Binary logistic regression

Herman Kamper

Model

Discriminative modelling in general: $P(y = k|\mathbf{x}; \mathbf{w})$

Binary classification: $y \in \{0, 1\}$

Want to predict probability of being in a particular class:
$P(y = 1|\mathbf{x}; \mathbf{w})$

We could just fit a linear model: $f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$

But this could give predictions outside $[0, 1]$ for some test inputs (invalid probabilities).

Let us use the sigmoid function to force the output to lie in the $[0, 1]$ range:

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

We interpret

$$f(\mathbf{x}; \mathbf{w}) = P(y = 1|\mathbf{x}; \mathbf{w})$$

implying

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

Loss

Data: $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ with $y \in \{0, 1\}$

E.g. for the Iris dataset we could have

$$\left(\begin{bmatrix} 3.5 \\ 1 \end{bmatrix}, 0 \right), \left(\begin{bmatrix} 6.5 \\ 2.25 \end{bmatrix}, 1 \right), \ldots, \left(\begin{bmatrix} 5.0 \\ 1.5 \end{bmatrix}, 0 \right)$$

To fit $\mathbf{w}$, we use maximum likelihood estimation:[1]

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

Or, equivalently, we minimise the negative log likelihood:

$$J(\mathbf{w}) = -\log L(\mathbf{w}) =$$

with

$$P(y|\mathbf{x}; \mathbf{w}) = \begin{cases} f(\mathbf{x}; \mathbf{w}) & \text{if } y = 1 \\ 1 - f(\mathbf{x}; \mathbf{w}) & \text{if } y = 0 \end{cases}$$

$$=$$

$$= \Big(\sigma(\mathbf{w}^\top \mathbf{x})\Big)^{y} \Big(1 - \sigma(\mathbf{w}^\top \mathbf{x})\Big)^{1-y}$$

[1] *Non-examinable:* Because we are doing discriminative modelling, the likelihood is based on the joint of the outputs $\{y^{(1)}, y^{(2)}, \ldots, y^{(N)}\}$ conditioned on being given the inputs $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$. You would arrive at the same result if you used the joint over the input-output pairs $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$ and then assumed a uniform prior $p(\mathbf{x})$ over the inputs.

This means we can write the loss as:

Optimisation

We use maximum likelihood estimation, or equivalently we want to minimise the negative log likelihood:

$$
\begin{aligned}
J(\mathbf{w}) &= -\log \prod_{n=1}^{N} P(y^{(n)}|\mathbf{x}^{(n)};\mathbf{w}) \\
&= -\sum_{n=1}^{N} \left[y^{(n)} \log \sigma(\mathbf{w}^\top \mathbf{x}^{(n)}) + (1 - y^{(n)}) \log \left(1 - \sigma(\mathbf{w}^\top \mathbf{x}^{(n)})\right)\right]
\end{aligned}
$$

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:

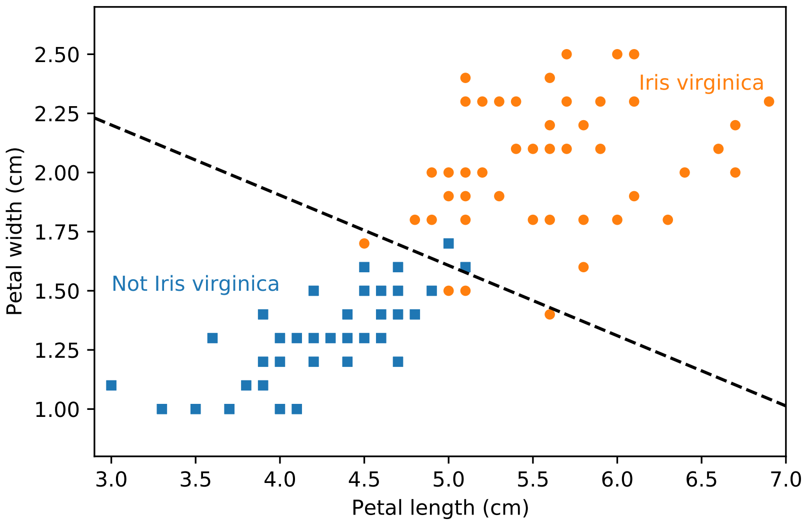
$$
\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{n=1}^{N} \left(y^{(n)} - f(\mathbf{x}^{(n)};\mathbf{w})\right) \mathbf{x}^{(n)}
$$

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 a closed-form solution (as in linear regression).

So instead we use gradient descent:

$$
\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}
$$

Binary logistic regression on Iris dataset

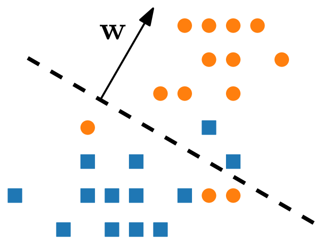


Binary logistic regression summary

- Prediction function: $f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \dfrac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$
- Interpret function as: $f(\mathbf{x}; \mathbf{w}) = P(y = 1 | \mathbf{x}; \mathbf{w})$
- With labels $y \in \{0, 1\}$, minimise the negative log likelihood:

$$
\begin{aligned}
J(\mathbf{w}) &= -\log \prod_{n=1}^{N} P(y^{(n)} | \mathbf{x}^{(n)}; \mathbf{w}) \\
&= -\sum_{n=1}^{N} \left[y^{(n)} \log f(\mathbf{x}^{(n)}; \mathbf{w}) + (1 - y^{(n)}) \log \left(1 - f(\mathbf{x}^{(n)}; \mathbf{w})\right) \right]
\end{aligned}
$$

- Gradient: $\dfrac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{n=1}^{N} \left(y^{(n)} - f(\mathbf{x}^{(n)}; \mathbf{w}) \right) \mathbf{x}^{(n)}$



Decision boundary

The decision boundary is the values of $\mathbf{x}$ for which $f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}) = 0.5$, i.e. $\mathbf{w}^\top \mathbf{x} = 0$.

Here it might be easier to explicitly include the bias term, i.e. $f(\mathbf{x}; \mathbf{w}) = \sigma(w_0 + \mathbf{w}^\top \mathbf{x}) = 0.5$.

Let's first consider the 2-D case.
Do the following:

1. Sketch the line $w_0 + w_1 x_1 + w_2 x_2 = 0$ in the x_1-x_2 plane.
2. Sketch the vector $\mathbf{w} = \begin{bmatrix} w_1 & w_2 \end{bmatrix}^\top$ 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.

Decision boundary

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

$$\begin{aligned} w_1 x_1 + w_2 x_2 + \ldots + w_D x_D &= 0 \\ \mathbf{w}^\top \mathbf{x} &= 0 \end{aligned}$$

If we think of $\mathbf{w}$ as a vector in $\mathbf{x}$-space, then the $\mathbf{x}$ vectors on the decision 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:

$$w_0 + \mathbf{w}^\top \mathbf{x} = 0$$

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

Interpreting gradient descent

The weights $\mathbf{w}$ is a vector orthogonal to the decision boundary.

Let's pretend we have a single training example with a positive label $y^{(n)} = 1$.

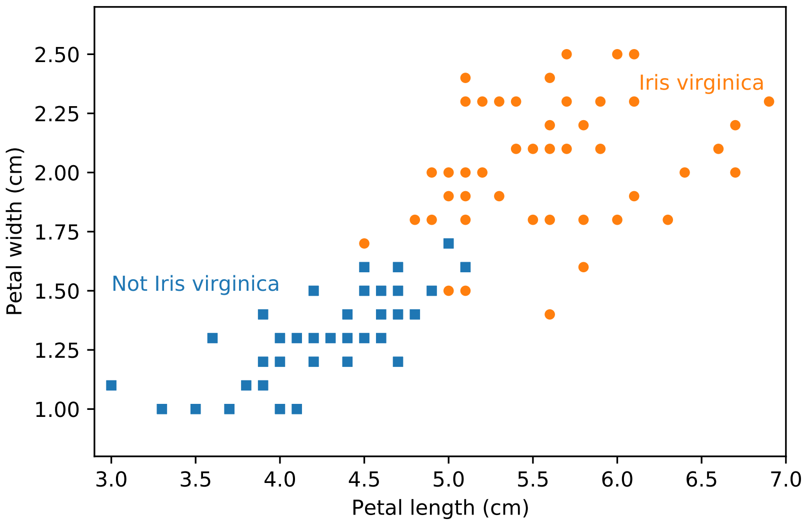
How does this single example affect the decision boundary in the gradient descent update step?

We also pretend we don't have a bias term w_0.

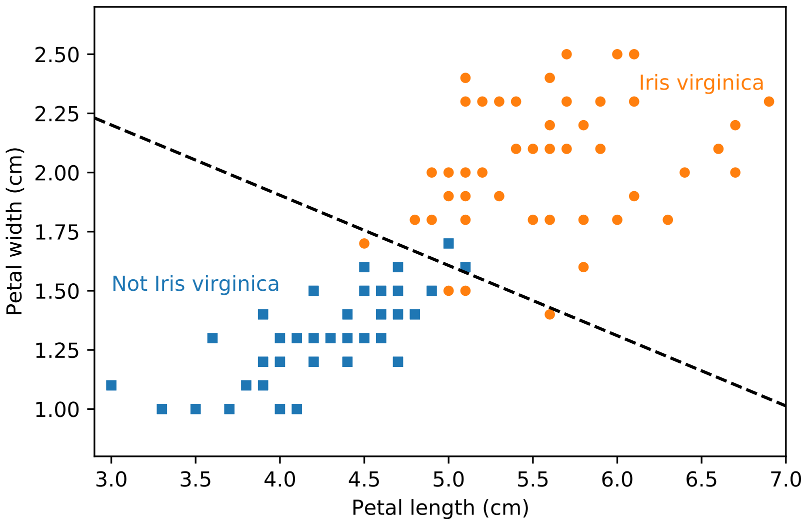
$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{n=1}^{N} \left(y^{(n)} - f(\mathbf{x}^{(n)}; \mathbf{w}) \right) \mathbf{x}^{(n)}$$

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \eta \left. \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \right|_{\mathbf{w}^{(\text{old})}}$$

Binary logistic regression on Iris dataset

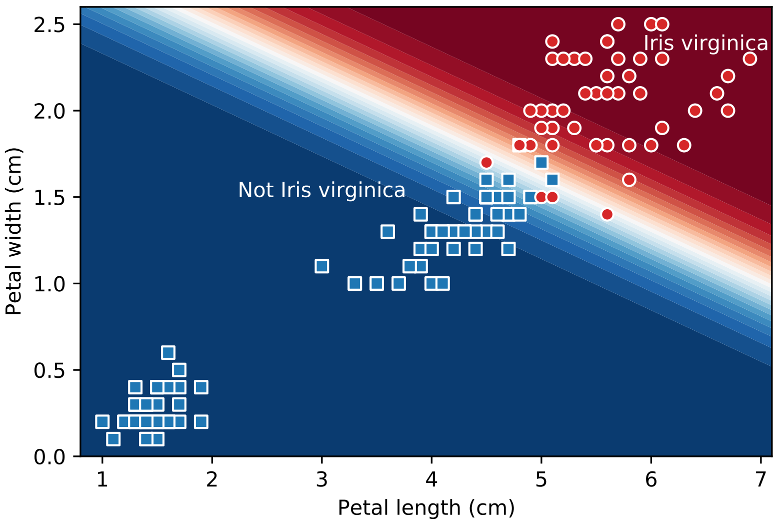


Binary logistic regression on Iris dataset

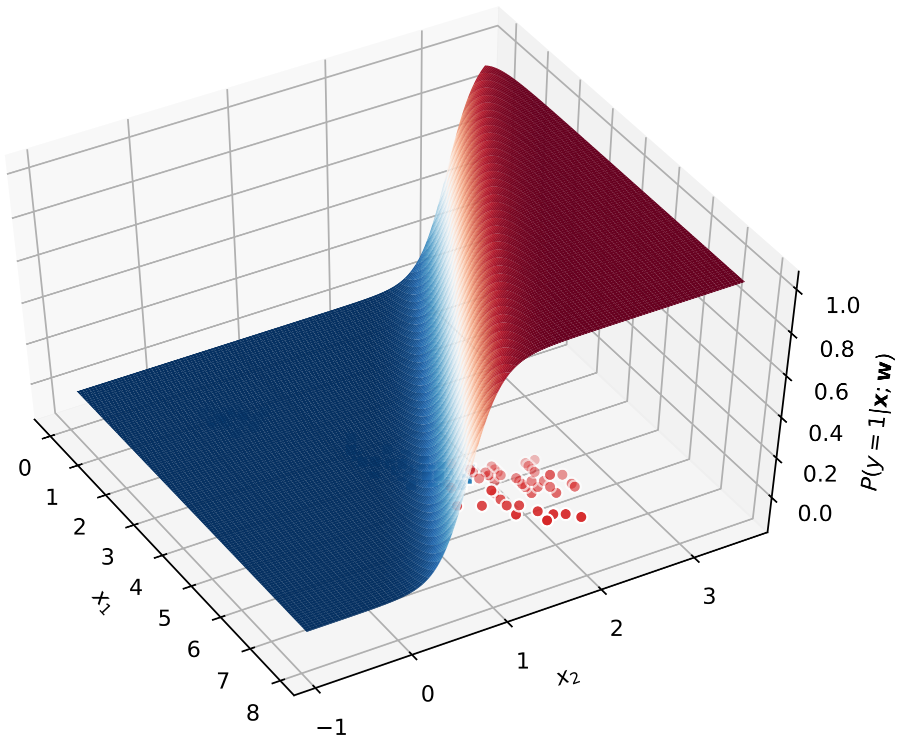


(See demo.)

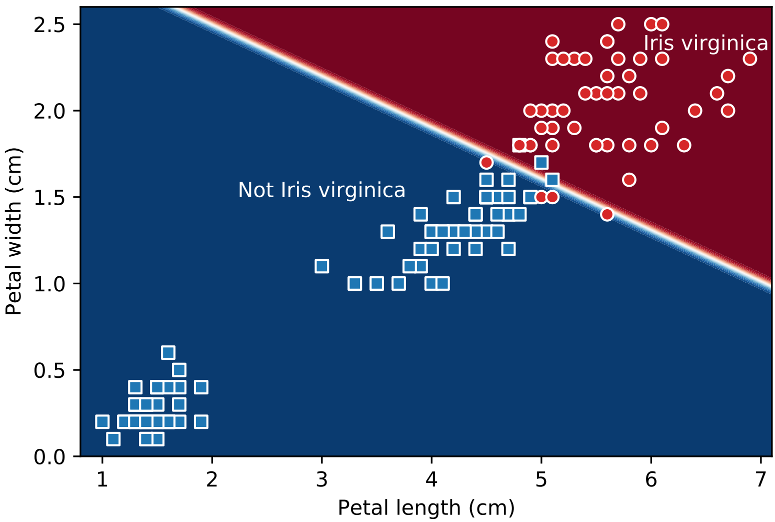
Visualising probabilities



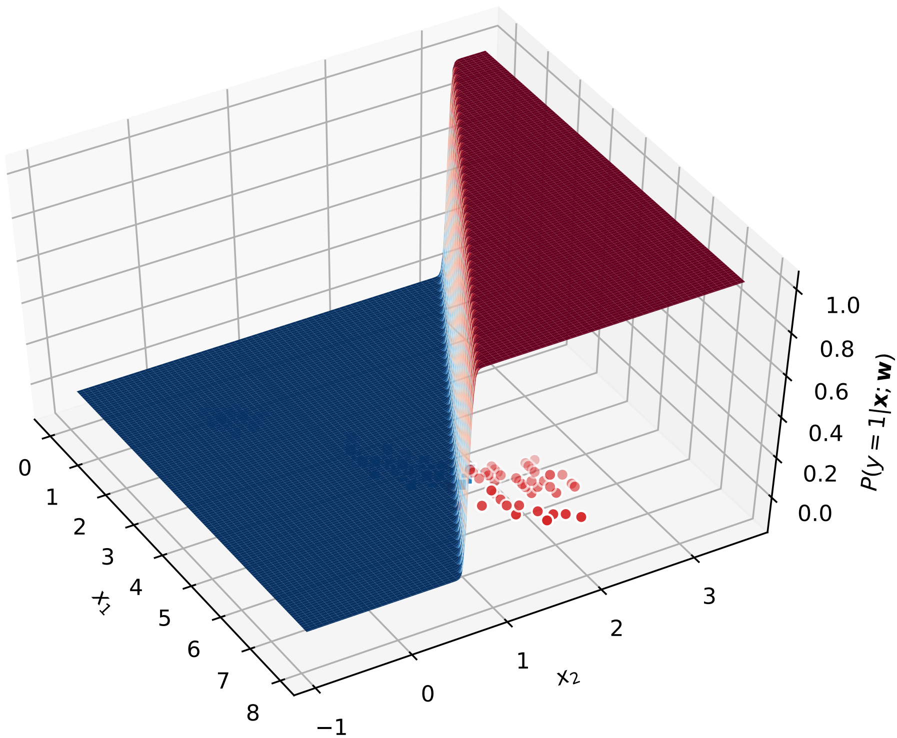
Probability surface



Probability surface with large $||\mathbf{w}||$



Probability surface with large $||\mathrm{w}||$



Weight vector summary

The bias term w_0 offsets the decision boundary.

The direction of $\mathbf{w}$ influences the direction of the decision boundary: $\mathbf{w}$ is orthogonal to the decision boundary.

The length of $\mathbf{w}$, i.e. $||\mathbf{w}||$, influences the "steepness" of the decision boundary.

For very large $||\mathbf{w}||$, even points that are very close to the decision boundary is assigned very high or very low probabilities $P(y = 1|\mathbf{x}; \mathbf{w})$.

With a small $||\mathbf{w}||$, the probability assignment is more gradual.

Logistic regression with basis functions and regularisation

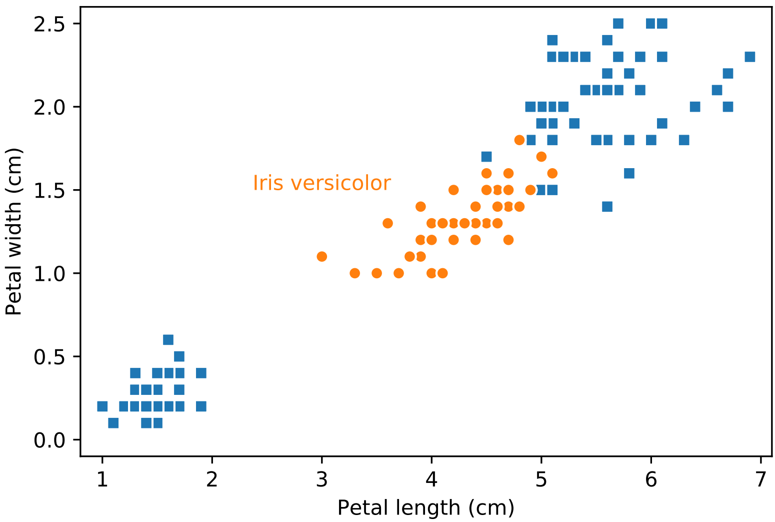
Basis functions

Anywhere we wrote an $\mathbf{x}$ in the previous videos, the feature vector $\mathbf{x}$ can be replaced with basis functions $\boldsymbol{\phi}(\mathbf{x})$.

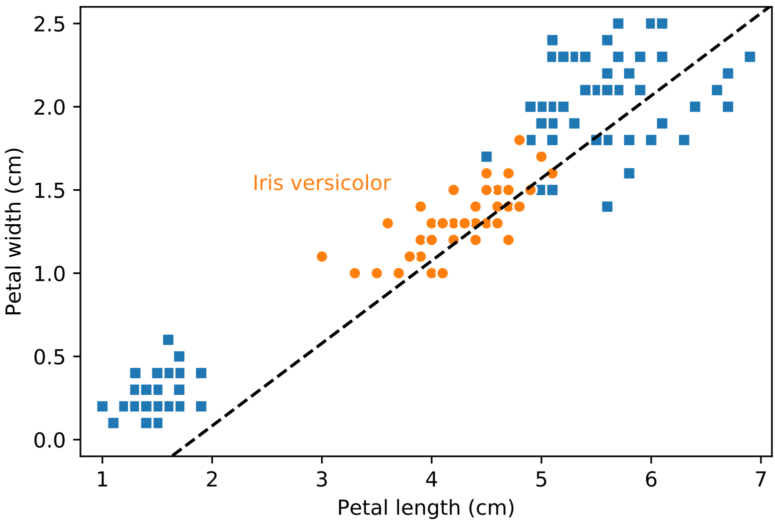
Regularisation

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

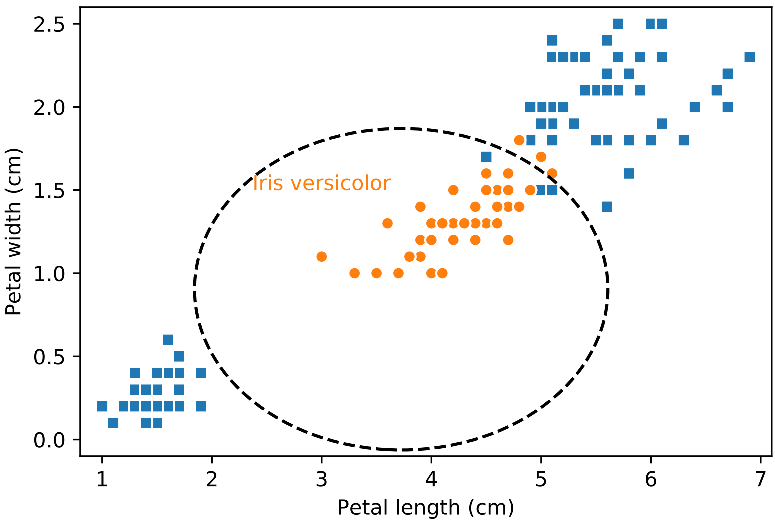
Logistic regression for non-separable classes



Logistic regression for non-separable classes

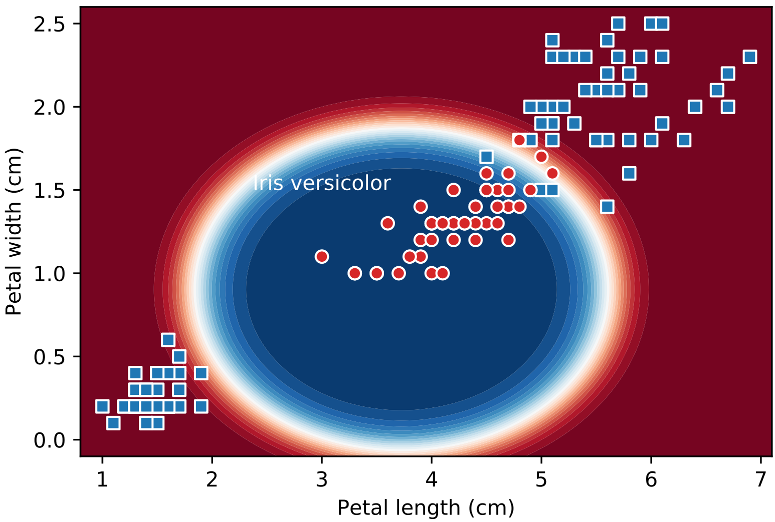


Logistic regression with basis functions



$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_2^2 \end{bmatrix}^\top$$

Logistic regression with basis functions



$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_2^2 \end{bmatrix}^\top$$

Videos covered in this note

- Logistic regression 1: Model and loss (14 min)
- Logistic regression 2: Optimisation (7 min)
- Logistic regression 3: The decision boundary and weight vector (21 min)
- Logistic regression 4: Basis functions and regularisation (6 min)

Reading

- ISLR 4.3 intro
- ISLR 4.3.1
- ISLR 4.3.2
- ISLR 4.3.3
- ISLR 4.3.4