CS446 Introduction to Machine Learning (Spring 2015)
University of Illinois at Urbana-Champaign
http://courses.engr.illinois.edu/cs446

LECTURE 10:
LARGE MARGIN CLASSIFIERS

Prof. Julia Hockenmaier
juliahmr@illinois.edu

Today's class

Large margin classifiers:

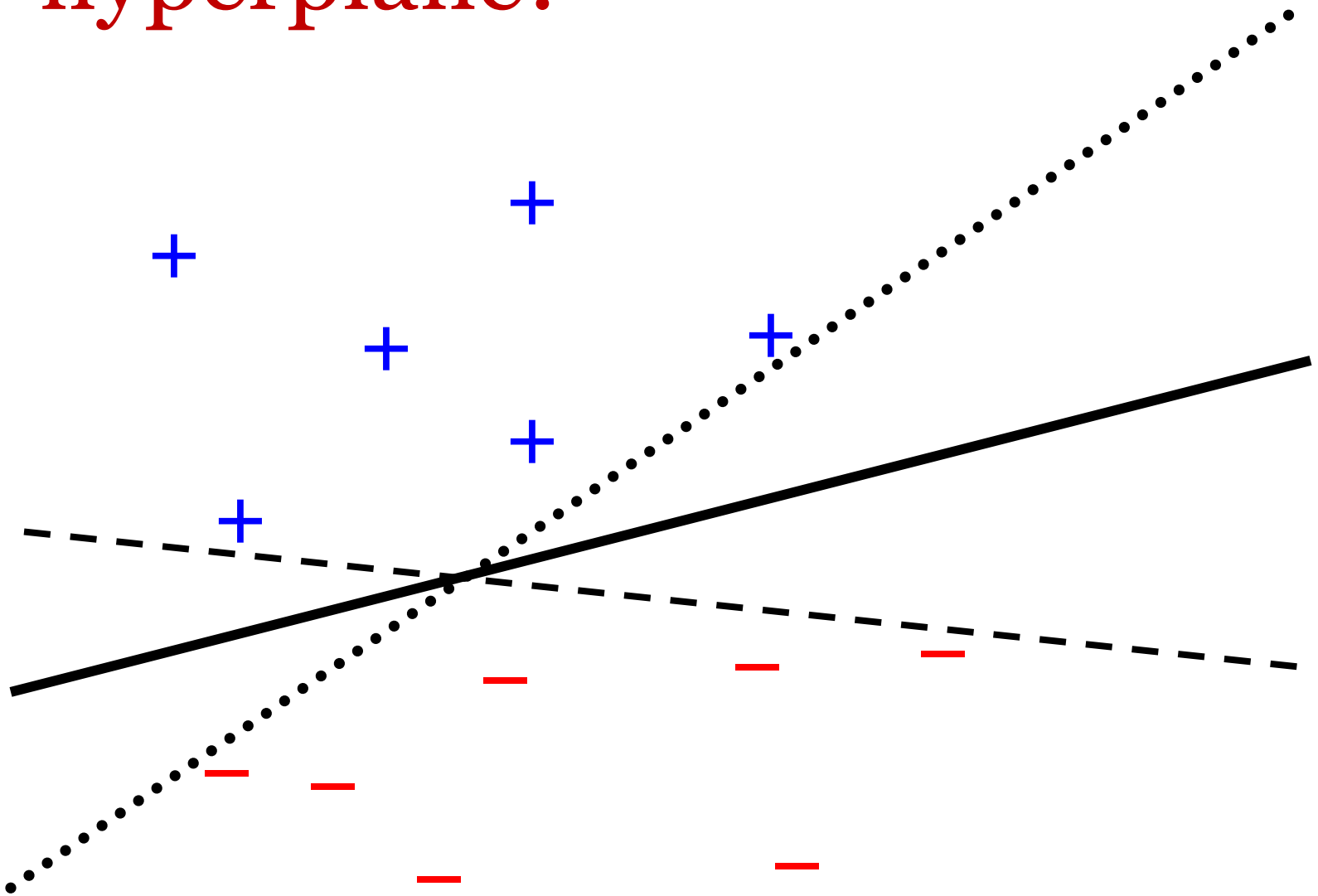
- Why do we care about the margin?
- Perceptron with margin
- Support Vector Machines

Dealing with outliers:

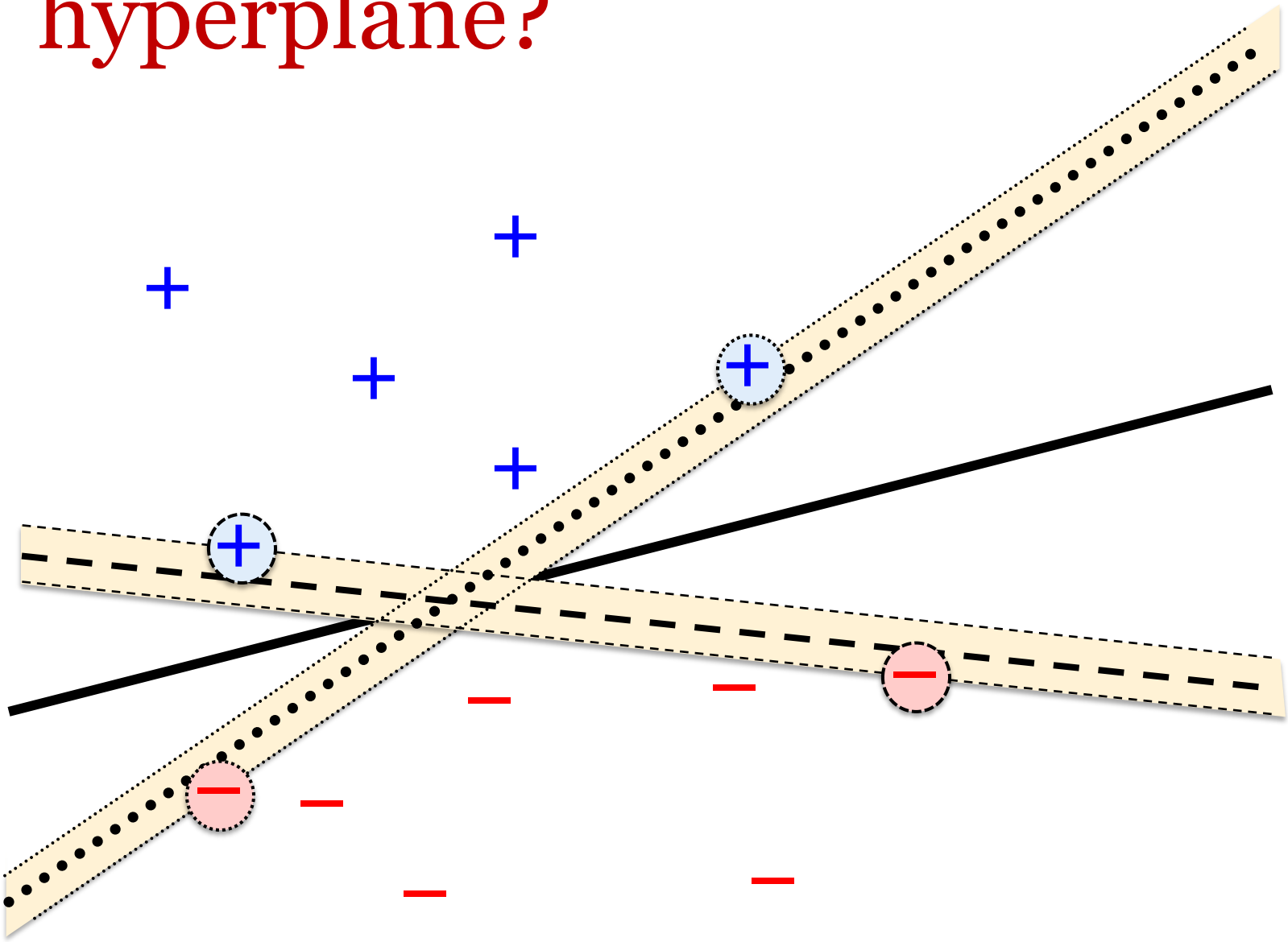
- Soft margins

Large margin classifiers

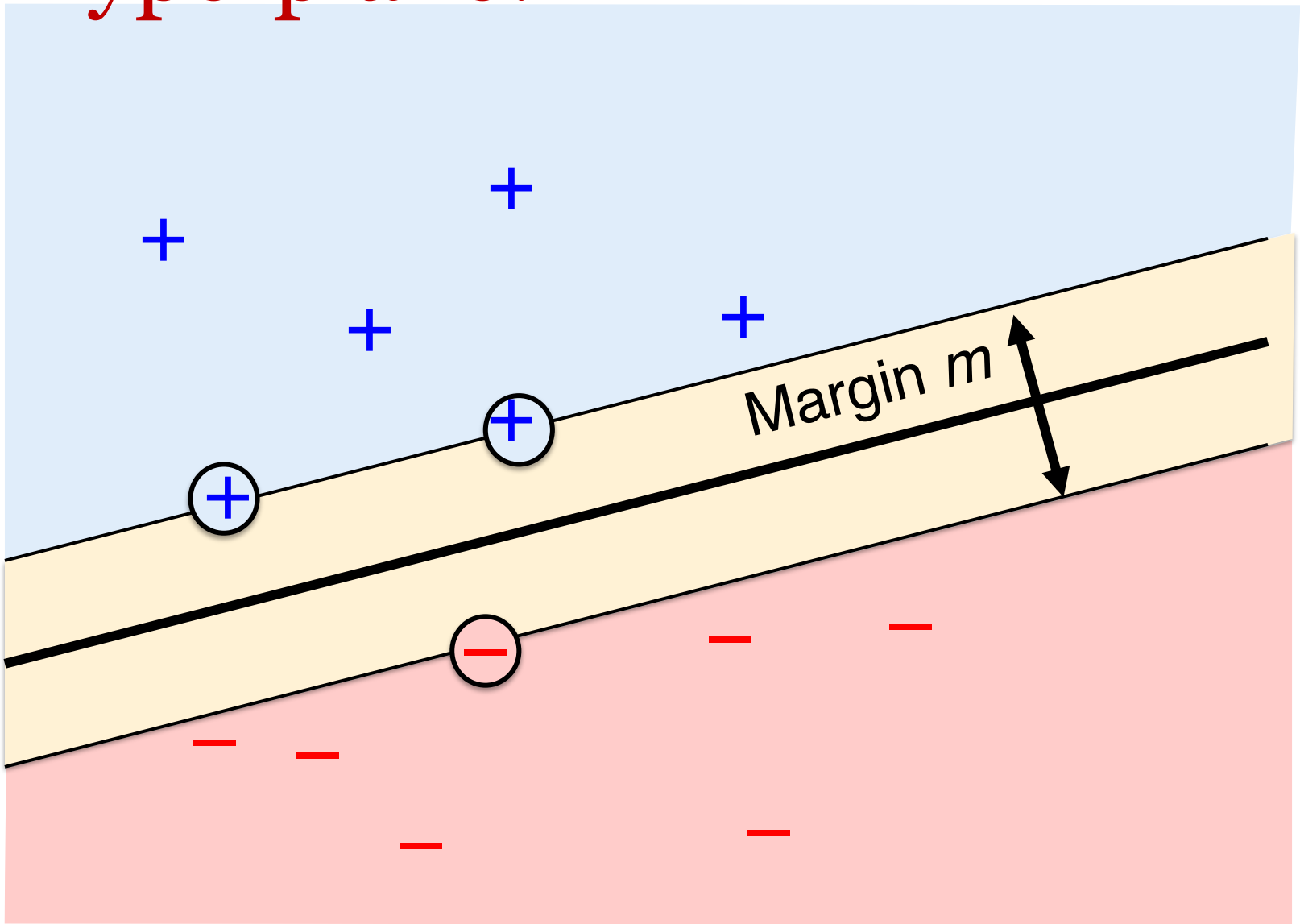
What's the best separating hyperplane?



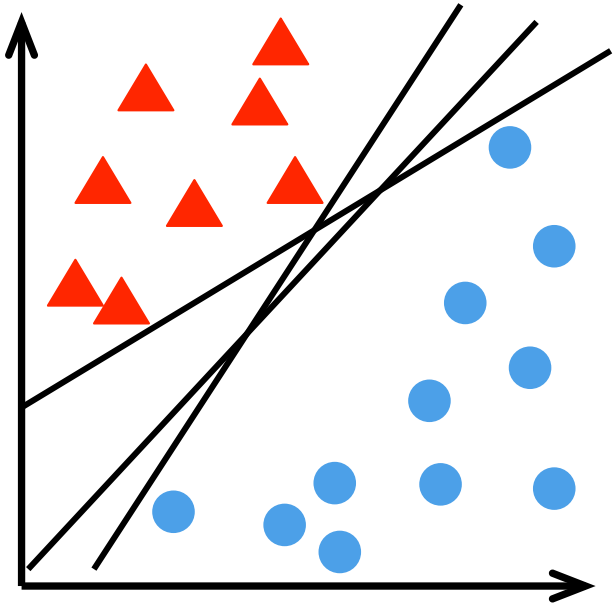
What's the best separating hyperplane?



What's the best separating hyperplane?

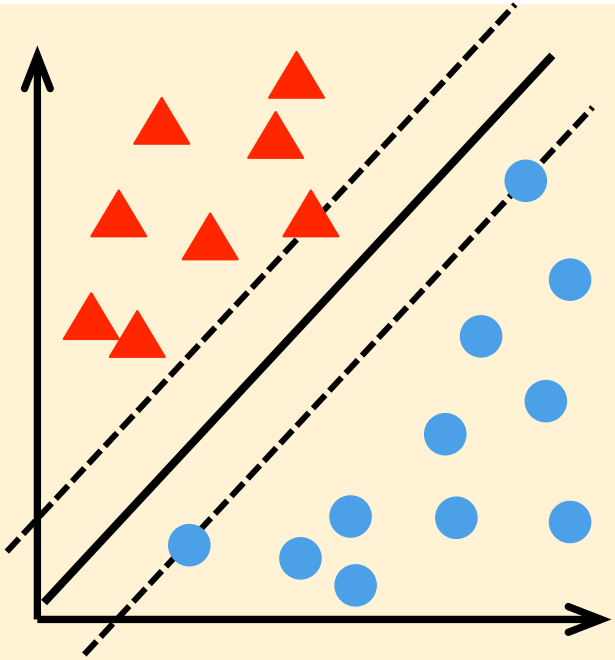


Maximum margin classifiers



These decision boundaries are very close to some items in the training data.
They have small margins.

Minor changes in the data could lead to different decision boundaries



This decision boundary is as far away from any training items as possible.
It has a large margin.

Minor changes in the data result in (roughly) the same decision boundary

Maximum margin classifier

Margin = the distance of the decision boundary to the closest items in the training data.

We want to find a classifier whose decision boundary is furthest away from the nearest data points. (This classifier has the largest margin).

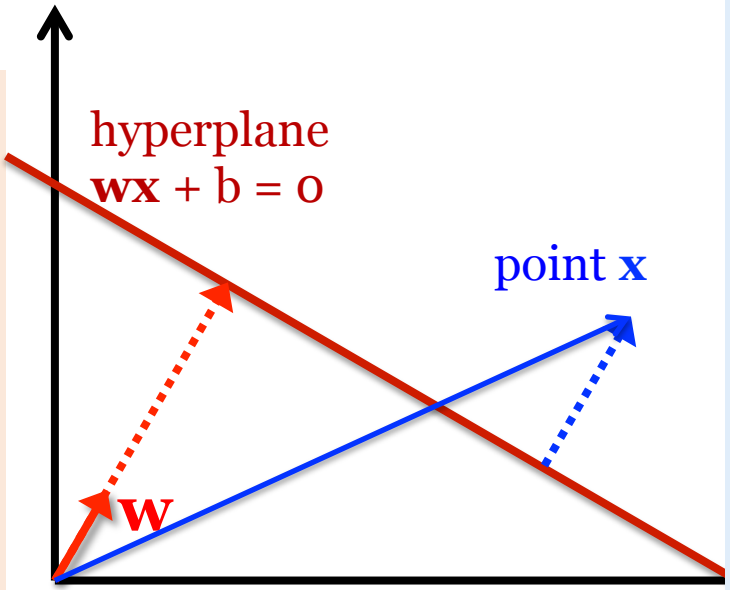
This additional requirement (*bias)* reduces the *variance* (i.e. reduces overfitting).

Margins

Margins

Distance of hyperplane $\mathbf{wx} + b = 0$ to origin:

$$\frac{-b}{\|\mathbf{w}\|}$$



Absolute distance of point $\mathbf{x}$ to hyperplane $\mathbf{wx} + b = 0$:

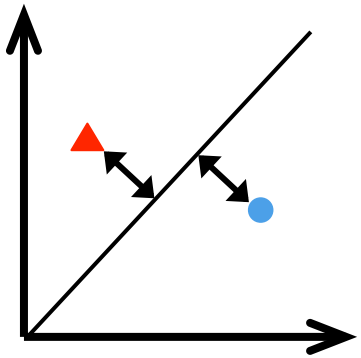
$$\frac{|\mathbf{wx} + b|}{\|\mathbf{w}\|}$$

Decision boundary:

Hyperplane with $f(\mathbf{x}) = 0$

i.e. $\mathbf{wx} + b = 0$

Margin



If the data are linearly separable,

$y^{(i)}(\mathbf{w}\mathbf{x}^{(i)} + b) > 0$

Euclidean distance of $\mathbf{x}^{(i)}$ to the decision boundary:

$$\frac{y^{(i)} f(\mathbf{x}^{(i)})}{\|\mathbf{w}\|} = \frac{y^{(i)}(\mathbf{w}\mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|}$$

Functional vs. Geometric margin

Geometric margin (Euclidean distance) of hyperplane $\mathbf{wx} + b = 0$ to point $\mathbf{x}^{(i)}$:

$$\frac{y^{(i)} f(\mathbf{x}^{(i)})}{\|\mathbf{w}\|} = \frac{y^{(i)}(\mathbf{w}\mathbf{x}^{(i)} + b)}{\|\mathbf{w}\|}$$

Functional margin of hyperplane $\mathbf{wx} + b = 0$ to point $\mathbf{x}^{(i)}$:

$$\gamma = y^{(i)}\, \mathrm{f}(\mathbf{x}^{(i)})$$

i.e. $$\gamma = y^{(i)}\, (\mathbf{w}\mathbf{x}^{(i)} + b)$$

Rescaling **w** and b

Rescaling **w** and b by a factor k to $k\boldsymbol{w}$ and kb does not change the geometric margin (Euclidean distance):

Geometric margin of **x** to **wx** + b = 0

$$\frac{y^{(i)}(\mathbf{w}\mathbf{x}^{(i)}+b)}{\|\mathbf{w}\|} = \frac{y^{(i)}\left(\sum_n w_n x_n^{(i)}+b\right)}{\sqrt{\sum_n w_n w_n}}$$

...spell out **wx**, || **w** || ...

$$= \frac{ky^{(i)}\left(\sum_n w_n x_n^{(i)}+b\right)}{k\sqrt{\sum_n w_n w_n}}$$

...multiply by k/k...

$$= \frac{y^{(i)}\left(\sum_n kw_n x_n^{(i)}+kb\right)}{\sqrt{\sum_n kw_n kw_n}}$$

...move k inside...

$$= \frac{y^{(i)}(k\mathbf{w}\mathbf{x}^{(i)}+kb)}{\|k\mathbf{w}\|}$$

Geometric margin of **x** to k**wx** + kb = 0

Rescaling **w** and b

Rescaling **w** and b by a factor k *does* change the functional margin γ by a factor k:

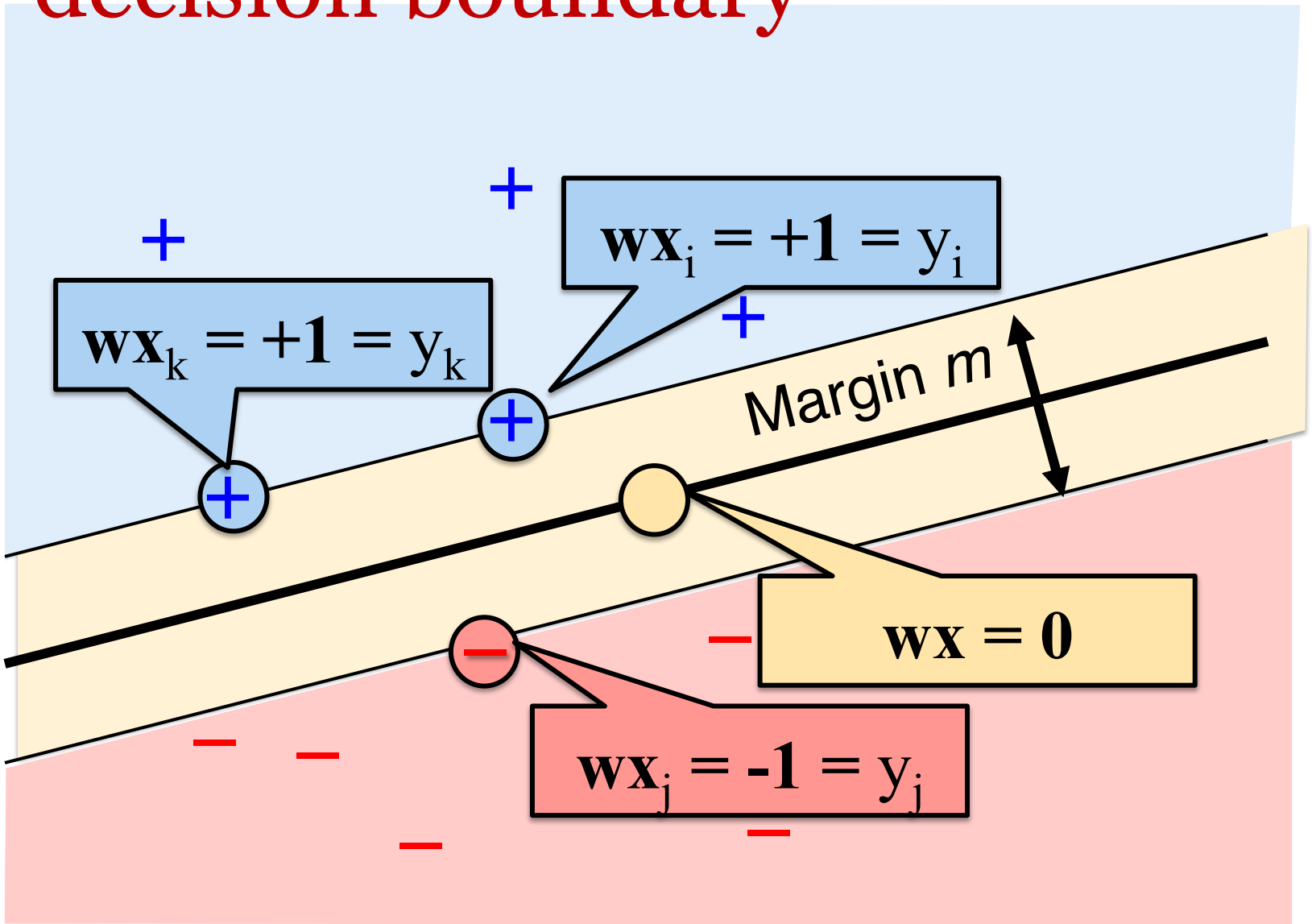
$$\gamma = y^{(i)} (\mathbf{w}\mathbf{x}^{(i)} + b)$$

$$k\gamma = y^{(i)} (k\mathbf{w}\mathbf{x}^{(i)} + kb)$$

The point that is closest to the decision boundary has functional margin γ_{min}

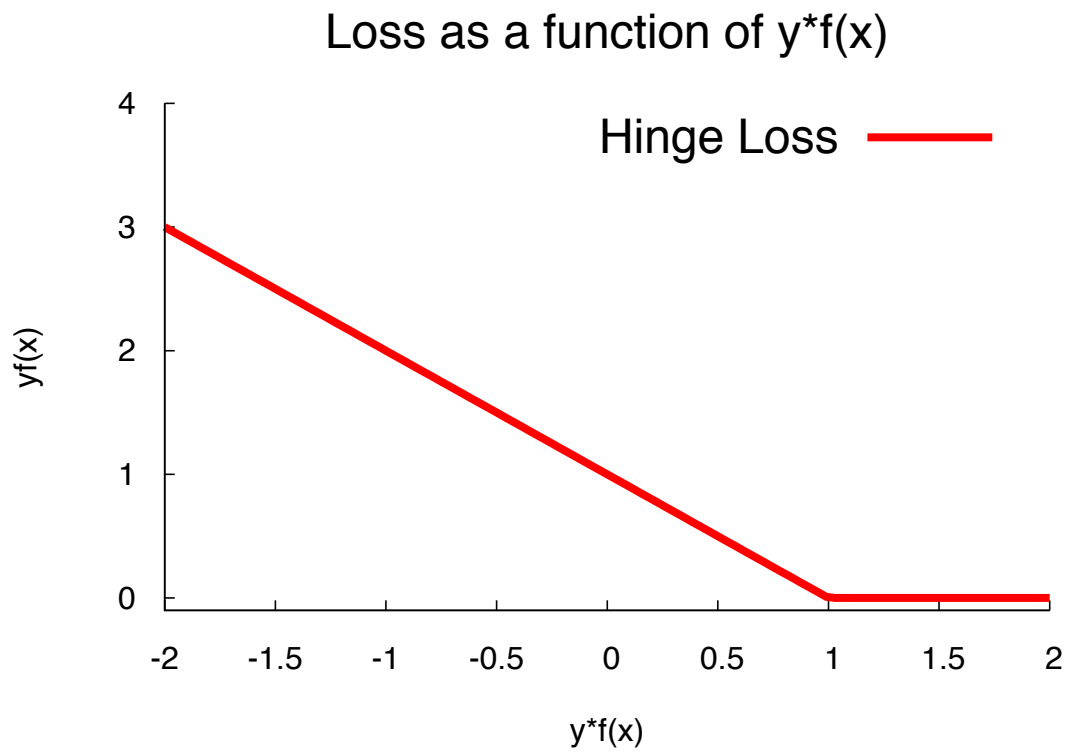
- **w** and b can be rescaled so that $\gamma_{min} = 1$
- When learning **w** and b, we can set $\gamma_{min} = 1$ (and still get the same decision boundary)

The maximum margin decision boundary



Hinge loss

$$L(y, f(\mathbf{x})) = \max(0, 1 - yf(\mathbf{x}))$$



Case 1: $f(\mathbf{x}) > 1$
$\mathbf{x}$ outside of margin
Hinge loss = 0

Case 2: $0 < yf(\mathbf{x}) < 1$:
$\mathbf{x}$ inside of margin
Hinge loss = $1-yf(\mathbf{x})$

Case 3: $yf(\mathbf{x}) < 0$:
$\mathbf{x}$ misclassified
Hinge loss = $1-yf(\mathbf{x})$

Perceptron with margin

Perceptron with Margin

Standard Perceptron update:

Update $\mathbf{w}$ if $y_m \cdot \mathbf{w} \cdot \mathbf{x_m} < 0$

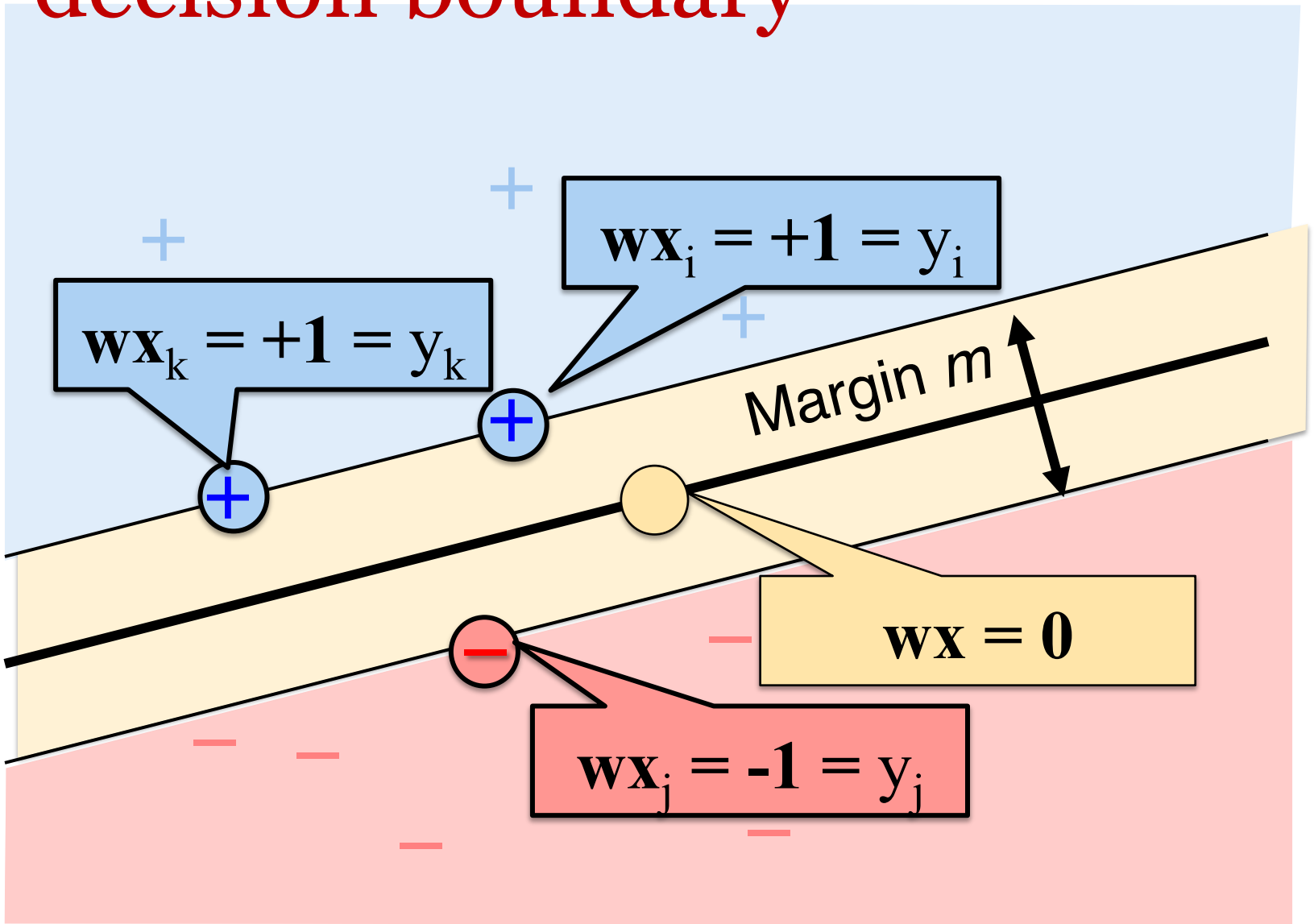
Perceptron with Margin update:

Define a functional margin $\gamma > 0$

Update $\mathbf{w}$ if $y_m \cdot \mathbf{w} \cdot \mathbf{x_m} < \gamma$

Support Vector Machines

The maximum margin decision boundary



The maximum margin decision boundary…

… is defined by two parallel hyperplanes:

– one that goes through the **positive** data points ($y_j = +1$) that are closest to the decision boundary, and

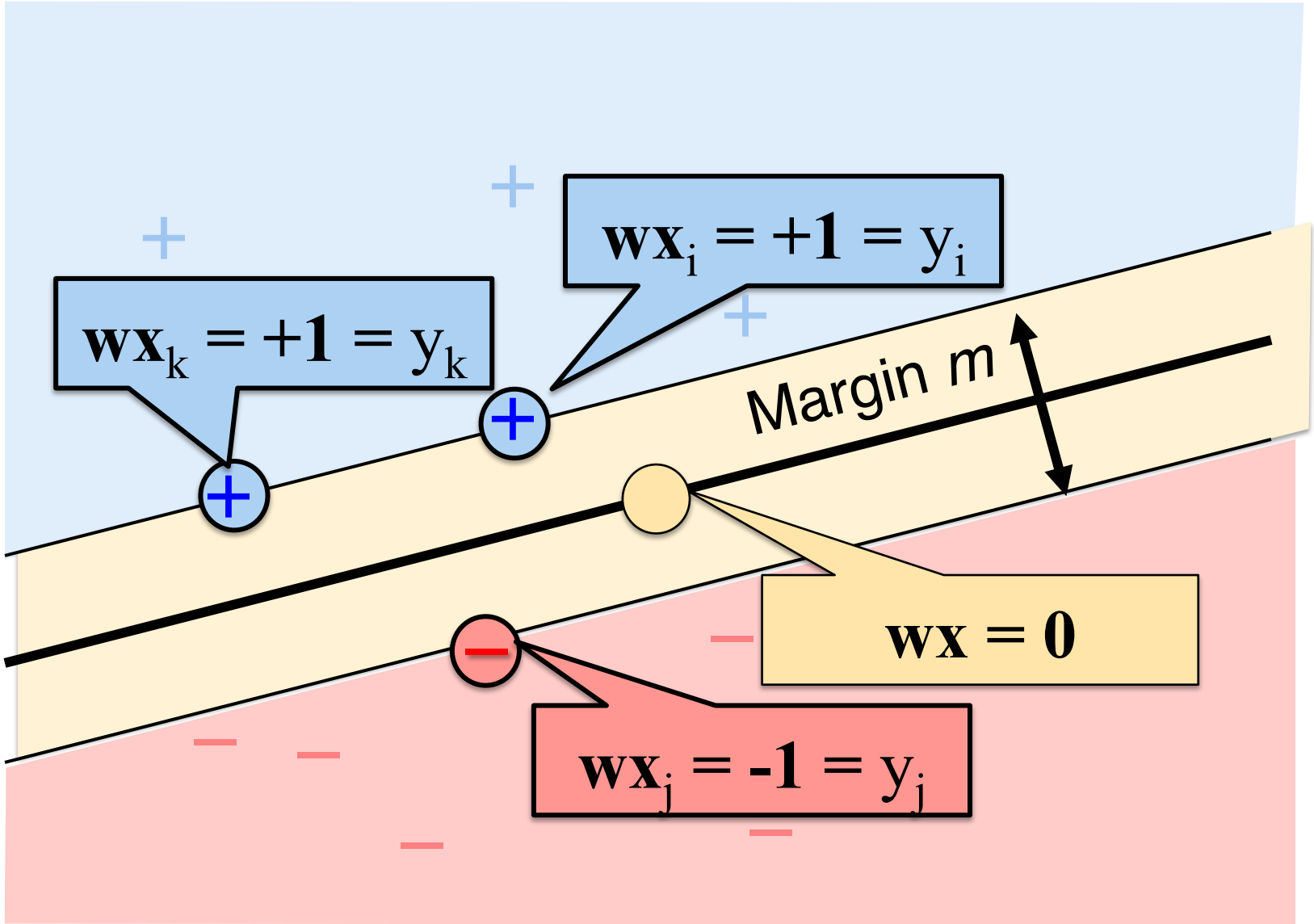
– one that goes through the **negative** data points ($y_j = -1$) that are closest to the decision boundary.

Support vectors

We can express the separating hyperplane in terms of the data points $\mathbf{x}_j$ closest to the decision boundary.

These data points are called the **support vectors.**

Support vectors



Perceptrons and SVMs: Differences in notation

Perceptrons:

- Weight vector has bias term w_0 (x_0 = dummy value 1)
- Decision boundary: $\mathbf{wx} = 0$

SVMs/Large Margin classifiers:

- Explicit bias term b; weight vector $\mathbf{w} = (w_1 \ldots w_n)$
- Decision boundary $\mathbf{wx} + b = 0$

Support Vector Machines

The functional margin of the data for (**w**, b) is determined by the points closest to the hyperplane

$$\gamma_{\min} = \min_{n} \left[y^{(n)}(\mathbf{w}\mathbf{x}^{(n)} + b) \right]$$

Distance of $\mathbf{x}^{(n)}$ to the hyperplane $\mathbf{w}\mathbf{x} = 0$: $\frac{|\mathbf{w}\mathbf{x} + b|}{\|\mathbf{w}\|}$

Learn **w** in an SVM = maximize the margin:

$$\underset{\mathbf{w}, b}{\operatorname{argmax}} \left\{ \frac{1}{\|\mathbf{w}\|} \min_{n} \left[y^{(n)}(\mathbf{w}\mathbf{x} + b) \right] \right\}$$

Support Vector Machines

Learn **w** in an SVM = maximize the margin:

$$\underset{\mathbf{w},\, b}{\operatorname{argmax}} \left\{ \frac{1}{\|\mathbf{w}\|} \min_{n} \left[y^{(n)} (\mathbf{w}\mathbf{x} + b) \right] \right\}$$

This is difficult to optimize.
Let's convert it to an equivalent problem that is easier.

Support Vector Machines

Learn **w** in an SVM = maximize the margin:

$$\underset{\mathbf{w},\, b}{\operatorname{argmax}} \left\{ \frac{1}{\|\mathbf{w}\|} \min_{n} \left[y^{(n)} (\mathbf{w}\mathbf{x} + b) \right] \right\}$$

Easier equivalent problem:

- We can always rescale **w** and b without affecting Euclidian distances.
- This allows us to set the functional margin to 1: $\min_n(y^{(n)}(\mathbf{w}\mathbf{x}^{(n)} + b) = 1$

Support Vector Machines

Learn **w** in an SVM = maximize the margin:

$$\underset{\mathbf{w},\, b}{\operatorname{argmax}} \left\{ \frac{1}{\|\mathbf{w}\|} \min_{n} \left[y^{(n)}(\mathbf{w}\mathbf{x} + b) \right] \right\}$$

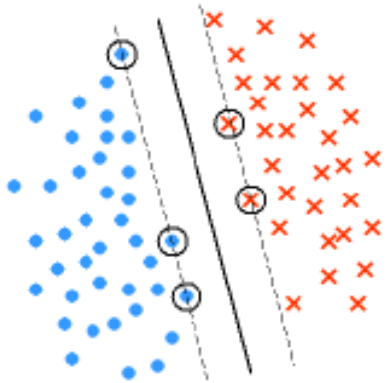
Easier equivalent problem: a quadratic program

- Setting $\min_n(y^{(n)}(\mathbf{w}\mathbf{x}^{(n)} + b) = 1$ implies $(y^{(n)}(\mathbf{w}\mathbf{x}^{(n)} + b) \geq 1$ for all n
- argmax(1/**ww**)= argmin(**ww**) = argmin(1/2·**ww**)

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2} \mathbf{w} \cdot \mathbf{w}$$
$$subject\ to$$
$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i$$

Support Vector Machines

The name "Support Vector Machine" stems from the fact that $\mathbf{w}^*$ is supported by (i.e. is the linear span of) the examples that are exactly at a distance $1/||\mathbf{w}^*||$ from the separating hyperplane. These vectors are therefore called **support vectors**.



Theorem: Let $\mathbf{w}^*$ be the minimizer of the SVM optimization problem for $S = \{(x_i, y_i)\}$.

Let $I = \{i: y_i (\mathbf{w}^* \mathbf{x}_i + b) = 1\}$.

Then there exist coefficients $\alpha_i > 0$ such that:

$$\mathbf{w}^* = \sum_{i \in I} \alpha_i y_i \mathbf{x}_i$$

The primal representation

The data items $\mathbf{x} = (x_1 \ldots x_n)$ have n features
The weight vector $\mathbf{w} = (w_1 \ldots w_n)$ has n elements

Learning:

Find a weight w_j for each feature x_j

Classification:

Evaluate **wx**

The dual representation

$$\mathbf{w} = \sum_j \alpha_j \mathbf{x}_j$$

Learning:

Find a weight α_j (≥ 0) for each data point $\mathbf{x}_j$

This requires computing the inner product $\mathbf{x}_i \mathbf{x}_j$ between all pairs of data items $\mathbf{x}_i$ and $\mathbf{x}_j$

Support vectors = the set of data points $\mathbf{x}_j$ with **non-zero weights** α_j

Classifying test data with SVM

In the primal:

Compute inner product between weight vector and test item

$\mathbf{wx} = \langle \mathbf{w}, \mathbf{x} \rangle$

In the dual:

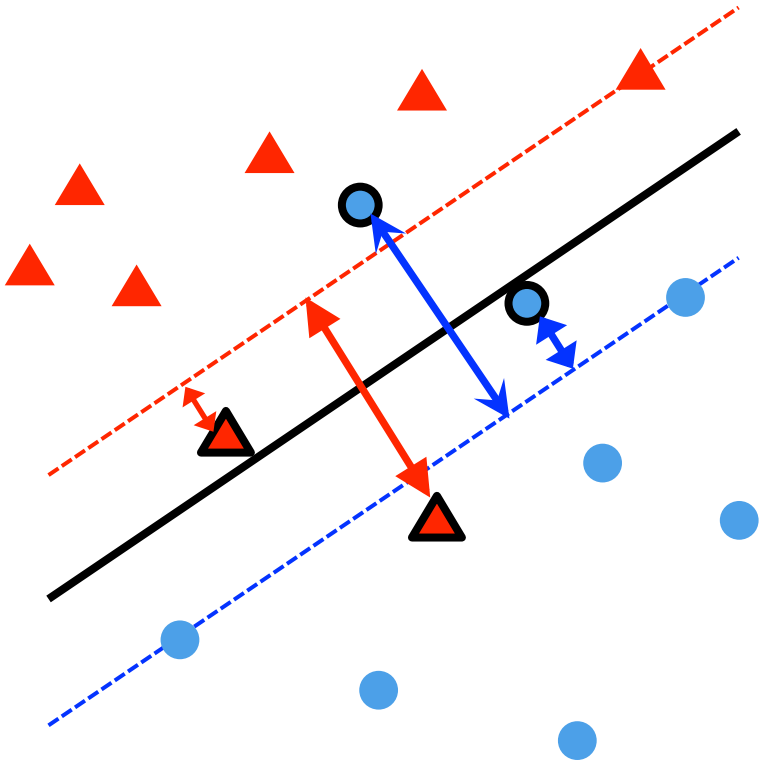
Compute inner products between the support vectors and test item

$\mathbf{wx} = \langle \mathbf{w}, \mathbf{x} \rangle = \langle \sum_j \alpha_j x_j, \mathbf{x} \rangle = \sum_j \alpha_j \langle x_j, \mathbf{x} \rangle$

Dealing with outliers: Soft margins

Dealing with outliers: Slack variables ξ_i

ξ_i measures by how much example $(\mathbf{x}_i, y_i)$ fails to achieve margin δ



Dealing with outliers: Slack variables ξ_i

If $\mathbf{x}_i$ is on correct side of the margin:

$\xi_i = 0$

otherwise

$\xi_i = |y_i - \mathbf{w}\mathbf{x}_i|$

If $\xi_i = 1$: $\mathbf{x}_i$ is on the decision boundary $\mathbf{w}\mathbf{x}_i = 0$

If $\xi_i > 1$: $\mathbf{x}_i$ is misclassified

Replace $y^{(n)}(\mathbf{w}\mathbf{x}^{(n)} + b) \geq 1$ (hard margin)

with $y^{(n)}(\mathbf{w}\mathbf{x}^{(n)} + b) \geq 1 - \xi^{(n)}$ (soft margin)

Soft margins

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$

$$subject\ to$$

$$\xi_i \geq 0 \ \ \forall i$$

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq (1-\xi_i)\forall i$$

ξ_i (slack): how far off is $\mathbf{x}_i$ from the margin?

C (cost): how much do we have to pay for misclassifying $\mathbf{x}_i$

We want to minimize $C\sum_i \xi_i$ and maximize the margin

C controls the tradeoff between margin and training error

Soft SVMs

Now the optimization problem becomes

$$\text{Min}_{\mathbf{w}}\ \tfrac{1}{2}\,||\mathbf{w}||^2 + C \sum_{(x,y)\in S} \max(0,\ 1 - y\,\mathbf{wx})$$

where the parameter C controls the tradeoff between choosing a large margin (small ||w||) and choosing a small hinge-loss.

Training SVMs

Traditional approach:

Solve quadratic program.

– This is very slow.

Current approaches:

Use variants of stochastic gradient descent or coordinate descent.

More on Tuesday!