Logistic Regression

Compare two of our frameworks for classification:

Bayes classification. Here we are given the joint density function $f_{X,Y}(\boldsymbol{x}, y)$, and then given $X = \boldsymbol{x}$, we choose the class k based on the conditional probabilities $\mathrm{P}\left(Y = k | X = \boldsymbol{x}\right)$.

Empirical risk minimization. We make no assumptions about the distribution. Given a set of training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, we solve an optimization program to find the classifier that has the best empirical performance (i.e. performs best on the training data).

You can also operate between these two extremes. One option is to try to fit a density function $f_{X,Y}$ to set of training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, then derive the Bayes classifier from this estimate. There is methodology in statistics for estimating density functions, but it is a tricky business — it invariably requires restrictive modeling assumptions (e.g. $X|Y$ is Gaussian with unknown mean and/or covariance) or enormous amounts of data.

But the Bayes classifier (and other techniques for predicting Y from $X = \boldsymbol{x}$) does not really need the joint distribution — it works from the conditional probabilities

$$\mathrm{P}\left(Y = k | X = \boldsymbol{x}\right), \quad \text{for all} \;\; k \in \mathcal{Y}, \;\; \boldsymbol{x} \in \mathbb{R}^D.$$

Logistic regression is technique for estimating these conditional probability to a set of observed data. For every k, we can think of $\mathrm{P}\left(Y = k | X = \boldsymbol{x}\right)$ as a function of $\boldsymbol{x}$; the "regression" is fitting this function to a set of data points. We will see, however, that both the function class and the loss function we want to use are different than in our discussions about regression using least-squares.

We start with the simple two-class problem, where $\mathcal{Y} = \{0, 1\}$. We are given data points $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ which we assume are independent and identically distributed, but come from an unknown distribution. We want to estimate the function

$$p(\boldsymbol{x}) = \mathrm{P}\left(Y = 1 | X = \boldsymbol{x}\right), \quad \boldsymbol{x} \in \mathbb{R}^D.$$

The estimate for $\mathrm{P}\left(Y = 0 | X = \boldsymbol{x}\right)$ is then $1 - p(\boldsymbol{x})$.

As before, we will define a loss function and a set of functions $\mathcal{P}$ and then solve the optimization program

$$\underset{\boldsymbol{p} \in \mathcal{P}}{\text{minimize}} \ \sum_{n=1}^{N} \ell(p(\boldsymbol{x}_n), y_n)$$

There are two main differences between this problem and our previous discussions about regression (fitting a function to data).

1. The function we are fitting is a probability. For it to make sense, we need $0 \leq p(\boldsymbol{x}) \leq 1$ everywhere. We will have to craft our function representation to make sure that this is true.
2. We are not given direct samples of the function. The y_n we observe are either 0 or 1, and can be interpreted as the outcomes of a Bernoulli random variable (weighted coin toss). This means that the loss we use will be based on some kind of likelihood rather than the deviation of $p(\boldsymbol{x}_n)$ from y_n.

Function representation

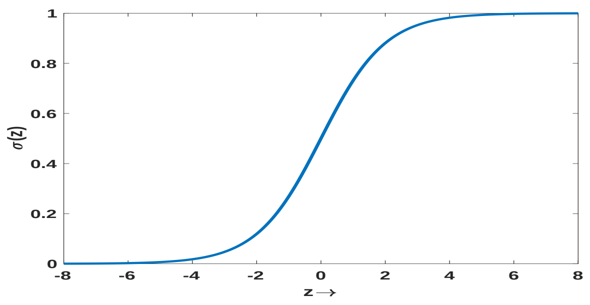
To address the first issue above, we need a tractable way for parameterizing a rich class of functions that takes values between 0 and 1.

One way of doing this is to take functions $\boldsymbol{f}$ from a set $\mathcal{F}$ and then pass them through a *logistic sigmoid*:

$$p(\boldsymbol{x}) = \sigma(f(\boldsymbol{x})), \quad \boldsymbol{f} \in \mathcal{F},$$

where

$\sigma(z) = \frac{1}{1+e^{-z}}$,



By construction, $0 \leq \sigma(f(\boldsymbol{x})) \leq 1$ no matter what $\boldsymbol{f}$ is.

There are of course many different "sigmoid" functions we might use to compress the range of $\boldsymbol{f}$ into the interval $[0, 1]$. The logistic sigmoid has two nice interpretations, though.

First, if we interpret $p(\boldsymbol{x}) = \sigma(f(\boldsymbol{x}))$ as a model for the probability that $Y = 1$ given $X = \boldsymbol{x}$, then

$$f(\boldsymbol{x}) = \sigma^{-1}(p(\boldsymbol{x})) = \log\left(\frac{p(\boldsymbol{x})}{1 - p(\boldsymbol{x})}\right),$$

is a model for the **log odds** of Y having the value 1.

Second, if it is the case that

$$\begin{aligned} X|Y = 0 &\sim \text{Normal}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}), \\ X|Y = 1 &\sim \text{Normal}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}), \\ \mathrm{P}\left(Y = 1\right) = \pi_1, \mathrm{P}\left(Y = 0\right) &= 1 - \pi_1 = \pi_0, \end{aligned}$$

then by Bayes rule

$$\mathrm{P}\left(Y=1|X=\boldsymbol{x}\right)=\frac{\pi_1 e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)}}{\pi_0 e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0)}+\pi_1 e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)}}$$
$$=\frac{1}{1+e^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)}}$$

where

$$\boldsymbol{w}=-2\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_0+\boldsymbol{\mu}_1),\quad b=\frac{1}{2}\left(\boldsymbol{\mu}_0^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0+\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1\right).$$

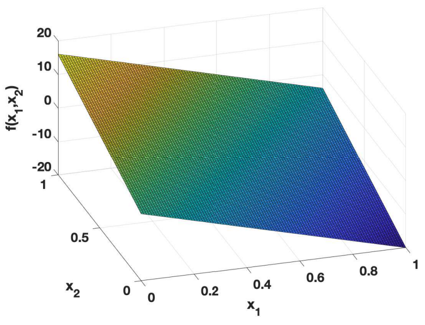
So logistic regression captures this case perfectly when $\mathcal{F}$ is class of all linear functions on $\mathbb{R}^D$.

The choice of function class $\mathcal{F}$ of course directly effects the probability functions $p(\boldsymbol{x})$ (and ultimately the classification regions) that can be expressed. This is best appreciated by illustration; below are some examples for $D=2$.

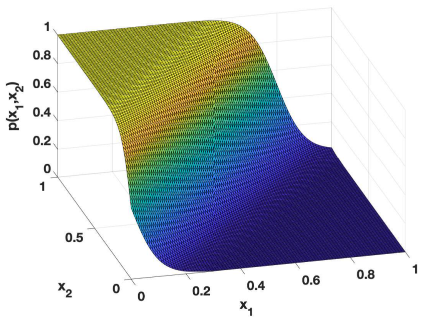
- f is linear:

$$f(\boldsymbol{x})=-20x_1+16x_2$$

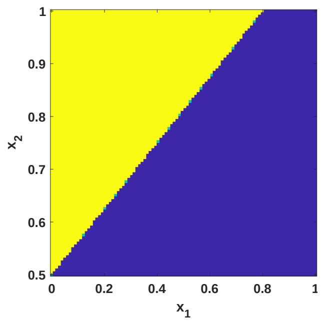
$f(\boldsymbol{x})$



$p(\boldsymbol{x})=\sigma(f(\boldsymbol{x}))$



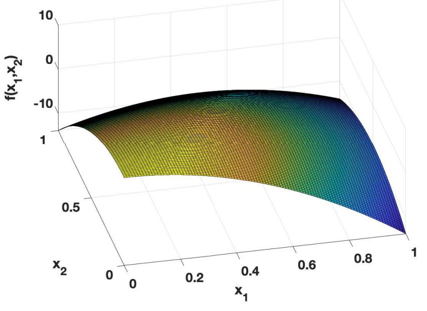
classification regions

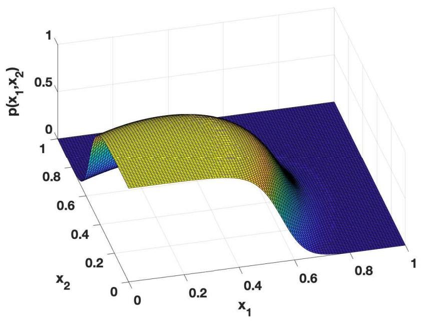


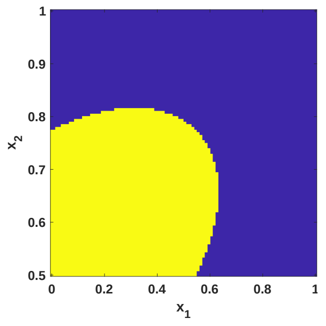
- f is a polynomial of degree 2:

$$f(\boldsymbol{x}) = -20x_1^2 - 20x_2^2 + 20x_1x_2 + 6$$

$f(\boldsymbol{x})$ $\qquad$ $p(\boldsymbol{x}) = \sigma(f(\boldsymbol{x}))$ $\qquad$ classification regions



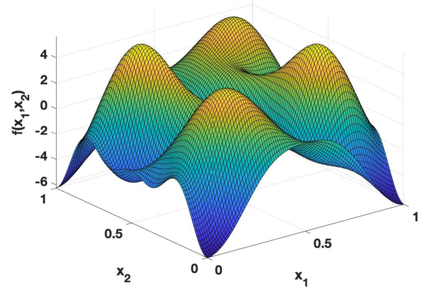


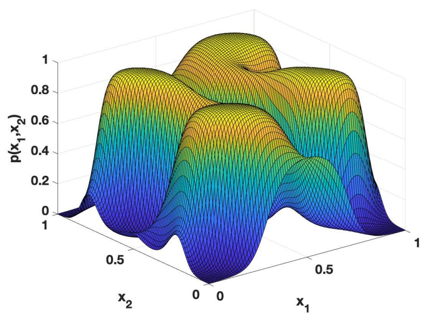


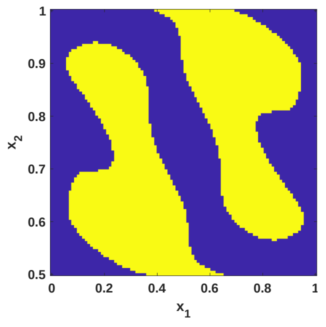
- f is a trigonometric polynomial of degree 3:

$$f(\boldsymbol{x}) = \sum_{k=0}^{3}\sum_{\ell=0}^{3} w_{i,j}\psi_k(x_1)\psi_\ell(x_2), \quad \psi_k(x) = \begin{cases} \cos(\pi k x), & k \text{ even}, \\ \sin(\pi(k+1)x), & k \text{ odd}. \end{cases}$$

$f(\boldsymbol{x})$ $\qquad$ $p(\boldsymbol{x}) = \sigma(f(\boldsymbol{x}))$ $\qquad$ classification regions







Loss function

As mentioned above, the observations we make cannot really be interpreted as sample values of the function that we are trying estimate.

A better interpretation is that they are outcomes of random variables drawn according to this underlying function. That is, y_n is a realization of a binary random variable Y_n where

$$Y_n \in \{0,1\}, \quad Y_n \sim \begin{cases} \mathrm{P}\left(Y_n = 0\right) = 1 - p(\boldsymbol{x}_n) \\ \mathrm{P}\left(Y_n = 1\right) = p(\boldsymbol{x}_n) \end{cases} . \tag{1}$$

The information that Y_n gives about the underlying conditional probability function p at the point $\boldsymbol{x}_n$ is very indirect. Viewing one sample in isolation simply tells us whether we expect $p(\boldsymbol{x}_n)$ to be closer to 1 (if $y_n = 1$) or 0 (if $y_n = 0$). But we can coherently combine this indirect information across all of the samples by treating the estimation of $p(\boldsymbol{x})$ as a maximum likelihood problem.

For a fixed $p(\boldsymbol{x})$, the data $\{(\boldsymbol{x}_n, y_n)\}$ assign a likelihood using the model (1). For a single point, the likelihood can be written as

$$L\left(\boldsymbol{p}; \boldsymbol{x}_n, y_n\right) = p(\boldsymbol{x}_n)^{y_n}(1 - p(\boldsymbol{x}_n))^{1-y_n},$$

and so the log likelihood is

$$\ell\left(\boldsymbol{p}; \boldsymbol{x}_n, y_n\right) = y_n \log(p(\boldsymbol{x}_n)) + (1 - y_n)\log(1 - p(\boldsymbol{x}_n)). \tag{2}$$

The joint log likelihood for all of the samples is simply the above summed over n. We thus find the maximum likelihood $\boldsymbol{p}$ by solving[1]

$$\underset{\boldsymbol{p}}{\text{minimize}} \;\; \sum_{n=1}^{N} -y_n \log(p(\boldsymbol{x}_n)) - (1 - y_n)\log(1 - p(\boldsymbol{x}_n)).$$

The loss function in (2) is often times referred to as the **cross entropy loss**. Writing this in terms of our function model $\mathcal{F}$ yields

$$\underset{\boldsymbol{f}\in\mathcal{F}}{\text{minimize}} \;\; \sum_{n=1}^{N} -y_n \log(\sigma(f(\boldsymbol{x}_n))) - (1 - y_n)\log(1 - \sigma(f(\boldsymbol{x}_n))).$$

[1]Maximizing the log likelihood is the same as minimizing the negative log likelihood.

The last step is to explicitly incorporate our model for $\mathcal{F}$. Here, we will assume that $\mathcal{F}$ is subspace of functions spanned by P basis vectors $\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_P$, so

$$f(\boldsymbol{x}) = \sum_{i=1}^{P} w_i \psi_i(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\Psi}(\boldsymbol{x}), \quad \boldsymbol{\Psi}(\boldsymbol{x}) = \begin{bmatrix} \psi_1(\boldsymbol{x}) \\ \psi_2(\boldsymbol{x}) \\ \vdots \\ \psi_P(\boldsymbol{x}) \end{bmatrix}.$$

Then the optimization program above can be re-written as a search for the expansion coefficients (or "weights") $\boldsymbol{w}$,

$$\underset{\boldsymbol{w} \in \mathbb{R}^P}{\text{minimize}} \ \sum_{n=1}^{N} -\ell(\boldsymbol{w}; \boldsymbol{x}_n, y_n),$$

where

$$\ell(\boldsymbol{w}; \boldsymbol{x}_n, y_n) = y_n \log(\sigma(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\Psi}(\boldsymbol{x}_n))) + (1 - y_n) \log(1 - \sigma(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\Psi}(\boldsymbol{x}_n))).$$

As it turns out, this is a convex function in $\boldsymbol{w}$, and so there are any number of techniques (e.g. gradient descent) that can be used to compute the minimum.

Multiple classes

The discussion above can be easily extending to multiple classes. If $Y \in \{1, \ldots, K\}$, then we need to estimate K different conditional probabilities. The function $\boldsymbol{p}$ we are trying to fit to the data is vector-valued and the outputs must be positive and sum to 1. That is, $\boldsymbol{p} : \mathbb{R}^D \to \Delta^K$, where Δ^K is the unit simplex in $\mathbb{R}^K$,

$$\Delta^K = \left\{ \boldsymbol{v} \in \mathbb{R}^K \ : \ v_k \geq 0, \quad \sum_{k=1}^{K} v_k = 1 \right\}.$$

As before, we will generate such $\boldsymbol{p}$ by creating vector-valued functions $\boldsymbol{f} : \mathbb{R}^D \to \mathbb{R}^K$ that take arbitrary values at their outputs, then passing them through a nonlinearity that maps onto the simplex. The analog to the logistic sigmoid used in the two-class case is the **softmax** function

$$\boldsymbol{s}(\boldsymbol{z}) = \frac{1}{\sum_{k=1}^K e^{-z_k}} \begin{bmatrix} e^{-z_1} \\ e^{-z_2} \\ \vdots \\ e^{-z_K} \end{bmatrix}.$$

With the basis $\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_P$ fixed, we can build up K different functions using K different sets of weights, collected into the $P \times K$ matrix $\boldsymbol{W}$. That is, we consider functions $\boldsymbol{f} : \mathbb{R}^D \to \mathbb{R}^K$ of the form

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Psi}(\boldsymbol{x}).$$

The (negative of the) loss function becomes

$$\ell(\boldsymbol{W}; \boldsymbol{x}_n, y_n) = \log(\boldsymbol{s}(\boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Psi}(\boldsymbol{x}_n))_{y_n}),$$

where $\boldsymbol{s}(\boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Psi}(\boldsymbol{x}_n))_{y_n}$ is the entry of the vector $\boldsymbol{s}(\boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Psi}(\boldsymbol{x}_n))$ corresponding to the class label y_n. The optimization program

$$\underset{\boldsymbol{W}}{\text{minimize}} \ \sum_{n=1}^N -\log(\boldsymbol{s}(\boldsymbol{W}^{\mathrm{T}} \boldsymbol{\Psi}(\boldsymbol{x}_n))_{y_n})$$

is again convex in the matrix variable $\boldsymbol{W}$.

A note on modern methods

State-of-the-art methods (i.e. deep neural networks) for classification follow the same framework as above, but with different function classes $\mathcal{F}$. Instead of using a subspace model, the functions $\boldsymbol{f}$ are parameterized with multiple "layers" of linear weights followed by nonlinearities. With these models, minimizing the cross-entropy loss (or really any other kind of loss) is no longer a convex program. This means that you are never confident you have found the absolute best model, but in practice, running gradient descent tends to produce function estimates that are very effective.