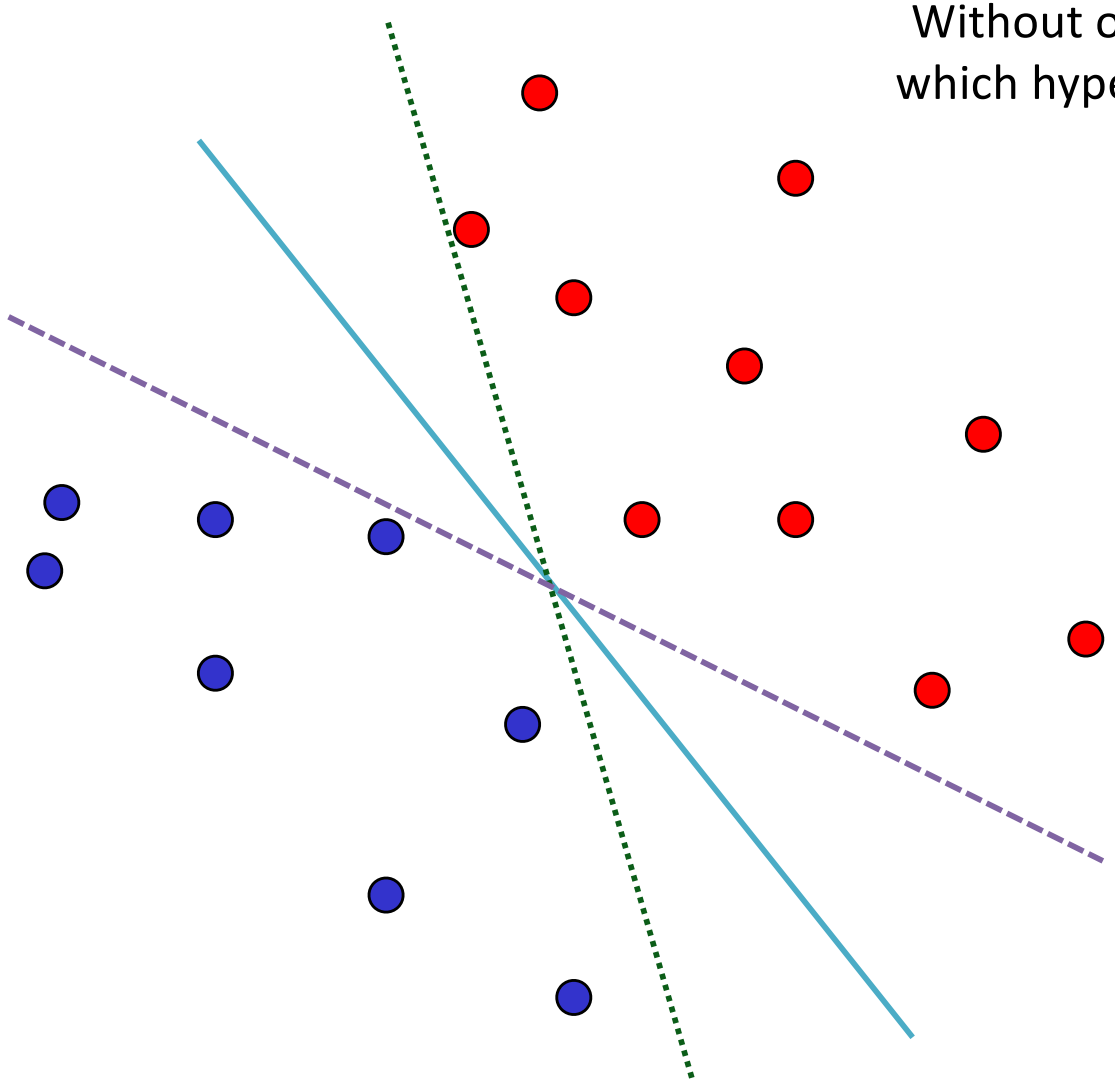
Support Vector Machines (SVMs)

Math: Subgradients

Math: Lagrange Multipliers for Constrained Optimization

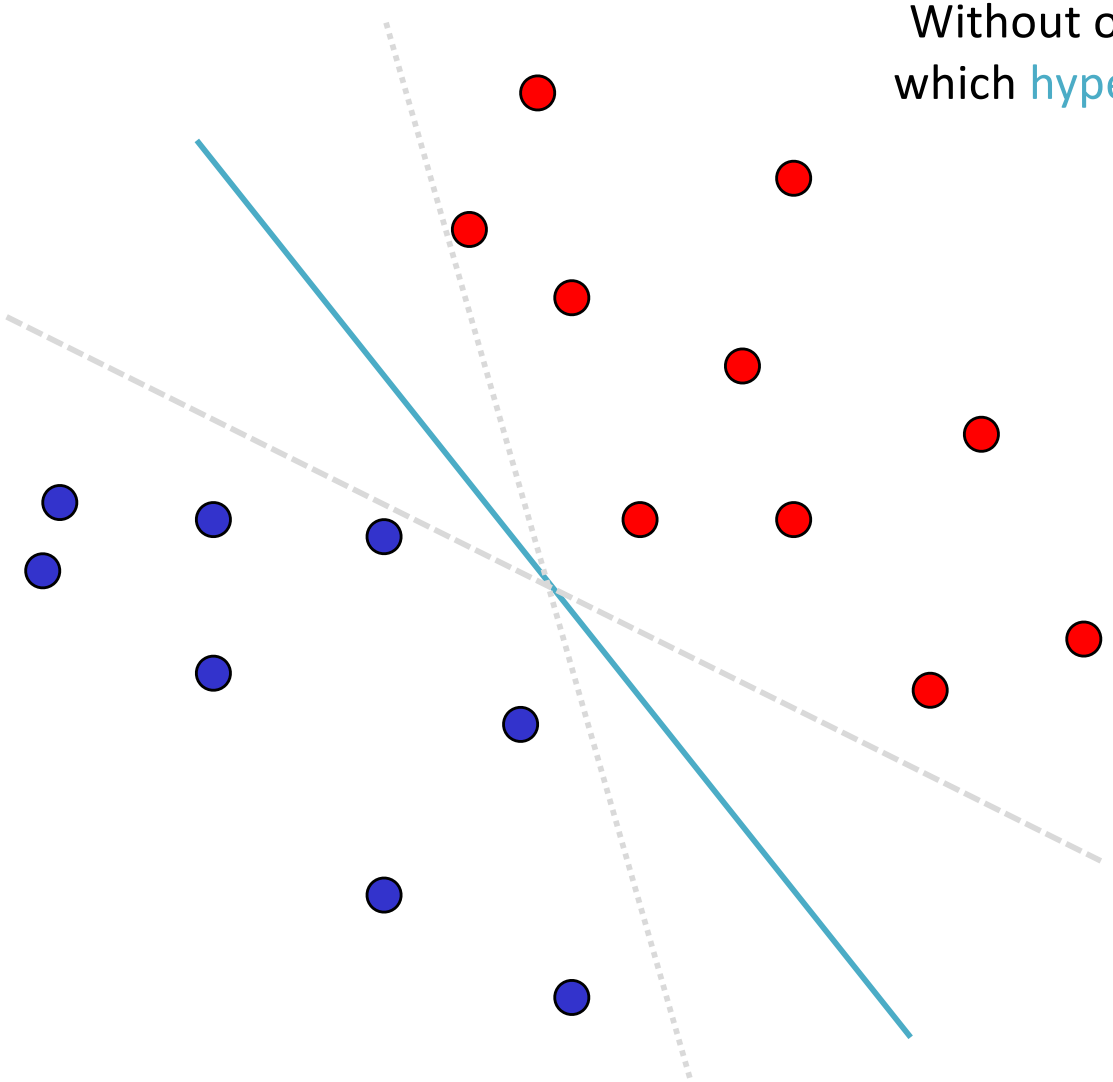
SVMs Revisited: Dual Formulation

Picking a good hyperplane



Without other information, which hyperplane would you pick?

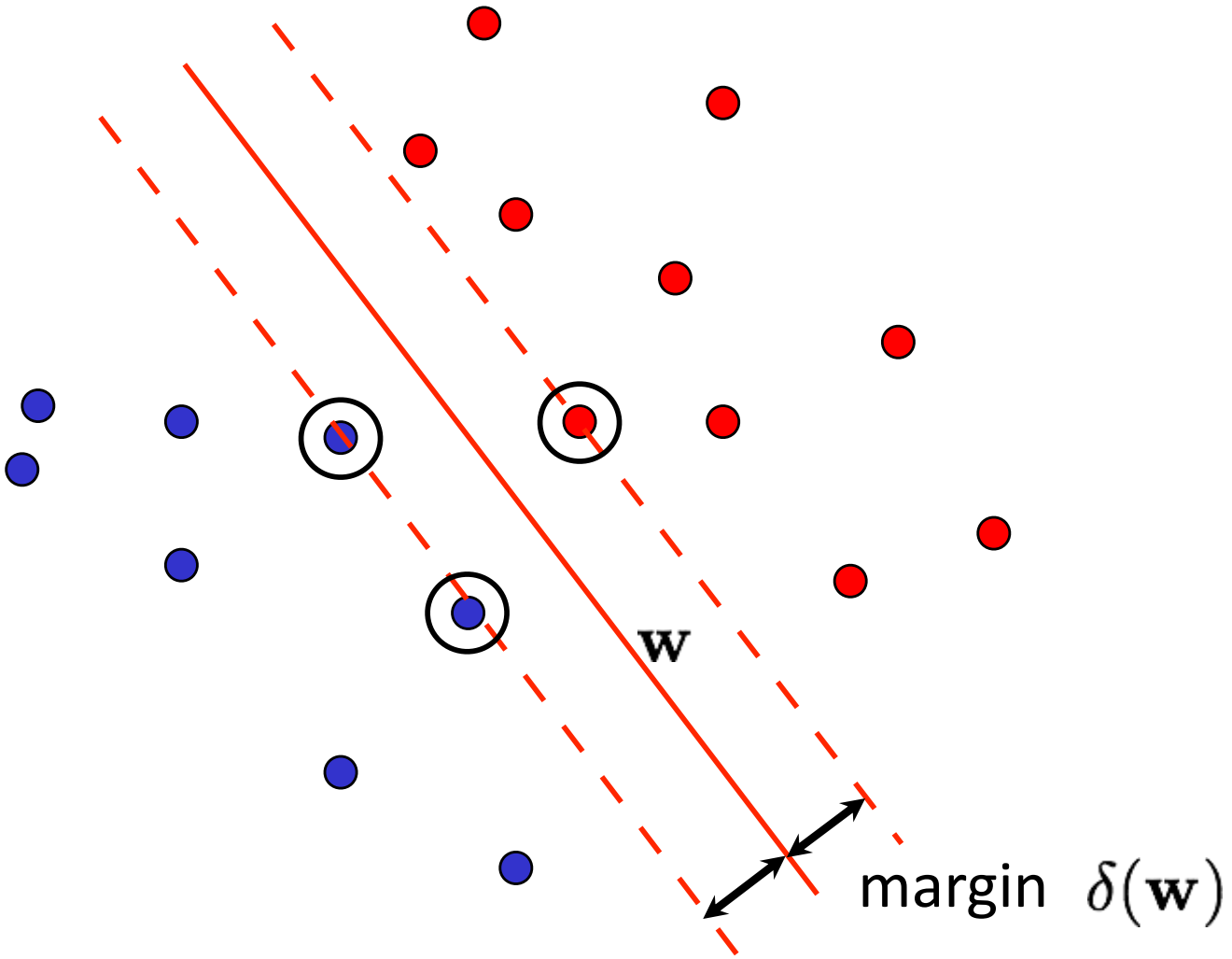
Picking a good hyperplane



Without other information, which hyperplane would you pick?

Support Vector Machines (SVMs)

Maximize the distance to the nearest point (margin),
while correctly classifying all the points



Optimization for SVMs

Separable case: hard margin SVM

$$\min_{\mathbf{w}} \frac{1}{\delta(\mathbf{w})}$$

maximize margin

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1, \forall n$$

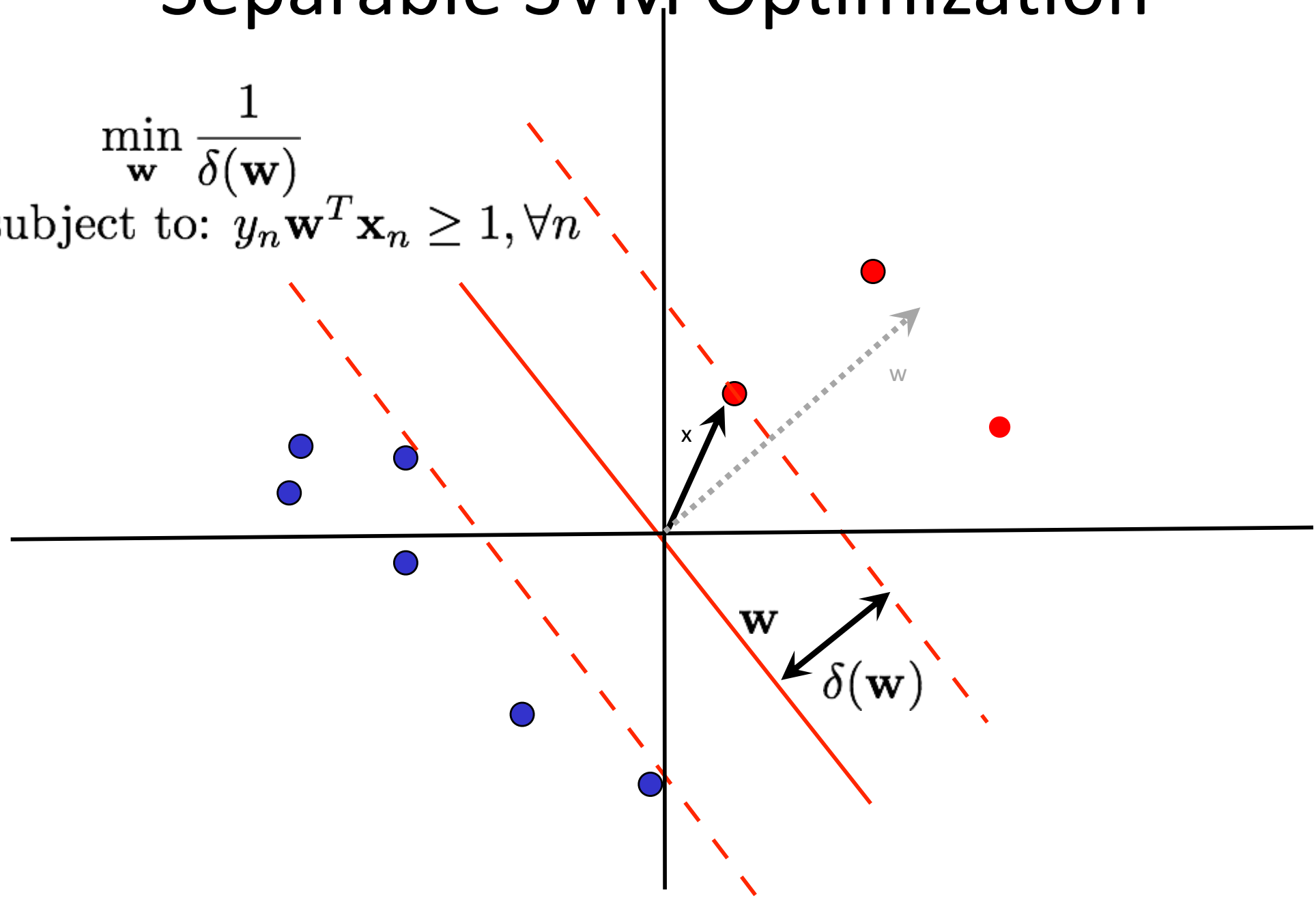
(assuming y = +1 or y = -1)

separate by a non-trivial margin

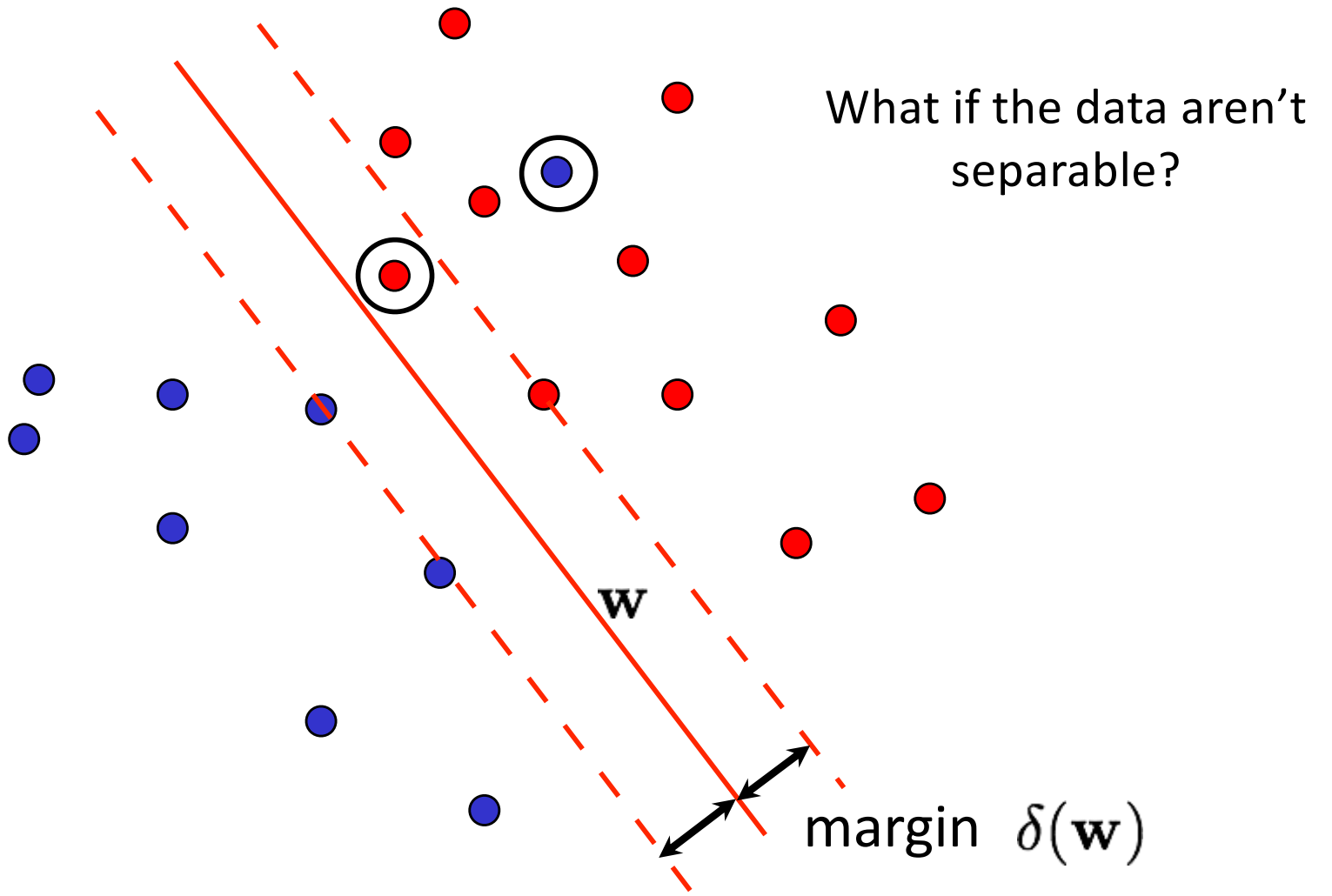
Separable SVM Optimization

$$\min_{\mathbf{w}} \frac{1}{\delta(\mathbf{w})}$$

subject to: $y_n \mathbf{w}^T \mathbf{x}_n \geq 1, \forall n$



Support Vector Machines (SVMs)



Optimization for SVMs

Separable case: hard margin SVM

$$\min_{\mathbf{w}} \frac{1}{\delta(\mathbf{w})}$$

maximize margin

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1, \forall n$$

separate by a non-trivial margin

Non-separable case: soft margin SVM

$$\min_{\mathbf{w}} \frac{1}{\delta(\mathbf{w})} + C \sum_n \xi_n$$

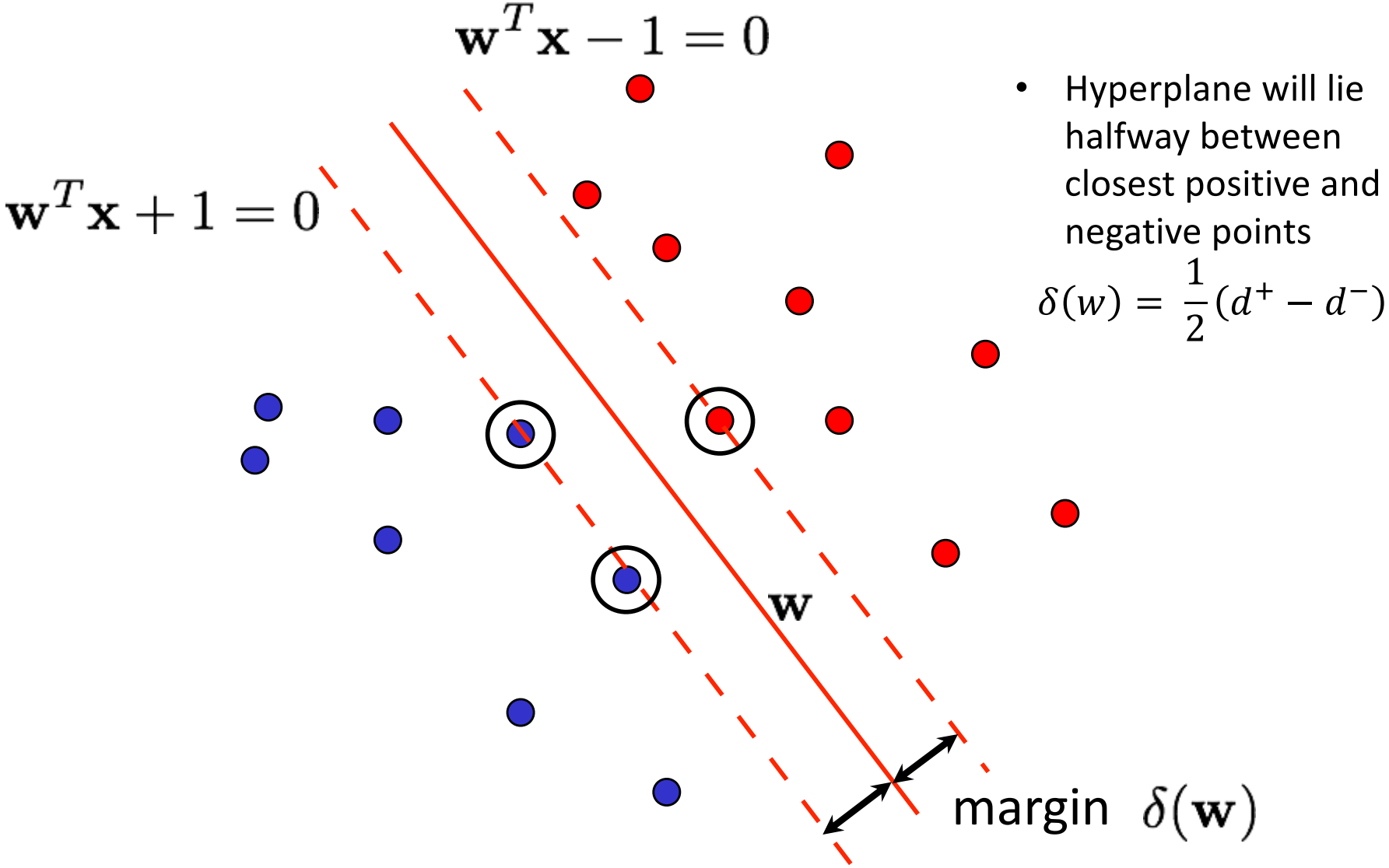
maximize margin minimize slack

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1 - \xi_n, \forall n$$

allow some slack

$$\xi_n \geq 0$$

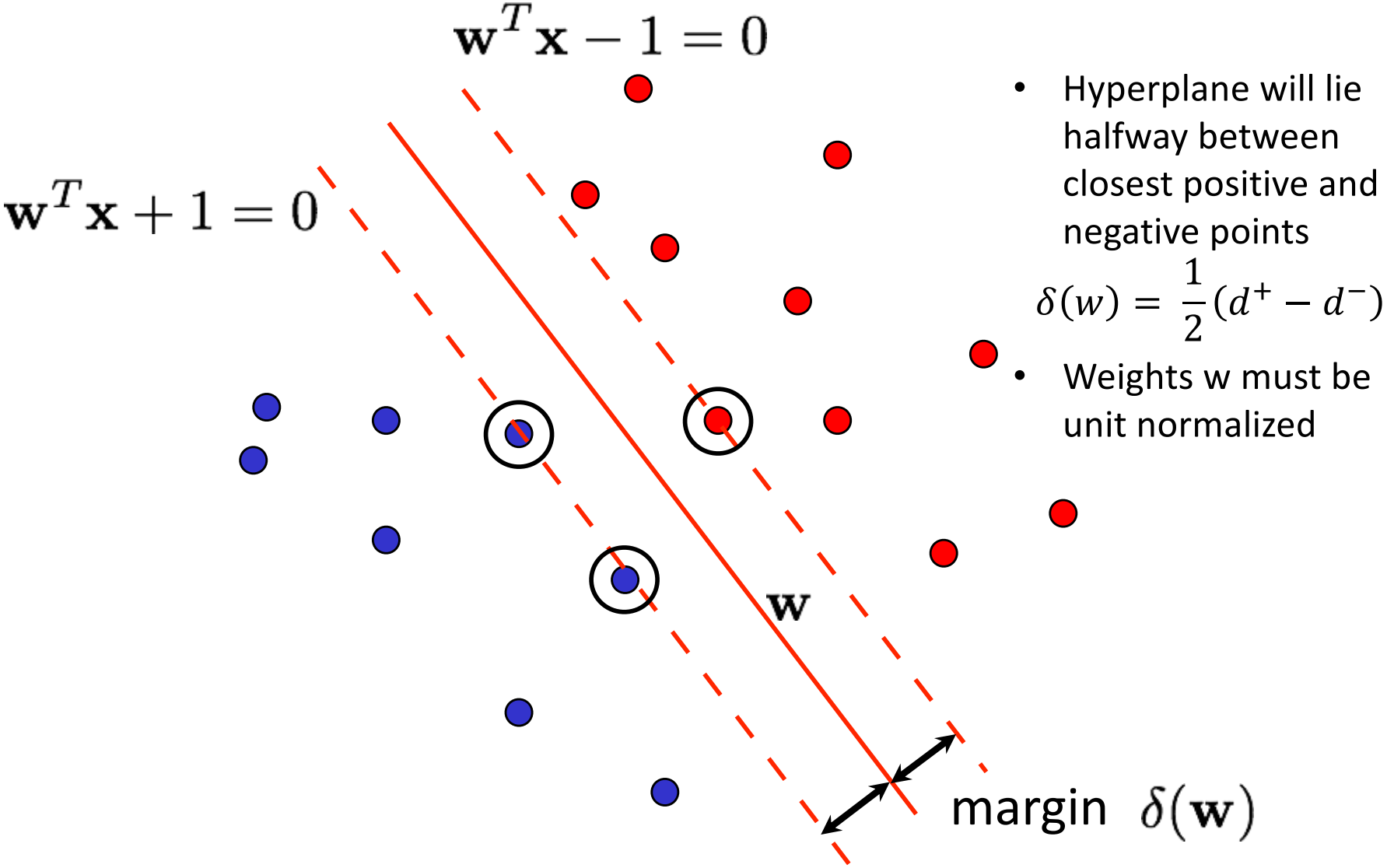
Maximizing Margin = Minimizing norm



- Hyperplane will lie halfway between closest positive and negative points

$$\delta(w) = \frac{1}{2}(d^+ - d^-)$$

Maximizing Margin = Minimizing norm

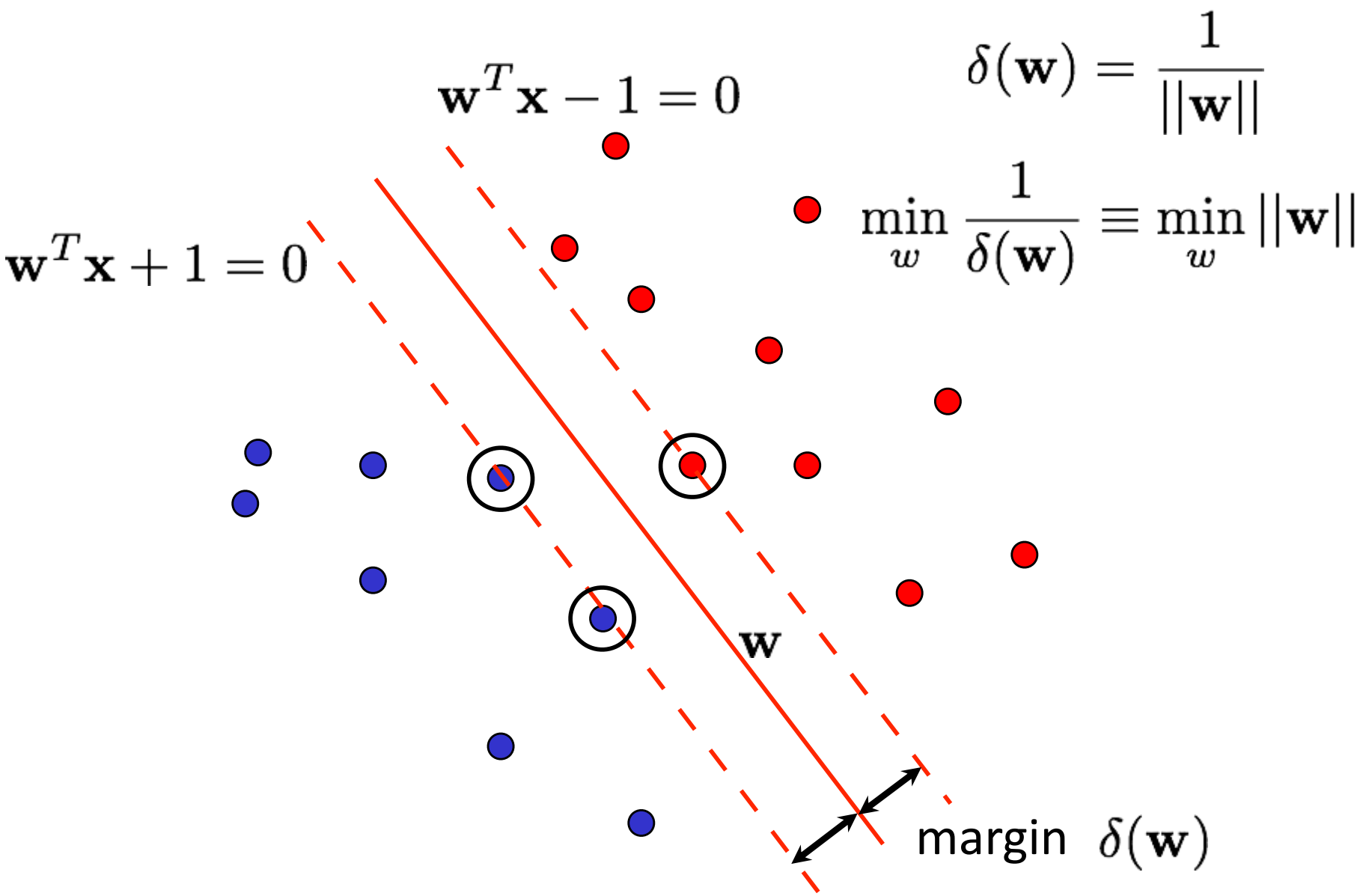


- Hyperplane will lie halfway between closest positive and negative points

$$\delta(w) = \frac{1}{2}(d^+ - d^-)$$

- Weights w must be unit normalized

Maximizing Margin = Minimizing norm



$$\delta(\mathbf{w}) = \frac{1}{||\mathbf{w}||}$$

$$\min_w \frac{1}{\delta(\mathbf{w})} \equiv \min_w ||\mathbf{w}||$$

Equivalent optimization for SVMs

Separable case: hard margin SVM

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2$$

maximize margin

squaring and half for convenience

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1, \forall n$$

separate by a non-trivial margin

Non-separable case: soft margin SVM

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C \sum_n \xi_n$$

maximize margin minimize slack

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1 - \xi_n, \forall n$$

allow some slack

$$\xi_n \geq 0$$

Slack variables

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_n \xi_n$$

soft margin SVM

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1 - \xi_n, \forall n$$

$$\xi_n \geq 0$$

Given weights + bias, can you derive the optimal slack for the n^{th} example?

Slack variables

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_n \xi_n$$

soft margin SVM

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1 - \xi_n, \forall n$$

$$\xi_n \geq 0$$

Given weights + bias, can you derive the optimal slack for the n^{th} example?

$y_n \mathbf{w}^T \mathbf{x}_n$ = 0.8, ξ_n = ?

$y_n \mathbf{w}^T \mathbf{x}_n$ = -1 , ξ_n = ?

$y_n \mathbf{w}^T \mathbf{x}_n$ = 2.5, ξ_n = ?

Slack variables

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_n \xi_n$$

soft margin SVM

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1 - \xi_n, \forall n$$

$$\xi_n \geq 0$$

Given weights + bias, can you derive the optimal slack for the n^{th} example?

$y_n \mathbf{w}^T \mathbf{x}_n$ = 0.8, ξ_n = ? 0.2

$y_n \mathbf{w}^T \mathbf{x}_n$ = -1 , ξ_n = ? 2.0

$y_n \mathbf{w}^T \mathbf{x}_n$ = 2.5, ξ_n = ? 0

Slack variables

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_n \xi_n$$

soft margin SVM

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1 - \xi_n, \forall n$$

$$\xi_n \geq 0$$

Given weights + bias, can you derive the optimal slack for the n^{th} example?

$y_n \mathbf{w}^T \mathbf{x}_n$ = 0.8, ξ_n = ? 0.2

$y_n \mathbf{w}^T \mathbf{x}_n$ = -1 , ξ_n = ? 2.0

$y_n \mathbf{w}^T \mathbf{x}_n$ = 2.5, ξ_n = ? 0

$$\xi_n = \begin{cases} 0 & y_n \mathbf{w}^T \mathbf{x}_n \geq 1 \\ 1 - y_n \mathbf{w}^T \mathbf{x}_n & \text{otherwise} \end{cases}$$

Slack variables

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_n \xi_n$$

soft margin SVM

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1 - \xi_n, \forall n$$

$$\xi_n \geq 0$$

Given weights + bias, can you derive the optimal slack for the n^{th} example?

$y_n \mathbf{w}^T \mathbf{x}_n$ = 0.8, ξ_n = ? 0.2

$y_n \mathbf{w}^T \mathbf{x}_n$ = -1 , ξ_n = ? 2.0

$y_n \mathbf{w}^T \mathbf{x}_n$ = 2.5, ξ_n = ? 0

$$\xi_n = \begin{cases} 0 & y_n \mathbf{w}^T \mathbf{x}_n \geq 1 \\ 1 - y_n \mathbf{w}^T \mathbf{x}_n & \text{otherwise} \end{cases}$$

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_n \max(0, 1 - y_n \mathbf{w}^T \mathbf{x}_n)$$

Same as hinge loss with squared norm regularization!

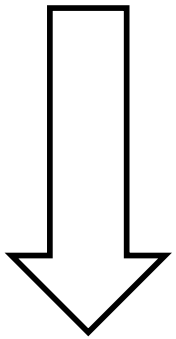
Slack variables

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_n \xi_n$$

soft margin SVM

$$\text{subject to: } y_n \mathbf{w}^T \mathbf{x}_n \geq 1 - \xi_n, \forall n$$

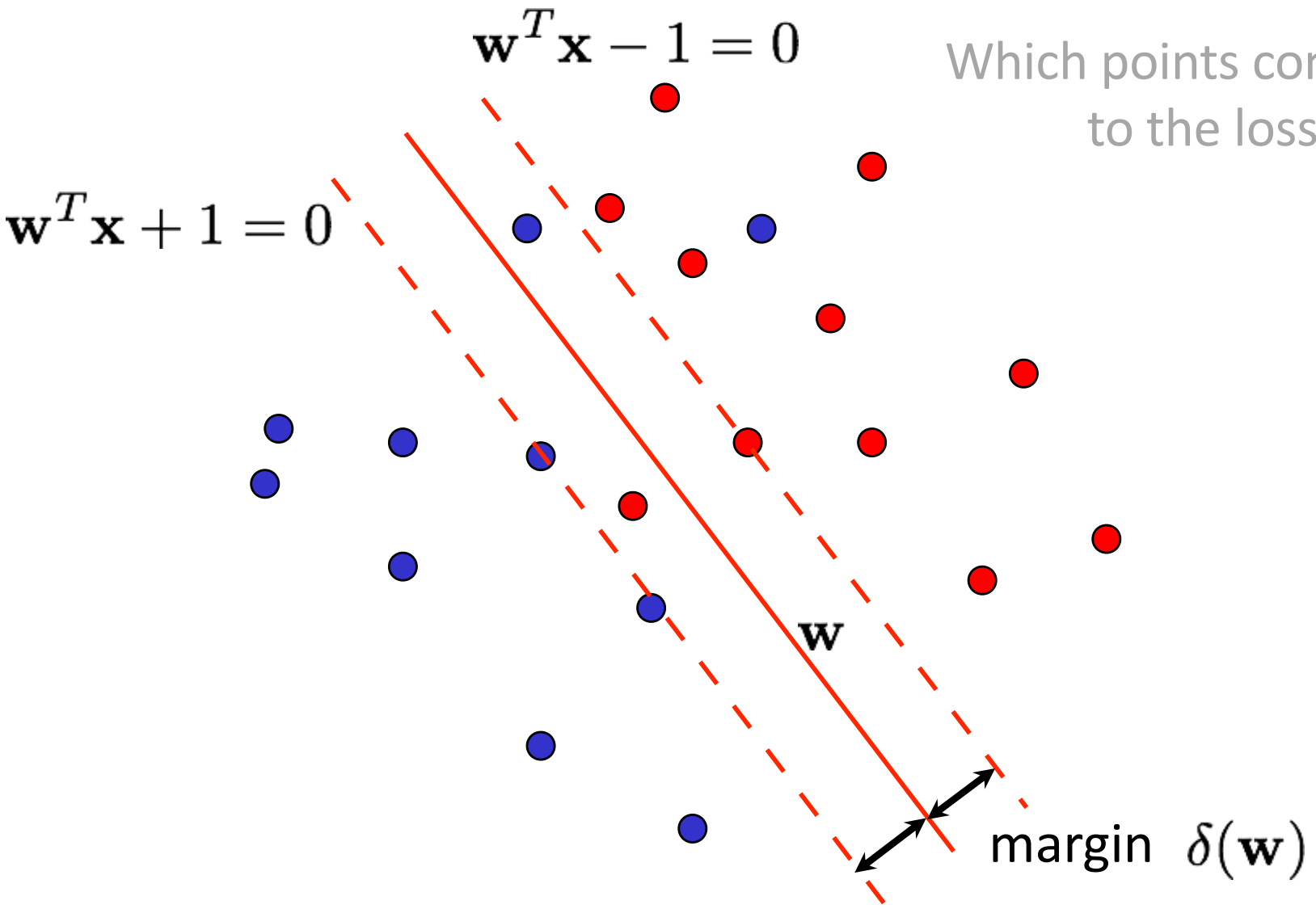
$$\xi_n \geq 0$$



$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_n \max(0, 1 - y_n \mathbf{w}^T \mathbf{x}_n)$$

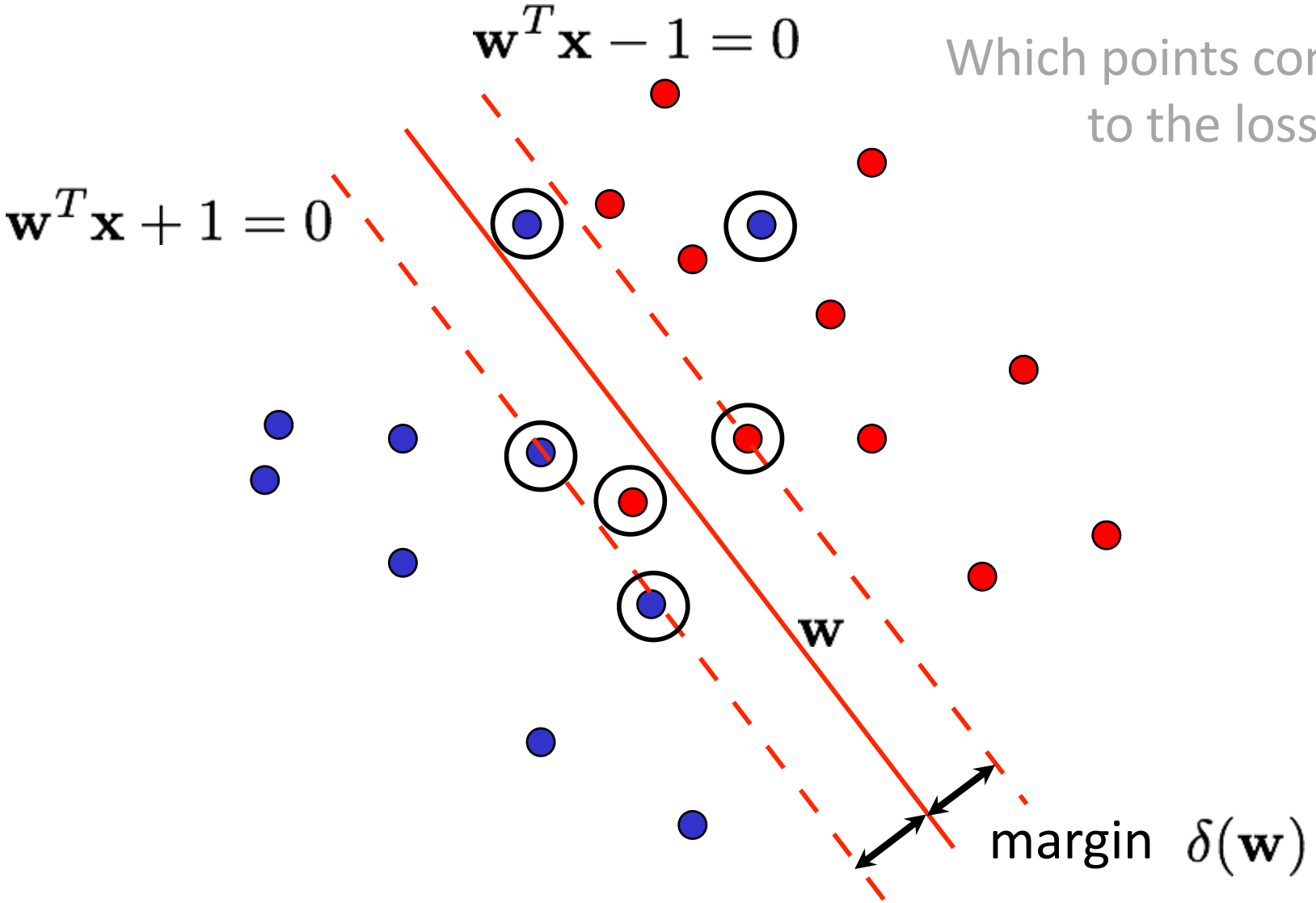
Same as hinge loss with squared norm regularization!

Support vectors



Which points contribute to the loss?

Support vectors



Which points contribute to the loss?

Optimization for linear models

Under suitable conditions*, provided you pick the step sizes appropriately, the convergence rate of gradient descent is O(1/N)

i.e., if you want a solution within 0.001 of the optimal you have to run the gradient descent for N=1000 iterations.

For linear models (hinge/logistic/exponential loss) and squared-norm regularization there are off-the-shelf solvers that are fast in practice: SVM^{perf}, LIBLINEAR, PEGASOS

SVM^{perf}, LIBLINEAR use a different optimization method

* the function is strongly convex: $f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{m}{2} \|y - x\|_2^2$

Outline

Recap

Kernel Methods & Feature Mapping

Support Vector Machines (SVMs)

Math: Subgradients

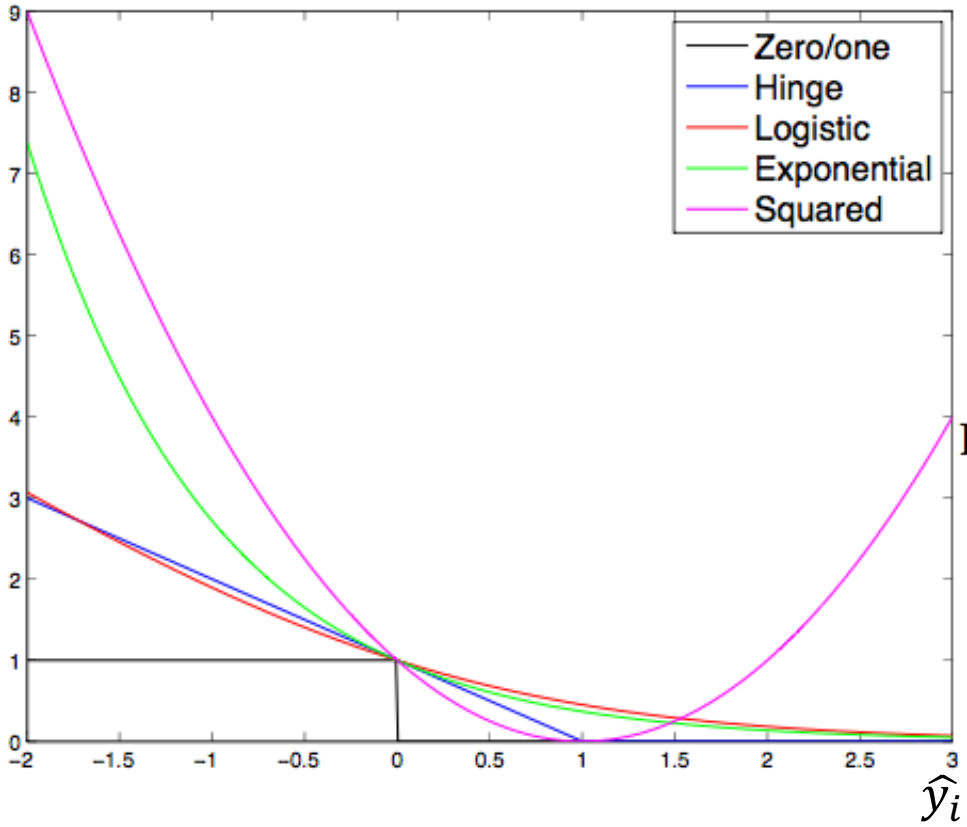
Math: Lagrange Multipliers for Constrained Optimization

SVMs Revisited: Dual Formulation

Reprise from earlier in the semester: Convex surrogate loss functions

Surrogate loss: replace Zero/one loss by a smooth function

Easier to optimize if the surrogate loss is convex



$$y = +1 \qquad \hat{y} \leftarrow \mathbf{w}^T \mathbf{x}$$

$$\text{Zero/one:} \quad \ell^{(0/1)}(y, \hat{y}) = \mathbf{1}[y\hat{y} \leq 0]$$

$$\text{Hinge:} \quad \ell^{(\text{hin})}(y, \hat{y}) = \max\{0, 1 - y\hat{y}\}$$

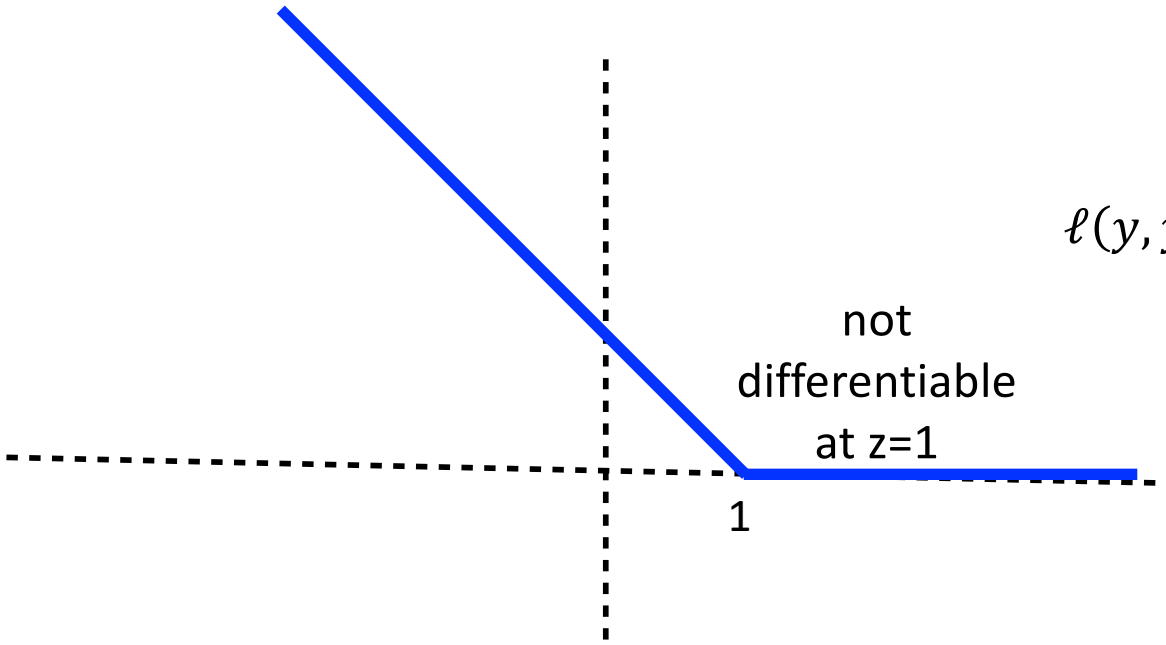
$$\text{Logistic:} \quad \ell^{(\log)}(y, \hat{y}) = \frac{1}{\log 2} \log\left(1 + \exp[-y\hat{y}]\right)$$

$$\text{Exponential:} \quad \ell^{(\exp)}(y, \hat{y}) = \exp[-y\hat{y}]$$

$$\text{Squared:} \quad \ell^{(\text{sqr})}(y, \hat{y}) = (y - \hat{y})^2$$

Courtesy Hamed Pirsiavash, CIML

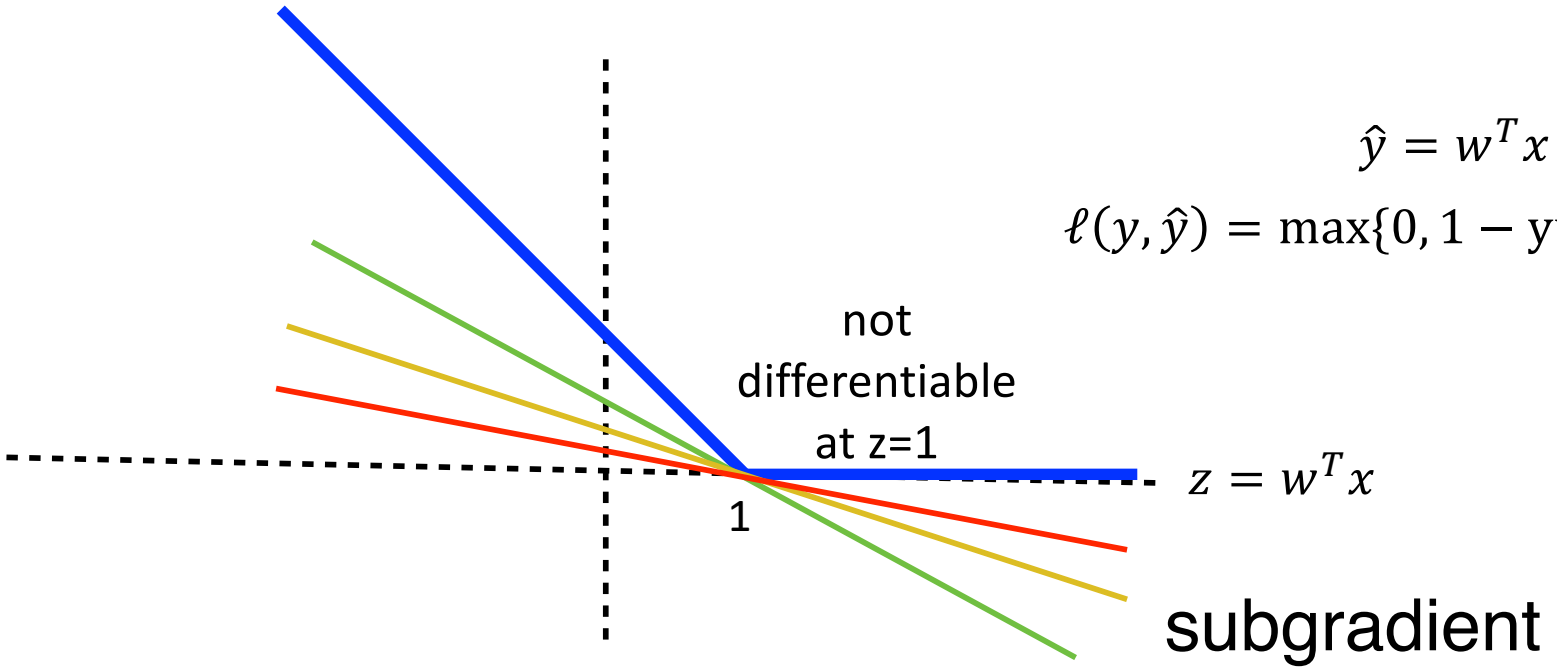
Gradient of Hinge Loss?



$$\hat{y} = w^T x$$

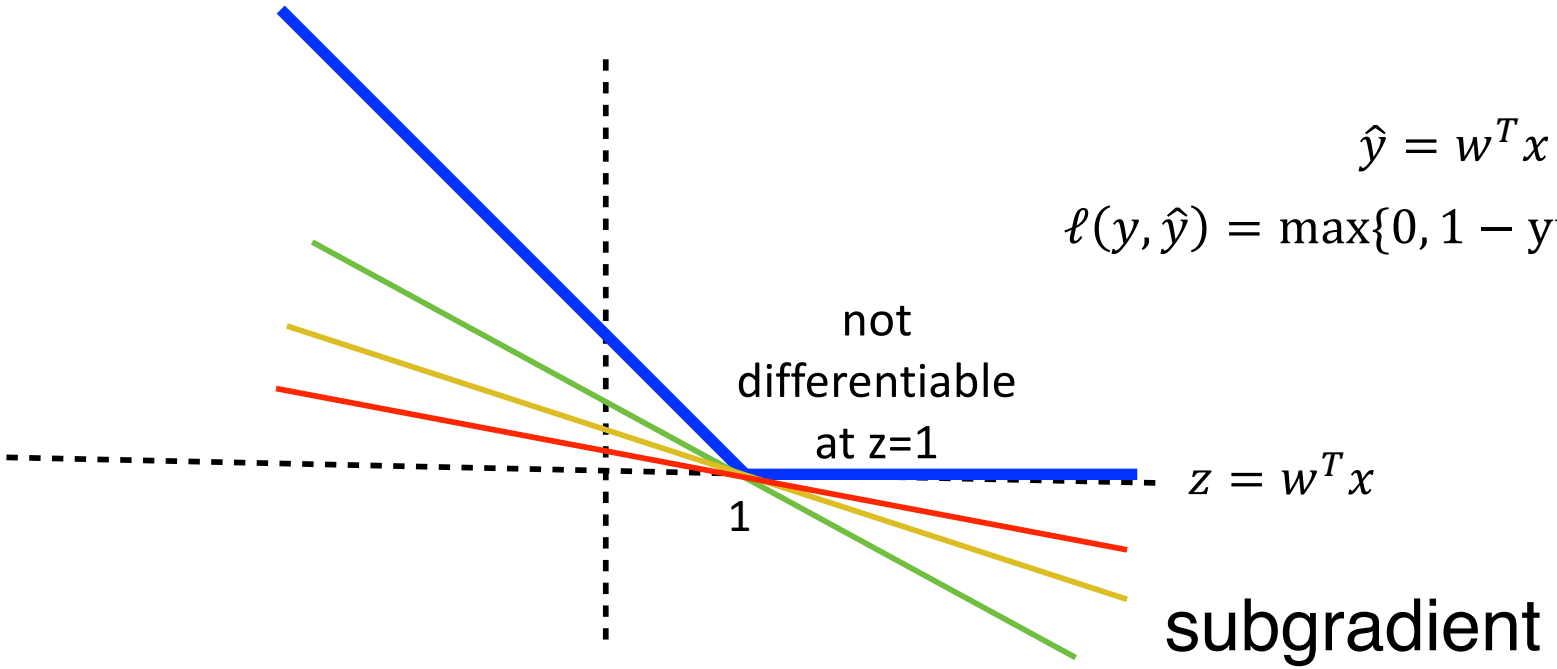
$$\ell(y, \hat{y}) = \max\{0, 1 - yw^T x\}$$

*Sub*gradient of Hinge Los



Subgradient is any direction that is below the function

*Sub*gradient of Hinge Los



Subgradient is any direction that is below the function

For the hinge loss a possible subgradient is:

$$\frac{d\ell^{\text{hinge}}}{d\mathbf{w}} = \begin{cases} 0 & \text{if } y\mathbf{w}^T\mathbf{x} > 1 \\ -y\mathbf{x} & \text{otherwise} \end{cases}$$

Example: Hinge loss

$$\mathcal{L}(\mathbf{w}) = \sum_n \max(0, 1 - y_n \mathbf{w}^T \mathbf{x}_n) + \frac{\lambda}{2}||\mathbf{w}||^2$$ objective

Example: Hinge loss

$$\mathcal{L}(\mathbf{w}) = \sum_n \max(0, 1 - y_n \mathbf{w}^T \mathbf{x}_n) + \frac{\lambda}{2}||\mathbf{w}||^2$$ objective

$$\frac{d\mathcal{L}}{d\mathbf{w}} = \sum_n -\mathbf{1}[y_n \mathbf{w}^T \mathbf{x}_n \leq 1] y_n \mathbf{x}_n + \lambda \mathbf{w}$$ subgradient

Example: Hinge loss

$$\mathcal{L}(\mathbf{w}) = \sum_n \max(0, 1 - y_n \mathbf{w}^T \mathbf{x}_n) + \frac{\lambda}{2}||\mathbf{w}||^2$$ objective

$$\frac{d\mathcal{L}}{d\mathbf{w}} = \sum_n -\mathbf{1}[y_n \mathbf{w}^T \mathbf{x}_n \leq 1] y_n \mathbf{x}_n + \lambda \mathbf{w}$$ subgradient

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \left(\sum_n -\mathbf{1}[y_n \mathbf{w}^T \mathbf{x}_n \leq 1] y_n \mathbf{x}_n + \lambda \mathbf{w} \right)$$ update

Example: Hinge loss

$$\mathcal{L}(\mathbf{w}) = \sum_n \max(0, 1 - y_n \mathbf{w}^T \mathbf{x}_n) + \frac{\lambda}{2}||\mathbf{w}||^2$$ objective

$$\frac{d\mathcal{L}}{d\mathbf{w}} = \sum_n -\mathbf{1}[y_n \mathbf{w}^T \mathbf{x}_n \leq 1] y_n \mathbf{x}_n + \lambda \mathbf{w}$$ subgradient

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \left(\sum_n -\mathbf{1}[y_n \mathbf{w}^T \mathbf{x}_n \leq 1] y_n \mathbf{x}_n + \lambda \mathbf{w} \right)$$ update

loss term

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y_n \mathbf{x}_n$$

only for points $y_n \mathbf{w}^T \mathbf{x}_n \leq 1$

perceptron update $y_n \mathbf{w}^T \mathbf{x}_n \leq 0$

regularization term

$$\mathbf{w} \leftarrow (1 - \eta\lambda)\mathbf{w}$$

shrinks weights towards zero

Outline

Recap

Kernel Methods & Feature Mapping

Support Vector Machines (SVMs)

Math: Subgradients

Math: Lagrange Multipliers for Constrained Optimization

SVMs Revisited: Dual Formulation

Lagrange multipliers

Assume an original optimization problem

$$\min_{x} \quad x^2$$

$$st. \quad x \geq b$$

Lagrange multipliers

Assume an original optimization problem

$$\min_{x} \quad x^2$$
$$st. \quad x \geq b$$

We convert it to a new optimization problem:

$$\min_{x} \max_{\alpha} \quad L(x, \alpha)$$
$$st. \quad \alpha \geq 0$$
$$L(x, \alpha) = x^2 - \alpha(x - b)$$

Lagrange multipliers: an equivalent problem?

$$\min_x \quad x^2$$
$$st. \quad x \geq b$$

$$\min_x \max_\alpha \quad L(x, \alpha)$$
$$st. \quad \alpha \geq 0$$
$$L(x, \alpha) = x^2 - \alpha(x - b)$$

$$x < b \quad \Rightarrow \quad (x - b) < 0 \quad \Rightarrow \quad \max_\alpha -\alpha(x - b) = \inf$$

Lagrange multipliers: an equivalent problem?

$$\min_x \quad x^2$$
$$st. \quad x \geq b$$

$$\min_x \max_\alpha \quad L(x, \alpha)$$
$$st. \quad \alpha \geq 0$$
$$L(x, \alpha) = x^2 - \alpha(x - b)$$

$$x < b \quad \Rightarrow \quad (x - b) < 0 \quad \Rightarrow \quad \max_\alpha -\alpha(x - b) = \inf$$
$$x > b \quad \Rightarrow \quad (x - b) > 0 \quad \Rightarrow \quad \max_\alpha -\alpha(x - b) = 0, \alpha = 0 \quad \Rightarrow \quad L(x, \alpha) = x^2$$

Lagrange multipliers: an equivalent problem?

$$\min_x \quad x^2$$
$$st. \quad x \geq b$$

$$\min_x \max_\alpha \quad L(x, \alpha)$$
$$st. \quad \alpha \geq 0$$
$$L(x, \alpha) = x^2 - \alpha(x - b)$$

$$x < b \quad \Rightarrow \quad (x - b) < 0 \quad \Rightarrow \quad \max_\alpha -\alpha(x - b) = \inf$$
$$x > b \quad \Rightarrow \quad (x - b) > 0 \quad \Rightarrow \quad \max_\alpha -\alpha(x - b) = 0, \alpha = 0 \quad \Rightarrow \quad L(x, \alpha) = x^2$$
$$x = b \quad \Rightarrow \quad (x - b) = 0 \quad \Rightarrow \quad \alpha = \text{any thing} \quad \Rightarrow \quad L(x, \alpha) = x^2$$

Outline

Recap

Kernel Methods & Feature Mapping

Support Vector Machines (SVMs)

Math: Subgradients

Math: Lagrange Multipliers for Constrained Optimization

SVMs Revisited: Dual Formulation

Dual formulation for SVM

SVM problem:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 + C\sum_n \xi_n$$

$$\text{subj. to} \quad y_n\left(\boldsymbol{w}\cdot\boldsymbol{x}_n + b\right) \geq 1 - \xi_n$$

$$\xi_n \geq 0$$

Dual formulation for SVM

SVM problem:

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

$$\text{subj. to} \quad y_n\,(w \cdot x_n + b) \geq 1 - \xi_n$$

$$\xi_n \geq 0$$

With Lagrange multipliers:

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2}\,||w||^2 + C\sum_n \xi_n - \sum_n \beta_n \xi_n$$
$$- \sum_n \alpha_n \left[y_n\,(w \cdot x_n + b) - 1 + \xi_n\right]$$

Dual formulation for SVM

SVM problem:

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

$$\text{subj. to} \quad y_n\left(w \cdot x_n + b\right) \geq 1 - \xi_n$$

$$\xi_n \geq 0$$

With Lagrange multipliers:

$$\min_{w,b,\xi} \max_{\alpha \geq 0} \max_{\beta \geq 0} \mathcal{L}(w, b, \xi, \alpha, \beta)$$

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2}||w||^2 + C\sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n\left(w \cdot x_n + b\right) - 1 + \xi_n\right]$$

Dual formulation for SVM

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \max_{\boldsymbol{\alpha}\geq 0} \max_{\boldsymbol{\beta}\geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\,||\boldsymbol{w}||^2 + C\sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n\left(\boldsymbol{w}\cdot\boldsymbol{x}_n + b\right) - 1 + \xi_n\right]$$

1. minimize wrt. **w**:

$$\nabla_w \mathcal{L} = \boldsymbol{w} - \sum_n \alpha_n y_n \boldsymbol{x}_n = 0 \quad \Longleftrightarrow \quad \boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{x}_n$$

Is this form familiar?

Dual formulation for SVM

$$\min_{w,b,\xi} \max_{\alpha \geq 0} \max_{\beta \geq 0} \mathcal{L}(w, b, \xi, \alpha, \beta)$$

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2} ||w||^2 + C \sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n \left(w \cdot x_n + b \right) - 1 + \xi_n \right]$$

2. substitute **w**:

$$\mathcal{L}(b, \xi, \alpha, \beta) = \frac{1}{2} \left\| \sum_m \alpha_m y_m x_m \right\|^2 + C \sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n \left(\left[\sum_m \alpha_m y_m x_m \right] \cdot x_n + b \right) - 1 + \xi_n \right]$$

Dual formulation for SVM

$$\min_{w,b,\xi} \max_{\alpha \geq 0} \max_{\beta \geq 0} \mathcal{L}(w, b, \xi, \alpha, \beta)$$

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2} ||w||^2 + C \sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n \left(w \cdot x_n + b \right) - 1 + \xi_n \right]$$

2. substitute **w**:

$$\mathcal{L}(b, \xi, \alpha, \beta) = \frac{1}{2} \left\| \sum_m \alpha_m y_m x_m \right\|^2 + C \sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n \left(\left[\sum_m \alpha_m y_m x_m \right] \cdot x_n + b \right) - 1 + \xi_n \right]$$

(feature) expand this out further

Dual formulation for SVM

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \max_{\boldsymbol{\alpha}\geq 0} \max_{\boldsymbol{\beta}\geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\,||\boldsymbol{w}||^2 + C\sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n\left(\boldsymbol{w}\cdot\boldsymbol{x}_n + b\right) - 1 + \xi_n\right]$$

3. minimize wrt **b**:

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_n \alpha_n y_n = 0$$

Dual formulation for SVM

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \max_{\boldsymbol{\alpha}\geq 0} \max_{\boldsymbol{\beta}\geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n \left(\boldsymbol{w}\cdot\boldsymbol{x}_n + b\right) - 1 + \xi_n\right]$$

3. minimize wrt **b**:

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_n \alpha_n y_n = 0$$

at optima, this term is zero (eliminated)

Dual formulation for SVM

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \max_{\boldsymbol{\alpha}\geq 0} \max_{\boldsymbol{\beta}\geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\,||\boldsymbol{w}||^2 + C\sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n\left(\boldsymbol{w}\cdot\boldsymbol{x}_n + b\right) - 1 + \xi_n\right]$$

4. minimize wrt $\boldsymbol{\xi_n}$:

$$\frac{\partial \mathcal{L}}{\partial \xi_n} = C - \beta_n - \alpha_n \quad \Longleftrightarrow \quad C - \beta_n = \alpha_n$$

Dual formulation for SVM

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \max_{\boldsymbol{\alpha}\geq 0} \max_{\boldsymbol{\beta}\geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\,||\boldsymbol{w}||^2 + C\sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n\left(\boldsymbol{w}\cdot\boldsymbol{x}_n + b\right) - 1 + \xi_n\right]$$

5. do algebra & rewrite, with expansion from **w**

$$-\frac{1}{2}\sum_n\sum_m \alpha_n\alpha_m y_n y_m \boldsymbol{x}_n\cdot\boldsymbol{x}_m + \sum_n (C-\beta_n)\xi_n - b\sum_n \alpha_n y_n - \sum_n \alpha_n(\xi_n - 1)$$

Dual formulation for SVM

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \max_{\boldsymbol{\alpha}\geq 0} \max_{\boldsymbol{\beta}\geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\,||\boldsymbol{w}||^2 + C\sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n\left(\boldsymbol{w}\cdot\boldsymbol{x}_n + b\right) - 1 + \xi_n\right]$$

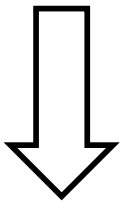
6. rewrite with kernel

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_n \alpha_n - \frac{1}{2}\sum_n\sum_m \alpha_n \alpha_m y_n y_m K(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

Dual formulation for SVM

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \max_{\boldsymbol{\alpha}\geq 0} \max_{\boldsymbol{\beta}\geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\,||\boldsymbol{w}||^2 + C\sum_n \xi_n - \sum_n \beta_n \xi_n - \sum_n \alpha_n \left[y_n\left(\boldsymbol{w}\cdot\boldsymbol{x}_n + b\right) - 1 + \xi_n\right]$$



$$\min_{\boldsymbol{\alpha}} \quad -\mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_n\sum_m \alpha_n\alpha_m y_n y_m K(\boldsymbol{x}_n, \boldsymbol{x}_m) - \sum_n \alpha_n$$

$$\text{subj. to} \quad 0 \leq \alpha_n \leq C$$