

Intro. to machine learning (CSI 5325)

Lecture 21: Support vector learning

Greg Hamerly

Some content from Andrew Moore
<http://www.cs.cmu.edu/~awm/tutorials>.

- 1 Linear classifiers
- 2 Classifier margin
- 3 Margin width
- 4 Quadratic programming solution

SVM overview

Support vector machines are a type of classifier that

- is based on the simple linear classifier
- works well in high dimension
- obtains best performance in many applications
- adapts well to non-linear decision surfaces
- relies heavily on more advanced math such as optimization theory, linear algebra, and quadratic programming

Linear (hyperplane) classifiers

$$\begin{aligned} f(\mathbf{x}, \mathbf{w}, b) &= \text{sign}(\mathbf{w} \cdot \mathbf{x} - b) \\ &= \mathbf{w} \cdot \mathbf{x} - b \geq 0 \\ &= \mathbf{w} \cdot \mathbf{x} \geq b \end{aligned}$$

- \mathbf{w} and \mathbf{x} are d -dimensional vectors
- b is the 'bias,' a scalar

Learning task: find \mathbf{w} and b so that f will best fit a set of labeled examples $\langle \mathbf{x}_i, y_i \rangle$, where $y_i \in \{+1, -1\}$.

Some interpretations

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

Together, \mathbf{w} and b form a hyperplane of $d - 1$ dimensions.

- the hyperplane divides \mathbf{R}^d into two parts (two classes)

\mathbf{w} is perpendicular to the hyperplane, and points in the direction for positive classification.

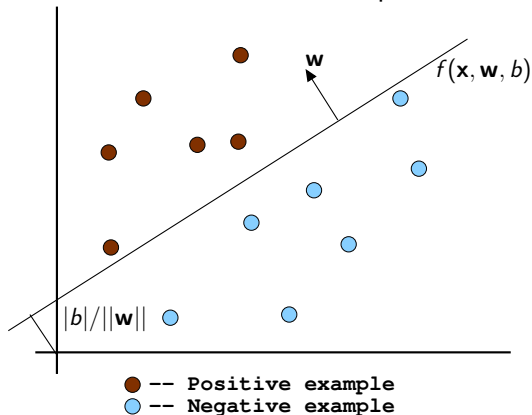
$|b|/||\mathbf{w}||$ is the distance from the origin to the closest point on the hyperplane.

What does it look like?

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

f is a plane that divides \mathbf{R}^d

Two dimensional example:



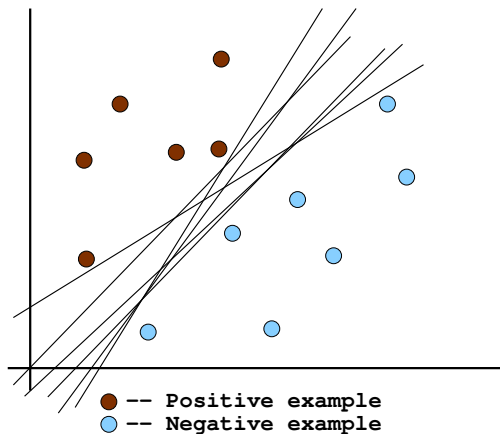
How should we choose \mathbf{w} and b ?

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

Which hyperplane should we choose?

There are infinite possibilities...

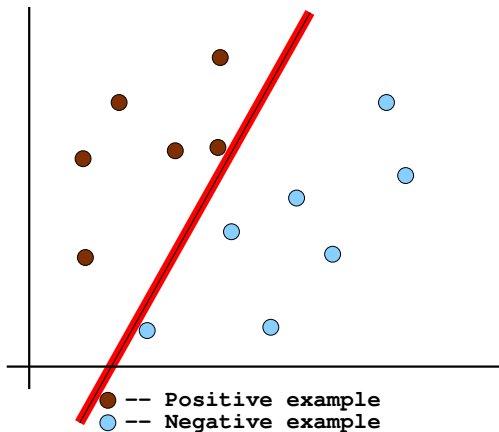
...if the examples are separable



Classifier margin

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

The *margin* of a classifier is the width that the boundary could be increased before hitting an example.

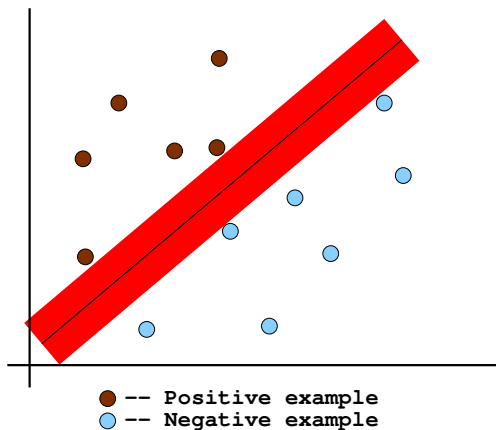


Maximizing the margin

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

The *maximum margin* linear classifier is the hyperplane with the maximum margin.

This is the simplest kind of support vector machine (SVM).

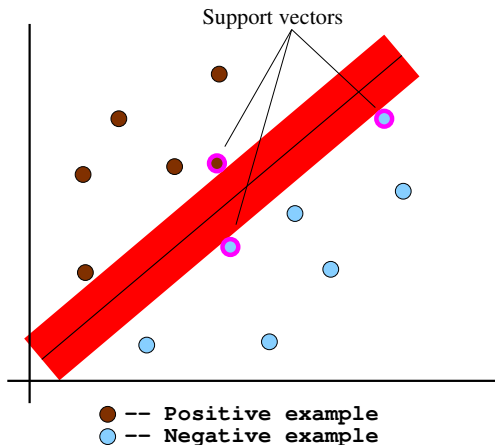


Support vectors

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

So-called 'support vectors' are the examples which the margin touches.

Note: the max-margin hyperplane is uniquely defined by the support vectors alone.

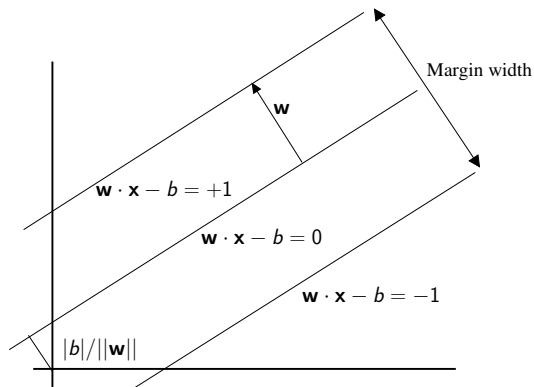


Why maximize the margin?

Why should we maximize the margin?

- Intuitively this feels safest among the various hyperplanes.
- This gives us the smallest chance that a mistake in the boundary location will lead to a misclassification.
- LOOCV is easy since the model is immune to removing any non-support-vector examples.
- There's some theory (based on VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
- Empirically it works very well.

Perpendicular \mathbf{w}



Claim: \mathbf{w} is perpendicular to the decision plane. Why?

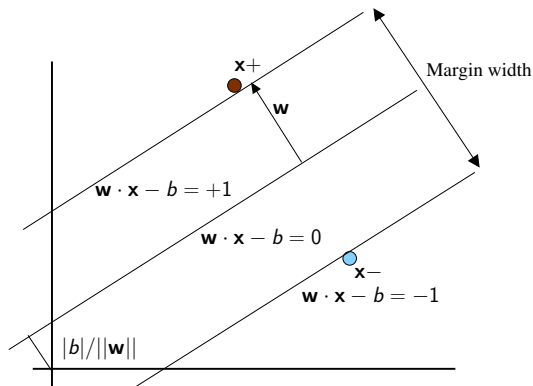
- Take any two points \mathbf{u} and \mathbf{v} that are on the decision plane. What is $\mathbf{w} \cdot (\mathbf{u} - \mathbf{v})$?

The margin width

What is the margin width?

Well, first:

- let \mathbf{x}^+ be any point on the 'plus plane'
- let \mathbf{x}^- be the closest point to \mathbf{x}^+ on the 'minus plane'
- claim: $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$ for some λ . Why?

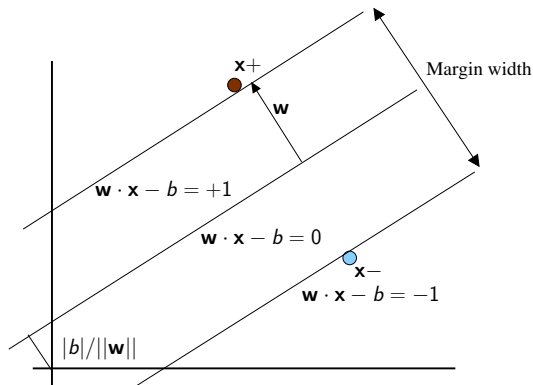


The margin width M (2)

We know:

- $\mathbf{w} \cdot \mathbf{x}^+ - b = +1$
- $\mathbf{w} \cdot \mathbf{x}^- - b = -1$
- $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$
- $\|\mathbf{x}^+ - \mathbf{x}^-\| = M$

So we can get M in terms of \mathbf{w} and b .



The margin width M (3)

We know:

- $\mathbf{w} \cdot \mathbf{x}^+ - b = +1$
- $\mathbf{w} \cdot \mathbf{x}^- - b = -1$
- $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$
- $\|\mathbf{x}^+ - \mathbf{x}^-\| = M$

Therefore:

$$\begin{aligned} \mathbf{w} \cdot (\mathbf{x}^- + \lambda \mathbf{w}) - b &= 1 \\ \mathbf{w} \cdot \mathbf{x}^- + \lambda \mathbf{w} \cdot \mathbf{w} - b &= 1 \\ -1 + \lambda \mathbf{w} \cdot \mathbf{w} &= 1 \\ \lambda &= \frac{2}{\mathbf{w} \cdot \mathbf{w}} \end{aligned}$$

Now combine λ on the right with facts on the left...

$$\begin{aligned} M &= \|(\mathbf{x}^- + \lambda \mathbf{w}) - \mathbf{x}^-\| \\ &= \|\lambda \mathbf{w}\| \\ &= \|2/(\mathbf{w} \cdot \mathbf{w})\mathbf{w}\| \end{aligned}$$

The margin width M (4)

Recall that $\|\mathbf{x}\| \equiv \sqrt{\mathbf{x} \cdot \mathbf{x}}$

$$\begin{aligned}
 M &= \|2/(\mathbf{w} \cdot \mathbf{w})\mathbf{w}\| \\
 &= \frac{2}{\mathbf{w} \cdot \mathbf{w}} \|\mathbf{w}\| \\
 &= \frac{2}{\mathbf{w} \cdot \mathbf{w}} \sqrt{\mathbf{w} \cdot \mathbf{w}} \\
 &= \frac{2}{\sqrt{\mathbf{w} \cdot \mathbf{w}}} \\
 &= \frac{2}{\|\mathbf{w}\|}
 \end{aligned}$$

Phew! So the margin is inversely related to the length of \mathbf{w} .

Therefore, maximizing M will involve minimizing $\|\mathbf{w}\|$.

Learning the maximum margin classifier

Given a guess of \mathbf{w} and b , we can

- compute whether all data points are correctly classified
- compute the margin width M

So we just need an algorithm to search the space of \mathbf{w} and b to find the widest margin that matches all the datapoints.

How?

- Gradient descent? Simulated annealing? Matrix inversion? EM? Newton's method?

Learning via quadratic programming

- QP is a well-studied class of optimization algorithms to
- maximize a quadratic function of real-valued variables
 - subject to linear constraints

Quadratic programming solution

$$\text{Find: } \underset{\mathbf{u}}{\operatorname{argmax}} \quad c + \mathbf{d}^T \mathbf{u} + \frac{\mathbf{u}^T \mathbf{R} \mathbf{u}}{2}$$

Subject to k linear **inequality** constraints:

$$a_{11}u_1 + a_{12}u_2 + \dots + a_{1m}u_m \leq b_1$$

$$a_{21}u_1 + a_{22}u_2 + \dots + a_{2m}u_m \leq b_2$$

$$\vdots \quad \vdots \quad \vdots$$

$$a_{n1}u_1 + a_{n2}u_2 + \dots + a_{nm}u_m \leq b_k$$

And subject to e linear **equality** constraints:

$$a_{(k+1)1}u_1 + a_{(k+1)2}u_2 + \dots + a_{(k+1)m}u_m = b_{(k+1)}$$

$$a_{(k+2)1}u_1 + a_{(k+2)2}u_2 + \dots + a_{(k+2)m}u_m = b_{(k+2)}$$

$$\vdots \quad \vdots \quad \vdots$$

$$a_{(k+e)1}u_1 + a_{(k+e)2}u_2 + \dots + a_{(k+e)m}u_m = b_{(k+e)}$$

Quadratic programming solution

Of course, rather than implement our own solver, we will use off-the-shelf QP solvers.

Now, how do we cast SVM training as a QP problem?

Casting SVM learning as QP

Assume there are n datapoints, each $\langle \mathbf{x}_i, y_i \rangle$ where $y_i \in \{+1, -1\}$.

What should our quadratic optimization criterion be?

Casting SVM learning as QP

Assume there are n datapoints, each $\langle \mathbf{x}_i, y_i \rangle$ where $y_i \in \{+1, -1\}$.

What should our quadratic optimization criterion be?

- minimize $\mathbf{w} \cdot \mathbf{w}$

Casting SVM learning as QP

Assume there are n datapoints, each $\langle \mathbf{x}_i, y_i \rangle$ where $y_i \in \{+1, -1\}$.

What should our quadratic optimization criterion be?

- minimize $\mathbf{w} \cdot \mathbf{w}$

What constraints will we have?

Casting SVM learning as QP

Assume there are n datapoints, each $\langle \mathbf{x}_i, y_i \rangle$ where $y_i \in \{+1, -1\}$.

What should our quadratic optimization criterion be?

- minimize $\mathbf{w} \cdot \mathbf{w}$

What constraints will we have?

- $\mathbf{w} \cdot \mathbf{x}_i - b \geq 1$ if $y_i = 1$
- $\mathbf{w} \cdot \mathbf{x}_i - b \leq -1$ if $y_i = -1$

Casting SVM learning as QP

Assume there are n datapoints, each $\langle \mathbf{x}_i, y_i \rangle$ where $y_i \in \{+1, -1\}$.

What should our quadratic optimization criterion be?

- minimize $\mathbf{w} \cdot \mathbf{w}$

What constraints will we have?

- $\mathbf{w} \cdot \mathbf{x}_i - b \geq 1$ if $y_i = 1$
- $\mathbf{w} \cdot \mathbf{x}_i - b \leq -1$ if $y_i = -1$

How many such constraints will we have?

Casting SVM learning as QP

Assume there are n datapoints, each $\langle \mathbf{x}_i, y_i \rangle$ where $y_i \in \{+1, -1\}$.

What should our quadratic optimization criterion be?

- minimize $\mathbf{w} \cdot \mathbf{w}$

What constraints will we have?

- $\mathbf{w} \cdot \mathbf{x}_i - b \geq 1$ if $y_i = 1$
- $\mathbf{w} \cdot \mathbf{x}_i - b \leq -1$ if $y_i = -1$

How many such constraints will we have?

- n

That's it!