1 Binary Logistic Regression

Model

- Binary classification: $y \in \{0, 1\}$
- Want to predict probability of being in a particular class: $P(y = 1|\mathbf{x}; \mathbf{w})$
- Could fit a linear model: $f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T\mathbf{x}$
- But this could give predictions outside $[0, 1]$ for some test inputs (invalid probabilities)
- Use the sigmoid function to force the output to lie in the $[0, 1]$ range:

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

- Interpret $f(\mathbf{x}; \mathbf{w}) = P(y = 1|\mathbf{x}; \mathbf{w})$, implying $P(y = 0|\mathbf{x}; \mathbf{w}) = 1 - f(\mathbf{x}; \mathbf{w})$

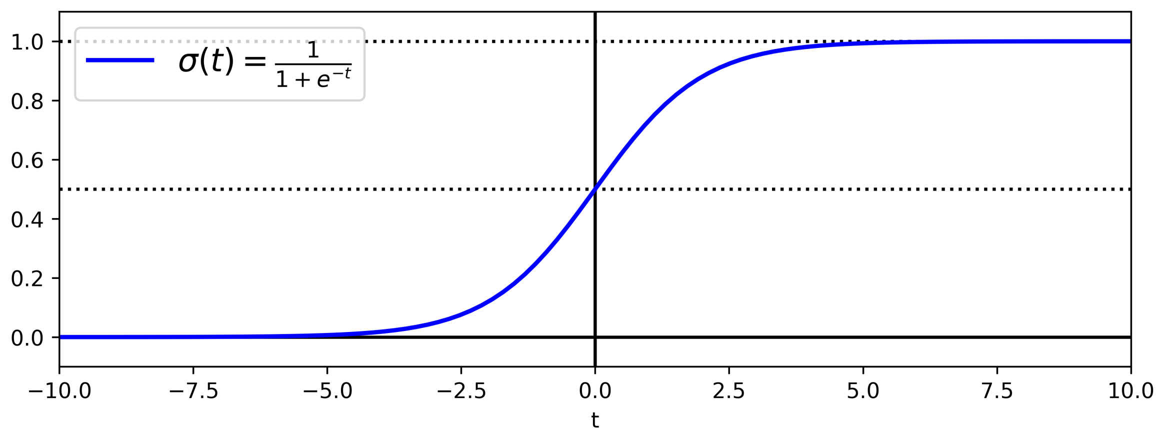

Figura 1: Function used to force the output to lie in the $[0, 1]$ range

Loss Funciton

We observe data $\{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, with $y \in \{0, 1\}$, Using maximum likehood:

$$\begin{aligned} L(\mathbf{w}) &= P(y^{(1)}|\mathbf{x}^{(1)};\mathbf{w}) \cdot P(y^{(2)}|\mathbf{x}^{(2)};\mathbf{w}) \cdots P(y^{(n)}|\mathbf{x}^{(n)};\mathbf{w}) \\ &= \prod_{n=1}^{N} P(y^{(n)}|\mathbf{x}^{(n)};\mathbf{w}) \end{aligned}$$

minimising the negative log likehood

$$\begin{aligned} J(\mathbf{w}) = -\log L(\mathbf{w}) &= -\log \prod_{n=1}^{N} P(y^{(n)}|\mathbf{x}^{(n)};\mathbf{w}) = -\sum_{n=1}^{N} \log P(y^{(n)}|\mathbf{x}^{(n)};\mathbf{w}) \\ (*)\ P(y|\mathbf{x};\mathbf{w}) &= \begin{cases} f(\mathbf{x};\mathbf{w}) & if \quad y=1 \\ 1-f(\mathbf{x};\mathbf{y}) & if \quad y=0 \end{cases} = \begin{cases} \sigma(\mathbf{w}^T;\mathbf{x}) & if \quad y=1 \\ 1-\sigma(\mathbf{w}^T;\mathbf{x}) & if \quad y=0 \end{cases} \\ \implies P(y|\mathbf{x};\mathbf{w}) &= \sigma(\mathbf{w}^T;\mathbf{x})^y(1-\sigma(\mathbf{w}^T;\mathbf{x}))^{1-y} \\ &= -\sum_{n=1}^{N} \log\left[\sigma(\mathbf{w}^T;\mathbf{x}^{(n)})^y(1-\sigma(\mathbf{w}^T;\mathbf{x}^{(n)})^{1-y^{(n)}})\right] \\ &= -\sum_{n=1}^{N} [\log \sigma(\mathbf{w}^T;\mathbf{x}^{(n)})^y + (1-y^{(n)}) \cdot \log(1-\sigma(\mathbf{w}^T;\mathbf{x}^{(n)}))] \end{aligned}$$

1.1 Gradient Descent

- We have some function $J(\mathbf{w})$ that we want to minimise w.r.t parameters $\mathbf{w}$;
- Idea: Start with a random $\mathbf{w}$ and then keep updating it to reduce $J(\mathbf{w})$;
- This method could get stuck in a local minimum;
- As we get closer to the minimum, the step sizes automatically gets smaller.

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k \cdot \frac{\partial J}{\partial \mathbf{w}} \tag{1}$$

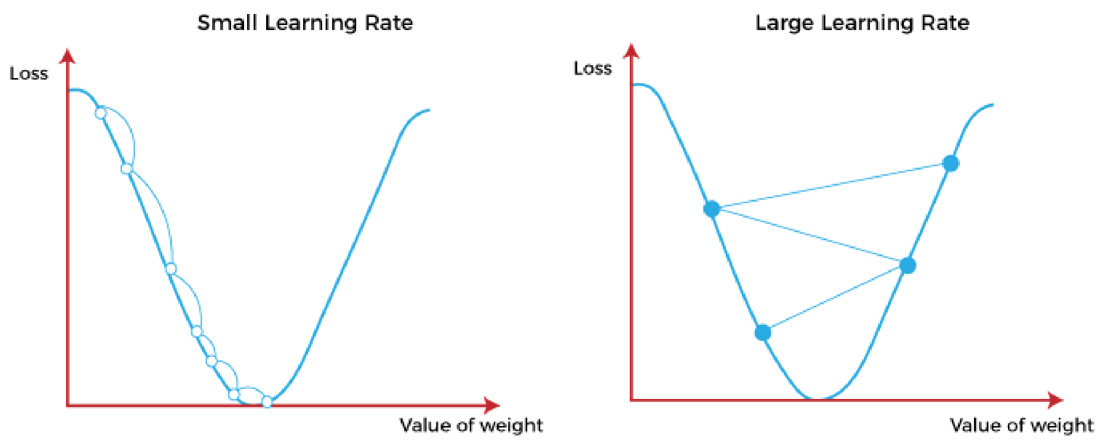

Figura 2: Potential problems

Returning to Loss Function, we use maximum likehood estimation, or equivalently we want to minimise the negative log likehood:

$$J(\mathbf{w}) = -\log \prod_{n=1}^{N} P(y^{(n)}|\mathbf{x}^{(n)};\mathbf{w}) = -\sum_{n=1}^{N} [\log \sigma(\mathbf{w}^T;\mathbf{x}^{(n)})^y + (1-y^{(n)}) \cdot \log(1-\sigma(\mathbf{w}^T;\mathbf{x}^{(n)}))]$$

To minimise this loss, we need the gradients $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$. Using vector and matrix derivatives, we can show that:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{n=1}^{N} (y^{(n)} - f(\mathbf{x}^{(n)};\mathbf{w}))\mathbf{x}^{(n)}$$

To optimise the loss, you could try setting $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 0$. But you will see this does not give closed-form solution (as in linear regression). So instead we use gradient descent (1).

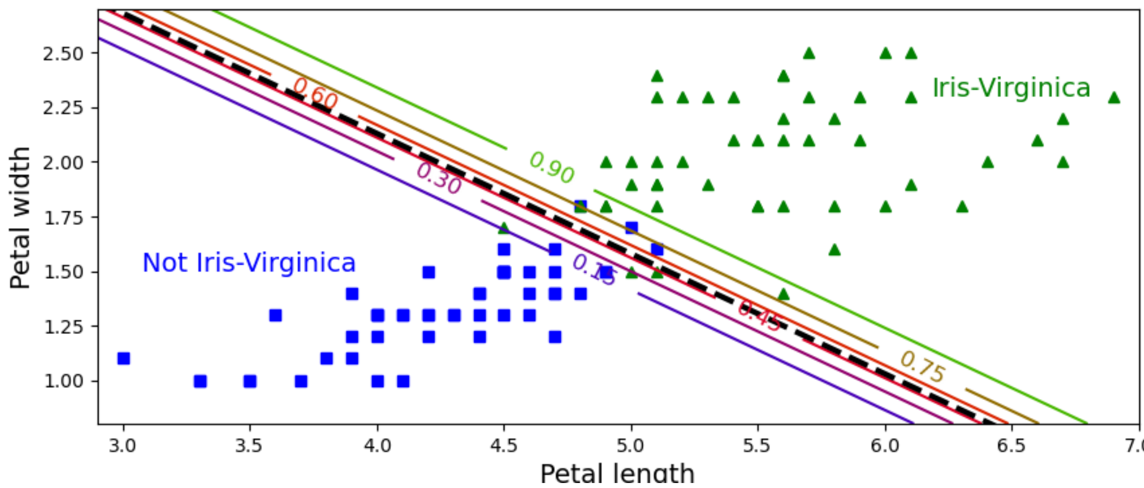

Figura 3: Iris-Virginica Prediction in Logistic Regression.

1.2 Decision Boundary

The decision boundary is the value od $\mathbf{x}$ for which $f(\mathbf{x};\mathbf{w}) = \sigma(\mathbf{w}^T, \mathbf{x}) = 0.5 \implies \mathbf{w}^T \cdot \mathbf{x} = 0$. Here it might be easier to explicity include the bias term, i.e. $f(\mathbf{x};\mathbf{w}) = \sigma(w_0 + \mathbf{w}^T\mathbf{x}) = 0.5$. Let's first consisder the 2-D case.

1. Sketch the line $w_0 + w_1x_1 + w_2x_2 = 0$ in the $x_1 - x_2$ plane;
2. Sketch the vector $\mathbf{w} = [w_1 w_2]^T$ in the same plane;
3. Redraw the line in (1), but pretend $w_0 = 0$;
4. Prove that the line in (3) is orthogonal to the line in (2).

This proves that $\mathbf{w}$ is $\perp$ to the decision boundary.

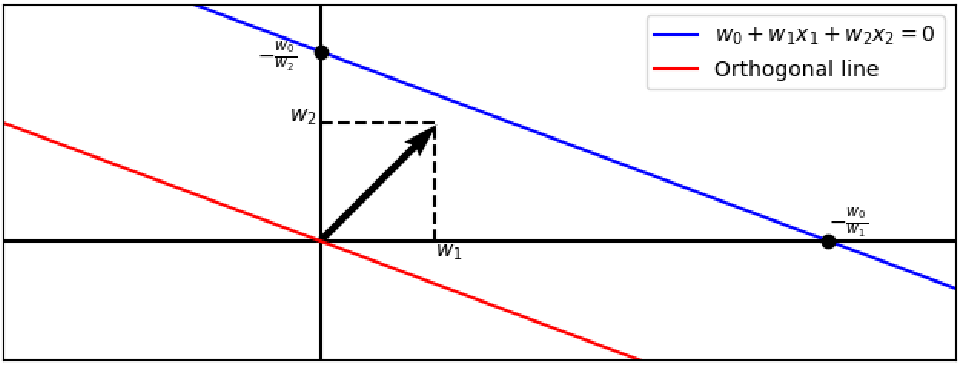

Figura 4: Boundary Decision in 2D

we can extend the above to higher dimensions. If we first ignore the bias term, the decision is given by:

$$\begin{aligned} w_1x_1 + w_2x_2 + \cdots + w_Dx_D = 0 \\ = \mathbf{w}^T\mathbf{x} = 0 \end{aligned}$$

If we thinck of $\mathbf{w}$ as a vector in $\mathbf{x}$-space, then the $\mathbf{x}$ vectors on the decisions boundary is orthogonal to $\mathbf{w}$, since their dot product is zero: $\mathbf{w} \cdot \mathbf{x} = 0$. We can add the bias back in:

$$w_0 + \mathbf{w}^T\mathbf{x} = 0 \tag{2}$$

This has the effect of offshering the decision boundary in $\mathbf{x}$-space.

The lenght of $\mathbf{w}$, i.e. $||\mathbf{w}||$, inlfuences the "steepness"of the decision boundary, for very large $||\mathbf{w}||$, even points that are very close to the decision boundary will be assigned very high or low probabilities $P(y = 1|\mathbf{x}, \mathbf{w})$, with small $||\mathbf{w}||$, probability assignment will be more graual.

1.3 Basis function and regularisation

Basis funcitons:

Anywhere we wrote an $\mathbf{x}$ the features vector $\mathbf{x}$ can be replaced with basis functions $\phi(\mathbf{x})$.

Regularization:

As in linear regression, we can perform regularised logistic regression by penalising the weights:

$$\begin{aligned} J(\mathbf{w}) &= -\log L(\mathbf{w}) + \lambda \sum_{d=1}^{D} w_d^2 \\ &= -\sum_{n=1}^{N} [y^{(n)} \log \sigma(\mathbf{w}^T\mathbf{x}^{(n)}) + (1 - y^{(n)}) \log (1 - \sigma(\mathbf{w}^T\mathbf{x}^{(n)}))] + \lambda \sum_{d=1}^{D} w_d^2 \end{aligned}$$

1.4 Multiclass Logistic Regression

One-vs-All Classification

We define a binary classificator to each class, after that, we attribute a new sample with greater probabilitie.

In this strategy, each classificator is trained using all samples of the training set.

Applying One-vs-All method,

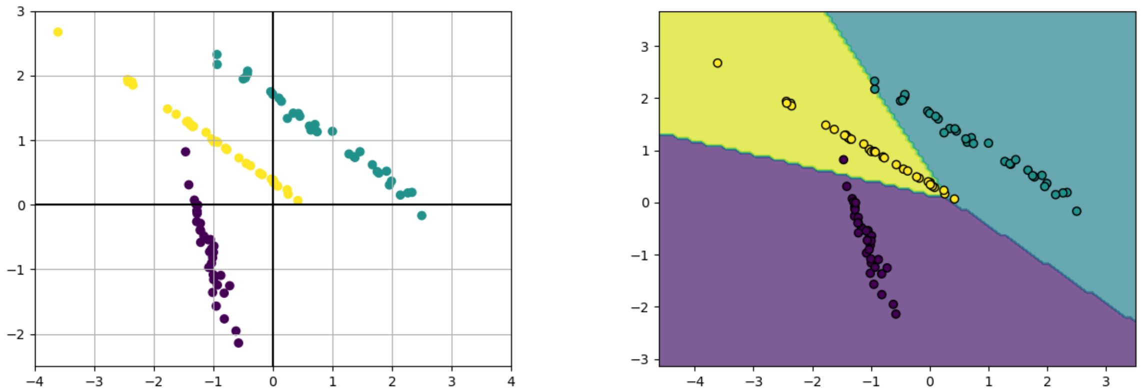

Figura 5: One-vs-Rest method

Softmax Regression

For binary regression we had $f(\mathbf{w};\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$ with $y \in [0,1]$. We interpreted the output as $P(y=1|\mathbf{x};\mathbf{x})$, implying $P(y=0|\mathbf{x}^T;\mathbf{w}) = 1 - f(\mathbf{x};\mathbf{w})$. For the multiclass setting we now have $y \in \{1,2,\dots,K\}$. Ideia: instead of just outputting a single value for the positive class, let's output a vector of probabiblities for aech class:

$$\mathbf{f}(\mathbf{x};\mathbf{W}) = \begin{bmatrix} P(y=1|\mathbf{x};\mathbf{W}_1) \\ P(y=2|\mathbf{x};\mathbf{W}_2) \\ \vdots \\ P(y=K|\mathbf{x};\mathbf{W}_K) \end{bmatrix}$$

We will now build up to a model that does this.

Each element in $\mathbf{f}(\mathbf{x};\mathbf{W})$ should be a "score"for how well input $\mathbf{x}$ matches that class, for input $\mathbf{x}$, let's set the score for class k to $\mathbf{w}_k^T\mathbf{x}$. But probabilities need to be positive, so let's take te exponential $e^{\mathbf{w}_k^T\mathbf{x}}$, but probabilities need to sum to one, normalising, we have

$$P(y=k|\mathbf{x};\mathbf{w}_k) = \frac{e^{\mathbf{w}_k^T\mathbf{x}}}{\sum\limits_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}}} \tag{3}$$

This gives us the softmax regression model:

$$\mathbf{f}(\mathbf{x};\mathbf{W}) = \frac{1}{\sum\limits_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}}} \begin{bmatrix} e^{\mathbf{w}_1^T} \\ e^{\mathbf{w}_2^T} \\ \vdots \\ e^{\mathbf{w}_k^T} \end{bmatrix}$$

Optimization

Fit model using maximum likelihood. Equivalent to minimising the negative log likelihood:

$$J(\mathbf{W}) = -\log L(\mathbf{W}) = -\sum_{n=1}^{N} \log P(y^{(n)}|\mathbf{x}^{(n)}; \mathbf{W})$$
$$= -\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{I}\{y^{(n)} = k\} \log \frac{e^{\mathbf{x}_k^T \mathbf{x}^{(n)}}}{\sum\limits_{j=1}^{K} e^{\mathbf{w}_j^T \mathbf{x}^{(n)}}}$$

Derivates:

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_k} = -\sum_{n=1}^{N}\sum_{k=1}^{K} \left(\mathbb{I}\{y^{(n)} = k\} - f_k(\mathbf{x}^{(n)}; \mathbf{w}_k)\right) \mathbf{x}^{(n)}$$

Using these derivates, we can minimise the loss using gradient descent.

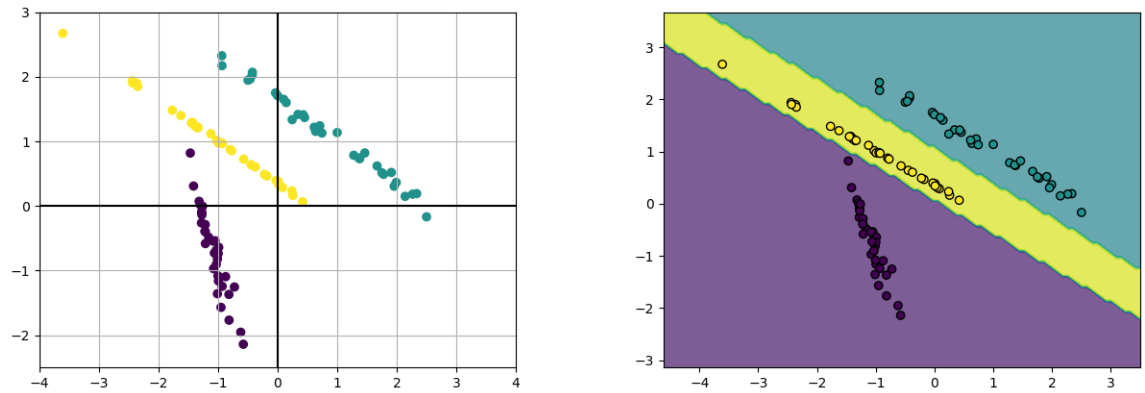

Figura 6: Softmax method