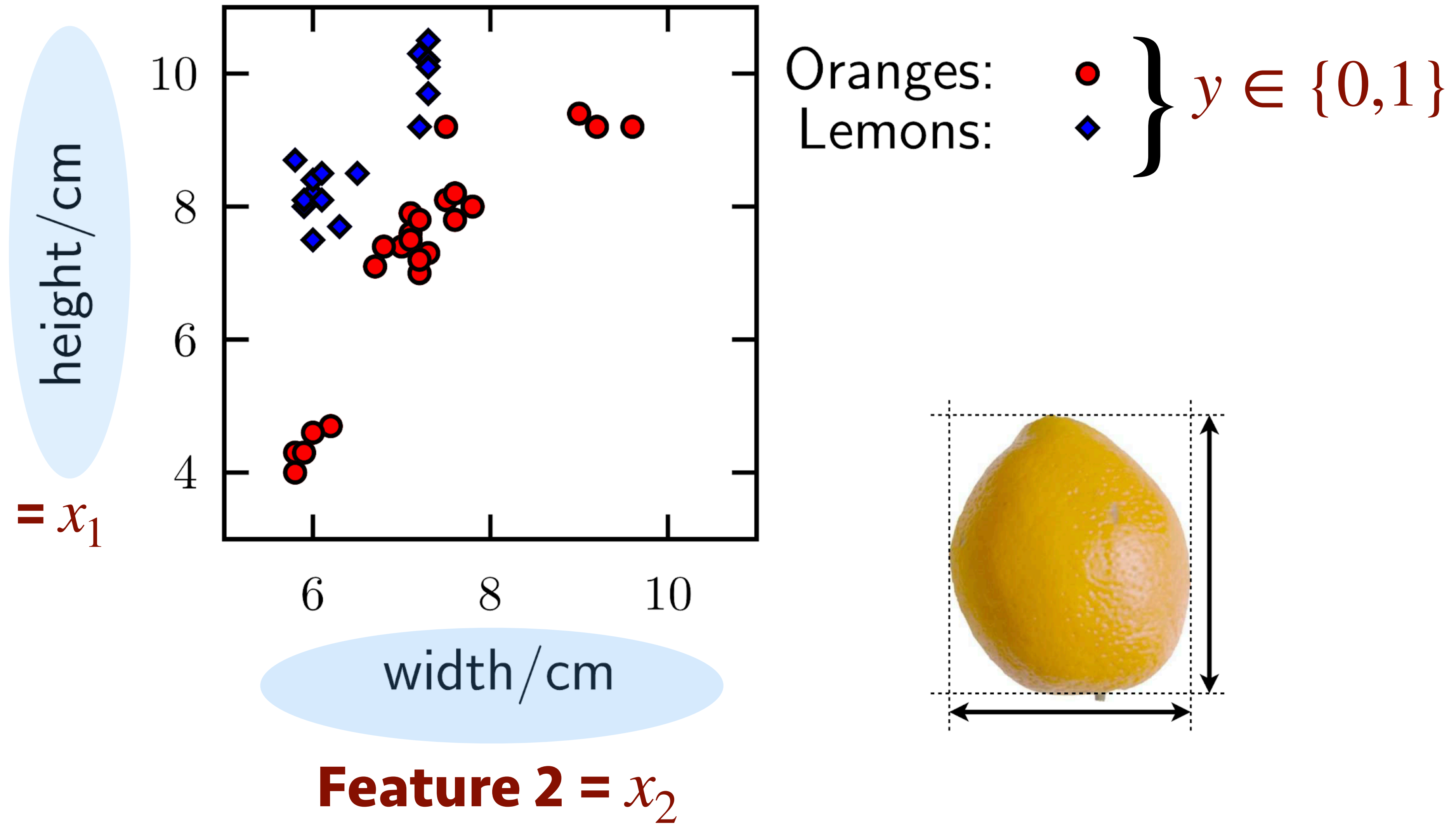
COMPSCI 5100
ML & AI for Data Science

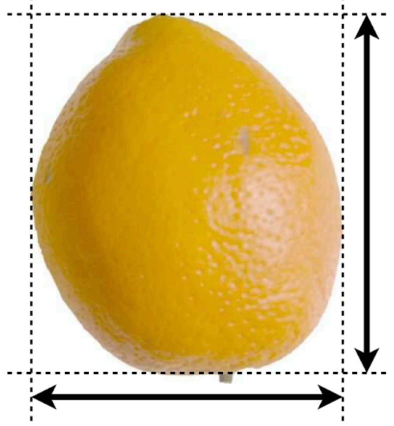
Ali Gooya

Ali.gooya@glasgow.ac.uk

Logistic regression

- A popular statistical model used for **binary** classification
- Can be derived from **Linear Regressio**





Least squares for classification

- Training data: $(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2)\cdots(\mathbf{x}_N, t_N)$, where $\mathbf{x}_i = [x_{i_1}, x_{i_2}]^T$,
- t_i = target label of the i^{th} sample, $t_i \in \{-1,1\}$
- Classification is done by specifying the plane separating two classes (decision boundary)
- The decision boundary is then given by $y(\mathbf{x}) = cte$ (0)
- $y(\mathbf{x})$ denotes the model's prediction (discriminant function).
- For example, let's use a linear regression model:

$$y_i = w_0 + w_1 x_{i_1} + w_2 x_{i_2} = w_o + \mathbf{w}^T \mathbf{x}_i$$

In this case, $y(\mathbf{x}) = 0$ is a line in the plane of x_{i_1}, x_{i_2}

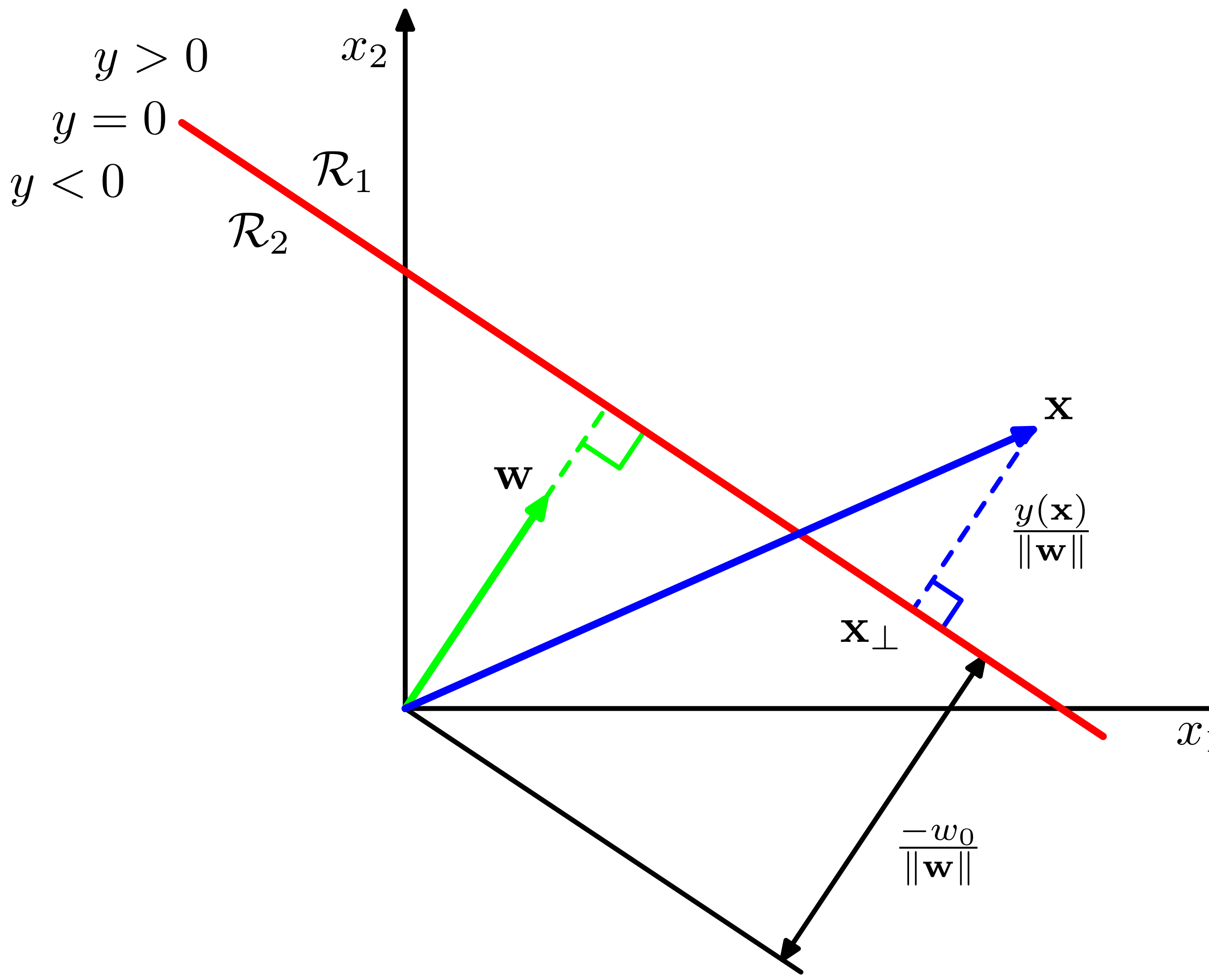
The parameters can be obtained by minimising:

$$J(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i=1:N} (y_i - t_i)^2$$

Geometry of the linear regression

- Distance of $\mathbf{x}$ from decision boundary $y = 0$ is proportional to $y(\mathbf{x})$
- $\mathbf{w}$ is normal to the decision boundary.
- Hence, the cost function penalises the distance of points from the decision plane

Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.



Why bother with logistic regression?

- Classification based on least-squares is prone to outliers.
- See (C. Bishop's PRML Book, Ch. 4)

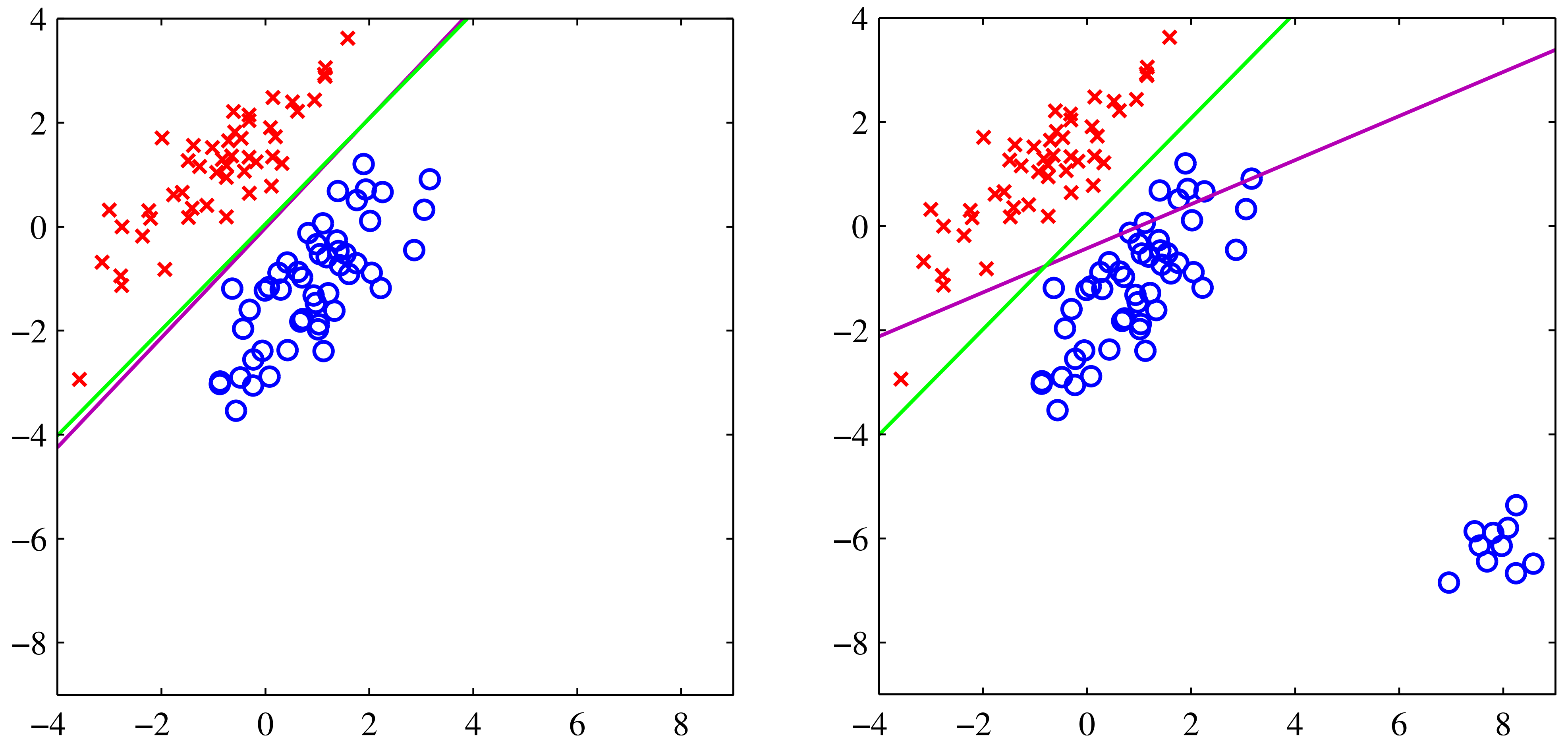


Figure 4.4 The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve), which is discussed later in Section 4.3.2. The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

Logistic regression

. Without loss of generality, assume that $t \in \{0,1\}$

. We use an activation function to make the classifier more robust to the outliers: $y(\mathbf{x}) = \sigma(w_0 + \mathbf{w}^T\mathbf{x})$

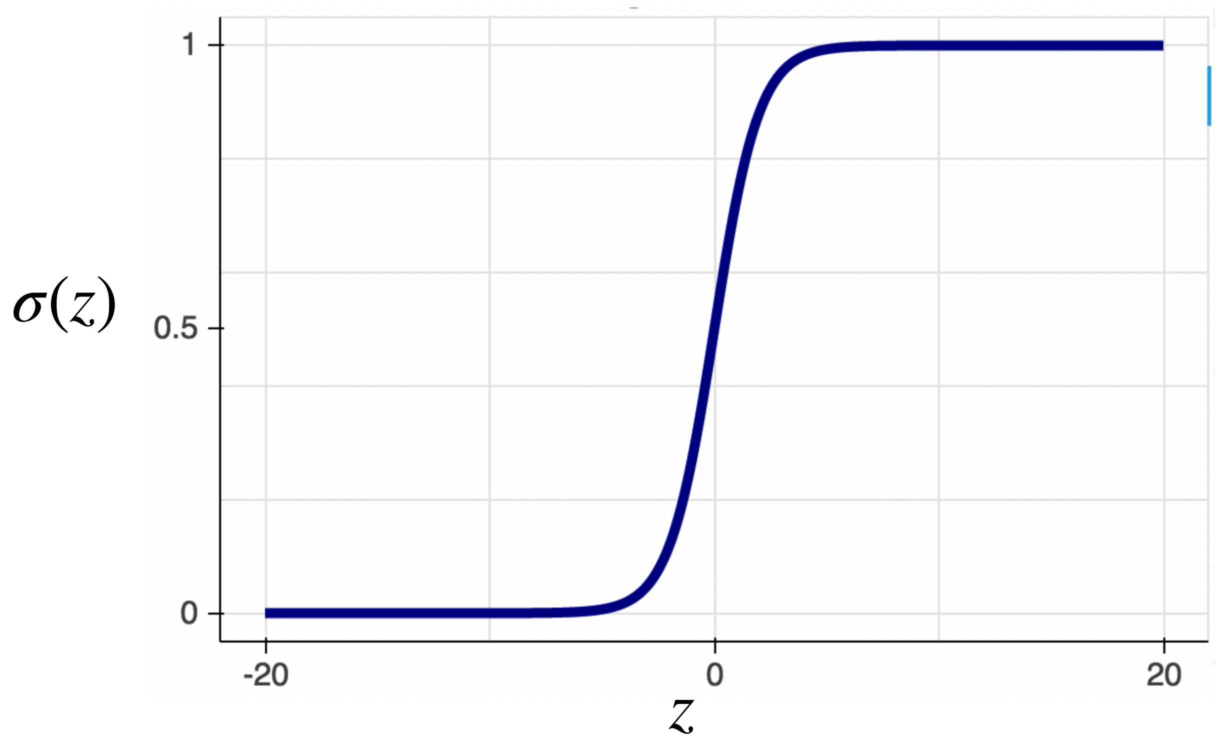
$\sigma(z) = \dfrac{1}{1+e^{-z}}$ is called the logistic function (sigmoid function).

. For simplicity:

$\mathbf{w} \leftarrow [w_0, \mathbf{w}]^T$

$\mathbf{x} \leftarrow [1,\mathbf{x}]^T$

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



Logistic regression

- **Logistic regression** outputs can be interpreted as posterior class probabilities:

$$y(\mathbf{x}) = \sigma(\mathbf{w}^{\mathsf{T}}\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} = P\left(t = 1 \,|\, \mathbf{x}, \mathbf{w}\right)$$

$$P\left(t = 0 \,|\, \mathbf{x}, \mathbf{w}\right) = 1 - P\left(t = 1 \,|\, \mathbf{x}, \mathbf{w}\right)$$

- How do we learn the parameters $\mathbf{w}$?
    - We need a cost function $J(\mathrm{w})$ over which we can optimise

Cost function

Likelihood for individual label

$$p\left(t_i = 1 \mid \mathbf{x}_i, \mathbf{w}\right) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}_i)$$

$$p\left(t_i = 0 \mid \mathbf{x}_i, \mathbf{w}\right) = 1 - \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}_i)$$

Combining

$$p(t_i \mid \mathbf{x}_i, \mathbf{w}) = [\sigma(\mathbf{w}^T\mathbf{x}_i)]^{t_i}[1 - \sigma(\mathbf{w}^T\mathbf{x}_i)]^{(1-t_i)} = y_i^{t}(1 - y_i)^{(1-t_i)}$$

Log-likelihood

$$\log p(t_i \mid \mathbf{x}, \mathbf{w}) = t_i \log[\sigma(\mathbf{w}^T\mathbf{x}_i)] + (1 - t_i)\log[1 - \sigma(\mathbf{w}^T\mathbf{x}_i)]$$

Cost function

- Minimise the negative of log-likelihood (or maximise log-likelihood)

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\, J(\mathbf{w})$$

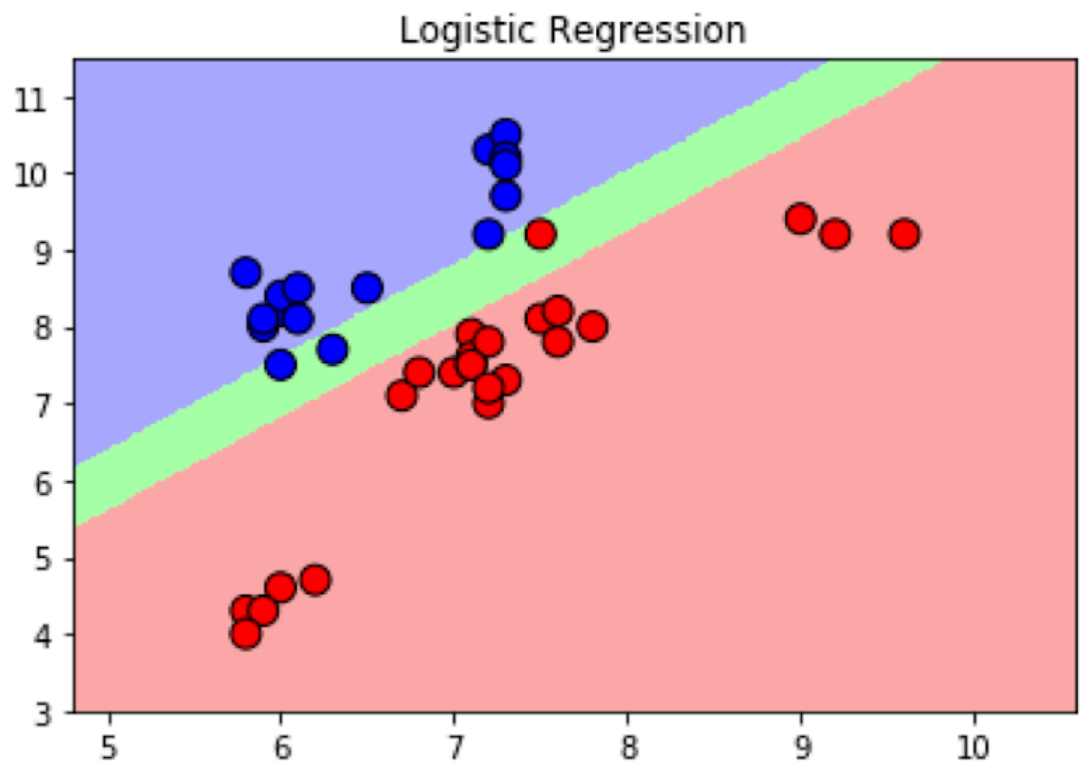
$$J(\mathbf{w}) = -\sum_i \log p(t_i \,|\, \mathbf{x}_i, \mathbf{w})$$

- Can be L2 regularised

$$J(\mathbf{w}) = -\sum_i \log p(t_i \,|\, \mathbf{x}_i, \mathbf{w}) + \lambda \,|\, \mathbf{w} \,|^2$$

```
In [59]: from sklearn.linear_model import LogisticRegression
         clf = LogisticRegression().fit(X, t)
         Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
         # Put the result into a color plot
         Z = Z.reshape(xx.shape)
         plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
         # Plot also the training points
         plt.scatter(X[:, 0], X[:, 1], c=t, cmap=cmap_bold,
                     edgecolor='k', s=100)
         plt.xlim(xx.min(), xx.max())
         plt.ylim(yy.min(), yy.max())
         plt.title("Logistic Regression")
         mean_cv_score = np.mean( cross_val_score(clf, X, t, cv=5) )
         print("5-fold averae CV error:", 1-mean_cv_score)
```

```
5-fold averae CV error: 0.025000000000000022
```



Example: orange and lemon data

Logistic regression

- To convert prob scores to discrete labels
  - Label = whichever class probability is higher
  - Label decided based on a threshold

$$y_{new} = 1 \text{ if } P\left(y_{new} = 1 \mid \mathbf{x}_{new}, \theta\right) \geq 0.6$$

- May also 'reject' data when the difference between the two classes is small
- Threshold may be computed using CV, but highly data-dependent

Logistic regression summary

Simple

- Linear model passed through a non-linear function
- Smooth cost function
- Learnable params = feature weights

...but

- Assumes linearity between dependent and independent variables.
- If the number of samples< feature dimension, we can have overfitting
- Too simple for complex data