Machine Learning I

Lecture 9: Linear Methods in Classification

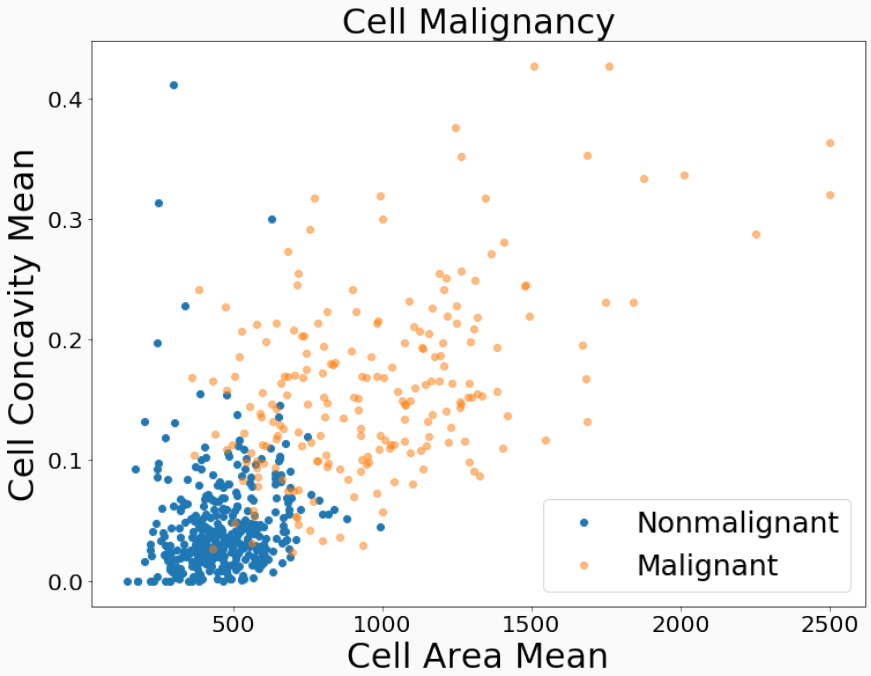
Nathaniel Bade

Northeastern University Department of Mathematics

Table of contents

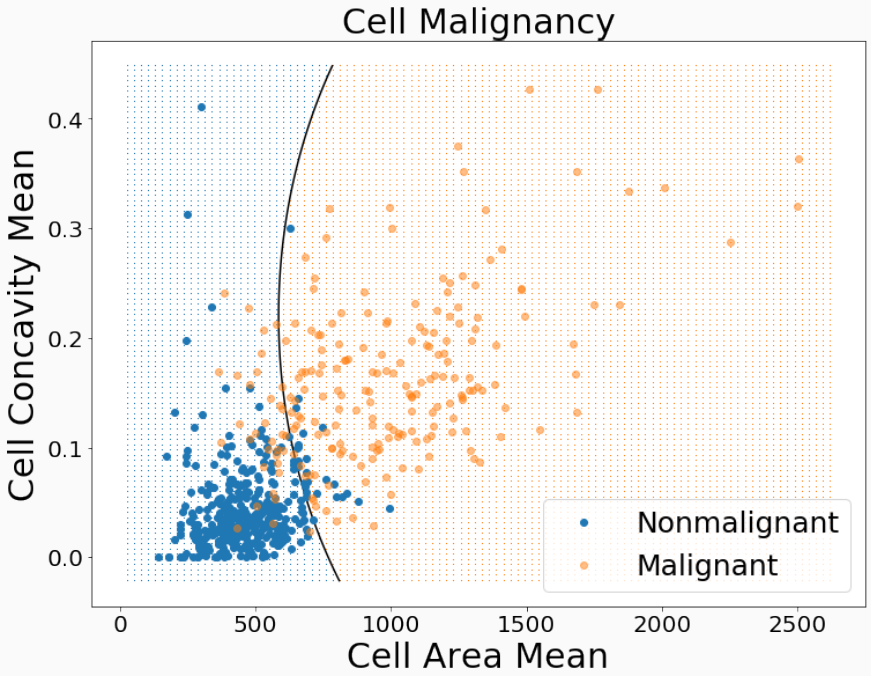
Binary Classification by a Halfspace and Linear Programming

Binary Classification



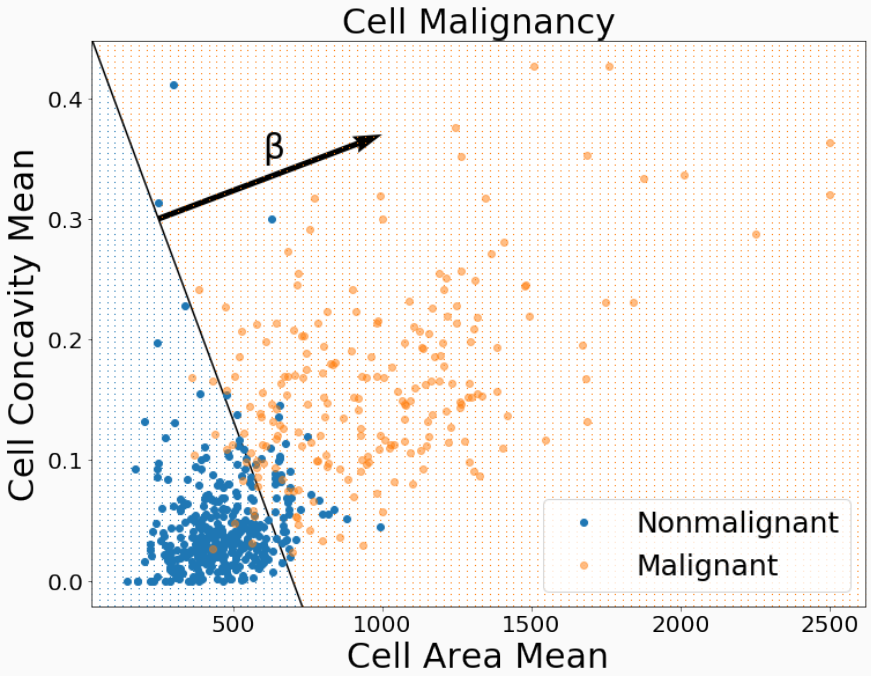
In binary classification, the target space $\mathcal{Y}$ is a categorical variable with two labels. For example, the data above (UW Breast Cancer Dataset) is attempting to classify tumor cells as malignant or nonmalignant based on the cells average area and average concavity.

Binary Classification



In binary classification, the target space $\mathcal{Y}$ is a categorical variable with two labels. For example, the data above (UW Breast Cancer Dataset) is attempting to classify tumor cells as malignant or nonmalignant based on the cells average area and average concavity.

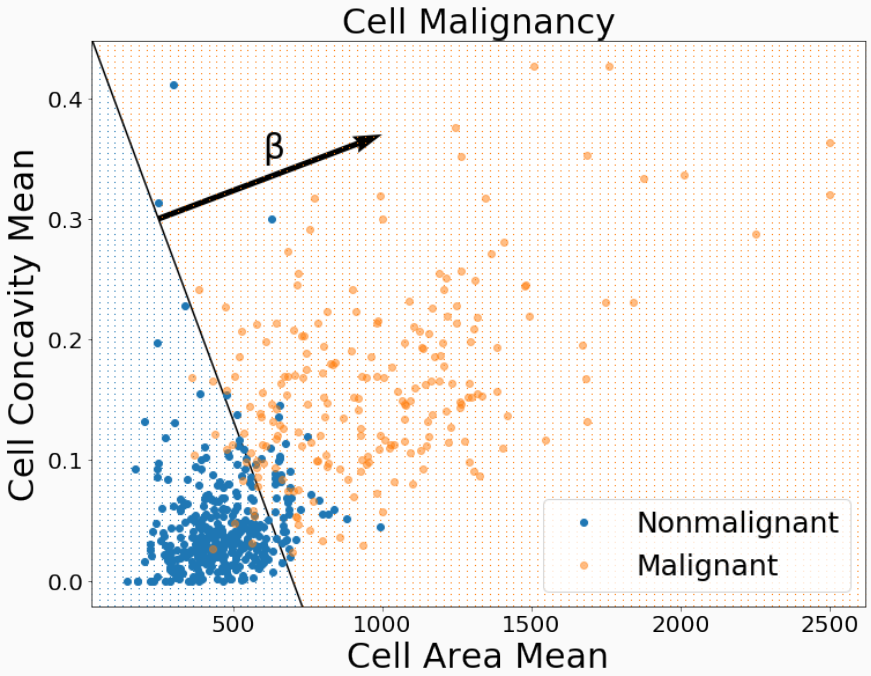
Halfspace Classifiers



The first and simplest hypothesis class to consider is the class of linear half-space classifiers. For binary variables, we can always pick $\mathcal{Y} = \{-1, +1\}$. For an input vector X, an a parameter vector β, the halfplace classifiers can be written

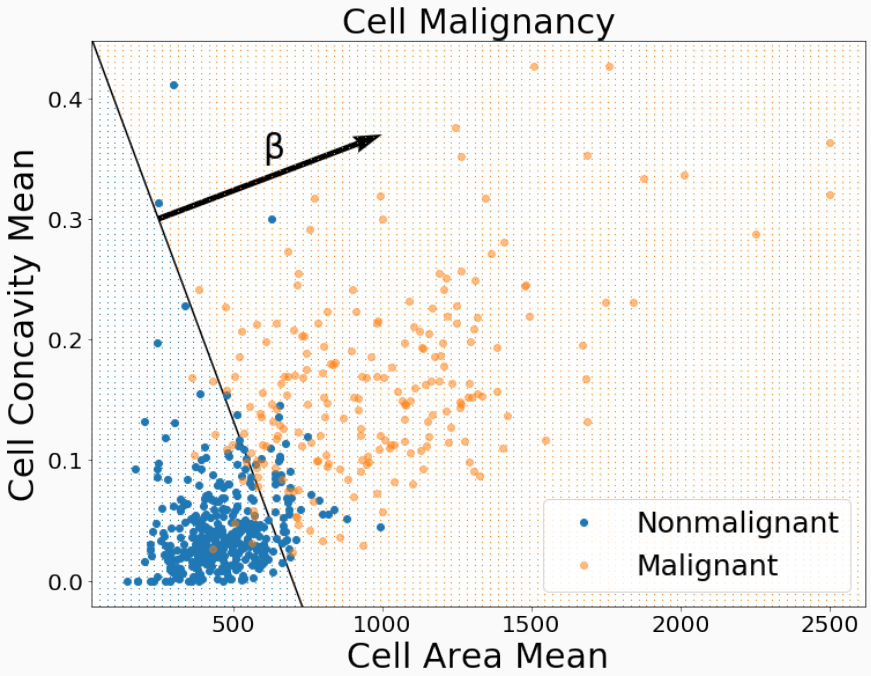
$$\hat{f} = \text{sign}(\beta_0 + X_1\beta_1 + \ldots X_p\beta_p) = \text{sign}(\beta_0 + X^T\beta)\,.$$

Halfspace Classifiers



Then, if x_i lies on the side of $\hat{f}$ that β points towards, $\text{sign}(\beta_0 + X^T\beta)$ assigns it the label $\hat{y}_i = 1$. Otherwise it is assigned the label $\hat{y}_i = -1$.

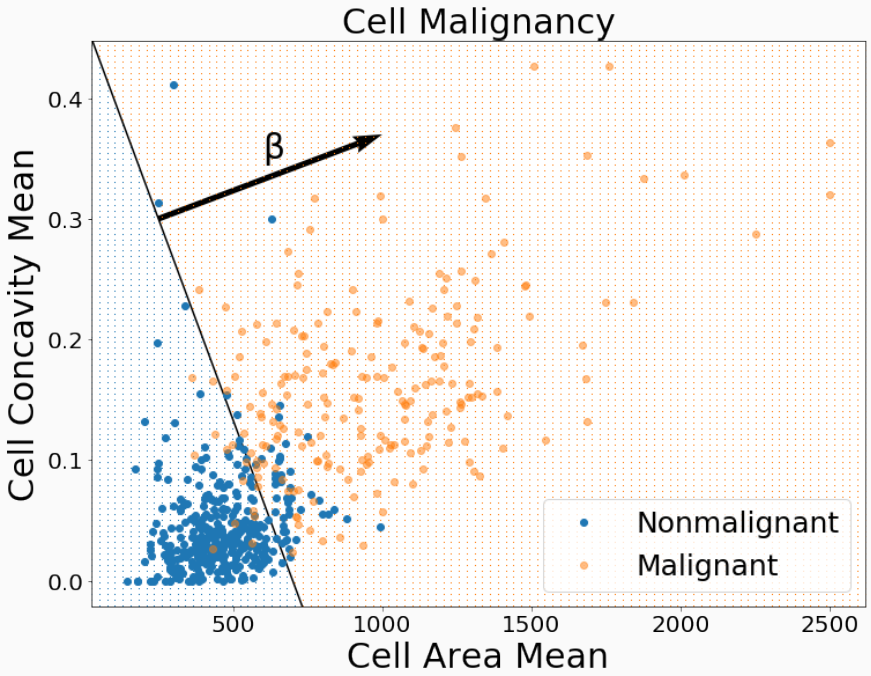
Halfspace Classifiers



With a bit of work, we can see that the VCdim of $\mathcal{H}$ is $p+1$, so the ERM procedure can find a predictor within ϵ of the best linear predictor with probability δ, provided training data on the order of

$$|\mathcal{T}| \approx O\left(\frac{d+\log(1/\delta)}{\epsilon}\right).$$

Halfspace Classifiers



The problem is implementing the ERM rule for 0-1 loss. If $\mathcal{H}$ is realizable, we call the labels **separable** and use **Linear Programming.** Unfortunately, implementing the ERM rule in the **nonseperable** case is known to be computationally hard.

Computationally efficient approaches to binary classification (eg. logistic regression, which we will see later) therefore fit to different loss functions.

Linear Programming

Linear programs are problems that can be expressed as maximizing a linear function, subject to linear inequalities. For $U \in \mathbb{R}^p$ a fixed input, $M \in \mathbb{R}^m$ a fixed offset vector and A a $m \times p$ matrix, we want to find

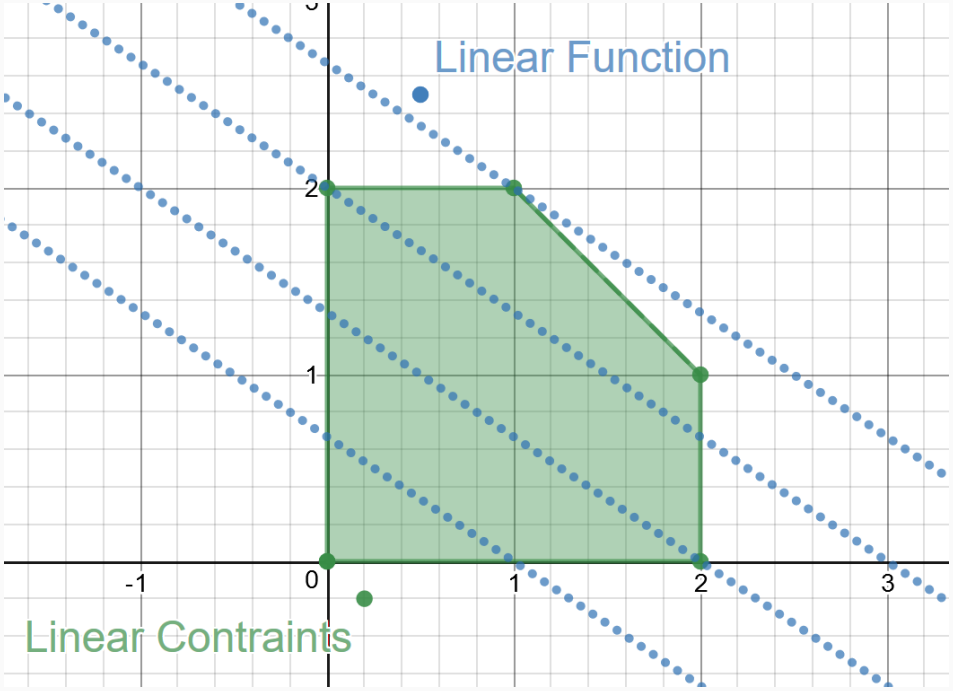
$$\max_{\beta \in \mathbb{R}^p} \langle U, \beta \rangle = \max_{\beta \in \mathbb{R}^p} U^T \beta \,,$$

subject to

$$A\beta \geq M \,.$$

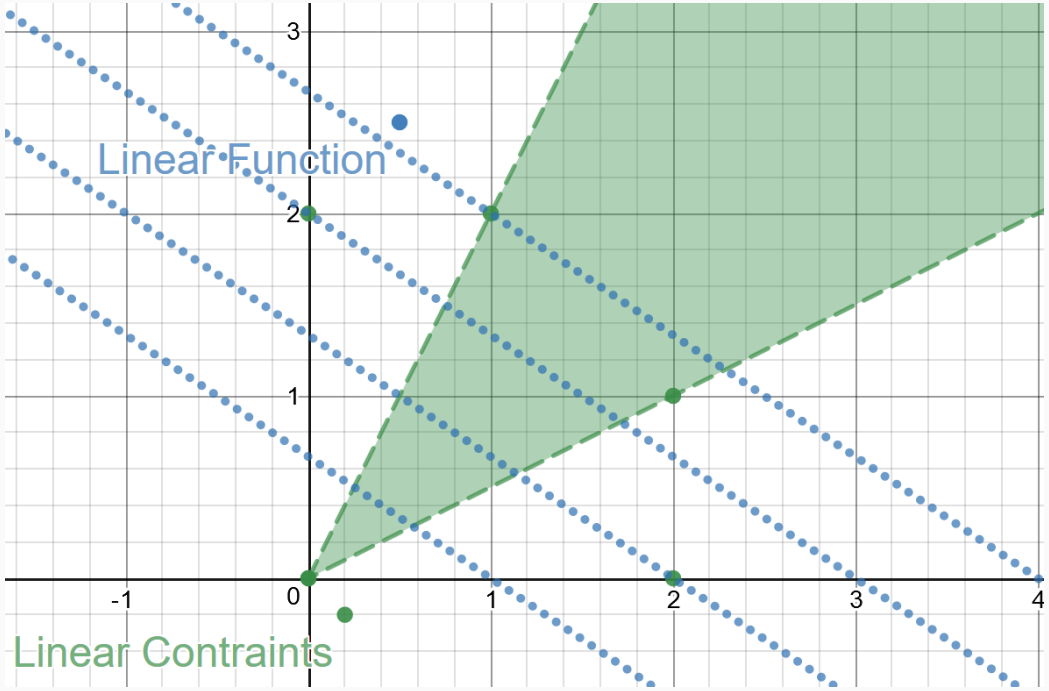
Linear programs can be solved efficiently and are implemented in a variety of packages in R, C, Matlab and Python.

Linear Programming



Linear programs can be compact problems or noncompact problems, but they are always **convex**, meaning any straight line between two points stays in their domain. Optimization over convex domains tends to be easier than over nonconvex domains since any pair of points can be interpolated between.

Linear Programming



Linear programs can be compact problems or noncompact problems, but they are always **convex**, meaning any straight line between two points stays in their domain. Optimization over convex domains tends to be easier than over nonconvex domains since any pair of points can be interpolated between. Since the function to be maximized is linear, this means the maxima are on the boundary.

Linear Programming

Idea: In the realizable case for 0-1 loss, the ERM problem for halfspace classifiers can be solved as a linear program.

By realizability, ERM should have 0 errors on the training set (x_i, y_i), $i = 1, \ldots, N$, with $x_i \in \mathbb{R}^p$, $y_i \in \{\pm 1\}$. Assuming $\beta_0 = 0$ for simplicity, we are looking for $\beta \in \mathbb{R}^p$ such that

$$\text{sign}(x_i^T \beta) = y_i \quad \forall i = 1, \ldots N$$

But this is equivalent to finding a β such that

$$y_i(x_i^T \beta) > 0 \quad \forall i = 1, \ldots N\,.$$

We need a strict inequality to have a linear program since the power comes from assuming the max is on the boundary.

Linear Programming

By realizability, there does exist a β_* with

$$y_i(x_i^T \beta_*) > 0 \quad \forall i = 1, \ldots N\,.$$

Setting $\gamma = \min_i(x_i^T \beta_*)$, the inequality above is equivalent to

$$\frac{1}{\gamma} y_i(x_i^T \beta_*) \geq 1 \quad \forall i = 1, \ldots N\,.$$

so the problem of finding an ERM classifier is a linear program.

Letting $A_{ij} = y_i x_{ij}$ and $M = \langle 1, \ldots, 1 \rangle$, the constraint becomes $A\beta \geq M$. Since any β that satisfies this equation is an equal candidate, we set $U = 0$. This system can be solved using any available LP solvers.

Binary Classifacation via the Perception Algorithm

Perception

One of the most important algorithms in machine learning is Rosenblatts perceptron algorithm. The perceptron algorithm works iteratively to minimize the distance of misclassified points to the decision boundary.

Initially, we set all $\beta^{(j)}$ to 0 and then at each step the perception finds an example of a mislabeled instance x_i. We then generate $\beta^{(j+1)}$ by

$$\beta^{(j+1)} = \beta^{(j)} + y_i x_i \,.$$

Since

$$y_i \langle x_i, \beta^{(j+1)} \rangle = y_i \langle x_i, \beta^{(j)} + y_i x_i \rangle = y_i \langle x_i, \beta^{(j)} \rangle + ||x_i||^2 > 0 \,,$$

the new halfplane "more correctly" classifies x_i. The algorithm stops when all points are classified.

`https://www.youtube.com/watch?v=xpJHhHwR4DQ`

Perception

Theorem: Assume that (x_i, y_i) are separable. Let $R = \max_i ||x_i||$ and

$$B = \min\{||\beta|| \ : \ y_i \langle x_i, \beta \rangle \geq 1 \, \forall i\} \, .$$

Then the perceptron algorithm stops after at most $(RB)^2$ iterations.

A couple of points about the perceptron:

- When the data is separable, there are many solutions and which one is found depends on the starting value.
- The finite number of steps can be large, practically, if the gap is small the time to find it is large.
- When the data are not seperable, the algorithm does not converge, and instead falls into a cycle.

Perception

The perceptron is neither an explicit solution to a vanishing derivative, a direct gradient decent nor Newtons method. If we consider the loss function

$$\ell(\beta, \beta_0) = -\sum_{i=1}^{N} y_i (x_i^T \beta + \beta_0)\,,$$

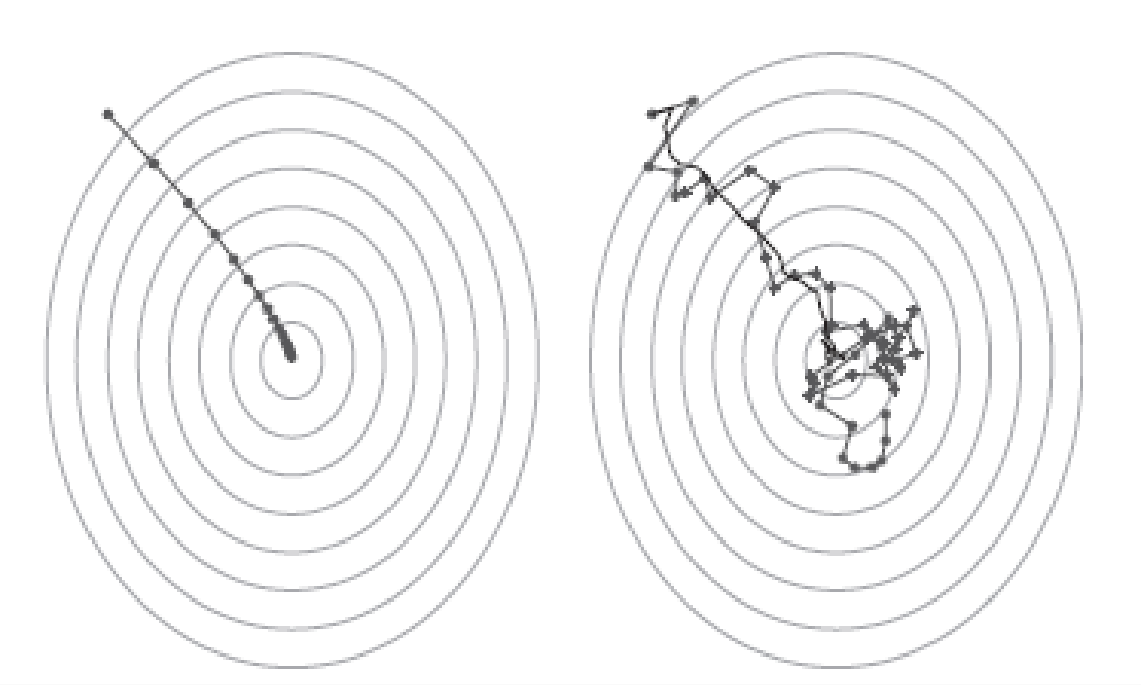
the gradient is

$$\frac{\partial \ell}{\partial \beta} = -\sum_{i=1}^{N} y_i x_i\,, \qquad \frac{\partial \ell}{\partial \beta} = -\sum_{i=1}^{N} y_i\,.$$

In gradient decent, we would update β and β_0 using the full gradient:

$$\begin{pmatrix} \beta^{(j+1)} \\ \beta_0^{(j+1)} \end{pmatrix} = \begin{pmatrix} \beta^{(j)} \\ \beta_0^{(j)} \end{pmatrix} + \rho \begin{pmatrix} \sum_{i=1}^{N} y_i x_i \\ \sum_{i=1}^{N} y_i \end{pmatrix}\,, \quad \rho > 0\,.$$

Perception



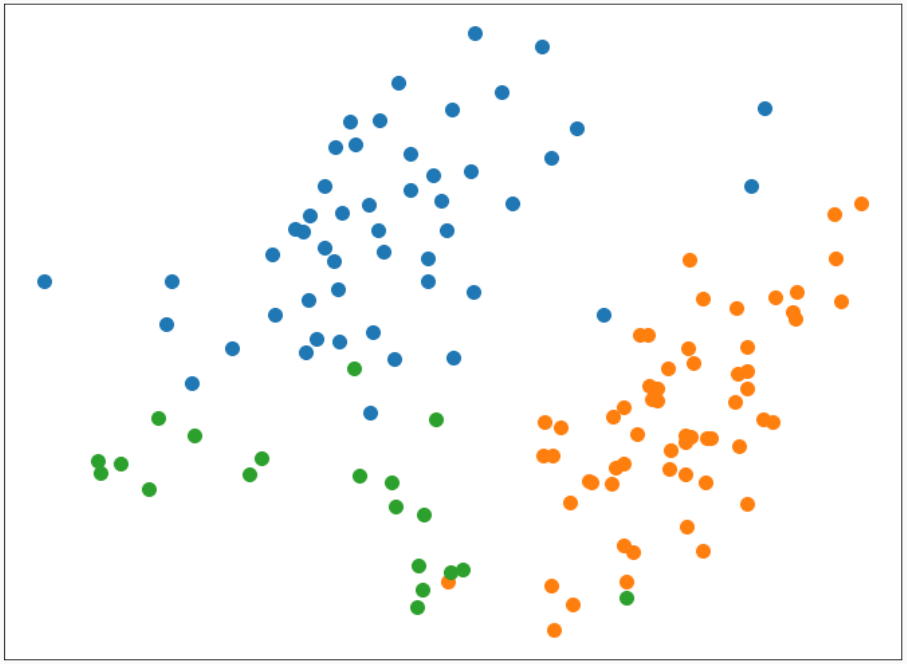
The perceptron's update picks out only a single derivative from a privileged subset:

$$\begin{pmatrix} \beta^{(j+1)} \\ \beta_0^{(j+1)} \end{pmatrix} = \begin{pmatrix} \beta^{(j)} \\ \beta_0^{(j)} \end{pmatrix} + \begin{pmatrix} y_i x_i \\ y_i \end{pmatrix} .$$

This is a form of **stochastic gradient decent**, where we only require the expected value of the update to be in the gradient direction.

Multilabel Classification

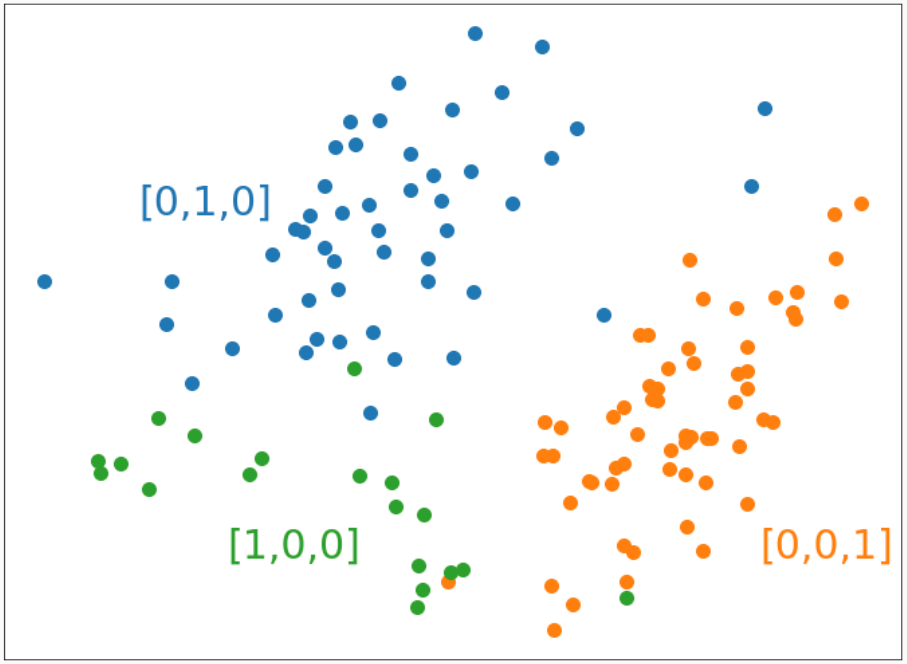
Perception



We will now expand our toolbox to include methods of classification for multiple labels. Suppose there are $|\mathcal{G}| = K$ classes, labeled $1, 2, \ldots, K$ and let $f_k(X)$ be the probability that Y takes the label k given X. Then fitting Y is just fitting K function

$$(\hat{f}_1(X), \ldots, \hat{f}_K(X)) \quad \text{given} \quad \hat{f}_1(X) + \ldots + \hat{f}_K(X) = 1\,,\; f_k(X) \geq 0\,.$$

Multilabel Classification



In practice this means that $Y = k$ is encoded in a K vector, where the k'th entry is 1 and the other entries are 0. If we encode this bitwise it is known as a **one-hot encoding**. If probability functions $\hat{f}_k$ have been fit, a label $\hat{y}_i$ can be predicted by taking

$$\hat{G}(x_i) = \underset{k \in \mathcal{G}}{\text{argmax}}\ \hat{f}(x_i)$$

Multilabel Classification: Loss functions

For multiple labels, the 0-1 loss function can be generalized. For any $K \times K$ matrix L that is 0 on the diagonal and positive off diagonal,

$$L = \begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix} \qquad \text{(for example)}$$

we can define the loss function $\ell(j, k) = L_{jk}$ where L_{jk} is the "price" for misclassifying k as j. The expected prediction error is

$$EPE = E[\ell(Y, \hat{G})] \,.$$

Of course, the most common error is to set all off diagonal entries to 1, punishing all misclassifacations equally.

Multilabel Classification: Bayes Classifier

The Bayes optimal predictor can be found by conditionalizing:

$$EPE = E_{\mathcal{T}}[\ell(Y, \hat{G})] = E_X E_{Y|X}[\ell(Y, \hat{G}(X))]$$
$$= E_X \left[\sum_{k=1}^{K} \ell[k, \hat{G}(X)] \cdot \mathbb{P}(\, k|X\,) \right] .$$

As before, it's clearly sufficient to minimize EPE pointwise, so

$$\hat{G}(x) = \underset{g \in \mathcal{G}}{\text{argmin}}\ E_X \left[\sum_{k=1}^{K} \ell[k, g] \cdot \mathbb{P}(\, k|X = x\,) \right] .$$

With 0-1 loss, this is more simply

$$\hat{G}(x) = \underset{g \in \mathcal{G}}{\text{argmin}}\ \mathbb{P}(g|X = x)\,.$$

Multilabel Classification: Bayes Classifier

In words, (for 0-1 loss) at each point x, the Bayes classifier returns the value g that is most probable at that point

$$\hat{G}(x) = \underset{g \in \mathcal{G}}{\operatorname{argmin}} \, \mathbb{P}(g|X = x) \, .$$

For more general loss, the Bayes classifier returns the value g that yields the smallest total potential for loss.

The error rate of the Bayes classifier is called the Bayes rate, and is the theoretical minimum of loss.

Multilabel Classification: Methods

Our goal of course is to get as close to the Bayes classifier as possible. We will first consider regression on each of the functions $\hat{f}_k$, modeling them as linear and functions.

The regression approach is a member of a class of methods that model discriminant functions δ_i. If we allow ourselves to make assumptions about the background distribution, we will see that we can derive quite good algorithms for 2d visualization by comparing probabilities.

Finally, we will look at using logistic regression to smoothly interpolate between discrete probabilities.

Regression on Categorical Variables

Multilable Regression

Given a classification problem with K labels, we encode the repose categories $\mathcal{G}$ into a K vector $Y = (Y_1, \ldots, Y_K)$, $Y_i \in \{0, 1\}$. The N training instances form a $N \times K$ matrix $\mathbf{Y}$ of 1's and 0's. Letting X denote the $p + 1$ vector with $X_0 = 1$ as usual, the linear model

$$Y = X^T \beta \,,$$

now has a $p + 1 \times K$ matrix of coefficients β. The loss

$$RSS(Y, G(X)) = \sum_{i=1}^{N} ||Y - G(X)||^2 = \sum_{i=1}^{N} ||Y - X^T \beta||^2$$

is minimized by $\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$.

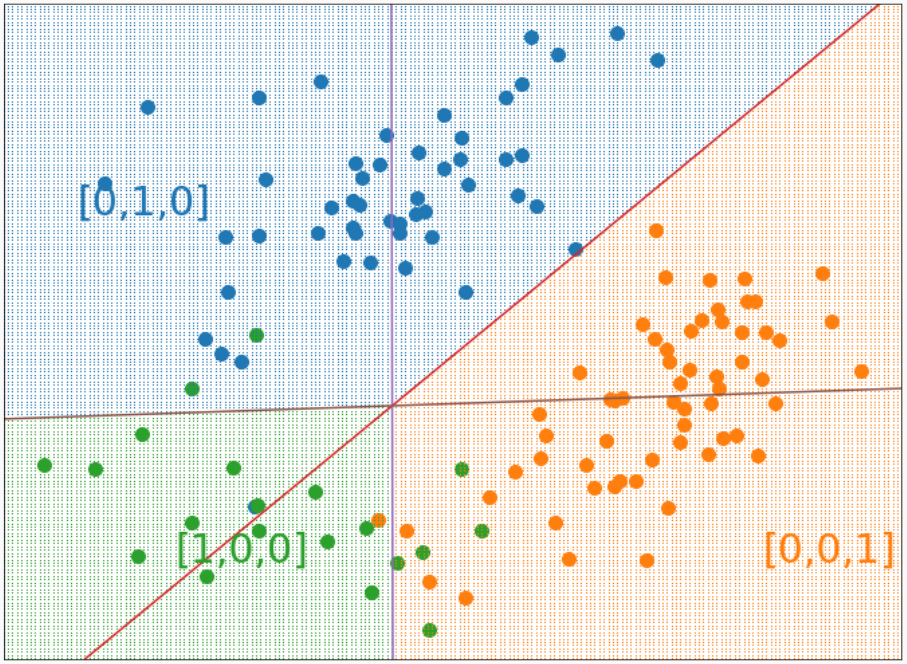
Multilable Regression

With a little bit of work (**exercise**, use centered vectors) one can show that

$$\sum_{k=1}^{K} \hat{f}_k(X) = 1 \quad \forall X\,,$$

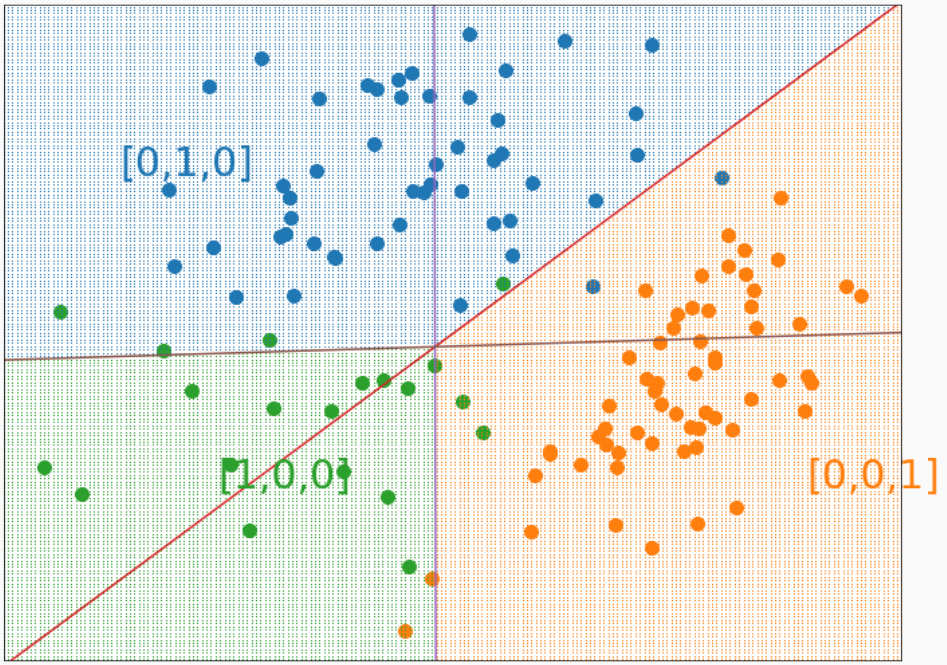
but there is no guarantee that the $\hat{f}_k$ are positive. This isn't necessarily fatal, in fact this kind of linear regression often works well, but we shouldn't understand the $\hat{f}_k$ as providing strict probabilities in any sense.

Multilabel Classification: Example



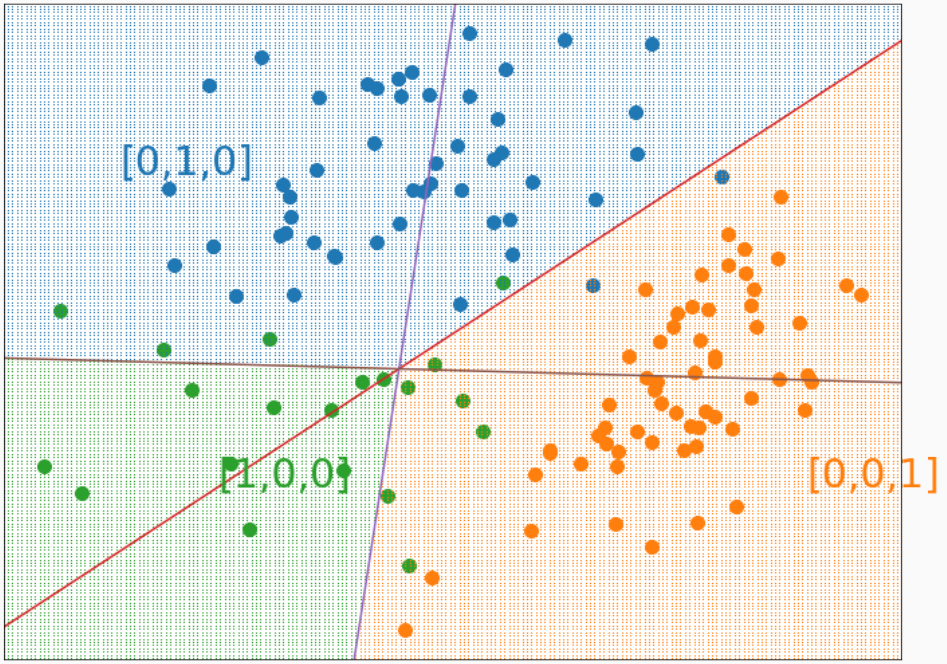
Also, while a linear decision boundary may not provide the best fit they have low variance relative to other classes. The regions above are defined by the regions in which $\hat{f}_k(x)$ is the largest for each k. The lines are the intersection lines $\hat{f}_k = \hat{f}_j$.

Multilabel Classification: Example



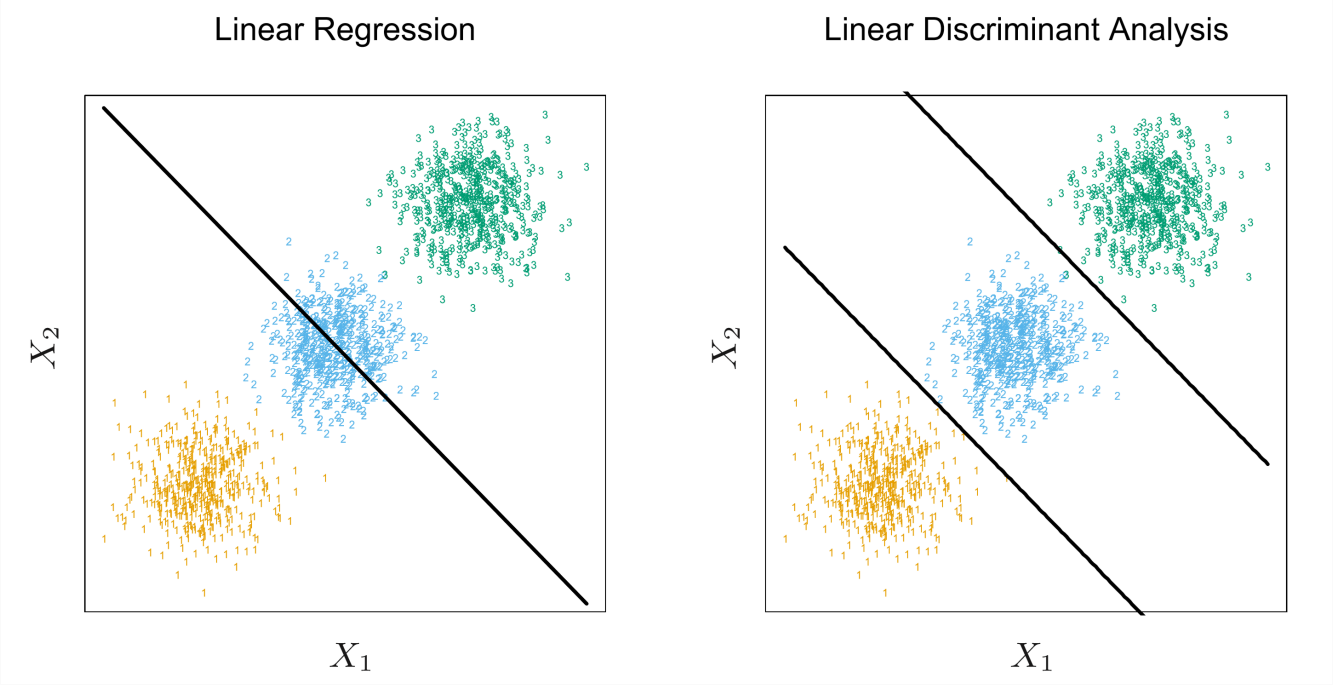
Also, while a linear decision boundary may not provide the best fit they have low variance relative to other classes. The regions above are defined by the regions in which $\hat{f}_k(x)$ is the largest for each k. The lines are the intersection lines $\hat{f}_k = \hat{f}_j$.

Multilabel Classification: Example



Also, while a linear decision boundary may not provide the best fit they have low variance relative to other classes. The regions above are defined by the regions in which $\hat{f}_k(x)$ is the largest for each k. The lines are the intersection lines $\hat{f}_k = \hat{f}_j$.

Multilabel Classification: Masking



The biggest problem with linear decision boundaries comes from masking. In the admittedly extreme example on the left, the three clusters are clearly distinct but the linear regressor misses the middle class completely. On the right we see the classes being successfully fit with a quadratic regression.

Multilabel Classification: Quadratic Fitting

A quick note about quadratic regression. While we will say more about nonlinear fitting later, quadratic regression can be though of a linear regression on the constructed features $X_i X_j$, for all i, j. If we're willing to add $p(p+1)/2$ new variables the quadratic regression

$$Y = \beta_0 + \sum_i X_i \beta_i + \sum_{i<j} X_i X_j \alpha_{ij}$$

can be done with a linear fit.

Linear Discriminant Analysis

Bayes Theorem and Class Densities

The Bayes optimal classifier $\mathbb{P}(G|X)$ tells us that we are interested in estimating the conditional probability. If we have some total probability π_k (**prior**) for the label k and a **class conditional density**

$$f_k(x) = \mathbb{P}(X = x|G = k)\,,$$

Bayes theorem gives

$$\mathbb{P}(G = k|X = x) = \frac{\mathbb{P}(X = x|G = k)\mathbb{P}(G = k)}{\mathbb{P}(X = x)} = \frac{f_k(x)\pi_k}{\sum_{j=1}^{K} f_j(x)\pi_j}\,.$$

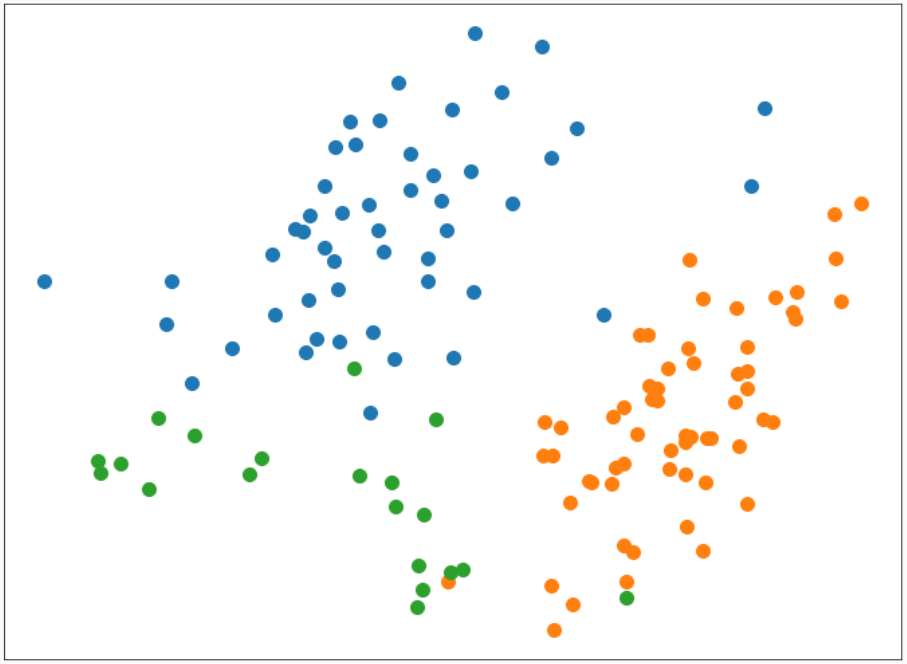
So knowing $f_k(x)$ and estimating the label proportions is almost good enough to know $\mathbb{P}(G|X)$.

Bayes Theorem and Class Densities

Many techniques are based on models for the class densities:

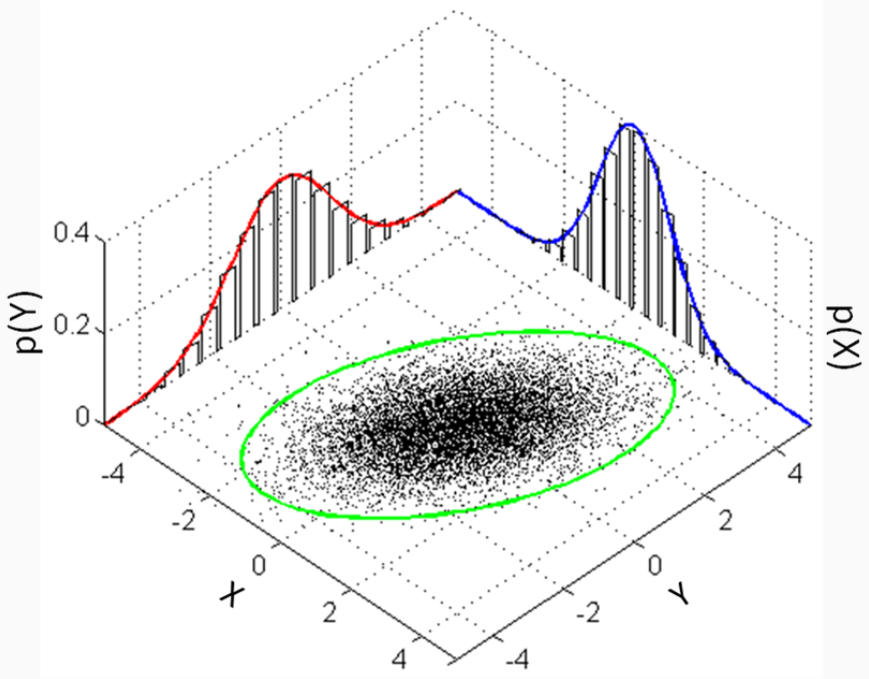
- Semiregular Gaussian class densities lead to linear and quadratic densities.
- Higher order nonlinear decision boundaries can be generated by non-parametric density estimates (ie sampled or bootstrapped densities).
- Naive Bayes techniques are a common variant of the above, where we additionally assume that the variables are independent.

Gaussian Densities



Given a multilabel classification problem, we may try to model each of the labels directly by guess their underlying probability distribution and using maximum likelihood, or other methods.

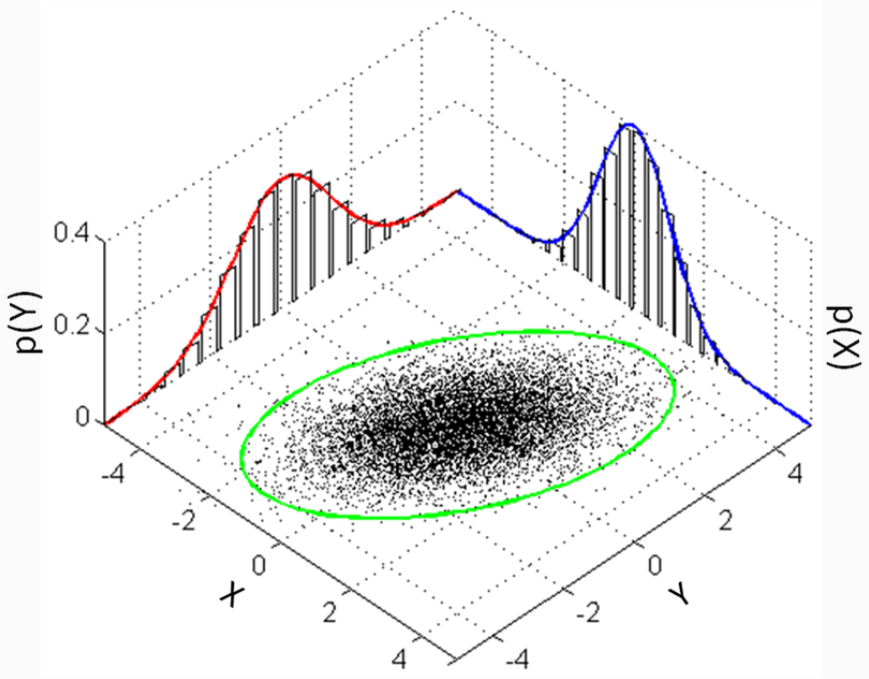
Gaussian Densities



Suppose we model each density $f_k(x)$ as a multivariate Gaussian with covariance matrix $\mathbf{\Sigma}_k$

$$f_k(x) = \left[(2\pi)^p |\mathbf{\Sigma}_k|\right]^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}_k^{-1}(x-\mu_k)\right).$$

Gaussian Densities



$$f_k(x) = \left[(2\pi)^p |\mathbf{\Sigma}_k|\right]^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}_k^{-1}(x-\mu_k)\right)$$

Above are the 3σ ellipse and marginal distributions for

$$\mu_k = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \mathbf{\Sigma} = \begin{pmatrix} 1 & 3/5 \\ 3/5 & 2 \end{pmatrix}.$$

Gaussian Densities

Suppose we model each density $f_k(x)$ as a multivariate Gaussian with covariance matrix $\mathbf{\Sigma}_k$

$$f_k(x) = \left[(2\pi)^p |\mathbf{\Sigma}_k|\right]^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}_k^{-1}(x-\mu_k)\right).$$

We want to compare the relative likelihoods for $f_k(x)$ at each point. Given the exponential, it's more natural to compare the ratio of the logs to 1.

$$\log \frac{\mathbb{P}(G=k|X=x)}{\mathbb{P}(G=j|X=x)} = \log \frac{f_k \pi_k}{f_j \pi_j} \stackrel{?}{\geq} 0$$

Linear Discriminant Functions

If the covaraince matrices are the same for each k, that is $\mathbf{\Sigma}_k = \mathbf{\Sigma}$ for all k, then

$$f_k(x) = \left[(2\pi)^p |\mathbf{\Sigma}|\right]^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}^{-1}(x-\mu_k)\right).$$

and both the constant and quadratic terms cancel

$$\begin{aligned}
\log \frac{f_k \pi_k}{f_j \pi_j} &= \log \frac{\pi_k}{\pi_j} - \frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}^{-1}(x-\mu_k) + \frac{1}{2}(x-\mu_j)^T \mathbf{\Sigma}^{-1}(x-\mu_j) \\
&= \log \frac{\pi_k}{\pi_j} - \frac{1}{2}(\mu_k+\mu_j)^T \mathbf{\Sigma}^{-1}(\mu_k-\mu_j) + x^T \mathbf{\Sigma}^{-1}(\mu_k-\mu_j) \\
&\geq 0\,.
\end{aligned}$$

This is a linear expression in x and so leads to linear decision boundaries.

Linear Discriminant Functions

The equation

$$0 \geq \log \frac{\pi_k}{\pi_j} - \frac{1}{2}(\mu_k + \mu_j)^T \mathbf{\Sigma}^{-1}(\mu_k - \mu_j) + x^T \mathbf{\Sigma}^{-1}(\mu_k - \mu_j)$$

can be rewritten as

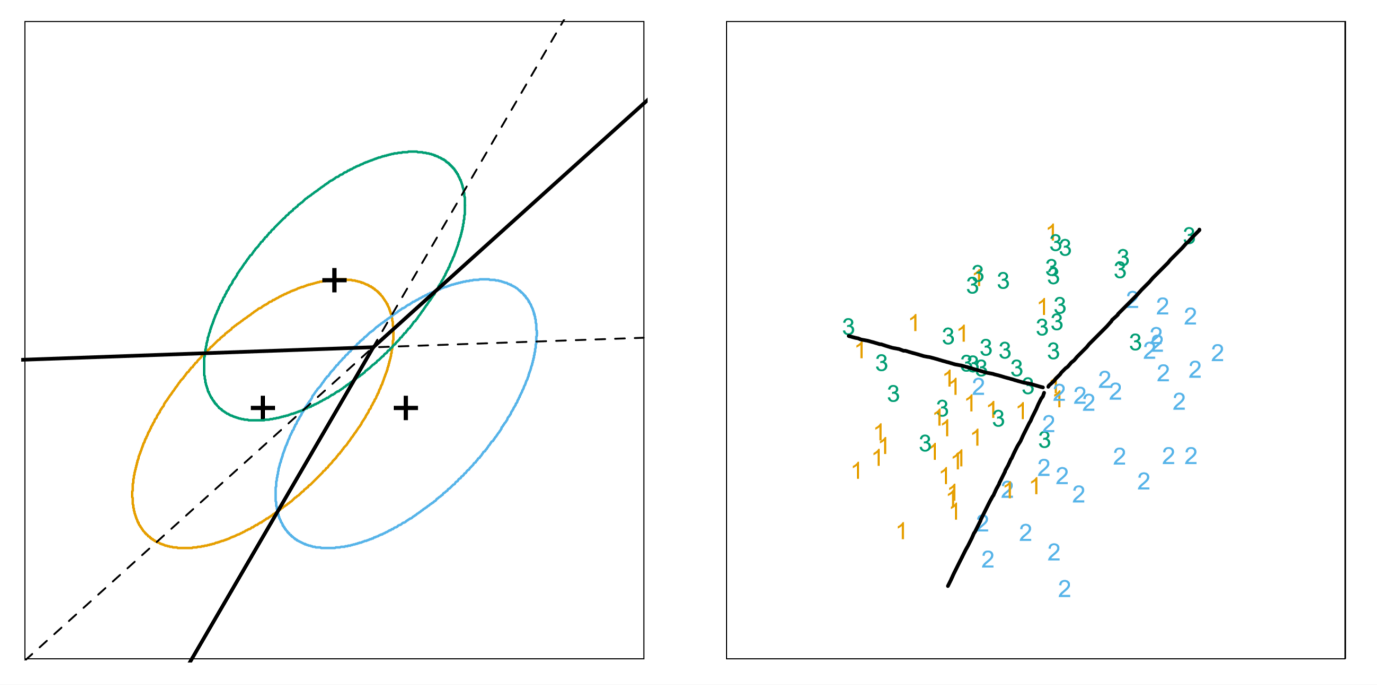
$$\delta_k(x) \geq \delta_k(x)\,,$$

where

$$\delta_k(x) = \log(\pi_k) - \frac{1}{2}\mu_k^T \mathbf{\Sigma}^{-1}\mu_k + x^T \mathbf{\Sigma}^{-1}\mu_k\,.$$

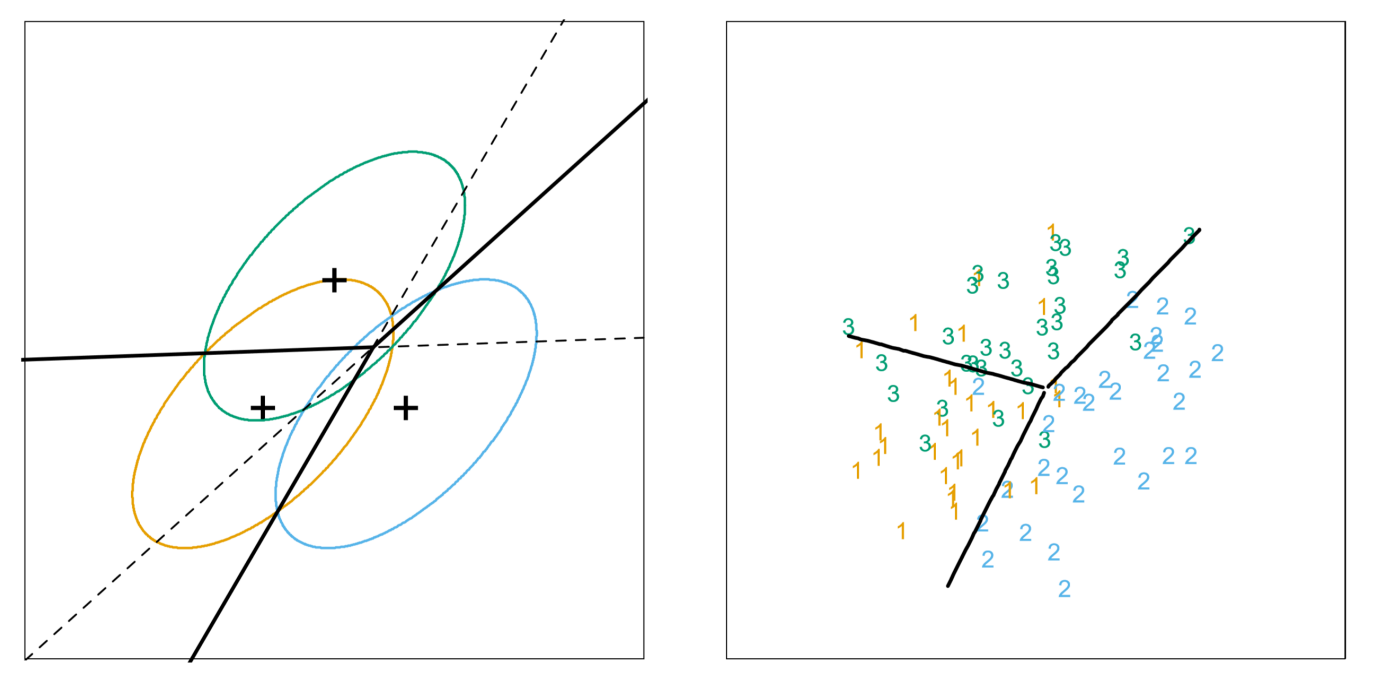
The label on x is given by $\text{argmax}_k \delta_k(x)$. The linear decision boundaries are $\delta_k(x) = \delta_j(x)$.

Linear Discriminant Functions



The label on x is given by $\text{argmax}_k \delta_k(x)$. The linear decision boundaries are $\delta_k(x) = \delta_j(x)$. Above, we see three Gaussian distributions 95% density contours, as well as the decision boundaries.

Linear Discriminant Functions



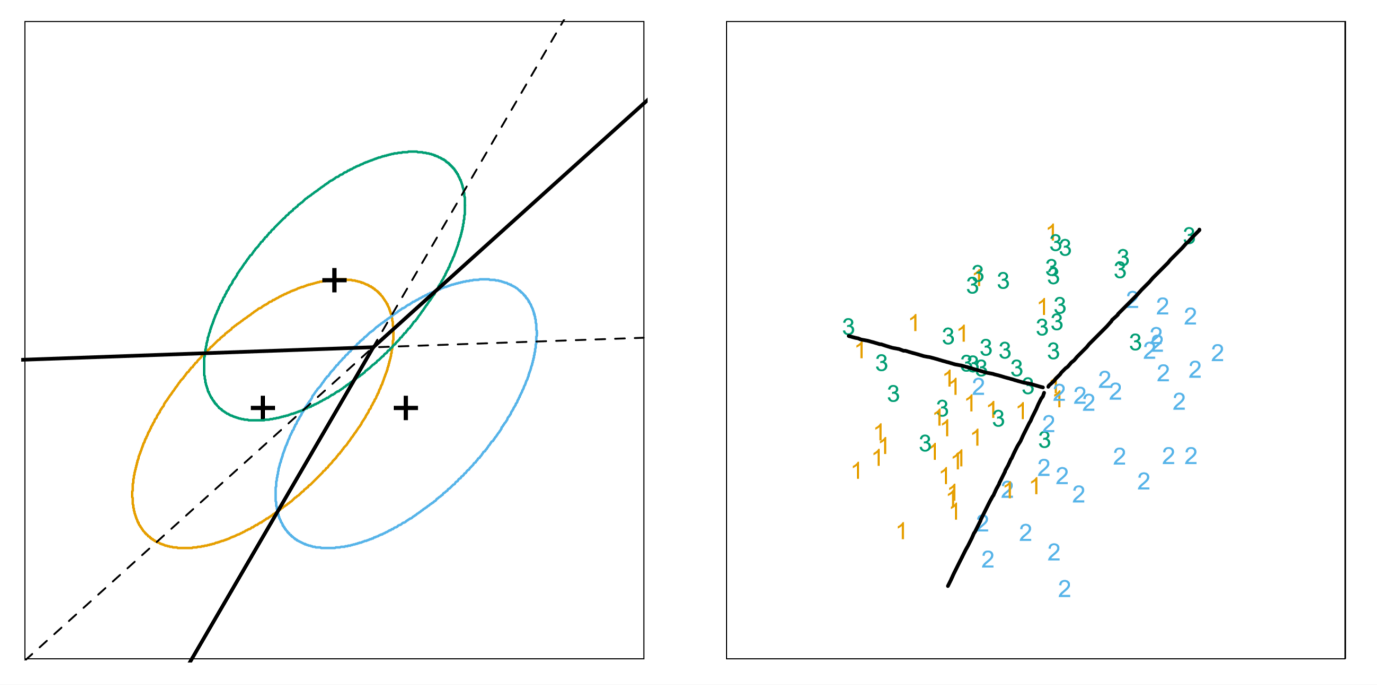
In practice, we need to estimate the parameters of the Gaussian distribution as follows:

$\hat{p}i_k = N_k/N$, where N_k is the number of observations of k.

$\hat{\mu}_k = \frac{1}{N_k} \sum_{y_i=k} x_i$ is the mean of k observations.

$\hat{\mathbf{\Sigma}} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{y_i=k} ||x_i - \hat{\mu}_k||^2$ estimates covariance.

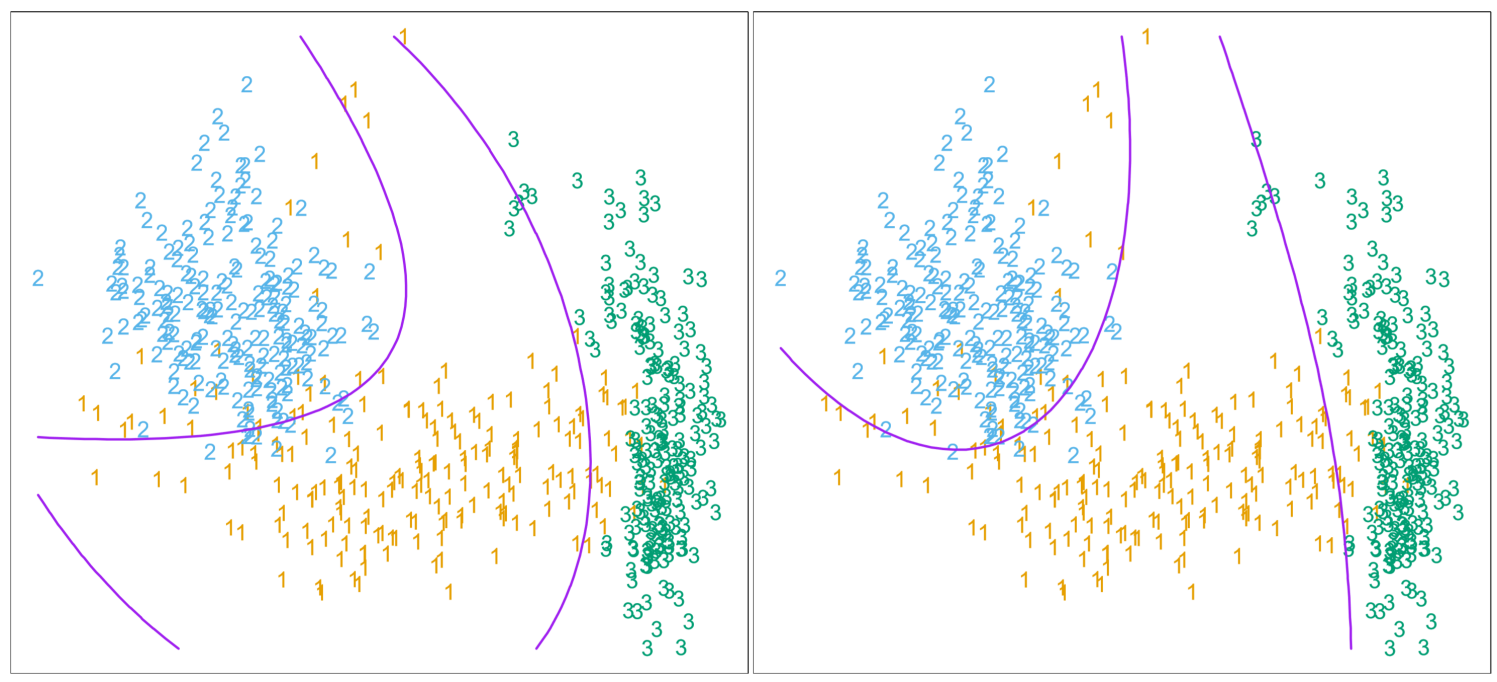
Linear Discriminant Functions



For binary classification it can be shown that the coefficient vector β from least squares is proportional to the coefficient vector of linear discriminant analysis (LDA). But the origin β_0 might be different.

For more labels, the correspondence between LDA and regression can be made by changing the loss function. This again shows the versatility of regression, since LDA can be shown to avoid masking problems.

Quadratic Discriminant Functions

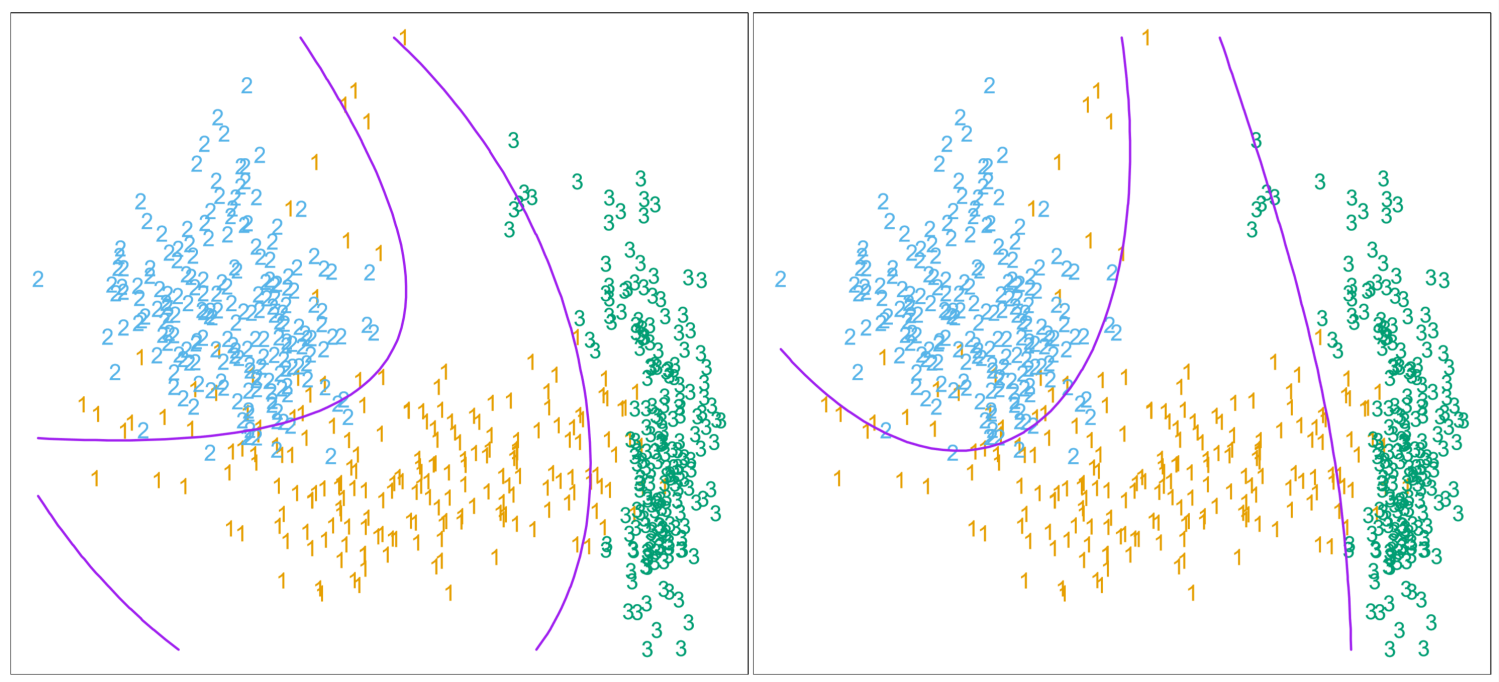


If we remove the requirement that the $\mathbf{\Sigma}_k$ are all equal, the discriminant functions become quadratic

$$\delta_k(x) = -\frac{1}{2}\log|\mathbf{\Sigma}_k| - \frac{1}{2}(x-\mu_k)^T\mathbf{\Sigma}_k^{-1}(x-\mu_k) + \log\pi_k\,.$$

This is known as **quadratic discriminant analysis** (QDA).

Quadratic Discriminant Functions



Here, the right plot uses LDA on X_1, X_2, X_1X_2, X_1^2 and X_2^2 while the right plot uses QDA by directly fitting the coefficients using gradient decent. Both techniques work very well on a wide variety of labeling tasks.

Logistic Regression

Logistic Regression

Logistic regression rises from two places:

- The desire to define a meaningful, smooth probability density functions for discrete data.
- The desire to define a linear model for the conditional probabilities $\mathbb{P}(G = j|X = x)$, while holding constant the fact that they sum to 1.

Taking our hint from the linear discriminant, we fix the probabilities relative to $G = K$ as linear functions

$$\log \frac{\mathbb{P}(G = j|X = x)}{\mathbb{P}(G = K|X = x)} = \beta_{j,0} + x^T \beta_j \quad \forall j = 1, \ldots, K-1 \,.$$

Logistic Regression

We can solve

$$\log \frac{\mathbb{P}(G = j|X = x)}{\mathbb{P}(G = K|X = x)} = \beta_{j,0} + x^T \beta_j \quad \forall j = 1, \ldots, K-1 \,.$$

for

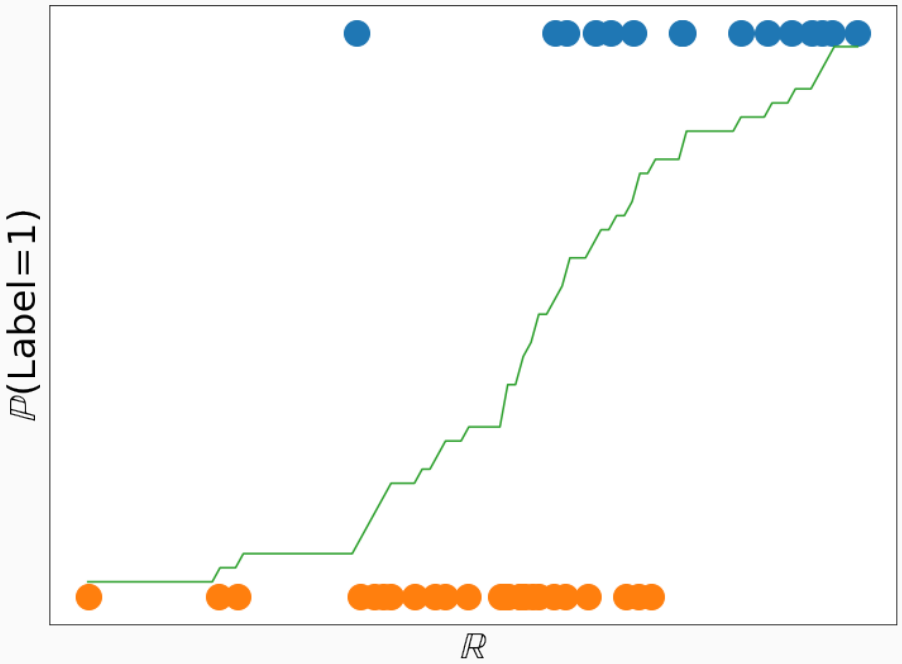
$$\mathbb{P}(G = j|X = x) = \frac{\exp(\beta_{j,0} + x^T \beta_j)}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell,0} + x^T \beta_\ell)} \,, \quad \forall j = 1, \ldots, K-1 \,.$$

and

$$\mathbb{P}(G = K|X = x) = \frac{1}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell,0} + x^T \beta_\ell)} \,.$$

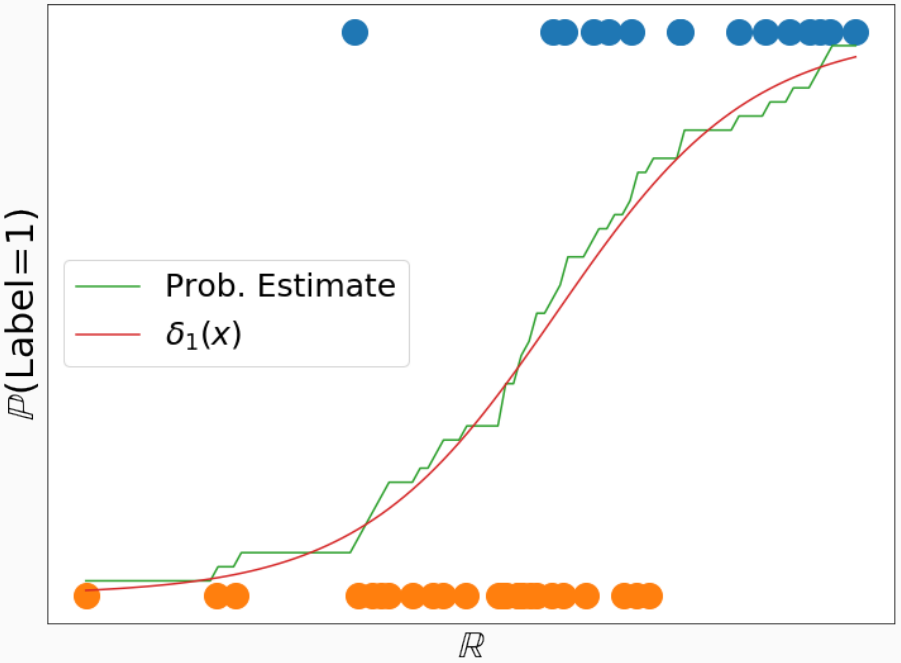
These clearly sum to 1.

One Variable Examples



In one variable, this gives a meaningful smoothing of the density of labels along a continuous axis.

One Variable Examples

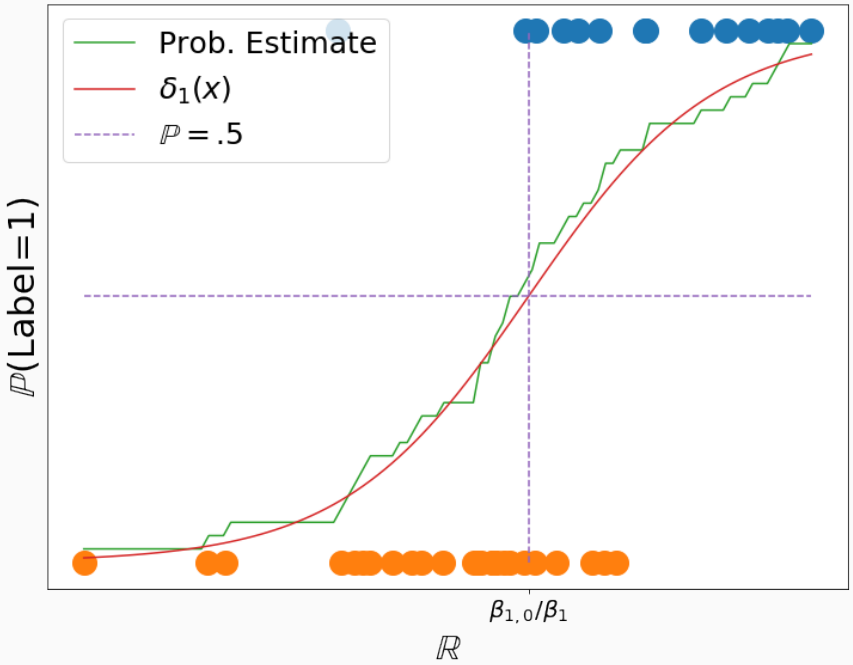


In one variable, this gives a meaningful smoothing of the density of labels along a continuous axis.

$$\delta_1 = \frac{\exp(\beta_{1,0} + x^T\beta_1)}{1 + \exp(\beta_{1,0} + x^T\beta_1)}.$$

In terms of these parameters, $\delta_1 = .5$ when $x = -\beta_{1,0}/\beta_1$

One Variable Examples

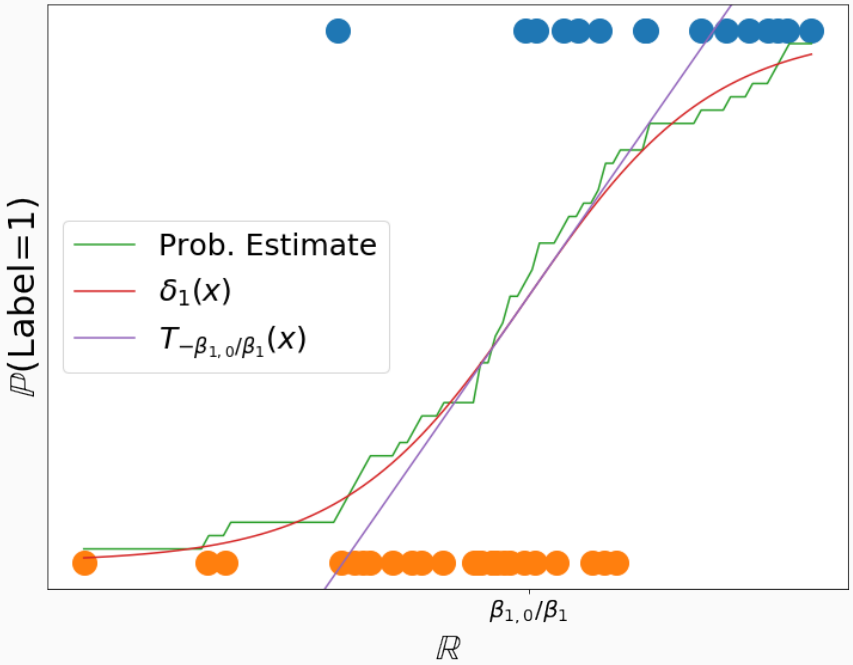


In one variable, this gives a meaningful smoothing of the density of labels along a continuous axis.

$$\delta_1 = \frac{\exp(\beta_{1,0} + x^T\beta_1)}{1 + \exp(\beta_{1,0} + x^T\beta_1)}.$$

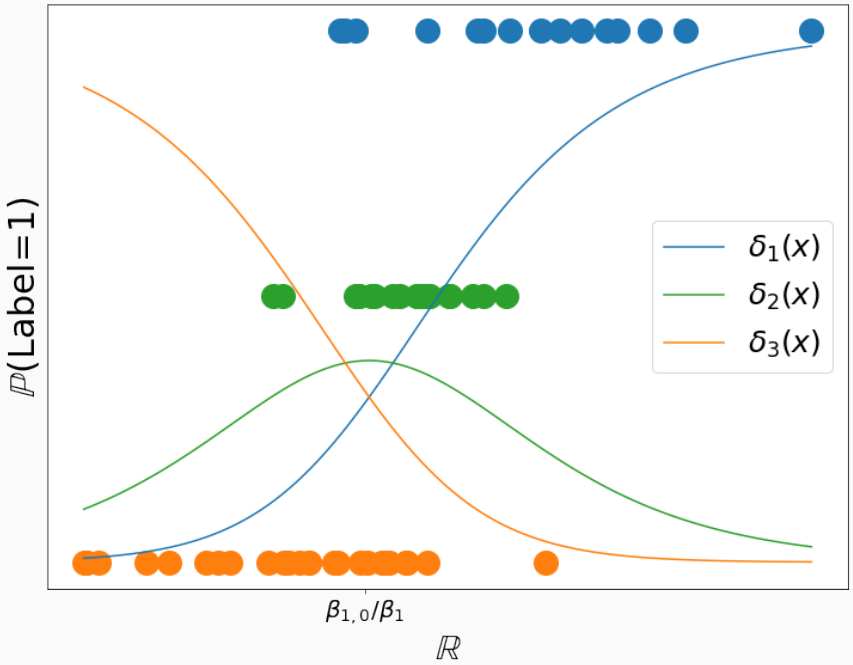
In terms of these parameters, $\delta_1 = .5$ when $x = -\beta_{1,0}/\beta_1$

One Variable Examples



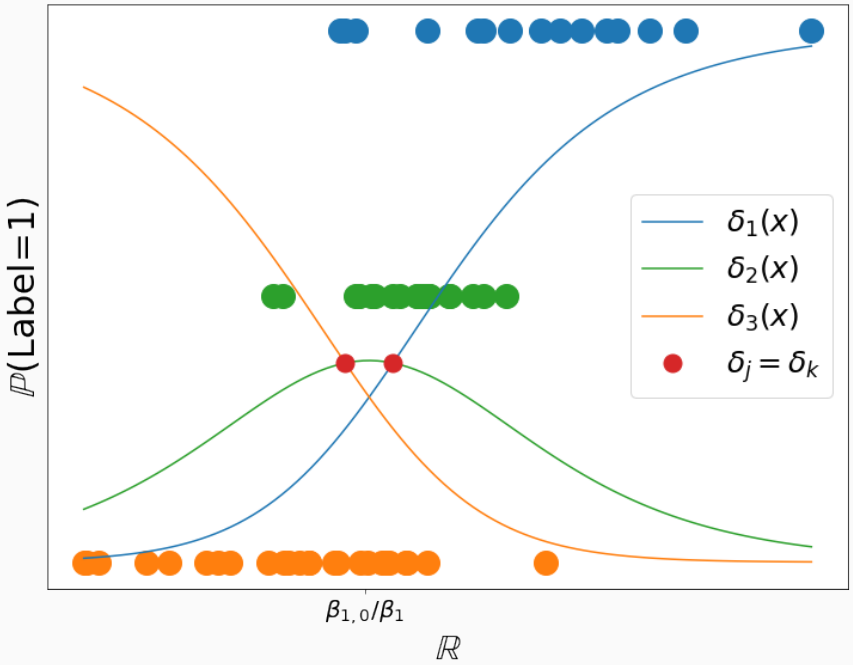
In terms of these parameters, $\delta_1 = .5$ when $x = -\beta_{1,0}/\beta_1$. Furthermore, this is an inflection point and the slope of $\delta_1(x)$ point is $\beta_1/4$.

One Variable Examples



For multiple labels, the dependence on the parameters is less explicit but can still be worked out. Notice that the $\delta_2(x)$ is defined explicitly in terms of δ_1 and δ_2.

One Variable Examples



For multiple labels, the dependence on the parameters is less explicit but can still be worked out. Notice that the $\delta_2(x)$ is defined explicitly in terms of δ_1 and δ_2. For more variables, the intersection points will become our linear decision boundaries $\delta_j(x) = \delta_k(x)$.

Fitting Logistic Regression

Logistic regression is usually fit using maximum likelihood. For N observations of data (x_i, y_i), the likelihood is

$$L(\beta) = \prod_{i=1}^{N} \mathbb{P}(G = y_i | X = x_i; \beta) \,.$$

To maximize $L(\beta)$ is it enough to maximize $\ell(\beta) = \log L(\beta)$.

Fitting Logistic Regression with Newtons Method

Newtons Method

Newtons Method is an iterative method of finding zeros of differential functions. In one variable, we try to find a zero of $f(x)$ by successive approximations of $f(x)$ by it's Taylor polynomial. Starting with x_0, we try to find an improved guess $x + \delta$ by Taylor expanding $f(x + \delta)$ around x:

$$f(x + \delta) \approx f(x) + \delta f'(x) = 0\,.$$

Solving for $\delta = -f(x)/f'(x)$, we have an "improved" guess $x_{new} = x - f(x)/f'(x)$ for the location of the zero. We then iterate until we have arrived at a zero. It can be proved that under reasonable assumptions on $f(x)$ we always will.

Multivariate Newtons Method

For higher dimensional functions, we proceed exactly as before: Let $f(x)$ be differentiable. Expanding $f(x+\delta)$ around x, we find

$$f(x+\delta) \approx f(x) + J(x)\delta = 0\,,$$

where $J_{ij} = \frac{\partial f_i}{\partial x_j}$ is the Jacobean matrix (gradient if $f(x) \in \mathbb{R}$). If $J(x)$ is invertible, we can solve for

$$x_{new} = x - J^{-1}(x)f(x)\,.$$

Optimization via Newtons Method

For a single valued multivariate function $f(x)$, we can use Newtons method to perform optimization. Since optimization is just finding $\nabla f(x) = 0$, we write

$$\nabla f(x + \delta) \approx \nabla f(x) + H_f(x)\delta = 0$$

where $H_f(x) = (\frac{\partial f}{\partial x_i \partial x_j})_{ij}$ is the Hessian matrix. At each iteration then we update x_{old} to

$$x_{new} = x_{old} - H_f^{-1}(x)\nabla f(x)\,.$$

Note, Newtons Method converges much faster than gradient decent, but requires computing higher derivatives which might not exist, or may be hard to compute. As a result, it is often not used.

Fitting Logistic Regression

We will discuss fitting the binary label case. Let y_i take probabilities $k \in \{0, 1\}$. Under the logistic assumption we set

$$\delta_1(x) = \frac{\exp(x^T\beta)}{1 + \exp(x^T\beta)}, \quad \delta_0(x) = \frac{1}{1 + \exp(x^T\beta)},$$

where we have again absorbed β_0 into β. Then

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{N} \log \mathbb{P}(G = y_i | X = x_i; \beta) \\
&= \sum_{i=1}^{N} y_i \log(\delta_1(x_i)) + (1 - y_i) \log(1 - \delta_0(x_i)) \\
&= \sum_{i=1}^{N} y_i x_i^T \beta - \log(1 + e^{x_i^T \beta}).
\end{aligned}$$

Fitting Logistic Regression

Denoting by $\mathbf{d}$ the vector whose i'th coordinate is $\delta_1(x_i)$ fitted to β^{old}, the gradient of

$$\ell(\beta) = \sum_{i=1}^{N} y_i x_i^T \beta - \log(1 + e^{x_i^T \beta}$$

is

$$\nabla \ell(\beta) = \mathbf{X}^T(\mathbf{y} - \mathbf{d}) .$$

Letting $\mathbf{W}$ be the $N \times N$ diagonal matrix with i'th entry $\delta_1(x_i)$ fitted to β^{old}, the Hessian is

$$H = \frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\mathbf{X}^T \mathbf{W} \mathbf{X} .$$

Then

$$\beta^{new} = \beta^{old} + (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{p}) .$$

Fitting Logistic Regression

Logistic regression is a power tool since Newtons Method can be used directly. In addition, it is an explanatory tool, since in general it's coefficients have readily available, explicit interpretations. As a result, Newtons Method is often used in data analysis and situation in which one would like to actually explain outcomes. But it is a power tool in machine learning, and a corner stone of the analysis of categorical variables.