The Bayes classifier

Theorem
The classifier $h^*(\mathbf{x}) := \arg\max_k \eta_k(\mathbf{x})$ satisfies

$$R(h^*) = \min R(h)$$

where the min is over all possible classifiers.

To calculate the Bayes classifier/Bayes risk, we need to know $\eta_k(\mathbf{x}) = \mathbb{P}[Y = k | X = \mathbf{x}]$

Alternatively, since $\pi_k f_{X|Y}(\mathbf{x}|k) = \eta_k(\mathbf{x}) f_X(\mathbf{x})$, to find the maximum $\eta_k(\mathbf{x})$ it is sufficient to know $\pi_k f_{X|Y}(\mathbf{x}|k)$

Linear discriminant analysis (LDA)

In linear discriminant analysis (LDA), we make the (strong) assumption that

$$X|Y = k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

for $k = 0, \ldots, K-1$

Here $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the multivariate Gaussian/normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$

$$\phi(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\}$$

Note: Each class has the same covariance matrix $\boldsymbol{\Sigma}$

Example

Suppose that $K = 2$

$$d_M^2(\mathbf{x}; \widehat{\boldsymbol{\mu}}_0, \widehat{\boldsymbol{\Sigma}}) - 2\log \widehat{\pi}_0 \overset{0}{\underset{1}{\lessgtr}} d_M^2(\mathbf{x}; \widehat{\boldsymbol{\mu}}_1, \widehat{\boldsymbol{\Sigma}}) - 2\log \widehat{\pi}_1$$

It turns out that by setting

$$\mathbf{w} = \widehat{\boldsymbol{\Sigma}}^{-1}(\widehat{\boldsymbol{\mu}}_1 - \widehat{\boldsymbol{\mu}}_0)$$

$$b = \frac{1}{2}\widehat{\boldsymbol{\mu}}_0^T \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_0 - \frac{1}{2}\widehat{\boldsymbol{\mu}}_1^T \widehat{\boldsymbol{\Sigma}}^{-1} \widehat{\boldsymbol{\mu}}_1 + \log \frac{\widehat{\pi}_1}{\widehat{\pi}_0}$$

we can re-write this as

$$\mathbf{w}^T \mathbf{x} + b \overset{0}{\underset{1}{\lessgtr}} 0$$ *linear classifier*

Challenges for LDA

The generative model is rarely valid

Moreover, the number of parameters to be estimated is

- class prior probabilities: $K - 1$
- means: Kd
- covariance matrix: $\frac{1}{2}d(d+1)$

If d is small and n is large, then we can accurately estimate these parameters (provably, using Hoeffding)

If n is small and d is large, then we have more parameters than observations, and will likely obtain very poor estimates
  - first apply a dimensionality reduction technique to reduce d (more on this later)
  - assume a more structured covariance matrix

Another possible escape

Recall from the very beginning of the lecture that the Bayes classifier can be stated either in terms of maximizing $\pi_k f_{X|Y}(\mathbf{x}|k)$ or $\eta_k(\mathbf{x})$

In LDA, we are estimating $\pi_k f_{X|Y}(\mathbf{x}|k)$, which is equivalent to the full joint distribution of (X,Y)

All we ***really*** need is to be able to estimate $\eta_k(\mathbf{x})$

- we don't need to know $f_X(\mathbf{x})$

LDA commits one of the cardinal sins of machine learning:

Never solve a more difficult problem as an intermediate step

Is there a better approach?

Another look at plugin methods

Suppose $K=2$

Define $\eta(\mathbf{x}) = \eta_1(\mathbf{x})$
$$= 1 - \eta_0(\mathbf{x})$$

In this case, another way to express the Bayes classifier is as

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

Note that we do not actually need to know the full distribution of (X,Y) to express the Bayes classifier

All we really need is to decide if $\eta(\mathbf{x}) \geq 1/2$

Gaussian case

Suppose that $K=2$ and that $X|Y=k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma)$

$$\eta(\mathbf{x}) = \frac{\pi_1 \phi(\mathbf{x}; \boldsymbol{\mu}_1, \Sigma)}{\pi_1 \phi(\mathbf{x}; \boldsymbol{\mu}_1, \Sigma) + \pi_0 \phi(\mathbf{x}; \boldsymbol{\mu}_0, \Sigma)}$$

$$= \frac{\pi_1 e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)}}{\pi_1 e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)} + \pi_0 e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_0)^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_0)}}$$

$$= \frac{1}{1 + \frac{\pi_0}{\pi_1} e^{\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_0)^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_0)}}$$

$$= \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$$

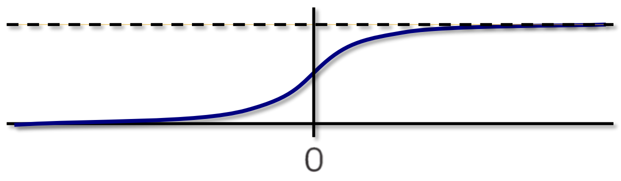
Logistic regression

This observation gives rise to another class of plugin methods, the most important of which is logisitic regression, which implements the following strategy

1. Assume $\eta(\mathbf{x}) = \dfrac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+b)}}$ $(\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R})$
2. Directly estimate $\mathbf{w}, b$ (somehow) from the data
3. Plug the estimate
$$\widehat{\eta}(\mathbf{x}) = \frac{1}{1+e^{-(\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b})}}$$
into the formula for the Bayes classifier

The logistic function

The function $\frac{1}{1+e^{-t}}$ is called a ***logistic*** function (or a ***sigmoid*** function in other contexts)



The logistic regression classifier

Denote the logistic regression classifier by

$$\widehat{h}(\mathbf{x}) = 1_{\{\widehat{\eta}(\mathbf{x}) \geq 1/2\}}(\mathbf{x})$$

Note that $\widehat{h}(\mathbf{x}) = 1 \iff \widehat{\eta}(\mathbf{x}) \geq \frac{1}{2}$

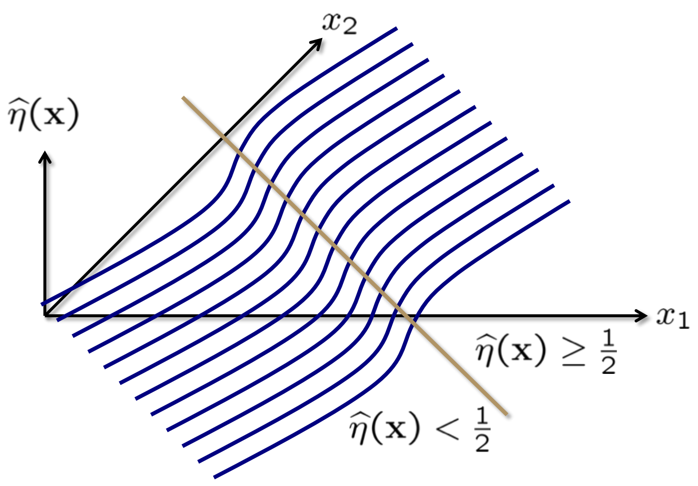
$$\iff \frac{1}{1+\exp\left(-(\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b})\right)} \geq \frac{1}{2}$$

$$\iff \exp\left(-(\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b})\right) \leq 1$$

$$\iff (\widehat{\mathbf{w}}^T\mathbf{x}+\widehat{b}) \geq 0$$

So $\widehat{h}(\mathbf{x}) = \begin{cases} 1 & \text{if } \widehat{\mathbf{w}}^T\mathbf{x}+b \geq 0 \\ 0 & \text{otherwise} \end{cases}$ ***linear classifier***

Example


x_2
$\widehat{\eta}(\mathbf{x})$
x_1
$\widehat{\eta}(\mathbf{x}) \geq \frac{1}{2}$
$\widehat{\eta}(\mathbf{x}) < \frac{1}{2}$

Estimating the parameters

Challenge: How to estimate the parameters for

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

One possibility: $\mathbf{w} = \widehat{\mathbf{\Sigma}}^{-1}(\widehat{\boldsymbol{\mu}}_1 - \widehat{\boldsymbol{\mu}}_0)$

$$b = \frac{1}{2}\widehat{\boldsymbol{\mu}}_0^T\widehat{\mathbf{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_0 - \frac{1}{2}\widehat{\boldsymbol{\mu}}_1^T\widehat{\mathbf{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_1 + \log\frac{\widehat{\pi}_1}{\widehat{\pi}_0}$$

Alternative: *Maximum likelihood estimation*

For convenience, let's let $\boldsymbol{\theta} = (b, \mathbf{w})$

Note that $\eta(\mathbf{x})$ is really a function of both $\mathbf{x}$ and $\boldsymbol{\theta}$, so we will use the notation $\eta(\mathbf{x};\boldsymbol{\theta})$ to highlight this dependence

The *a posteriori* probability of our data

Suppose that we knew $\boldsymbol{\theta}$. Then we could compute

$$\begin{aligned}\mathbb{P}[y_i|\mathbf{x}_i;\boldsymbol{\theta}] &= \mathbb{P}[Y_i = y_i|X_i = \mathbf{x}_i;\boldsymbol{\theta}] \\ &= \begin{cases}\eta(\mathbf{x}_i;\boldsymbol{\theta}) & \text{if } y_i = 1 \\ 1-\eta(\mathbf{x}_i;\boldsymbol{\theta}) & \text{if } y_i = 0\end{cases} \\ &= \eta(\mathbf{x}_i;\boldsymbol{\theta})^{y_i}(1-\eta(\mathbf{x}_i;\boldsymbol{\theta}))^{1-y_i}\end{aligned}$$

Because of independence, we also have that

$$\begin{aligned}\mathbb{P}[y_1,\ldots,y_n|\mathbf{x}_1,\ldots\mathbf{x}_n;\boldsymbol{\theta}] &= \prod_{i=1}^{n}\mathbb{P}[y_i|\mathbf{x}_i;\boldsymbol{\theta}] \\ &= \prod_{i=1}^{n}\eta(\mathbf{x}_i;\boldsymbol{\theta})^{y_i}(1-\eta(\mathbf{x}_i;\boldsymbol{\theta}))^{1-y_i}\end{aligned}$$

Maximum likelihood estimation

We don't actually know $\boldsymbol{\theta}$, but we do know $y_1,\ldots,y_n$

Suppose we view $y_1,\ldots,y_n$ to be fixed, and view $\mathbb{P}[y_1,\ldots,y_n|\mathbf{x}_1,\ldots\mathbf{x}_n;\boldsymbol{\theta}]$ as just a function of $\boldsymbol{\theta}$

When we do this, $\mathcal{L}(\boldsymbol{\theta}) = \mathbb{P}[y_1,\ldots,y_n|\mathbf{x}_1,\ldots\mathbf{x}_n;\boldsymbol{\theta}]$ is called the ***likelihood*** (or likelihood function)

The method of ***maximum likelihood*** aims to estimate $\boldsymbol{\theta}$ by finding the $\boldsymbol{\theta}$ that ***maximizes*** the ***likelihood*** $\mathcal{L}(\boldsymbol{\theta})$

In practice, it is often more convenient to focus on maximizing the ***log-likelihood***, i.e., $\log\mathcal{L}(\boldsymbol{\theta})$

The log-likelihood

To see why, note that the likelihood in our case is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{n}\eta(\mathbf{x}_i;\boldsymbol{\theta})^{y_i}(1-\eta(\mathbf{x}_i;\boldsymbol{\theta}))^{1-y_i}$$

Thus, the log-likelihood is given by

$$\begin{aligned}\ell(\boldsymbol{\theta}) &= \log\mathcal{L}(\boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} y_i\log\eta(\mathbf{x}_i;\boldsymbol{\theta}) + (1-y_i)\log(1-\eta(\mathbf{x}_i;\boldsymbol{\theta}))\end{aligned}$$

It is often easier to work with summations instead of products, and since the log is a monotonic transformation, maximizing $\ell(\boldsymbol{\theta})$ is equivalent to maximizing $\mathcal{L}(\boldsymbol{\theta})$

Simplifying the log-likelihood

Notation

- $\widetilde{\mathbf{x}} = [1, x(1),\ldots,x(d)]^T$
- $\boldsymbol{\theta} = [b, w(1),\ldots,w(d)]^T$

This means that $\mathbf{w}^T\mathbf{x} + b = \boldsymbol{\theta}^T\widetilde{\mathbf{x}}$, which lets us write

$$\eta(\mathbf{x}_i;\boldsymbol{\theta}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T\widetilde{\mathbf{x}}_i}}$$

Thus, if we let $g(t) = \frac{1}{1+e^{-t}}$, then we can write

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} y_i\log g(\boldsymbol{\theta}^T\widetilde{\mathbf{x}}_i) + (1-y_i)\log(1-g(\boldsymbol{\theta}^T\widetilde{\mathbf{x}}_i))$$

Simplifying the log-likelihood

Facts:

$$\log g(t) = \log\left(\frac{1}{1+e^{-t}}\right) = -\log(1+e^{-t})$$

$$\begin{aligned}\log(1-g(t)) &= \log\left(1-\frac{1}{1+e^{-t}}\right)\\ &= \log\left(\frac{e^{-t}}{1+e^{-t}}\right) = -t-\log(1+e^{-t})\\ &= \log\left(\frac{1}{1+e^{t}}\right) = -\log(1+e^{t})\end{aligned}$$

Simplifying the log-likelihood

Facts:

$$\log g(t) = -\log(1+e^{-t})$$

$$\log(1-g(t)) = -t-\log(1+e^{-t}) = -\log(1+e^{t})$$

Thus

$$\begin{aligned}\ell(\boldsymbol\theta) &= \sum_{i=1}^{n} y_i \log g(\boldsymbol\theta^T\widetilde{\mathbf{x}}_i) + (1-y_i)\log(1-g(\boldsymbol\theta^T\widetilde{\mathbf{x}}_i))\\ &= \sum_{i=1}^{n} -y_i\log(1+e^{-\boldsymbol\theta^T\widetilde{\mathbf{x}}_i}) - \log(1+e^{\boldsymbol\theta^T\widetilde{\mathbf{x}}_i})\\ &\qquad\qquad -y_i(-\boldsymbol\theta^T\widetilde{\mathbf{x}}_i - \log(1+e^{-\boldsymbol\theta^T\widetilde{\mathbf{x}}_i}))\\ &= \sum_{i=1}^{n} y_i\boldsymbol\theta^T\widetilde{\mathbf{x}}_i - \log(1+e^{\boldsymbol\theta^T\widetilde{\mathbf{x}}_i})\end{aligned}$$

Maximizing the log-likelihood

How can we maximize

$$\ell(\boldsymbol\theta) = \sum_{i=1}^{n} y_i\boldsymbol\theta^T\widetilde{\mathbf{x}}_i - \log(1+e^{\boldsymbol\theta^T\widetilde{\mathbf{x}}_i})$$

with respect to $\boldsymbol\theta$?

Find a $\boldsymbol\theta$ such that $\nabla\ell(\boldsymbol\theta) = \begin{bmatrix}\frac{\partial\ell(\boldsymbol\theta)}{\partial\theta_1}\\ \vdots\\ \frac{\partial\ell(\boldsymbol\theta)}{\partial\theta_{d+1}}\end{bmatrix} = 0$

(i.e., compute the partial derivatives and set them to zero)

Computing the gradient

It is not too hard to show that

$$\begin{aligned}\nabla\ell(\boldsymbol\theta) &= \sum_{i=1}^{n}\nabla\left(y_i\boldsymbol\theta^T\widetilde{\mathbf{x}}_i - \log(1+e^{\boldsymbol\theta^T\widetilde{\mathbf{x}}_i})\right)\\ &= \sum_{i=1}^{n} y_i\widetilde{\mathbf{x}}_i - \widetilde{\mathbf{x}}_i e^{\boldsymbol\theta^T\widetilde{\mathbf{x}}_i}(1+e^{\boldsymbol\theta^T\widetilde{\mathbf{x}}_i})^{-1}\\ &= \sum_{i=1}^{n}\widetilde{\mathbf{x}}_i(y_i - g(\boldsymbol\theta^T\widetilde{\mathbf{x}}_i)) = 0\end{aligned}$$

This gives us $d+1$ equations, but they are ***nonlinear*** and have no closed-form solution

Optimization

Throughout signal processing and machine learning, we will very often encounter problems of the form

$$\underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}}\ f(\mathbf{x})$$

(or $\underset{\boldsymbol{\theta}\in\mathbb{R}^{d+1}}{\text{minimize}}\ -\ell(\boldsymbol{\theta})$ for today)

In many (most?) cases, we cannot compute the solution simply by setting $\nabla f(\mathbf{x}) = 0$ and solving for $\mathbf{x}$

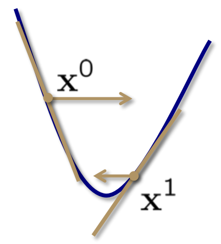
However, there are many powerful ***algorithms*** for finding $\mathbf{x}$ using a computer

Gradient descent

A simple way to try to find the minimum of our objective function is to iteratively *"roll downhill"*

From $\mathbf{x}^0$, take a step in the direction of the negative gradient

$$\mathbf{x}^1 = \mathbf{x}^0 - \alpha_0 \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^0} \qquad \alpha_0 : \text{"step size"}$$
$$\mathbf{x}^2 = \mathbf{x}^1 - \alpha_1 \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^1}$$
$$\vdots$$



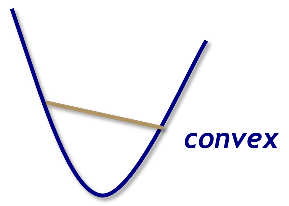
Convergence of gradient descent

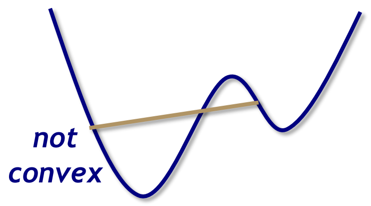
The core iteration of gradient descent is to compute

$$\mathbf{x}^{j+1} = \mathbf{x}^j - \alpha_j \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^j}$$

Note that if $\nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^j} = 0$, then we have found the minimum and $\mathbf{x}^{j+1} = \mathbf{x}^j$, so the algorithm will terminate

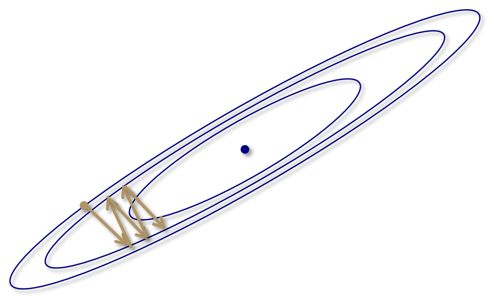
If f is convex and sufficiently smooth, then gradient descent (with a fixed step size α) is guaranteed to converge to the global minimum of f





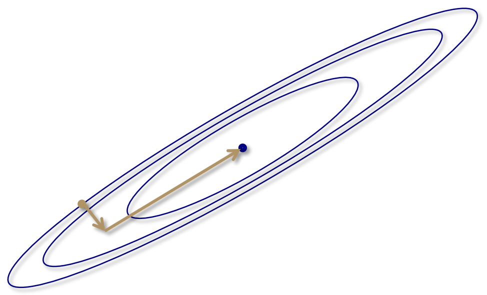
Step size matters!

Even though gradient descent provably converges, it can potentially take a while



Step size matters!

Even though gradient descent provably converges, it can potentially take a while



Newton's method

Also known as the Newton-Raphson method, this approach can be viewed as simply using the second derivative to automatically select an appropriate step size

$$\mathbf{x}^{j+1} = \mathbf{x}^j - \left(\nabla^2 f(\mathbf{x})\right)^{-1} \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^j}$$

Hessian matrix

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix}$$

Optimization for logistic regression

The negative log-likelihood in logistic regression is a convex function

Both gradient descent and Newton's method are common strategies for setting the parameters in logistic regression

Newton's method is much faster when the dimension d is small, but is impractical when d is large

Why?

More on this on next week's homework

Comparison of plugin methods

Naïve Bayes, LDA, and logistic regression are all ***plugin methods*** that result in ***linear*** classifiers

Naïve Bayes

- plugin method based on density estimation
- scales well to high-dimensions and naturally handles mixture of discrete and continuous features

Linear discriminant analysis

- better if Gaussianity assumptions are valid

Logistic regression

- models only the distribution of $Y|X$, not (X,Y)
- valid for a larger class of distributions
- fewer parameters to estimate

Beyond plugin methods

Plugin methods can be useful in practice, but ultimately they are very limited

- There are always distributions where our assumptions are violated
- If our assumptions are wrong, the output is totally unpredictable
- Can be hard to verify whether our assumptions are right
- Require solving a more difficult problem as an intermediate step

For most of the remainder of this course will focus on ***nonparametric*** methods that avoid making such strong assumptions about the (unknown) process generating the data