Linear Classifiers

Linear classifiers

For convenience, assume $Y \in \{-1, +1\}$.

A *linear classifier* has the form

$$f(x) = \text{sign}\{w^T x + b\}$$

where $a \in \mathbb{R}^d$, $b \in \mathbb{R}$

In addition to being extremely useful in their own right, linear classifiers are at the heart of many other discrimination rules, including SVMs, decision trees, and neural networks.

Approaches to linear classification are numerous. In fact, a study of linear classifiers exposes one to many, if not most, of the various algorithms and principles used in supervised learning.

~~We will study~~

Linear classification methodologies include the following, many of which we will cover:

- LDA
- Fisher's method
- Least squares regression
- Logistic regression
- The perceptron
- Widrow-Hoff / LMS
- Linear programming
- Maximum margin (linear SVM)

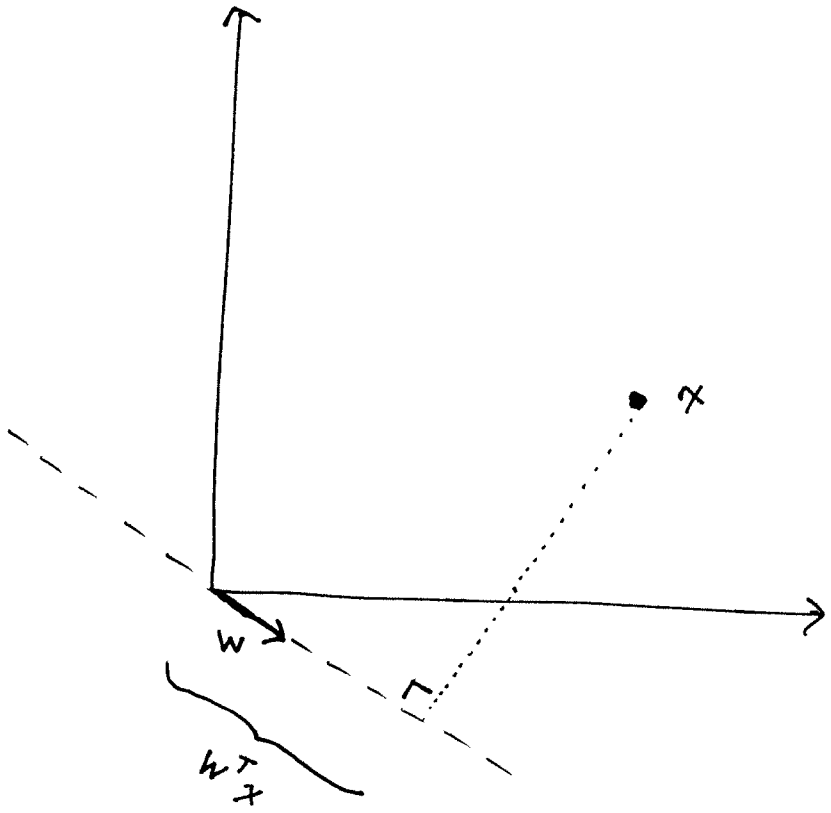
Fisher's Method

Suppose

$$f(x) = \text{sign}\{ w^T x + b \}$$

is a linear classifier. Without loss of generality, we may assume $\|w\| = 1$ (why?)

What is the geometric interpretation of $w^T x$?

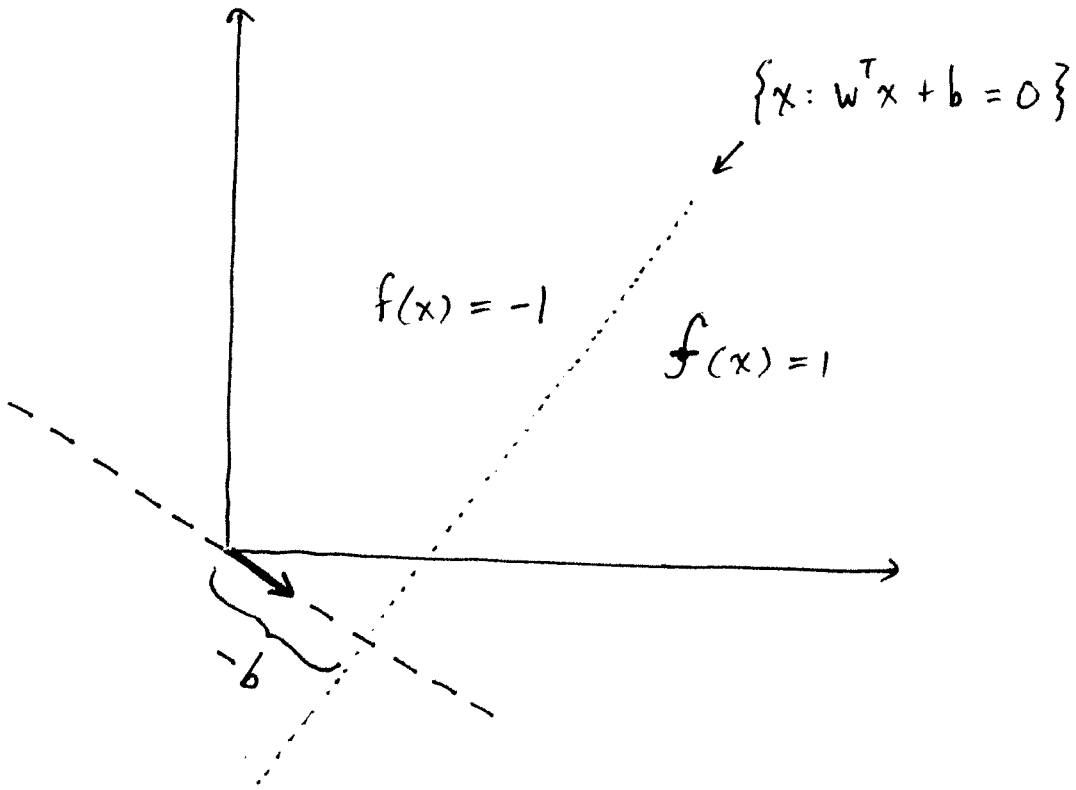


The projection of x onto w is

$$w(w^T w)^{-1} w^T x = w \cdot (w^T x)$$

↑ coefficient of projection

w is the normal vector to the hyperplane



We may think of a linear classifier as a two-step procedure:

1. Compute the scalar statistic

$$\tilde{x} = w^T x$$

2. Threshold $\tilde{x}$

Sir Ronald Fisher, the famous statistician, suggested the following principle for determining a good w given training data $(x_1, y_1), \ldots, (x_n, y_n)$:

Fisher's criterion

Choose w to maximize the <u>between-class scatter</u> of $t_i = w^T x_i$ while minimizing the <u>within-class scatter</u>.

Definitions

$$I_+ = \{i : y_i = +1\}, \quad I_- = \{i : y_i = -1\}$$

$$n_\pm = |I_\pm|, \quad n_- = |I_-|$$

$$m_\pm = \frac{1}{n_+} \sum_{i \in I_+} x_i, \quad m_- = \frac{1}{n_-} \sum_{i \in I_-} x_i$$

$$\tilde{m}_+ = \frac{1}{n_+} \sum_{i \in I_+} \tilde{x}_i, \quad \tilde{m}_- = \frac{1}{n_-} \sum_{i \in I_-} x_i$$

$$\tilde{s}_+^2 = \sum_{i \in I_+} (\tilde{x}_i - \tilde{m}_+)^2, \quad \tilde{s}_-^2 = \sum_{i \in I_-} (\tilde{x}_i - \tilde{m}_-)^2$$

Fisher's concrete proposal was to maximize

$$J(w) = \frac{|\tilde{m}_+ - \tilde{m}_-|^2}{\tilde{s}_+^2 + \tilde{s}_-^2} = \frac{\text{between-class scatter}}{\text{within-class scatter}}$$

Exercise | Express $J(w)$ as a function of w

94 4. Linear Methods for Classification

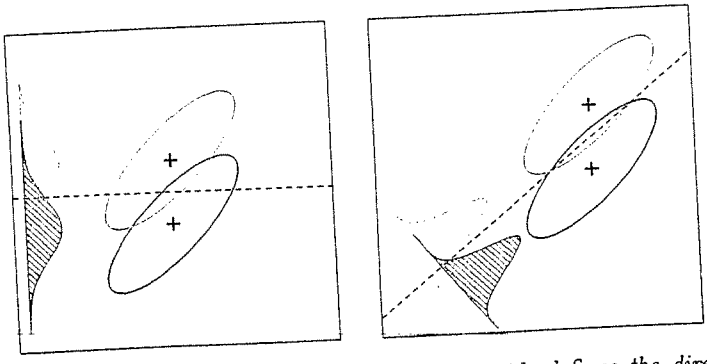


FIGURE 4.9. *Although the line joining the centroids defines the direction of greatest centroid spread, the projected data overlap because of the covariance (left panel). The discriminant direction minimizes this overlap for Gaussian data (right panel).*

↑ Hastie, Tibshirani, & Friedman

Solution

- $(\tilde{m}_+ - \tilde{m}_-)^2 = (w^T m_+ - w^T m_-)^2$

$$= w^T (m_+ - m_-)(m_+ - m_-)^T w$$

$$= w^T S_B w$$

↑ between class scatter matrix

- $\tilde{s}_+^2 = \sum_{i \in I_+} (\tilde{x}_i - \tilde{m}_+)^2 = \sum_{i \in I_+} (w^T x_i - w^T m_+)^2$

$$= \sum_{i \in I_+} w^T (x_i - m_+)(x_i - m_+)^T w$$

$$= w^T \left[\sum_{i \in I_+} (x_i - m_+)(x_i - m_+)^T \right] w$$

$$= w^T S_+ w$$

- $\tilde{s}_-^2 = w^T S_- w$

- $\tilde{s}_+^2 + \tilde{s}_-^2 = w^T (S_+ + S_-) w$

$$= w^T S_W w$$

↑ within class scatter matrix

$$\Rightarrow \boxed{J(w) = \frac{w^T S_B w}{w^T S_W w}}$$

"generalized Rayleigh quotient"

Note that $J(w) = J(\lambda w)$ for any $\lambda \in \mathbb{R}, \lambda \neq 0$.

Thus

$$\max_w \; J(w) = \max \; w^T S_B w$$
$$\text{s.t.} \quad w^T S_W w = 1$$

Using Lagrange multiplier theory, w is the solution of

$$\max \; w^T S_B w - \lambda (w^T S_W w - 1)$$

for some λ. Taking the derivative and setting to 0, we obtain

$$S_B w = \lambda S_W w$$

or

$$S_W^{-1} S_B w = \lambda w.$$

How should we solve for λ?

Note $S_W^{-1} S_B$ is not symmetric $\Rightarrow$ can't find eigenvalues.

We could make the transformation

$$w \longrightarrow S_W^{\frac{1}{2}} v$$

leading to

$$\underbrace{S_W^{\frac{1}{2}} S_B S_W^{\frac{1}{2}}}_{\text{symmetric}} v = \lambda v$$

However, there is a direct solution.

For any w, $S_B w$ is a multiple of _____.

Therefore,

$$S_W^{-1} S_B w = \lambda w$$

$$\Rightarrow w \propto \_\_\_\_\_.$$

Since we don't really care about whether w is normalized, we can take

$$w = S_W^{-1}(m_+ - m_-)$$

Recall

$$S_W = \sum_{i \in I_-} (x_i - m_-)^2 + \sum_{i \in I_+} (x_i - m_+)^2$$

$$= n \cdot \hat{\Sigma}$$

↑ pooled covariance estimate

Therefore Fisher's method is the same as LDA, at least in the choice of w.

Fisher's method is usually called <u>Fisher's Linear Discriminant</u>.

Linear Regression

In regression we observe (x_i, y_i), $i = 1, \ldots, n$, where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$. It is assumed

$$Y = h(X) + \epsilon$$

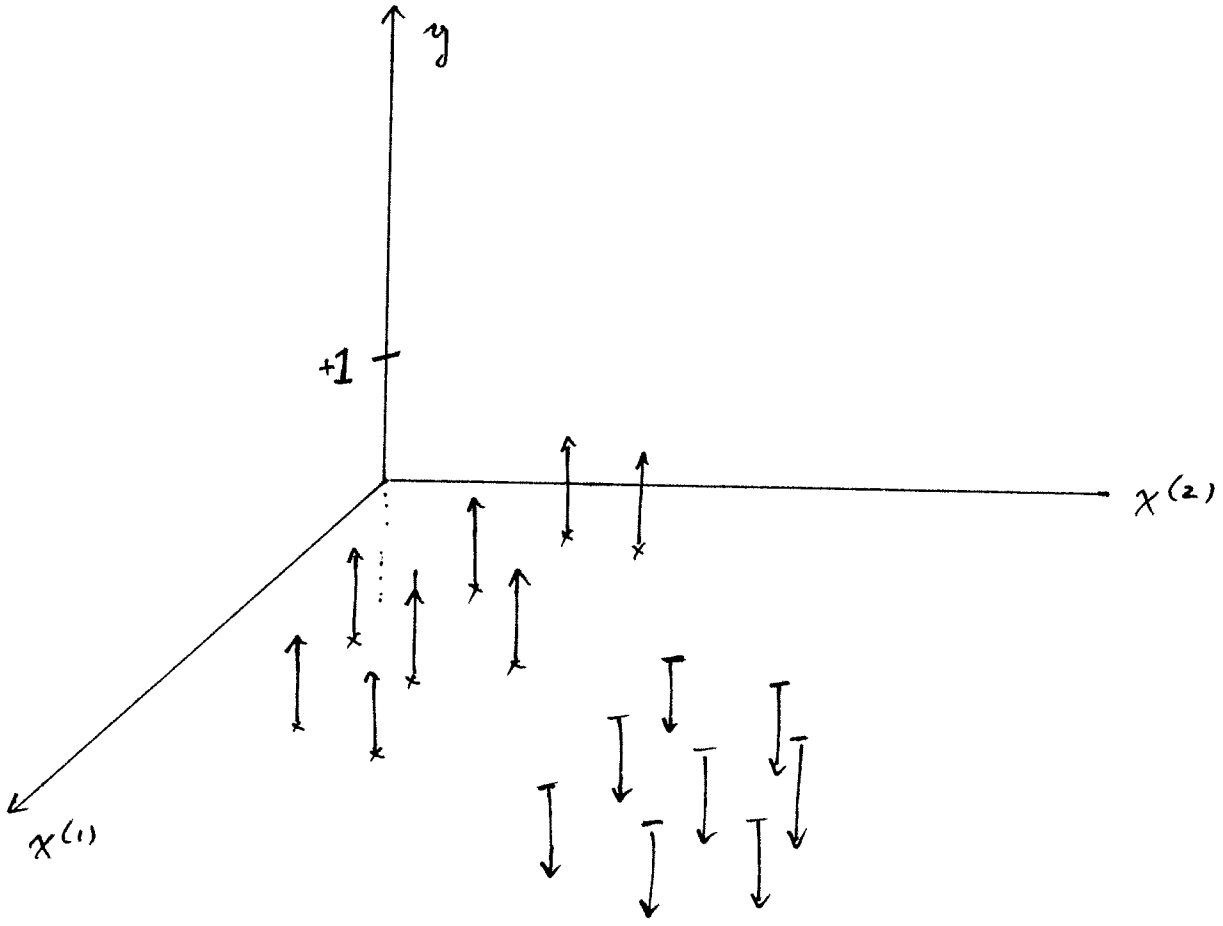
where h is a deterministic function and ϵ is zero-mean noise. The goal is to estimate the regression function

$$h(x) = E\{Y \mid X = x\}.$$

In linear regression it is assumed that

$$h(x) = w^T x + b$$

To apply linear regression to classification we treat the labels $y_i \in \{-1, 1\}$ as the response variables in a regression problem.



True, the linear assumption is flawed, but we are only interested in the final decision

$$\text{sign}\{w^T x + b\}.$$

The standard approach to linear regression is to minimize the squared error

$$\sum \left(y_i - (w^T x_i + b) \right)^2.$$

Let's write this in matrix notation:

$$\underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \underline{w} = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \underline{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & & & \\ 1 & x_{n1} & \cdots & x_{nd} \end{bmatrix}$$

Then we seek $\underline{w}$ minimizing

$$\| \underline{y} - \underline{X}\,\underline{w} \|^2$$

$$\Longrightarrow \quad \underline{w} = \left(\underline{X}^T \underline{X} \right)^{-1} \underline{X}^T \underline{y}$$

← pseudo inverse

Fact: If we take

$$y_i \in \left\{ -\frac{n}{n_-}, \frac{n}{n_+} \right\}$$

then the resulting classifier coincides with LDA and Fisher's linear discriminant.

Logistic Regression

LDA generates a linear classifier by assuming the class-conditional distributions are Gaussian (with equal covariance). Logistic regression also gives rise to a linear classifier, but models the posterior class probabilities.

Recall the Bayes classifier:

$$f^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq \frac{1}{2} \\ 0 & \text{if } \eta(x) < \frac{1}{2} \end{cases}$$

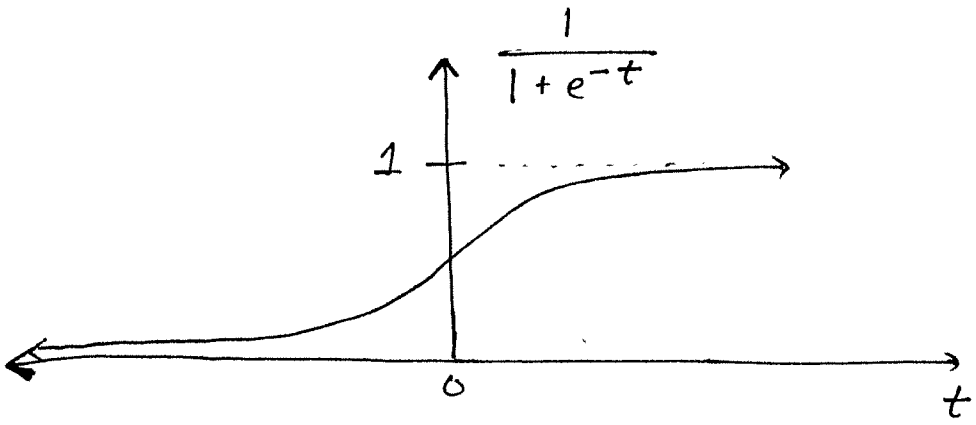
where

$$\eta(x) = \Pr\{ Y = 1 \mid X = x \}.$$

In logistic regression we assume

$$\eta(x) = \frac{1}{1 + \exp\{w^T x + b\}}$$

for some w, b. The goal is to find w, b, given training data, so that resulting classifier has low error probability.



With this model, the <u>plug in</u> classifier assigns class 1

$$\iff \eta(x) \geq \frac{1}{2} \iff \exp\{w^T x + b\} \leq 1$$

$$\iff w^T x + b \leq 0$$

So the classifier is <u>linear</u>:

$$\hat{f}(x) = \begin{cases} 1 & \text{if } w^T x + b \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

Maximum Likelihood Estimation

Typically the LR parameters are obtained by maximizing the conditional likelihood of Y/X:

$$\ell(\theta) = \prod_{i=1}^{n} \Pr\{Y = y_i \mid X = x_i, \theta\}$$

where $\theta = (w, b)$.

Exercise | Express the log-likelihood in terms of w, b. Simplify as much as possible.

Solution

$$\ell(\theta) = \prod_{i=1}^{n} \eta(x_i|\theta)^{y_i} \left(1-\eta(x_i|\theta)\right)^{1-y_i}$$

$$\Rightarrow \log \ell(\theta) = \sum_{i=1}^{n} y_i \log \eta(x_i|\theta) + (1-y_i)\log\left(1-\eta(x_i|\theta)\right)$$

Now $$\log \eta(x_i|\theta) = -\log\left(1+e^{w^T x_i + b}\right)$$

and $$\log\left(1-\eta(x_i|\theta)\right) = \log\left(\frac{e^{w^T x_i + b}}{1+e^{w^T x_i + b}}\right)$$

$$= w^T x_i + b - \log\left(1+e^{w^T x_i + b}\right)$$

So

$$\log \ell(\theta) = \sum_{i=1}^{n} y_i \left[-\log\left(1+e^{w^T x_i + b}\right)\right] + (1-y_i)\left[w^T x_i + b - \log\left(1+e^{w^T x_i + b}\right)\right]$$

$$= \sum_{i=1}^{n} (1-y_i)(w^T x_i + b) + \log\left(1+e^{w^T x_i + b}\right)$$

To maximize the likelihood, we can try setting the derivative equal to zero:

$$\frac{\partial \log l(\theta)}{\partial \theta} = \sum_{i=1}^{n} x_i \left((1-y_i) - \frac{e^{w^T x_i + b}}{1 + e^{w^T x_i + b}} \right)$$

$$= \sum_{i=1}^{n} x_i \left((1-y_i) - (1 - \eta(x_i | \theta)) \right)$$

$$= \sum_{i=1}^{n} x_i \left(\eta(x_i | \theta) - y_i \right) = 0$$

However, this is a <u>nonlinear</u> system of equations.

Typically, the log-likelihood is maximized using the Newton-Raphson algorithm:

$$\theta^{new} = \theta^{old} - \left(\frac{\partial^2 \log l(\theta)}{\partial \theta \, \partial \theta^T} \right)^{-1} \frac{\partial \log l(\theta)}{\partial \theta}$$

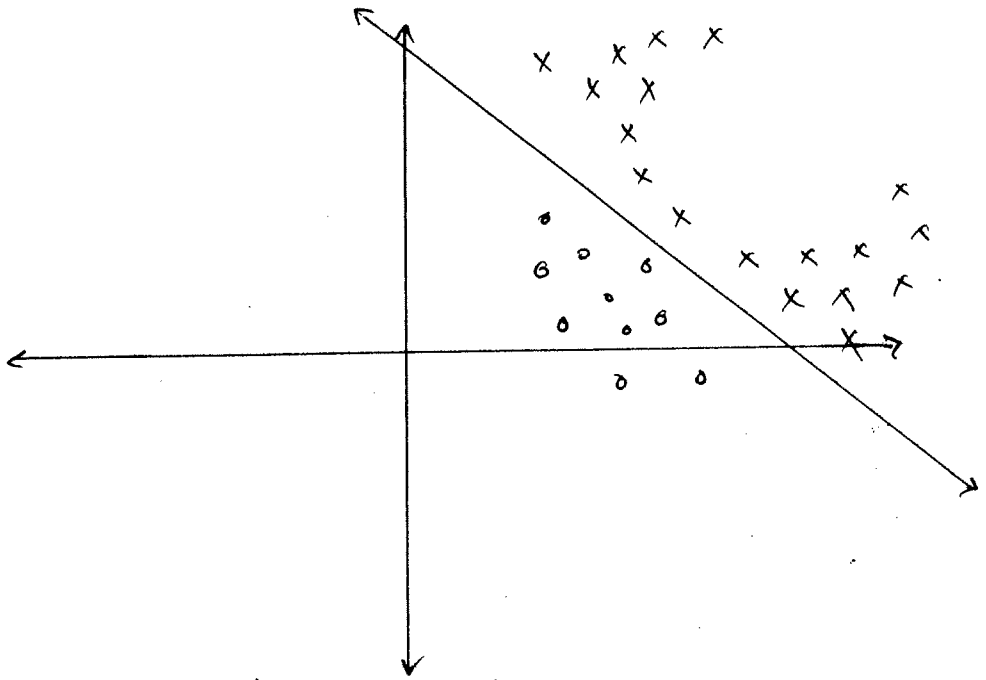
↑ Hessian matrix

Fortunately, the log-likelihood is concave, and so the algorithm converges, although overshooting can occur.

Separating Hyperplanes

So far the methods we have discussed for linear classification are plug-in rules or equivalent to plug-in rules. We will now consider methods that bypass distributional modeling and seek to estimate the decision boundary directly.

A <u>separating hyperplane</u> is any hyperplane that perfectly classifies the training data



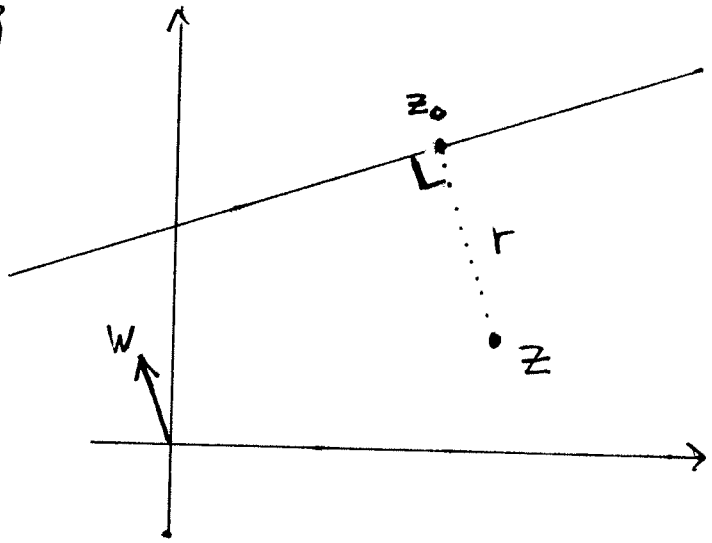
Assume such a hyperplane exists. How can we find one?

Some Geometry

Let $z \in \mathbb{R}^d$ and let w, b define a hyperplane

Question How far is z from the hyperplane,

$$\{x \in \mathbb{R}^d : w^T x + b = 0\}$$



Answer Write

$$z = z_0 + r \cdot \frac{w}{\|w\|}$$

where $w^T z_0 + b = 0$ and r may be negative.

Then
$$z + b = w^T z_0 + b + w^T \left(r \frac{w}{\|w\|} \right)$$

$$= r \cdot \|w\|$$

$$\Rightarrow \boxed{r = \frac{w^T z + b}{\|w\|}}$$

"signed distance"

Rosenblatt's Perceptron

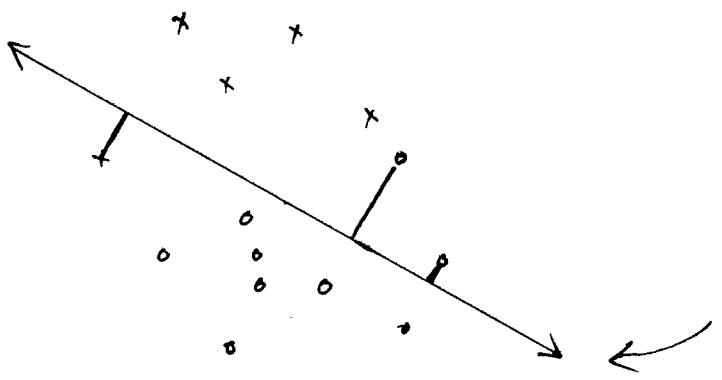
The perceptron learning algorithm seeks to minimize the total distance of misclassified points from the decision boundary.

Assume the classes are labeled $+1$ and -1. Then x_i is misclassified if and only if

$$y_i \left(w^T x_i + b \right) < 0.$$

Let $\mathcal{M}$ denote the set of misclassified points. Then the total (unsigned) distance of the the misclassified points to the decision boundary is proportional to

$$D(w, b) = -\sum_{i \in \mathcal{M}} y_i \left(w^T x_i + b \right)$$



current estimate of hyperplane

The perceptron learning algorithm minimizes $D(w,b)$ using stochastic gradient descent

The gradient of D is given by

$$\frac{\partial D}{\partial w} = -\sum_{i \in M} y_i x_i$$

$$\frac{\partial D}{\partial b} = -\sum_{i \in M} y_i$$

Instead of stepping in the opposite direction of the gradient, we visit each point in M in some (random) order and update the hyperplane with each step:

$$w^{new} = w^{old} + \gamma \cdot y_i x_i$$

$$b^{new} = b^{old} + \gamma \cdot y_i$$

where $\gamma > 0$ is the learning rate.

Facts

- If the data is linearly separable, then a separating hyperplane is obtained after a finite number of steps.

- This finite number can be large, especially when the gap between classes is small.

- The final solution depends on the initialization

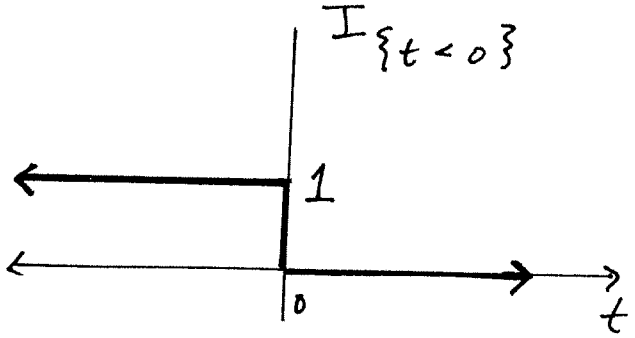
- If the data are not separable, the algorithm will not converge.

Other Gradient Descent Methods

Ideally we would like to select w, b to minimize the training error

$$\frac{1}{n}\sum_{i=1}^{n} I_{\{y_i(w^T x_i + b) < 0\}}.$$

Unfortunately, the function $I_{\{t<0\}}$ is not differentiable:



The basic idea is to replace $I_{\{t<0\}}$ by a function $\phi(t)$ that has qualitatively similar properties to $I_{\{t<0\}}$ but is differentiable or convex.

Then we can minimize

$$\frac{1}{n} \sum_{i=1}^{n} \phi(y_i (w^T x_i + b))$$

by gradient descent.

The function $\phi(t)$ is called a <u>loss function</u>.

Examples

1. Sigmoid loss

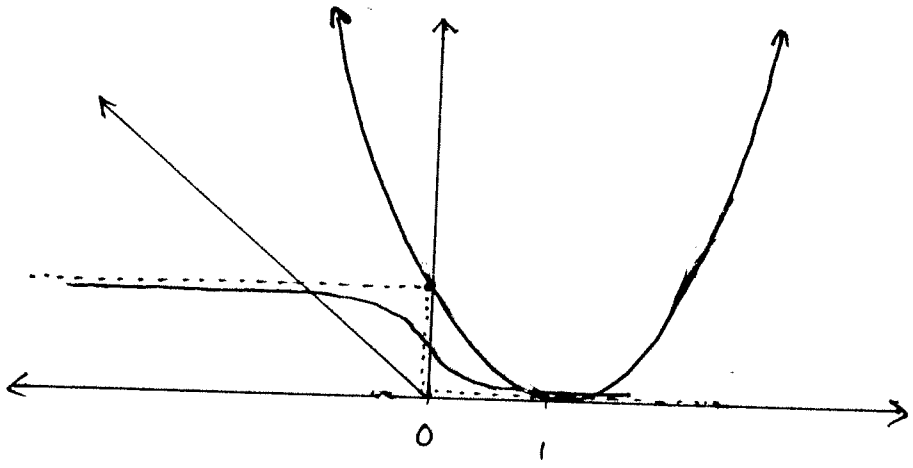
$$\phi(t) = \frac{1}{1+e^t}$$

2. hinge loss

$$\phi(t) = \begin{cases} 0 & \text{if } t \geq 0 \\ -t & \text{if } t < 0 \end{cases}$$

3. squared error loss

$$\phi(t) = (t-1)^2$$



Connections

- hinge loss + stochastic gradient descent

  $\Longrightarrow$ perceptron

- squared error loss

$$\left(y_i\left(w^T x_i + b\right) - 1\right)^2$$

$$= \left(\left(w^T x_i + b\right) - y_i\right)^2$$

$\Longrightarrow$ least squares regression.

When optimized by gradient descent, the algorithm is called the Widrow-Hoff/LMS rule.

- sigmoid $\Longrightarrow$ 1 layer neural network.

Remark

The perceptron algorithm finds a separating hyperplane if one exists, but otherwise it fails to converge.

The Widrow-Hoff / LMS algorithm always converges to the LMS optimal solution, but this is not necessarily a separating hyperplane even when one exists.

The best of both worlds is obtained by the Ho-Kashyap algorithm which is guaranteed to converge and find a separating hyperplane when one exists.

See Duda, Hart, and Stork (2001) for details.

Linear Programming

A *linear program* has the form

$$\min_{u} \quad c^T u$$
$$\text{s.t.} \quad Au \leq a$$

$\leq$ is interpreted component-wise

Here $u \in \mathbb{R}^p$, $c \in \mathbb{R}^p$

$A \in \mathbb{R}^{q \times p}$, $a \in \mathbb{R}^q$

It turns out we can compute a separating hyperplane using linear programming. Let's see how.

Recall the perceptron criterion:

$$D(w,b) = -\sum_{i \in M} y_i (w^T x_i + b)$$

proportional to (negative) distance to hyperplane

where M indexes the points that are misclassified.

As discussed previously, we may write

$$D(w,b) = \frac{1}{n}\sum_{i=1}^{n} \phi\left(y_i\left(w^T x_i + b\right)\right)$$

where

$$\phi(t) = \max\{-t, 0\} = \begin{cases} -t, & t \le 0 \\ 0, & t > 0 \end{cases}$$

This is optimized by solving

minimize sum of violations

$$\min_{w,b,\xi} \quad \frac{1}{n}\sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad y_i(w^T x_i + b) \geq -\xi_i$$

$$\xi_i \geq 0$$

which is a linear program. Unfortunately, there is a problem with this LP.

$w = 0$, $b = 0$, $\xi = 0$ is a trivial solution! This is not a useful hyperplane!

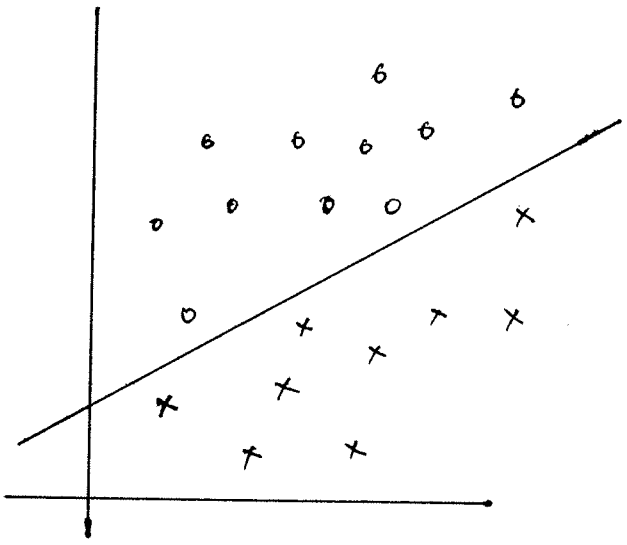
The constraint $w \neq 0$ is difficult to incorporate directly into an LP, so we proceed as follows:

If $(x_1, y_1), \ldots, (x_n, y_n)$ is linearly separable, then

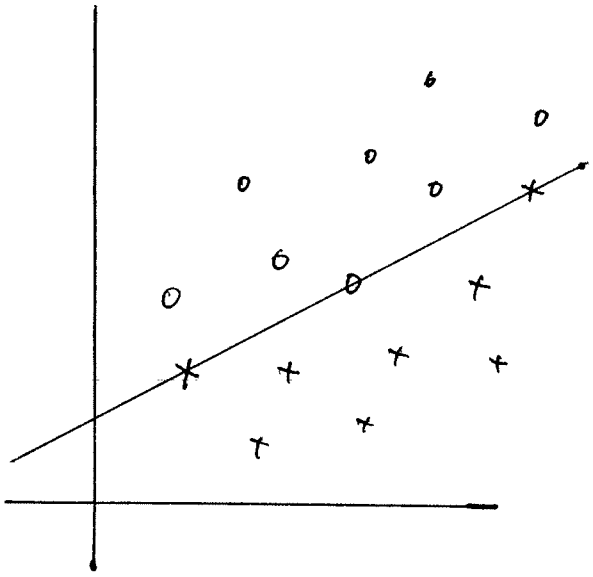
$$y_i (w^T x_i + b) > 0 \qquad \forall i$$

for some w, b.

linearly separable



not linearly separable



By rescaling w and b, a separating hyperplane satisfies

$$y_i (w^T x_i + b) \geq 1 \qquad \forall i.$$

The idea now is to minimize the sum of the violations of this constraint:

$$\min_{w, b, \xi} \ \frac{1}{n} \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad y_i (w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

This linear program enjoys the following properties: (Let w^*, b^*, ξ^* be the optimal solution)

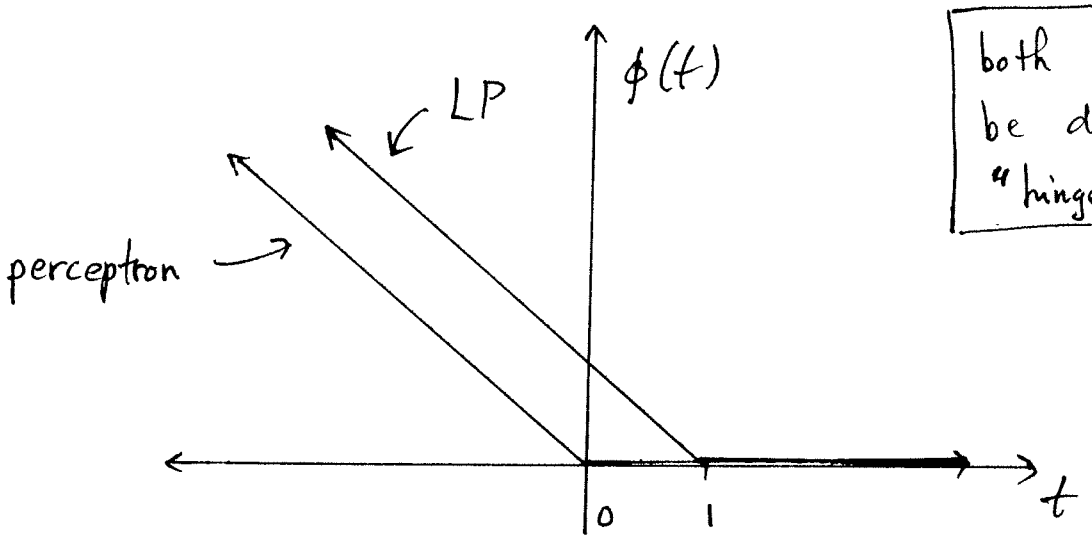
- A separating hyperplane exists iff $\sum \xi_i^* = 0$.
  If one exists, it will be found.
- If a separating hyperplane does not exist, w^*, b^* is still sensible
- $w^* = 0$ is not a solution.

Note that the perceptron corresponds to

$$\phi(t) = \max\{-t, 0\}$$

while our modified criterion corresponds to

$$\phi(t) = \max\{-(t-1), 0\}$$



both losses may be described as "hinge" losses

Unbalanced data

If the data is unbalanced ($n_- \gg n_+$ or $n_+ \gg n_-$) is advisable to change the objective function to

$$\frac{1}{n_-} \sum_{i \in I_-} \xi_i + \frac{1}{n_+} \sum_{i \in I_+} \xi_i$$

so that the minority class is not neglected.

Exercise] Recall the standard form of a linear program:

$$\begin{aligned} \min \quad & c^T u \\ \text{s.t.} \quad & Au \leq a \end{aligned}$$

Express the LP for linear classification in this form.

Solution The variable is

$$u = [w_1 \; w_2 \; \dots \; w_d \; b \; \xi_1 \; \xi_2 \; \dots \; \xi_n]^T.$$

The objective function is $c^T u$ where

$$c = [0 \; 0 \; \dots \; 0 \; 0 \; \tfrac{1}{n} \; \tfrac{1}{n} \; \dots \; \tfrac{1}{n}]^T.$$

The constraint is

$$-\begin{bmatrix} y_1 x_{11} & \dots & y_1 x_{1d} & y_1 & 1 & 0 & 0 & \dots & 0 \\ y_2 x_{21} & \dots & y_2 x_{2d} & y_2 & 0 & 1 & 0 & \dots & 0 \\ \vdots & & & & & & & & \\ y_n x_{n1} & \dots & y_n x_{nd} & y_n & 0 & 0 & 0 & \dots & 1 \\ & & & & 1 & & & 0 & \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & 0 & & \ddots & & \\ & & & & & & & 1 & \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \\ b \\ \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} \leq -\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

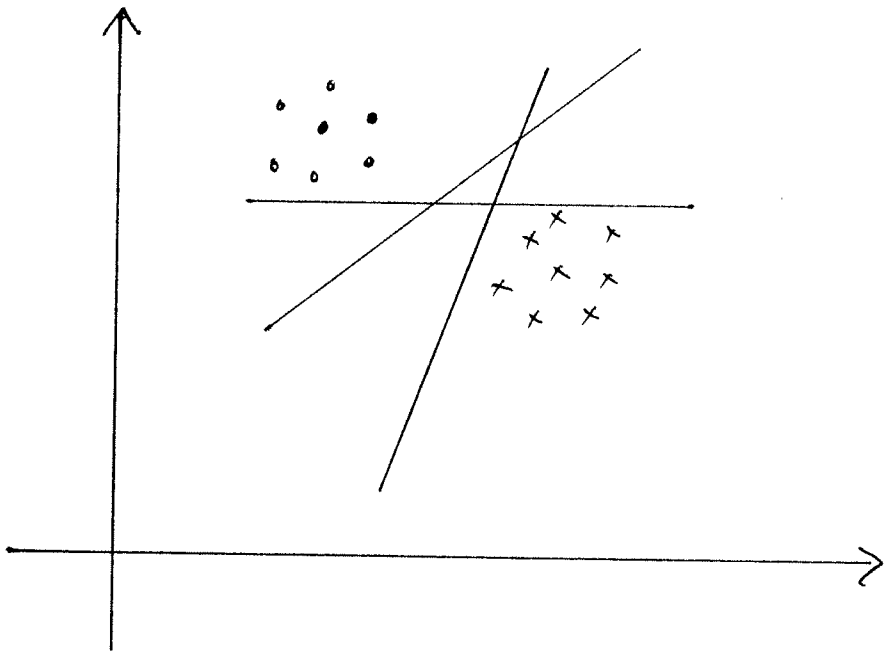
A $\quad$ u $\quad$ a

$2n \times (d+1+n)$ $\quad$ $(d+1+n) \times 1$ $\quad$ $2n \times 1$

Historical note: The LP linear classifier was developed by Olvi Mangasarian in 1965.

The Optimal Separating Hyperplane

The methods described thus far for finding a separating hyperplane do not pay attention to <u>which</u> hyperplane is produced. Yet if one separating hyperplane exists, then infinitely many do. So which one should we choose?



The Max-Margin Principle

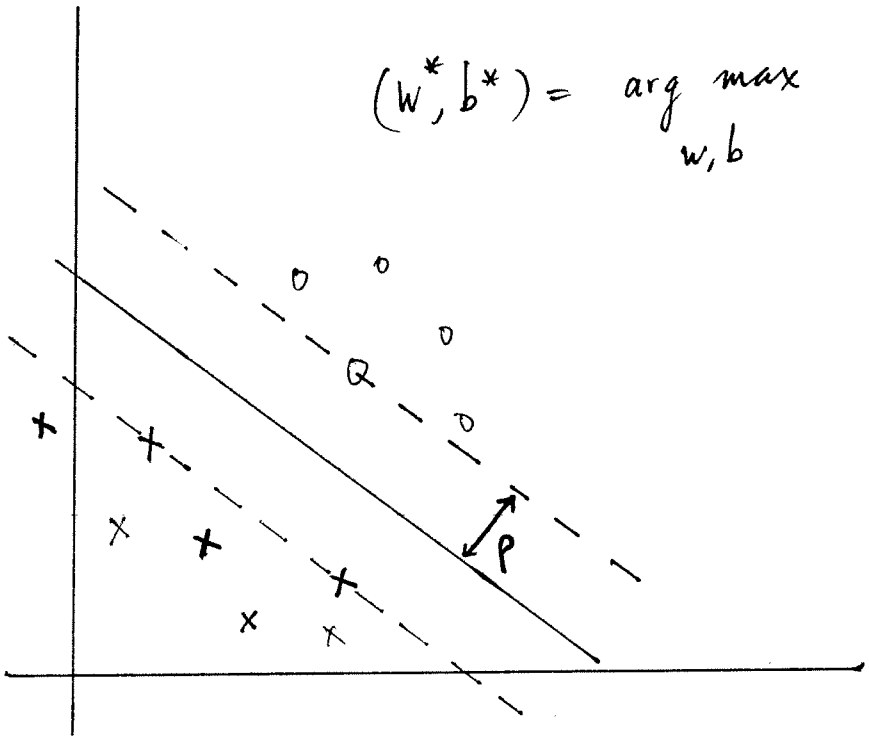
Definitions:

1. The <u>margin</u> ρ of a separating hyperplane is the distance from the hyperplane to the closest x_i

$$\rho(w,b) := \min_{i=1,\dots,n} \frac{|w^T x_i + b|}{\|w\|}$$

2. The <u>optimal separating hyperplane</u> is the separating hyperplane whose margin is maximal

$$(w^*, b^*) = \arg\max_{w,b} \rho(w,b)$$



larger margin $\Rightarrow$ better generalization

Canonical form

We may rescale any separating hyperplane so that it is in canonical form:

$$y_i (w^T x_i + b) \geq 1 \quad \forall i$$

$$y_i (w^T x_i + b) = 1 \quad \text{for some } i.$$

} canonical form for w, b.

Exercise Express the margin of a hyperplane in canonical form as a function of w and b.

Solution

$$\rho(w,b) = \min_{i=1,\dots,n} \frac{|w^T x_i + b|}{\|w\|} = \frac{1}{\|w\|}$$

Terminology: Those x_i such that $y_i(w^T x_i + b) = 1$ are called support vectors

Therefore, the optimal separating hyperplane is the solution of

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$\text{s.t. } y_i(w^T x_i + b) \geq 1, \quad i = 1,\dots,n$$

This is a quadratic program.

Equivalent terms:

- optimal separating hyperplane
- optimal margin "
- maximal (max) margin "
- linear support vector machine (separable case)

Nonseparable Data: The Optimal Soft-Margin Hyperplane

To accomodate nonseparable data, the optimal margin QP is modified by introducing slack variables $\xi_1, \ldots, \xi_n \geq 0$:

The results in the so-called optimal soft-margin hyperplane:

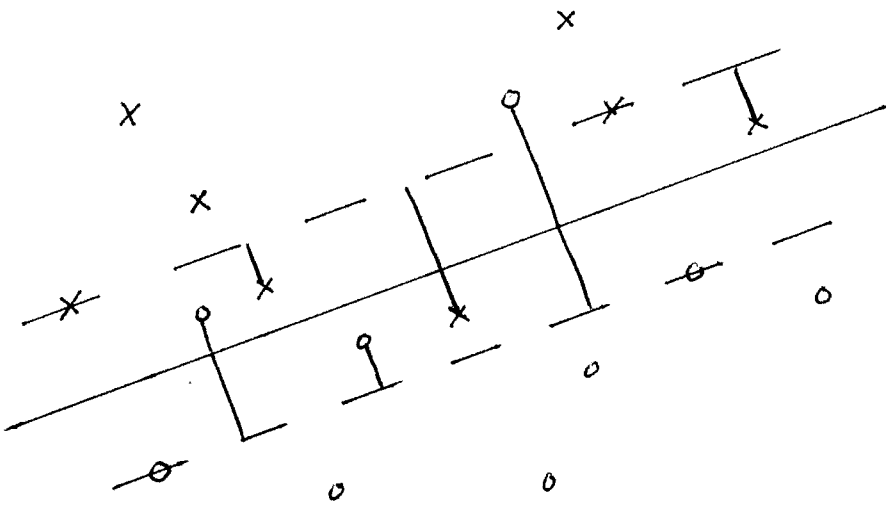
$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i\left(w^T x_i + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, n$$

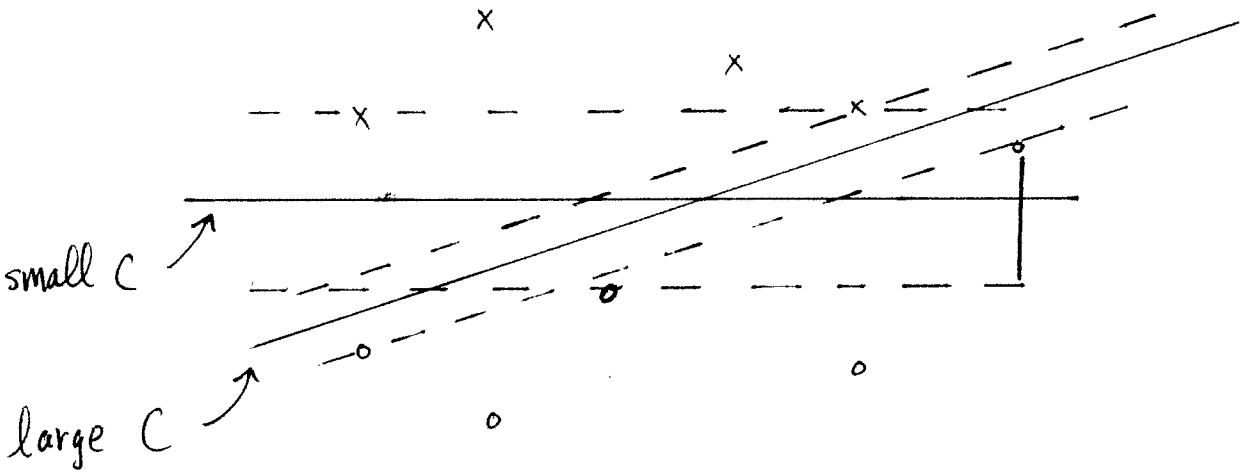
$$\xi_i \geq 0, \quad i = 1, \ldots, n$$

That is we allow points to lie within the margin, and try to maximize the margin while simultaneously minimizing the total margin violations.

$C > 0$ is a tradeoff parameter.



Even if the data is separable, the soft-margin classifier may not yield a separating hyperplane. It depends on C. For large enough C, a separating hyperplane will be found.



Therefore, smaller C may allow the optimal soft-margin hyperplane to be <u>robust</u> w/ respect to outliers.

Taking C too small can be dangerous, however.
What happens as $C \to 0$?

Summary of Linear Classifiers

<u>Plug in</u>

- LDA: equivalent to { Fisher's linear discriminant / LMS regression }
- Logistic regression

<u>Gradient descent</u>

- Perceptron (hinge loss)
- Widrow-Hoff/LMS (squared error loss)
- 1 layer neural net (sigmoid loss)

<u>Optimization Theoretic</u>

- Linear programming (perceptron w/ shifted hinge loss)
- Quadratic programming (optimal soft margin hyperplane)