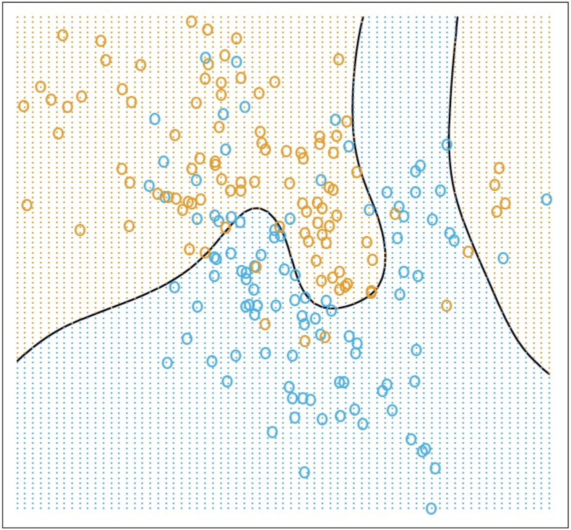
Topic 3: INTRO TO CLASSIFICATION

STAT 37710/CAAM 37710/CMSC 35400 Machine Learning
Risi Kondor, The University of Chicago



[Hastie, Tibshirani & Friedman]

Classification

Two-class classification in $\mathbb{R}^n$ is the prototypical supervised ML problem.

- Input space: $\mathcal{X}$ (in the simplest case, $\mathcal{X} = \mathbb{R}^n$)
- Output space: $\mathcal{Y} = \{-1, +1\}$ (in k-class case $\mathcal{Y} = \{1, \ldots, k\}$)
- Training set: $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$
- Goal: find a good hypothesis $f : \mathcal{X} \to \mathcal{Y}$.

We will look at four simple classifiers:

1. Gaussian Discriminant Analysis
2. Logistic regression
3. k –nearest neighbors
4. The perceptron

1. Gaussian discriminant analysis

Model based classification

1. Assume that $\mathbf{x}$ and y come from a joint distribution $p(\mathbf{x}, y)$ (similar to mixture models for clustering).
2. Define a marginal distribution for y (Bernoulli):

$$p_0(y = +1) = \pi \qquad p_0(y = -1) = 1 - \pi.$$

3. Define the conditionals for $\mathbf{x}$ given y :

$$p(\mathbf{x} | y = +1) = p_{+1}(\mathbf{x})$$
$$p(\mathbf{x} | y = -1) = p_{-1}(\mathbf{x}).$$

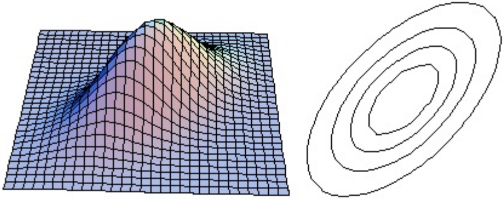
4. Estimate the parameters of p_+ , p_- and p_0 from the training data.
5. Classify future examples according to

$$\widehat{y} = \begin{cases} +1 & \text{if } p(y = +1 | \mathbf{x}) \geq p(y = -1 | \mathbf{x}) \\ -1 & \text{if } p(y = +1 | \mathbf{x}) < p(y = -1 | \mathbf{x}). \end{cases}$$

This is called the **Bayes optimal classifier**.

Gaussian Discriminant Analysis

Assume that $\mathcal{X} = \mathbb{R}^n$ and $p_+(\mathbf{x})$ and $p_-(\mathbf{x})$ are both multivariate normal (Gaussian) distributions:

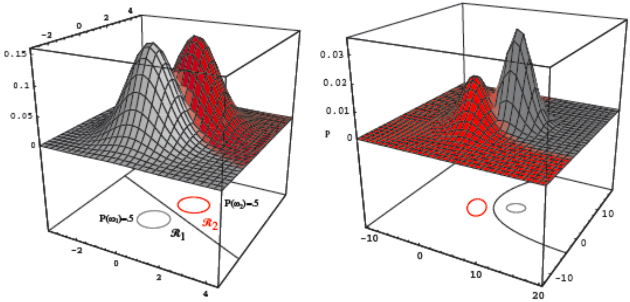


$$\boxed{\mathcal{N}(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})/2}}$$

where $\boldsymbol{\mu} \in \mathbb{R}^n$, and $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix.

$$\mathbb{E}(\mathbf{x}) = \boldsymbol{\mu} \qquad \text{Cov}(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\Sigma}_{i,j}$$

Gaussian Discriminant Analysis



$$p(\mathbf{x}, y) = \pi^{(1+y)/2}\,(1-\pi)^{(1-y)/2}\,\frac{1}{(2\pi)^{n/2}\,|\boldsymbol{\Sigma}_y|^{1/2}}\exp\left(-\frac{(\mathbf{x}-\boldsymbol{\mu}_y)^\top \boldsymbol{\Sigma}_y^{-1}(\mathbf{x}-\boldsymbol{\mu}_y)}{2}\right)$$

How do we learn (estimate) the parameters $\{\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}_{+1}, \boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}_{-1}, \pi\}$?

Easier than clustering because y is not latent, so no need for EM.

Likelihood for GDA

$$L(\pi, \boldsymbol{\mu}_{+1}, \boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}_{+1}, \boldsymbol{\Sigma}_{-1}) = \prod_{i=1}^{m} \pi^{(1+y_i)/2}(1-\pi)^{(1-y_i)/2} \dots$$

$$\frac{1}{(2\pi)^{n/2}|\,\boldsymbol{\Sigma}_{y_i}\,|^{1/2}}\ \exp\Big(-\big(\mathbf{x}_i - \boldsymbol{\mu}_{y_i}\big)^\top \boldsymbol{\Sigma}_{y_i}^{-1}\big(\mathbf{x}_i - \boldsymbol{\mu}_{y_i}\big)/2\Big)$$

Log-likelihood for GDA

$$\ell(\pi, \boldsymbol{\mu}_{+1}, \boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}_{+1}, \boldsymbol{\Sigma}_{-1}) = \textsf{cst} + \sum_{i=1}^{m} \left[\frac{1+y_i}{2} \log \pi + \frac{1-y_i}{2} \log(1-\pi) \right]$$
$$+ \sum_{i\,:\,y_i=1} \left[-\frac{1}{2} \log |\,\boldsymbol{\Sigma}_{+1}\,| - \frac{(\mathbf{x}-\boldsymbol{\mu}_{+1})^\top \boldsymbol{\Sigma}_{+1}^{-1} (\mathbf{x}-\boldsymbol{\mu}_{+1})}{2} \right]$$
$$+ \sum_{i\,:\,y_i=-1} \left[-\frac{1}{2} \log |\,\boldsymbol{\Sigma}_{-1}\,| - \frac{(\mathbf{x}-\boldsymbol{\mu}_{-1})^\top \boldsymbol{\Sigma}_{-1}^{-1} (\mathbf{x}-\boldsymbol{\mu}_{-1})}{2} \right]$$

Observation: the part of the likelihood relating to the $y=+1$ Gaussian completely separates from the part relating to the $y=-1$ Gaussian, which, in turn, separates from the Bernoulli part. So these three parts can be maximized separately.

MLE for two-Gaussians model

- MLE for π :

$$\widehat{\pi} = \frac{n_+}{n} = \frac{\text{number of traning examples with } y_i = 1}{\text{total number of training examples}}$$

- MLE for $(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}_{+1})$ and $(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}_{-1})$:

$$\widehat{\boldsymbol{\mu}}_{+1} = \frac{1}{n_{+1}} \sum_{i\,:\,y_i=+1} \mathbf{x}_i \qquad \widehat{\boldsymbol{\Sigma}}_{+1} = \frac{1}{n_{+1}} \sum_{i\,:\,y_i=+1} (\mathbf{x}_i - \boldsymbol{\mu}_{+1})(\mathbf{x}_i - \boldsymbol{\mu}_{+1})^\top$$

$$\widehat{\boldsymbol{\mu}}_{-1} = \frac{1}{n_{-1}} \sum_{i\,:\,y_i=-1} \mathbf{x}_i \qquad \widehat{\boldsymbol{\Sigma}}_{-1} = \frac{1}{n_{-1}} \sum_{i\,:\,y_i=-1} (\mathbf{x}_i - \boldsymbol{\mu}_{-1})(\mathbf{x}_i - \boldsymbol{\mu}_{-1})^\top$$

2. Logistic Regression

Logistic regression

- Assume that the conditional of y is

$$\mathbb{P}(y=1|\mathbf{x}) = h(\mathbf{x})$$
$$\mathbb{P}(y=-1|\mathbf{x}) = 1 - h(\mathbf{x}).$$

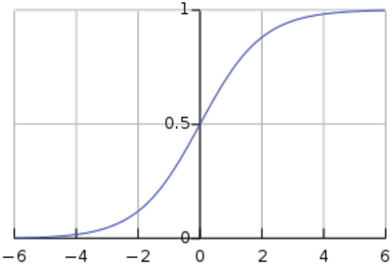
  In other words $y|\mathbf{x} \sim \mathrm{Bernoulli}(h(\mathbf{x})).$

- Set $h(\mathbf{x})$ to be a nice function $\mathbb{R}^d \to [0,1]$, in particular the **logistic function** (sigmoid function of $\boldsymbol{\theta} \cdot \mathbf{x}$)

$$h(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta} \cdot \mathbf{x}}}\,.$$

This is similar to how in model based regression we took $y|\mathbf{x} \sim \mathcal{N}(f(\mathbf{x}), \sigma^2))$.

The sigmoid function



The sigmoid function $g(z) = \frac{1}{1+e^{-z}}$ has the nice property that

$$\begin{aligned} g'(z) &= \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) = \frac{1}{(1+e^{-z})^2}\, e^{-z} = \frac{1}{1+e^{-z}}\, \frac{e^{-z}}{1+e^{-z}} \\ &= g(z)\, \frac{(1+e^{-z})-1}{1+e^{-z}} = g(z)\,(1-g(z)). \end{aligned}$$

MLE for logistic regression

Defining for convenience $u_i = \frac{1+y_i}{2}$, the likelihood is

$$L(\theta) = \prod_{i=1}^{m} h(\mathbf{x}_i)^{u_i} \, (1 - h(\mathbf{x}_i))^{1-u_i}.$$

The log-likelihood is

$$\ell(\theta) = \sum_{i=1}^{m} \big[u_i \log(h(\mathbf{x}_i)) + (1 - u_i) \log(1 - h(\mathbf{x}_i))\big].$$

Unlike in GDA, this *cannot* be maximized in closed form. $\rightarrow$ Try stochastic gradient descent.

Gradient descent

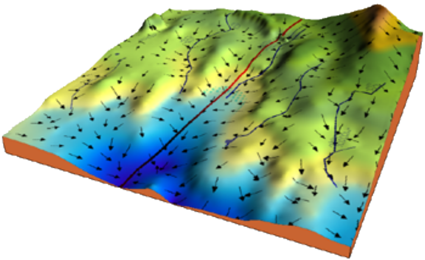
Given a loss function J, we can optimize it using gradient descent:

$$
\begin{aligned}
&\boldsymbol{\theta} \leftarrow \mathbf{0} \\
&\texttt{until(convergence)\{} \\
&\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla J(\boldsymbol{\theta}) \\
&\texttt{\}}
\end{aligned}
$$

α : a parameter called the **learning rate.**

Gradient descent is the “poor man’s algorithm” for optimization. Works in almost any case, but sometimes not very well. Choosing α can be tricky.

Recap: the gradient

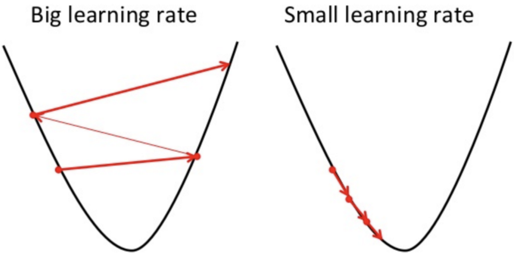


The **gradient** of a function $J : \mathbb{R}^p \to \mathbb{R}$ is the *vector* $\nabla J(\boldsymbol{u})$ with

$$[\nabla J(\boldsymbol{u})]_i = \frac{\partial}{\partial u_i} J(\boldsymbol{u}).$$

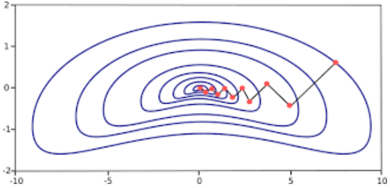
The gradient always points in the most uphill direction possible (picture shows negative gradient). At a minimum/maximum (or saddle point) $\nabla J = 0$.

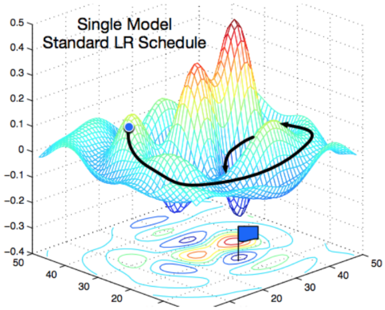
The learning rate α



Setting the learning rate can be quite tricky. Decrease learning rate as we get closer to the optimum?

Gradient descent





Gradient descent trajectories can be more complicated than you would expect.

Gradient of $\ell(\theta)$ in gory detail

$$\frac{\partial}{\partial[\boldsymbol{\theta}]_j}\ell(\theta) = \left(u\frac{1}{g(\boldsymbol{\theta}\cdot\mathbf{x})} - (1-u)\frac{1}{1-g(\boldsymbol{\theta}\cdot\mathbf{x})}\right)\frac{\partial}{\partial\theta_j}\,g(\boldsymbol{\theta}\cdot\mathbf{x})$$

$$= \left(u\frac{1}{g(\boldsymbol{\theta}\cdot\mathbf{x})} - (1-u)\frac{1}{1-g(\boldsymbol{\theta}\cdot\mathbf{x})}\right) g(\boldsymbol{\theta}\cdot\mathbf{x})(1-g(\boldsymbol{\theta}\cdot\mathbf{x}))\frac{\partial}{\partial\theta_j}\,(\boldsymbol{\theta}\cdot\mathbf{x})$$

$$= \left(u\frac{1}{g(\boldsymbol{\theta}\cdot\mathbf{x})} - (1-u)\frac{1}{1-g(\boldsymbol{\theta}\cdot\mathbf{x})}\right) g(\boldsymbol{\theta}\cdot\mathbf{x})(1-g(\boldsymbol{\theta}\cdot\mathbf{x}))\,[\mathbf{x}]_i$$

$$= \frac{u(1-g(\boldsymbol{\theta}\cdot\mathbf{x})) - (1-u)\,g(\boldsymbol{\theta}\cdot\mathbf{x})}{g(\boldsymbol{\theta}\cdot\mathbf{x})(1-g(\boldsymbol{\theta}\cdot\mathbf{x}))}\,g(\boldsymbol{\theta}\cdot\mathbf{x})(1-g(\boldsymbol{\theta}\cdot\mathbf{x}))\,[\mathbf{x}]_i$$

$$= (u(1-g(\boldsymbol{\theta}\cdot\mathbf{x})) - (1-u)\,g(\boldsymbol{\theta}\cdot\mathbf{x}))\,[\mathbf{x}]_i$$

$$= (u - g(\boldsymbol{\theta}\cdot\mathbf{x}))\,x_i \qquad \Rightarrow \qquad \nabla\ell(\theta) = (u - h(\mathbf{x}))\,\mathbf{x}$$

SGD for logistic regression

$\boldsymbol{\theta} \leftarrow \mathbf{0}$

$\texttt{until(convergence)\{}$

$\quad\texttt{for(}\ j = 1\ \ \texttt{to}\ \ m\ \texttt{)}$

$\qquad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\Big[(h(\mathbf{x}_i) - u_i)\ \mathbf{x}_i\Big]$

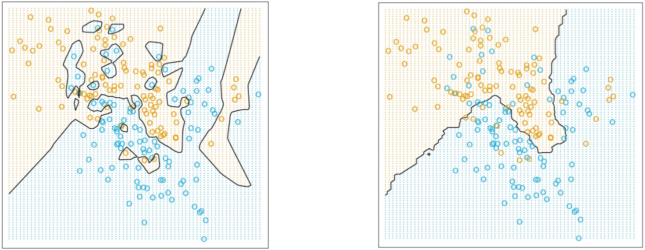
$\quad\texttt{\}}$

$\texttt{\}}$

Stochastic gradient descent performs gradient descent wrt the loss (negative likelihood) of individual examples or small randomly selected sets of examples, called minibatches.

3. k –nearest neighbors

k-nearest neighbors



1-nearest neighbor　　　15-nearest neighbor

The hypothesis returned by the k-NN classifier is

$$\widehat{f}(\mathbf{x}) = \operatorname{sgn}\left(\textstyle\sum_{i \in \mathrm{NN}_k(\mathbf{x})} y_i\right),$$

where $\mathrm{NN}_k(\mathbf{x}) = \{i_1, \ldots, i_k\}$ are the indices of the k points in S closest to $\mathbf{x}$. (When k is even and $\sum_{i \in \mathrm{NN}_k(\mathbf{x})} y_i = 0$, break ties arbitrarily.)

What effect does k have on the behavior of k-NN?

3. The Perceptron

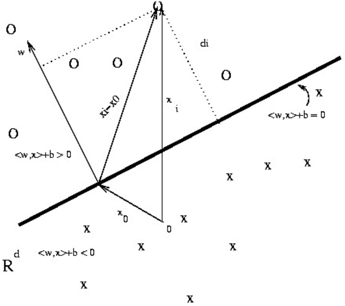
Linear classifiers

The Perceptron belongs to the wide class of linear classifiers.

Input space: $\mathcal{X} = \mathbb{R}^n$
Output space: $\mathcal{Y} = \{-1, +1\}$
Hypothesis (affine hyperplane):

$$f(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

- $\mathbf{w}$ is the normal to the **separating hyperplane**
- b is the **bias**

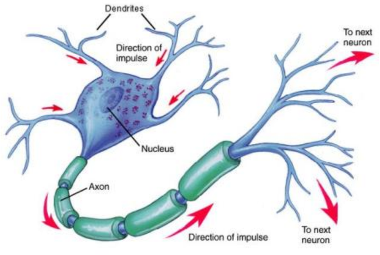


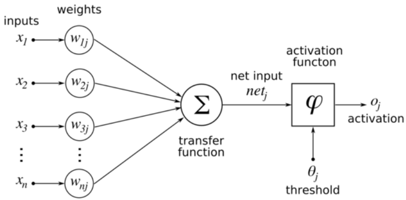
Empirical error : $\mathcal{E}_{\text{train}}(f) = \frac{1}{m} \sum_{i=1}^{m} \ell_{0/1}(f(\mathbf{x}_i), y_i)$

where $\ell_{0/1}(\widehat{y}, y) = 0$ if $\widehat{y} = y$, else $\ell_{0/1}(\widehat{y}, y) = 1$.

Of all possible hyperplanes that separate the data which one is best?

Inspiration: Artificial Neural Networks





$$o_j = \varphi(\theta_j + \sum_i w_{i,j}\, x_i),$$

An old idea going back to [McCulloch & Pitts, 1943].

The perceptron (Rosenblatt, 1958)

Consider the following a simplified scenario:

- $\|\mathbf{x}_i\| = 1$ for all i.
- Assume that all datapoints satisfy the ground truth concept

$$h_{\text{true}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{v} \cdot \mathbf{x} \geq 0 \\ -1 & \text{otherwise} \end{cases},$$

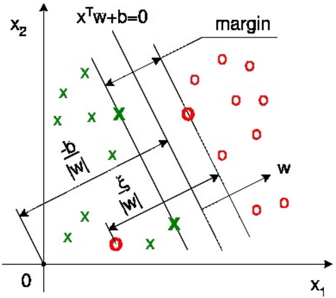
  where $\mathbf{v}$ is some fixed vector and w.l.o.g. $\|\mathbf{v}\| = 1$.
- Since h_{true} goes through the origin, set $b = 0$ for all h.
- We want to solve the problem in its online form.

The perceptron (Rosenblatt, 1958)

$\mathbf{w} \leftarrow 0$;
$t \leftarrow 1$;
`while(true){`
  `if` $\mathbf{w} \cdot \mathbf{x}_t \geq 0$ `predict` $\hat{y}_t = 1$; `else predict` $\hat{y}_t = -1$;
  `if` $((\hat{y}_t = -1)$ `and` $(y_t = 1))$ `let` $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$;
  `if` $((\hat{y}_t = 1)$ `and` $(y_t = -1))$ `let` $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$;
  $t \leftarrow t + 1$;
`}`

Note: In this form, the Perceptron is an **online** learning algorithm.

The margin



The projection of $\mathbf{x}$ onto the line orthogonal to decision boundary $\mathbf{w} \cdot \mathbf{x} + b = 0$ is

$$P\mathbf{x} = \mathbf{w} \cdot \mathbf{x} / \|\mathbf{w}\| .$$

On the boundary

$$\mathbf{w} \cdot \mathbf{x} = -b \quad \Rightarrow \quad P\mathbf{x} = \frac{-b}{\|\mathbf{w}\|}.$$

The margin of a correctly classified positive resp. negative example are

$$\delta_{(\mathbf{x},+1)} = P\mathbf{x} - \frac{-b}{\|\mathbf{w}\|}, \qquad \delta_{(\mathbf{x},-1)} = \frac{-b}{\|\mathbf{w}\|} - P\mathbf{x},$$

so the general formula is $\delta_{(\mathbf{x},y)} = y\,(\mathbf{w} \cdot \mathbf{x} + b)/\|\mathbf{w}\|$.

The perceptron convergence thm

Suppose that the perceptron is run on a sequence of examples with $\|\mathbf{x}_i\| = 1$ and $y_i(\mathbf{v} \cdot \mathbf{x}_i) \geq 0$ for some $\mathbf{v}$ with $\|\mathbf{v}\| = 1$. In addition, assume that the data obeys $|\mathbf{v} \cdot \mathbf{x}_i| \geq \delta$ for all $\mathbf{x}_t$ (**margin**) .

Claim 1: After M mistakes $\mathbf{w} \cdot \mathbf{v} \geq \delta M$.

Initially, $\mathbf{w} \cdot \mathbf{v} = 0$. After a false negative, $\mathbf{w} \leftarrow \boldsymbol{w} + \mathbf{x}_t$, and $(\mathbf{w} + \mathbf{x}_t) \cdot \mathbf{v} \geq (\mathbf{w} \cdot \mathbf{v}) + \delta$, so $\mathbf{w} \cdot \mathbf{v}$ increases by at least δ . Similarly for a false positive.

The perceptron convergence thm

Claim 2: After M mistakes, $\|\mathbf{w}\|^2 \leq M$

Initially $\|\mathbf{w}\| = 0$. On a false negative, $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$, and

$$\| \mathbf{w} + \boldsymbol{x} \|^2 = (\mathbf{w} + \mathbf{x}_t) \cdot (\mathbf{w} + \mathbf{x}_t) = \|\mathbf{w}\|^2 + 2\mathbf{w} \cdot \boldsymbol{x}_t + \|\mathbf{x}_t\|^2 .$$

The third term on the RHS is 1, while the second term must be negative, since we predicted 0. Therefore, $\|\mathbf{w}\|^2$ can increase by at most 1. Similarly for false positives.

The perceptron convergence thm

Theorem. If the perceptron is run a sequence of examples with $|\mathbf{v} \cdot \mathbf{x}_t| \geq \delta$ (plus our other assumptions), then it will make at most $1/\delta^2$ mistakes.

Proof.

$$\delta M \leq \mathbf{w} \cdot \mathbf{v} \leq \|\mathbf{w}\| \leq \sqrt{M}$$

$\blacksquare$

Affine hyperplanes

Question: What if we don't want the separator to go through the origin?

Add extra feature with $[\mathbf{x}_t]_{n+1} = 1$ for all t .

$$x_1 w_1 + x_2 w_2 + \dots x_n w_n \geq \theta$$

$$\iff$$

$$x_1 w_1 + x_2 w_2 + \dots x_n w_n + x_{n+1} w_{n+1} \geq 0$$

with $w_{n+1} = -\theta$

Question: How can we make linear threshold functions more expressive?

Towards multiclass

Rewrite perceptron as competition between $\mathbf{w}^+$ and $\mathbf{w}^-$

$\mathbf{w}^+ \leftarrow (0, 0, \ldots, 0)$;
$\mathbf{w}^- \leftarrow (0, 0, \ldots, 0)$;
$t \leftarrow 1$;
`while(1){`
  `if` $\mathbf{w}^+ \cdot \mathbf{x}_t \geq \mathbf{w}^- \cdot \mathbf{x}_t$ `predict` $\hat{y}_t = 1$; `else predict` $\hat{y}_t = 0$;
  `if` $((\hat{y}_t = 0)$ `and` $(y_t = 1))$ `let` $\mathbf{w}^+ \leftarrow \mathbf{w}^+ + \mathbf{x}_t$;
  `if` $((\hat{y}_t = 1)$ `and` $(y_t = 0))$ `let` $\mathbf{w}^- \leftarrow \mathbf{w}^- + \mathbf{x}_t$;
  $t \leftarrow t + 1$;
`}`

Towards multiclass

If $y_t \in \{1, 2, \ldots, \kappa\}$, maintain κ different weight vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_\kappa$.

On erroneously classifying example $\mathbf{x}_t$ as class i instead of class j :

- $\mathbf{w}_i \leftarrow \mathbf{w}_i - \mathbf{x}_t/2$
- $\mathbf{w}_j \leftarrow \mathbf{w}_j + \mathbf{x}_t/2$

[Collins & Duffy, 2002]