Applied Machine Learning (AML)

Logistic Regression

Oisin Mac Aodha • Siddharth N.

Linear Classification

Generative Versus Discriminative Classifiers

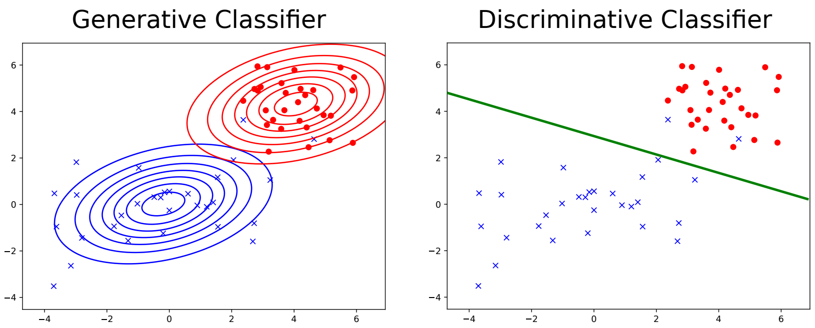
- **Generative** classifiers (e.g. Naive Bayes) model how a class 'generated' the feature vector $p(\boldsymbol{x}|y)$
- Which we then used for classification

$$p(y|\boldsymbol{x}) \propto p(\boldsymbol{x}|y)p(y)$$

- In contrast, **discriminative** classifiers do not waste effort modelling the generative process
- Instead, they model the posterior $p(y|\boldsymbol{x})$ *directly*

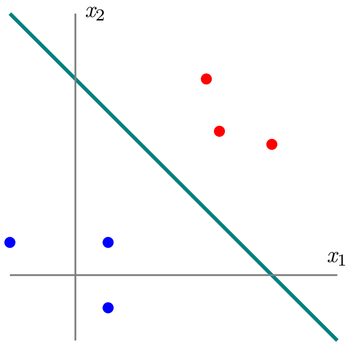
Generative Versus Discriminative Classifiers

- **Generative** approaches model the class conditional densities $p(\boldsymbol{x}|y)$ and priors $p(y)$
- **Discriminative** approaches directly model the posterior $p(y|\boldsymbol{x})$



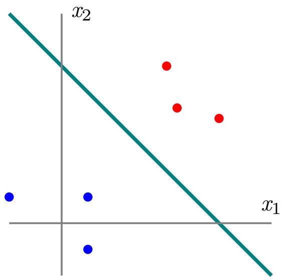
The Linear Classification Problem

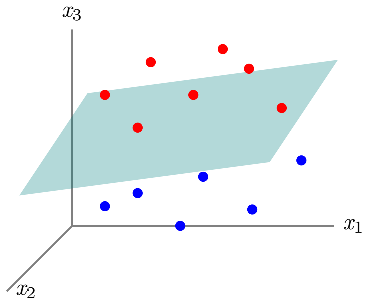
- In **binary linear classification** we are given some input features $\boldsymbol{x}$, with associated class labels y
- The goal is to estimate the parameters $\boldsymbol{w}$ of a hyperplane that can separate the data into the two classes
- The **decision boundary** is the boundary between these two regions, i.e. where the two classes are 'tied'



Linear Classifiers in Higher Dimensions

- In 2D, the decision boundary is represented as a **line**
- In 3D, the decision boundary is represented as a **plane**
- In higher dimensions, it is a **hyperplane**





Linear Classification Model

- In binary linear classification we have a set of input features vector $\boldsymbol{x} \in \mathbb{R}^D$ and binary class labels $y \in \{0, 1\}$

$$\begin{aligned} f(\boldsymbol{x}; \boldsymbol{w}) &= w_o + w_1 x_1 + \ldots + w_D x_D \\ &= w_o + \sum_{d=1}^{D} w_d x_d \\ &= \boldsymbol{w}^\intercal \phi(\boldsymbol{x}) \end{aligned}$$

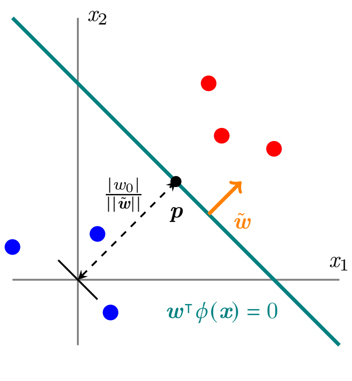
where $\boldsymbol{w} = [w_0, w_1, \ldots, w_D]^\intercal$ and $\phi(\boldsymbol{x}) = [1, x_1, \ldots, x_D]^\intercal$

- To make a prediction we can threshold the output of the function

$$\hat{y} = \begin{cases} 1 & \text{if } \boldsymbol{w}^\intercal \phi(\boldsymbol{x}) >= 0 \\ 0 & \text{if } \boldsymbol{w}^\intercal \phi(\boldsymbol{x}) < 0 \end{cases}$$

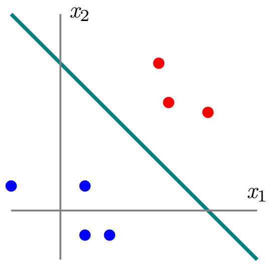
Geometric Perspective

- $\boldsymbol{w}^\intercal \phi(\boldsymbol{x}) = 0$ is the decision boundary
- Let $\tilde{\boldsymbol{w}}$ be the weights without the bias w_0, then $\tilde{\boldsymbol{w}}$ is normal to the decision boundary
- If $w_0 = 0$, $\boldsymbol{w}^\intercal \phi(\boldsymbol{x}) = 0$ is a line passing though the origin and orthogonal to $\tilde{\boldsymbol{w}}$
- When $w_0 \neq 0$, it shifts the location of the decision boundary
- If $\boldsymbol{p}$ is the point on the boundary closest to the origin, then the normal distance from the boundary to the origin is $\frac{|w_0|}{||\tilde{\boldsymbol{w}}||}$

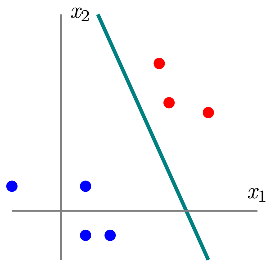


Linear Separability

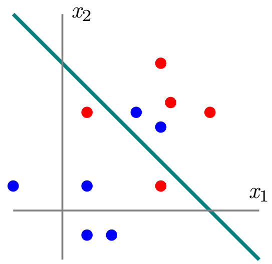
- If we can find a hyperplane to separate the data based on the class labels, the problem is said to be **linearly separable**



Linearly separable



Linearly separable



Linearly **non**-separable

Linear Separability

- If we can find a hyperplane to separate the data based on the classes, the problem is **linearly separable**

- Causes of non perfect separation
  - The linear model is too simple
  - Simple features that do not account for all variations
  - There is noise in the input features
  - There are errors in the class labels

Logistic Regression

Logistic Regression

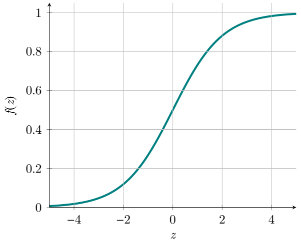
- One problem with our linear classifier, $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^\top \phi(\boldsymbol{x})$, is that the outputs are unbounded, i.e. $f(\boldsymbol{x}; \boldsymbol{w}) \in [-\infty, \infty]$
- We would like to model the posterior $p(y = 1|\boldsymbol{x})$ directly
- To do so, our model predictions need to be in the range $[0, 1]$
- One solution is to 'squash' outputs of $f(\boldsymbol{x}; \boldsymbol{w})$ so that they remain in the range $[0, 1]$

The Logistic Function

- We need a function that returns probabilities, i.e. its outputs are between 0 and 1
- The logistic function provides this

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

- As z goes from $-\infty$ to ∞, $\sigma(z)$ goes from 0 to 1,
- It has a 'sigmoid' shape, i.e. an 'S' like shape



Understanding the Logistic Function

- Here we provide some intuition for how the logistic function works

$$\sigma(z) = \frac{1}{1 + \exp(-z)} = \frac{\exp(z)}{\exp(z) + 1}$$

- As z becomes very *negative* we get

$$\sigma(z) = \frac{\text{small}}{1 + \text{small}} \sim 0$$

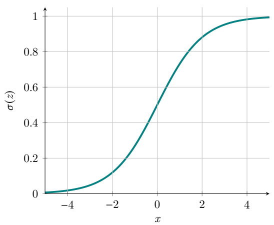
- As z becomes very *positive* we get

$$\sigma(z) = \frac{\text{large}}{1 + \text{large}} \sim 1$$

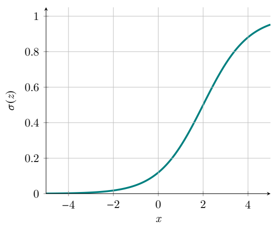
Shape of the Logistic Function

- Modifying the input to the logistic function changes the shape of the function, i.e. it changes the output

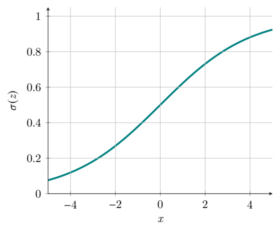
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



$z = x + 0$



$z = x - 2$



$z = 0.5x$

Logistic Regression

- Logistic regression = **linear weights** + **logistic squashing function**
- We model the class probabilities as

$$p(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^\intercal \phi(\boldsymbol{x}))$$

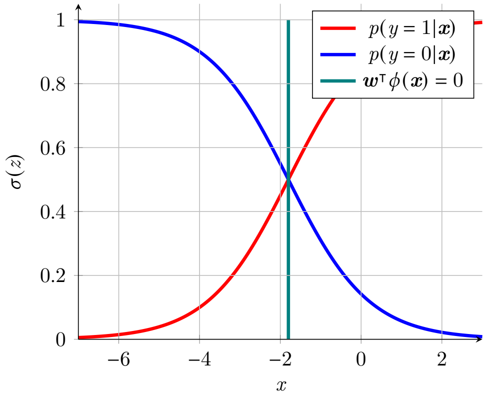
and thus

$$p(y = 0|\boldsymbol{x}) = 1 - \sigma(\boldsymbol{w}^\intercal \phi(\boldsymbol{x}))$$

- $\sigma(z) = 0.5$ when $z = 0$, hence the decision boundary is given by $\boldsymbol{w}^\intercal \phi(\boldsymbol{x}) = 0$
- The decision boundary is a $D - 1$ hyperplane for a D dimensional input space
- Despite the name, this is a model for **classification** not *regression*

Decision Boundary for Logistic Regression

- The **decision boundary** for logistic regression is where $p(y = 1|\boldsymbol{x}; \boldsymbol{w}) = p(y = 0|\boldsymbol{x}) = 0.5$
- The decision boundary occurs where $\boldsymbol{w}^\intercal \phi(\boldsymbol{x}) = 0$
- Logistic regression has a **linear** decision boundary

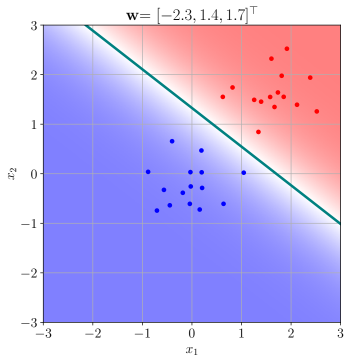


Logistic Regression

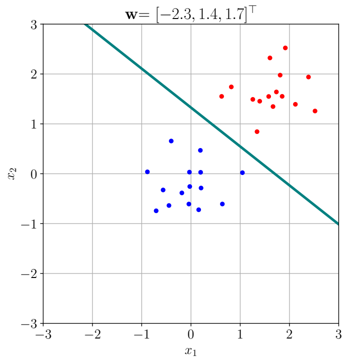
- Let $\tilde{\boldsymbol{w}} = [w_1, ..., w_D]^\intercal$, be the weight vector without the bias term
- The direction of the vector $\tilde{\boldsymbol{w}}$ affects the orientation of the hyperplane. The hyperplane is perpendicular to $\tilde{\boldsymbol{w}}$
- The bias parameter w_0 shifts the position of the hyperplane, but does not alter the orientation
- The magnitude of the weight vector $||\boldsymbol{w}||$ effects how certain the classifications are
- For small $||\boldsymbol{w}||$ most of the probabilities within the region of the decision boundary will be close to 0.5
- For large $||\boldsymbol{w}||$ probabilities in the same region will be close to 0 or 1

Impact of Weights on Classification

- Here we visualise what happens to the predictions when we change the weights



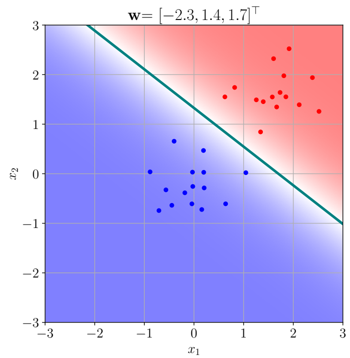
Standard model prediction



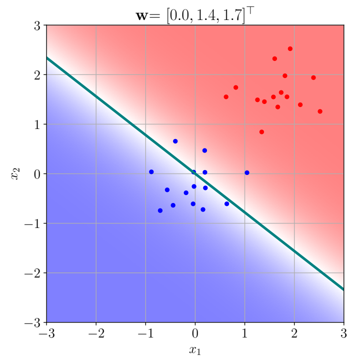
Input data

Impact of Weights on Classification

- On the right we set the bias to $w_0 = 0$



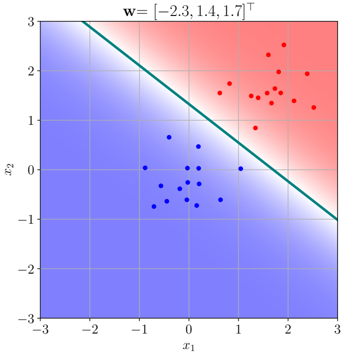
Standard model prediction



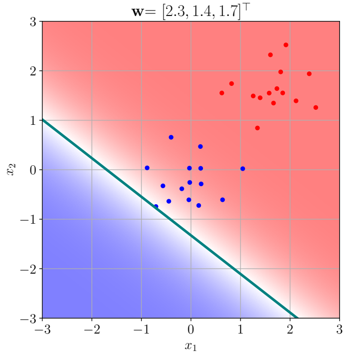
Zero bias

Impact of Weights on Classification

- On the right we set the bias to $w_0 = -w_0$



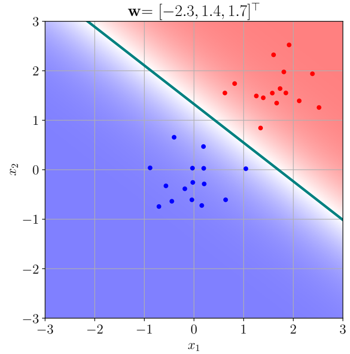
Standard model prediction



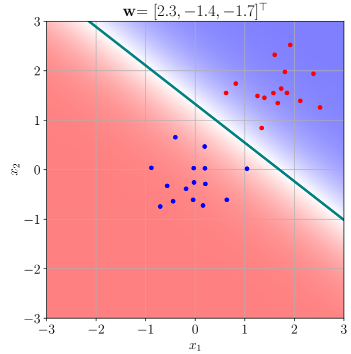
Negative bias

Impact of Weights on Classification

- On the right we negate all the weights $\boldsymbol{w} = -\boldsymbol{w}$



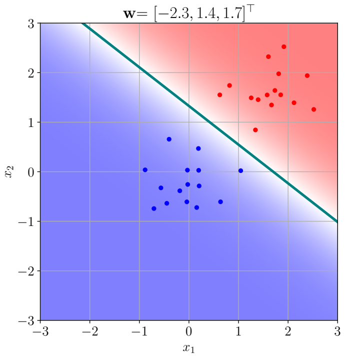
Standard model prediction



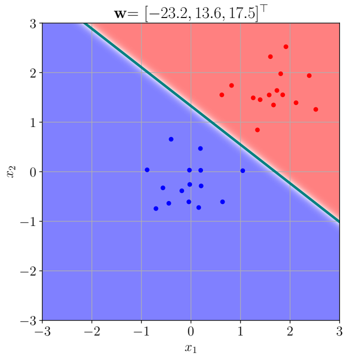
Negative weights

Impact of Weights on Classification

- On the right we scale the weights by a constant $\boldsymbol{w} = c\boldsymbol{w}$



Standard model prediction



Scaled weights

Learning Logistic Regression

Maximum Likelihood Estimation

- We want to estimate the parameters $\boldsymbol{w}$ of the logistic regression model using data
- We will do this via *maximum likelihood estimation*
- Main steps:
  - Write out the likelihood for the model
  - Find the derivatives of the negative log likelihood w.r.t the parameters
  - Adjust the parameters to minimise the negative log likelihood

Likelihood for Binary Classification

- We denote our dataset as $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ...(\boldsymbol{x}_N, y_N)\}$, where $y \in \{0, 1\}$
- We will assume data is independent and identically distributed (i.e. iid assumption)
- To simplify the notation, we will also assume that the bias term w_0 is absorbed into the weight vector, i.e. $\boldsymbol{w} = [w_0, w_1, ..., w_D]^\intercal$ and will let $\boldsymbol{x}_n = [1, x_{n1}, ..., x_{nD}]^\intercal$
- The likelihood is

$$p(\mathcal{D}|\boldsymbol{w}) = \prod_{n=1}^{N} p(y = y_n|\boldsymbol{x}_n; \boldsymbol{w})$$
$$= \prod_{n=1}^{N} p(y = 1|\boldsymbol{x}_n; \boldsymbol{w})^{y_n}(1 - p(y = 1|\boldsymbol{x}_n; \boldsymbol{w}))^{1-y_n}$$

Negative Log Likelihood

- The likelihood is

$$p(\mathcal{D}|\boldsymbol{w}) = \prod_{n=1}^{N} p(y = 1|\boldsymbol{x}_n; \boldsymbol{w})^{y_n}(1 - p(y = 1|\boldsymbol{x}_n; \boldsymbol{w}))^{1-y_n}$$

- Hence, the **negative log likelihood**, $\text{NLL}(\boldsymbol{w}) = -\frac{1}{N}\log p(\mathcal{D}|\boldsymbol{w})$, is given by

$$\text{NLL}(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \sigma(\boldsymbol{w}^\intercal \boldsymbol{x}_n) + (1 - y_n)\log(1 - \sigma(\boldsymbol{w}^\intercal \boldsymbol{x}_n))\right]$$

Maximising the Likelihood

- To find the maximum likelihood parameter estimate, we must solve

$$\frac{\partial \text{NLL}(\boldsymbol{w})}{\partial w_d} = 0$$

- It turns out that the likelihood has a unique optimum, i.e. it is *convex*
- Unfortunately, we cannot minimise the $NLL(\boldsymbol{w})$ directly using a closed form solution. Instead, we need to use a numerical optimisation method (i.e. gradient descent)
- To minimise it, we solve for the gradient

$$\frac{\partial \text{NLL}(\boldsymbol{w})}{\partial w_d} = \frac{1}{N}\sum_{n=1}^{N}(\sigma(\boldsymbol{w}^\intercal \boldsymbol{x}_n) - y_n)x_{nd}$$

Visualising the NLL Loss Surface

- NLL loss surface for binary logistic regression applied to the Iris dataset with one feature and one bias term

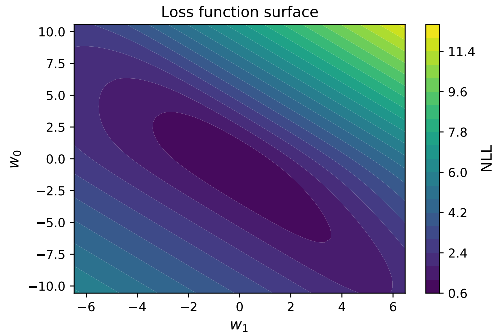
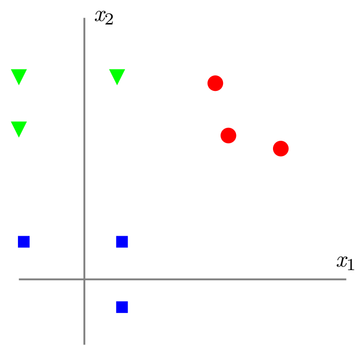


Figure adapted from Probabilistic Machine Learning: An Introduction, K. Murphy

Multiclass Classification

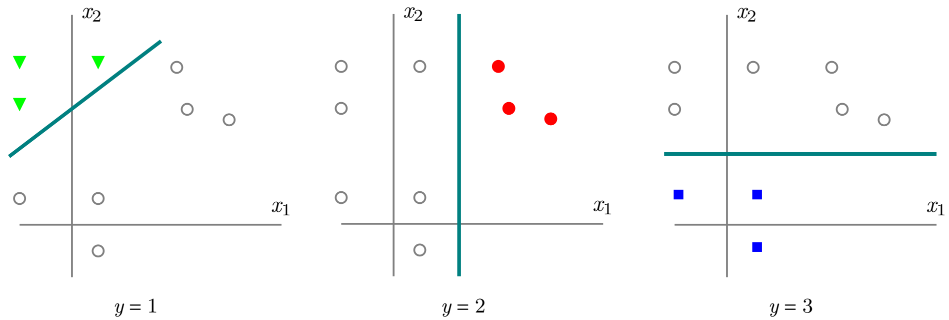
More Than Two Classes

- What if we have more than two classes, i.e. $y \in \{1, \ldots, C\}$?
- Binary classification is not directly applicable here. We need another approach



One-vs-Rest (OvR) Classification

- In OvR classification, the idea is to split the data into different "C" versus "not C" problems
- We train a *separate* classifier, with an associated weight vector $\mathbf{w}_c$, for each class



One-vs-Rest (OvR) Classification

- For each of the C classes we need to train a separate classifier, $p(y = c|\boldsymbol{x}) = \sigma(\boldsymbol{w}_c^\intercal \phi(\boldsymbol{x}))$
- To assign a new data point $\boldsymbol{x}$ to one of the classes, we need to evaluate it using each of the different per-class classifiers
- We select the maximum of the different classifiers as the predicted class, i.e.

$$\hat{y} = \arg\max_c \sigma(\boldsymbol{w}_c^\intercal \phi(\boldsymbol{x}))$$

- Note that the sum of the probabilities of the different classifiers is not constrained to be 1
- The OvR approach is a general one that can be applied to any binary classifier

Multinomial (Softmax) Logistic Regression

- An alternative approach is to create a single model which has parameters for all classes
- Multinomial logistic regression is an extension of binary logistic regression that can handle multiple classes using the *softmax* function

$$p(y = c|\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_c^\intercal \phi(\boldsymbol{x}))}{\sum_{k=1}^{C} \exp(\boldsymbol{w}_k^\intercal \phi(\boldsymbol{x}))}$$

- Note that $0 \leq p(y = c|\boldsymbol{x}) \leq 1$ and $\sum_{k=1}^{C} p(y = k|\boldsymbol{x}) = 1$

Properties of the Softmax Function

- The softmax function $\boldsymbol{s}()$ converts a vector of K real numbers, $\boldsymbol{z} \in \mathbb{R}^K$, into a probability distribution of K possible outcomes

$$\boldsymbol{s}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_{k=1}^{K} \exp(z_k)}$$

- It applies the standard exponential function to each element z_i and normalises these values by dividing by the sum of all these exponentials
- The normalisation ensures that the sum of the components of the output vector is 1, i.e. $\sum_{i=1}^{K} \boldsymbol{s}(\boldsymbol{z})_i = 1$

Summary

- We discussed linear classification
- We presented a discriminative approach for linear classification called *logistic regression*
- For a D dimensional input space, there are $D + 1$ parameters (i.e. weights) that need to be learned in binary classification
- We showed that we can derive an expression for estimating the parameters for this model using maximum likelihood estimation
- It is a simple model, but can be very effective. Often it should be one of the first models to try