

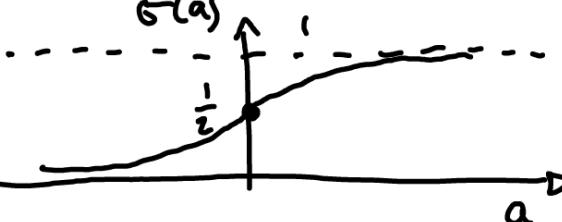
Binary logistic regression

Model and loss

Herman Kamper

<http://www.kamperh.com/>

Model

- In general: $P(y=k | \underline{x}; \underline{w})$
- 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: $f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$ *[Pretending $x_0 = 1$]*
- 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:

$$f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$$
- Interpret $f(\mathbf{x}; \mathbf{w}) = P(y = 1 | \mathbf{x}; \mathbf{w})$, implying $P(y = 0 | \mathbf{x}; \mathbf{w}) = 1 - f(\mathbf{x}; \mathbf{w})$

Loss function

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

E.g. for Iris data might have:

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

petal length
 ↓
 $x_1^{(1)}$
 ↓
 $x_2^{(1)}$
 ↓
 $y^{(1)}$
 petal width

not virginica
 ↓
 $x_1^{(2)}$
 ↓
 $x_2^{(2)}$
 ↓
 $y^{(2)}$
 virginica

Use maximum likelihood:

$$L(\underline{w}) = P(y^{(1)} | \underline{x}^{(1)}; \underline{w}) \cdot P(y^{(2)} | \underline{x}^{(2)}; \underline{w}) \cdot \dots \cdot P(y^{(N)} | \underline{x}^{(N)}; \underline{w}) = \prod_{n=1}^N P(y^{(n)} | \underline{x}^{(n)}; \underline{w})$$

Or minimise the negative log likelihood:

$$J(\underline{w}) = -\log L(\underline{w}) = -\log \prod_{n=1}^N P(y^{(n)} | \underline{x}^{(n)}; \underline{w}) = -\sum_{n=1}^N \log P(y^{(n)} | \underline{x}^{(n)}; \underline{w})$$

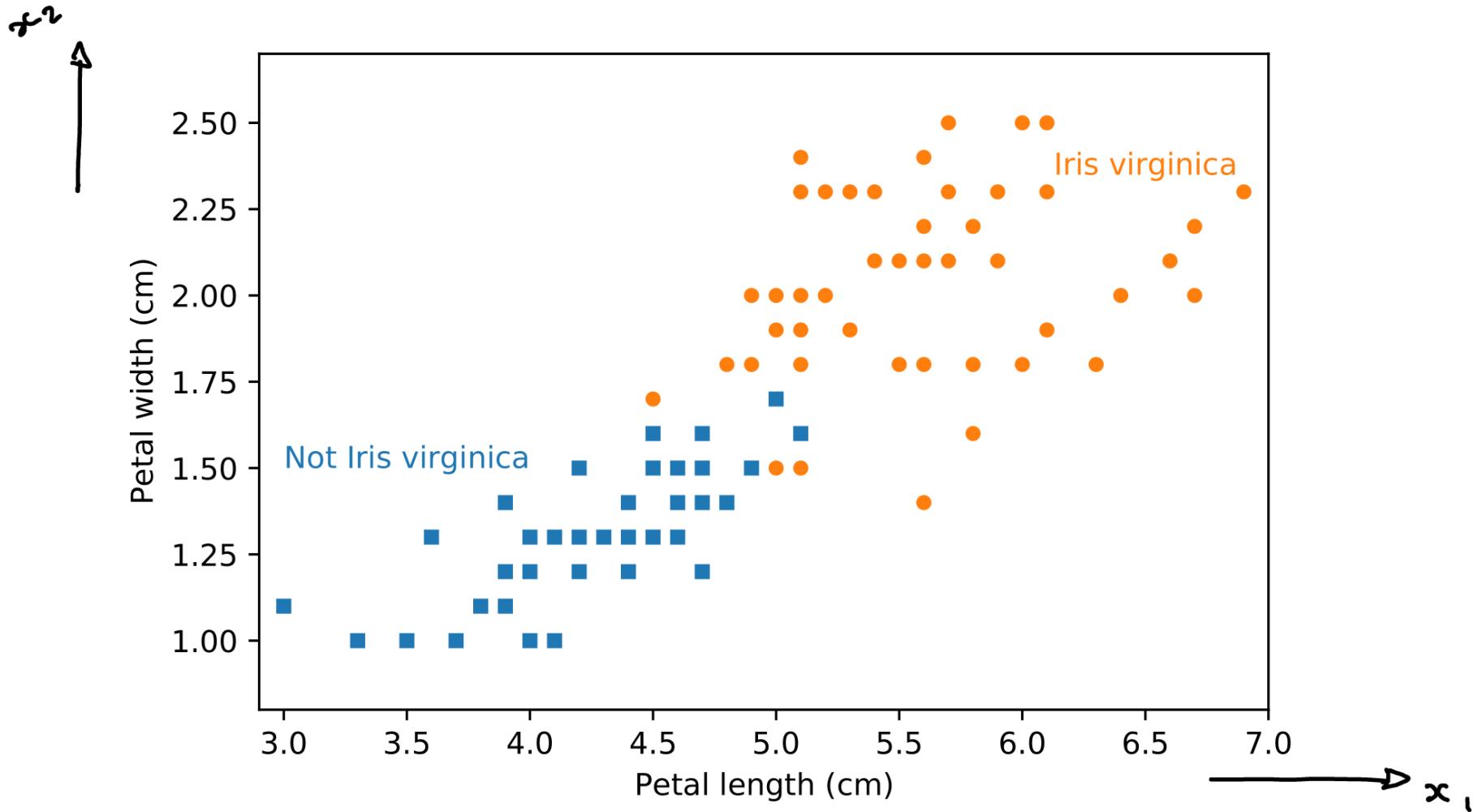
$$P(y | \underline{x}; \underline{w}) = \begin{cases} f(\underline{x}; \underline{w}) & \text{if } y=1 \\ 1 - f(\underline{x}; \underline{w}) & \text{if } y=0 \end{cases} = \begin{cases} \sigma(\underline{w}^\top \underline{x}) & \text{if } y=1 \\ 1 - \sigma(\underline{w}^\top \underline{x}) & \text{if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

Loss function

$$J(\underline{w}) = - \sum_{n=1}^N \log P(y^{(n)} | \underline{x}^{(n)}; \underline{w}) \quad \text{and} \quad P(y | \underline{x}; \underline{w}) = \sigma(\underline{w}^\top \underline{x})^y (1 - \sigma(\underline{w}^\top \underline{x}))^{1-y}$$

$$\begin{aligned} J(\underline{w}) &= - \sum_{n=1}^N \log [\sigma(\underline{w}^\top \underline{x}^{(n)})^{y^{(n)}} (1 - \sigma(\underline{w}^\top \underline{x}^{(n)}))^{(1-y^{(n)})}] \\ &= - \sum_{n=1}^N [y^{(n)} \log \sigma(\underline{w}^\top \underline{x}^{(n)}) + (1-y^{(n)}) \cdot \log (1 - \sigma(\underline{w}^\top \underline{x}^{(n)}))] \end{aligned}$$

Iris dataset



Binary logistic regression

Optimisation

Herman Kamper

<http://www.kamperh.com/>

Gradients and optimisation

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

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

... 1

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:

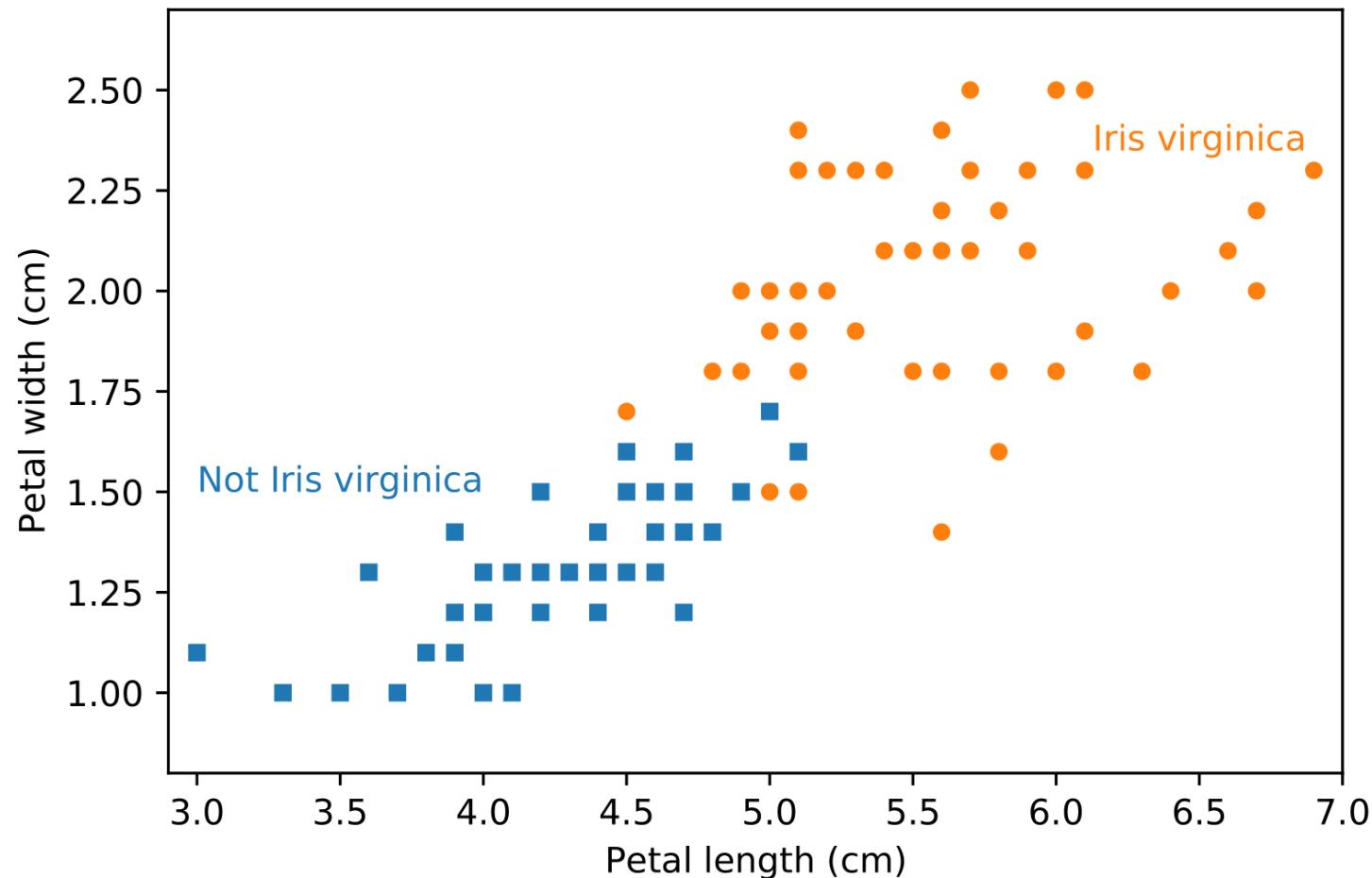
$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{n=1}^N (y^{(n)} - f(\mathbf{x}^{(n)}; \mathbf{w})) \mathbf{x}^{(n)}$$

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 a closed-form solution (as in linear regression). So instead we use gradient descent:

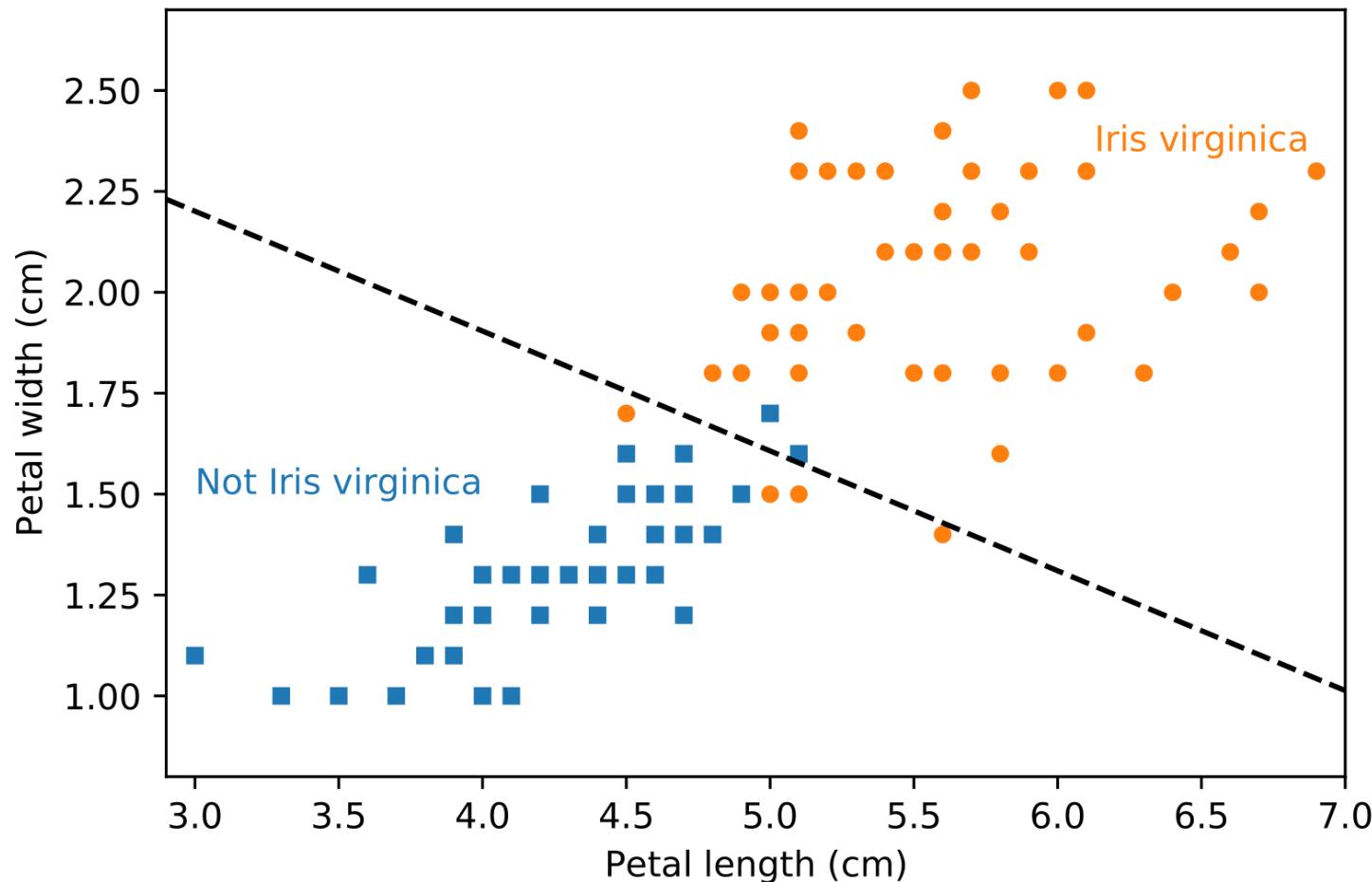
$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

[Watch the video
on gradient descent]

Binary logistic regression on the Iris dataset



Binary logistic regression on the Iris dataset

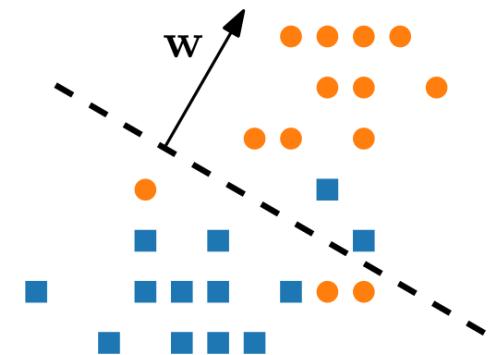


Binary logistic regression summary

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

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

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



Deriving the gradients

We need $\frac{\partial J}{\partial w}$. Instead of using ① directly, we make a small substitution to simplify getting the derivative:

$$z^{(n)} \in \{-1, 1\}$$

$$z^{(n)} = 2y^{(n)} - 1$$

Using $\sigma(-a) = 1 - \sigma(a)$, we can write ①:

$$J(w) = - \sum_{n=1}^N \log \sigma(z^{(n)} w^T \underline{x}^{(n)})$$

Useful identities: $\frac{d\sigma(a)}{da} = \sigma(a)(1 - \sigma(a))$

$$\frac{\partial g(u)}{\partial u} = \frac{\partial u}{\partial x} \frac{\partial g(u)}{\partial u}$$

$$\begin{aligned} \frac{\partial J(w)}{\partial w} &= - \sum_{n=1}^N \frac{\partial}{\partial w} \log \sigma(z^{(n)} w^T \underline{x}^{(n)}) \\ &= - \sum_{n=1}^N \left[\frac{\partial}{\partial w} \sigma(z^{(n)} \cdot w^T \underline{x}^{(n)}) \right] \cdot \frac{1}{\sigma(\dots)} \\ &= - \sum_{n=1}^N \left[\frac{\partial}{\partial w} (z^{(n)} w^T \underline{x}^{(n)}) \right] \cancel{\sigma(\dots)} (1 - \sigma(\dots)) \cancel{\frac{1}{\sigma(\dots)}} \\ &= - \sum_{n=1}^N z^{(n)} \underline{x}^{(n)} (1 - \sigma(z^{(n)} w^T \underline{x}^{(n)})) \end{aligned}$$

... ②

$$\begin{aligned} & -(1 - \sigma(-w^T \underline{x}^{(n)})) \\ &= -(1 - (1 - \sigma(w^T \underline{x}^{(n)}))) \\ &= -\cancel{(1 - 1)} + \sigma(w^T \underline{x}^{(n)}) \\ &= \sigma(w^T \underline{x}^{(n)}) \end{aligned}$$

Let's get an expression in terms of $y^{(n)}$:

$$\begin{aligned} z^{(n)}(1 - \sigma(\dots)) &= \begin{cases} -(1 - \sigma(-1 \cdot w^T \underline{x}^{(n)})) & \text{if } y^{(n)} = 0 \\ 1 - \sigma(w^T \underline{x}^{(n)}) & \text{if } y^{(n)} = 1 \end{cases} \\ &= \begin{cases} -\sigma(w^T \underline{x}^{(n)}) & \text{if } y^{(n)} = 0 \\ 1 - \sigma(w^T \underline{x}^{(n)}) & \text{if } y^{(n)} = 1 \end{cases} \\ &= y^{(n)} - \sigma(w^T \underline{x}^{(n)}) \end{aligned}$$

Now ② can be written as:

$$\frac{\partial J}{\partial w} = - \sum_{n=1}^N (y^{(n)} - \sigma(w^T \underline{x}^{(n)})) \underline{x}^{(n)}$$

Binary logistic regression

The decision boundary and weight vector

Herman Kamper

<http://www.kamperh.com/>

Decision boundary

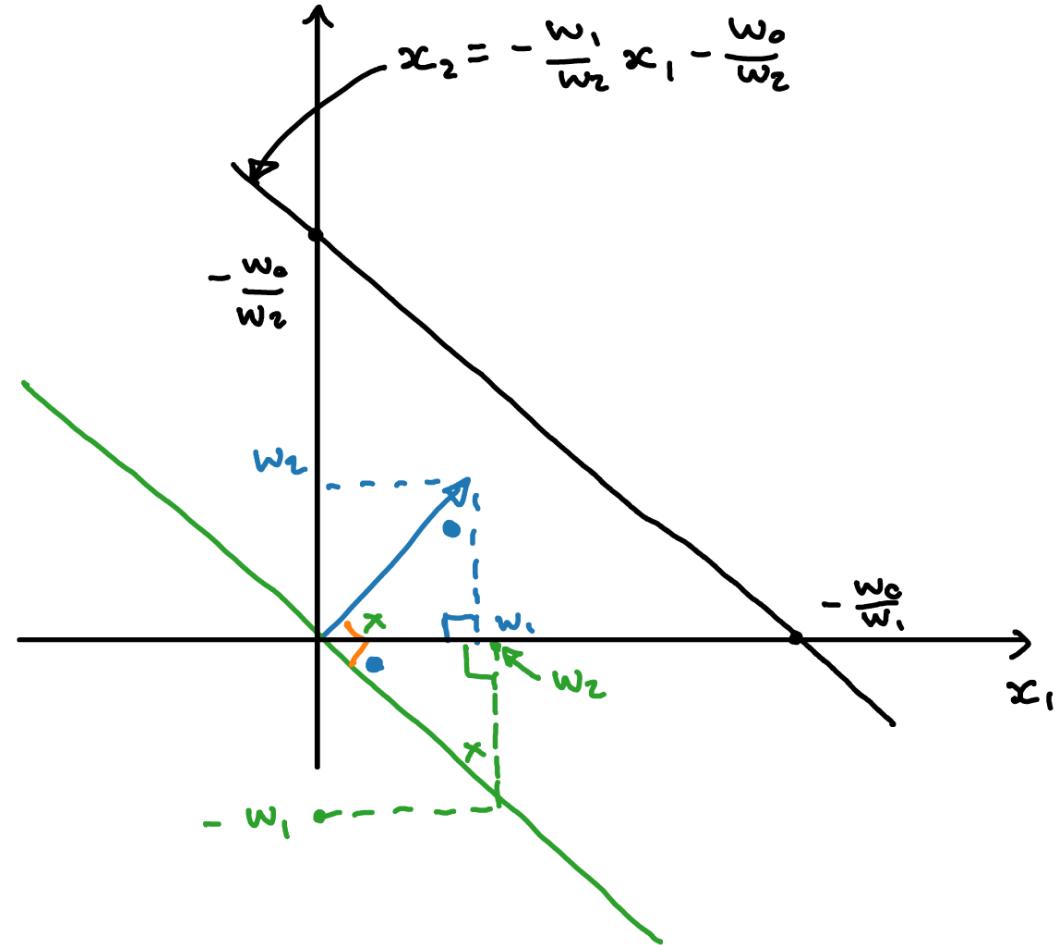
The decision boundary is the values of x for which $f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}) = 0.5$, i.e. $\underline{\mathbf{w}^\top \mathbf{x}} = 0$.

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 $w_0 + w_1x_1 + w_2x_2 = 0$ in the x_1 - x_2 plane.
2. Sketch the vector $\mathbf{w} = [w_1, w_2]^\top$ 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:

$$w_1x_1 + w_2x_2 + \dots + w_Dx_D = 0$$
$$\mathbf{w}^\top \mathbf{x} = 0$$

If we think of \mathbf{w} as a vector in \mathbf{x} -space, then the \mathbf{x} vectors on the decision boundary are orthogonal to \mathbf{w} , since their dot product is zero: $\mathbf{w} \cdot \mathbf{x} = 0$.

We can add the bias back in:

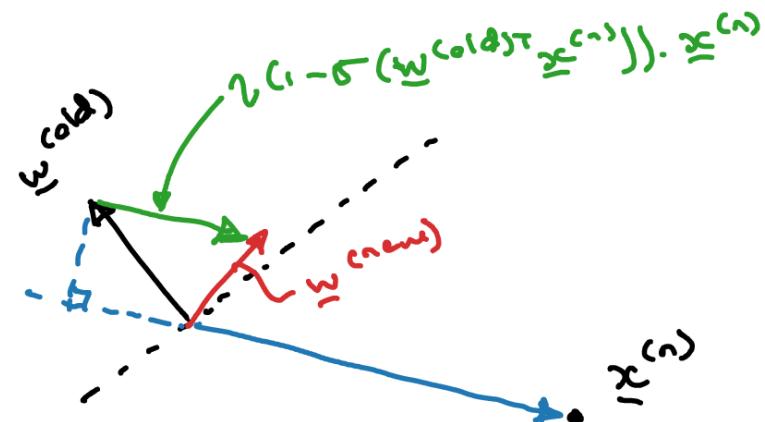
$$w_0 + \mathbf{w}^\top \mathbf{x} = 0$$

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

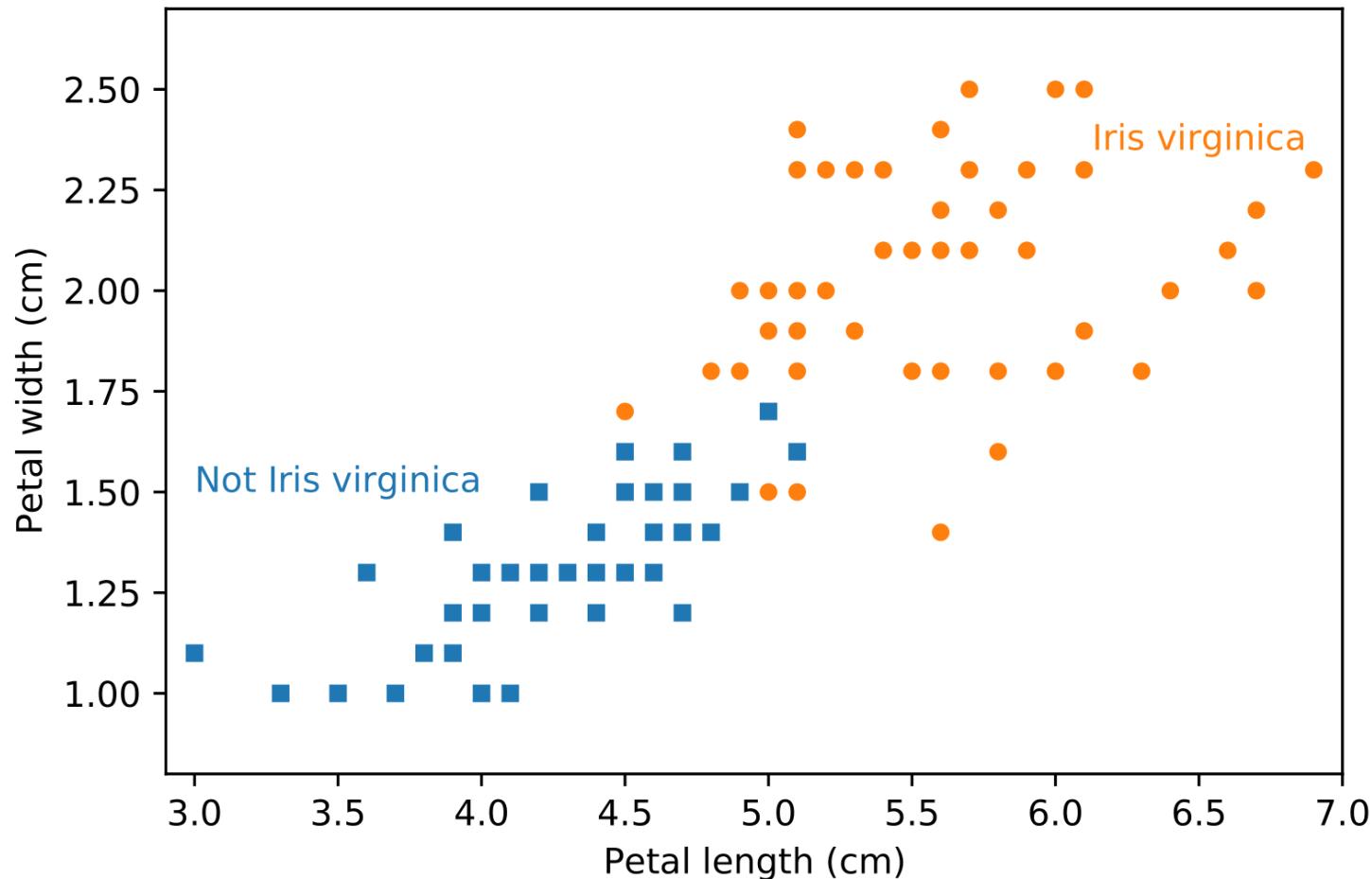
Interpreting gradient descent

- 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

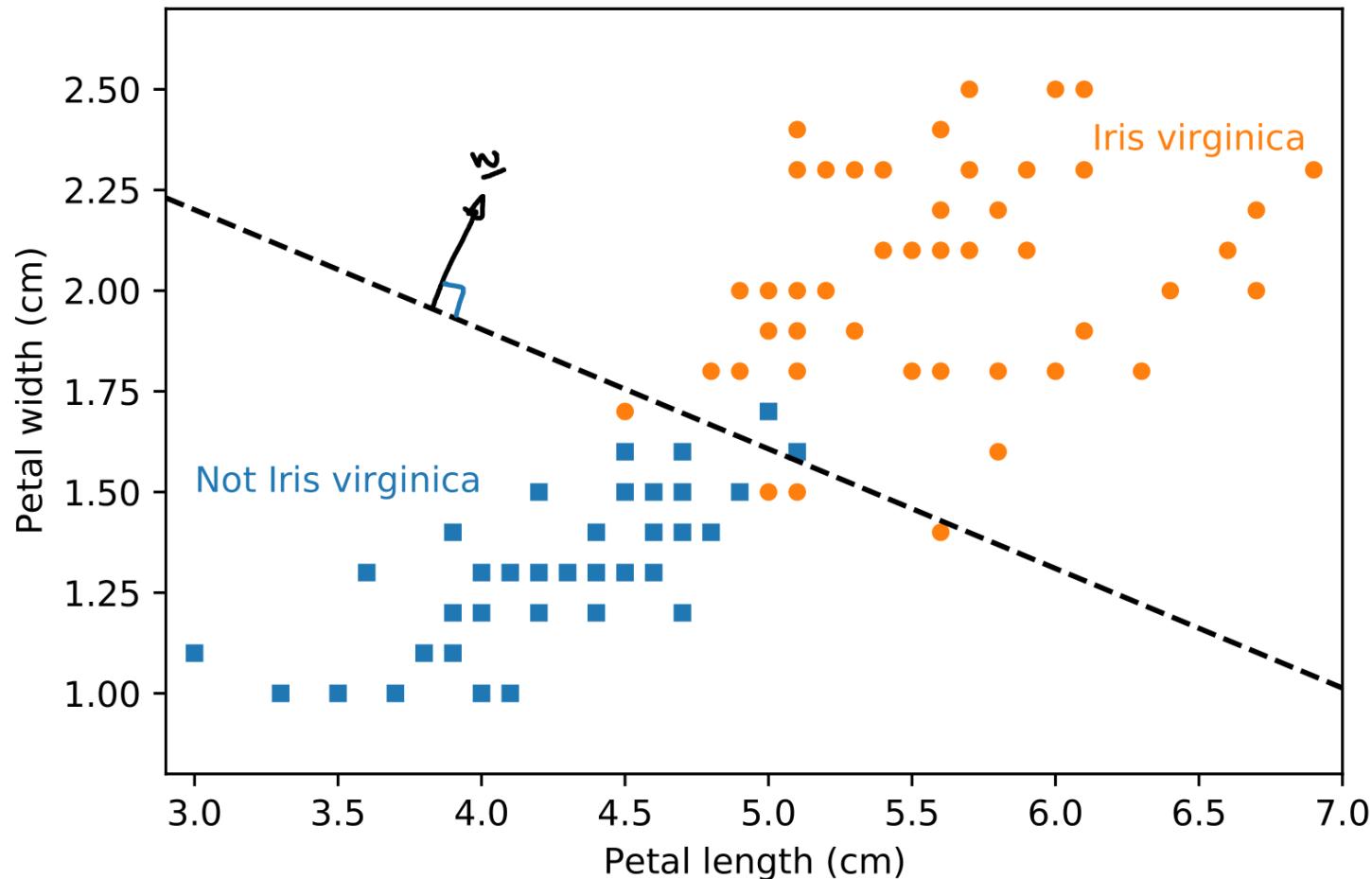
$$\frac{\partial J}{\partial w} = - \sum_{n=1}^N (y^{(n)} - \sigma(w^\top x^{(n)})) \cdot x^{(n)}$$
$$w^{(new)} \leftarrow w^{(old)} - \gamma \frac{\partial J}{\partial w}$$
$$w^{(new)} \leftarrow w^{(old)} + \gamma (1 - \sigma(w^{(old)\top} x^{(n)})) x^{(n)}$$



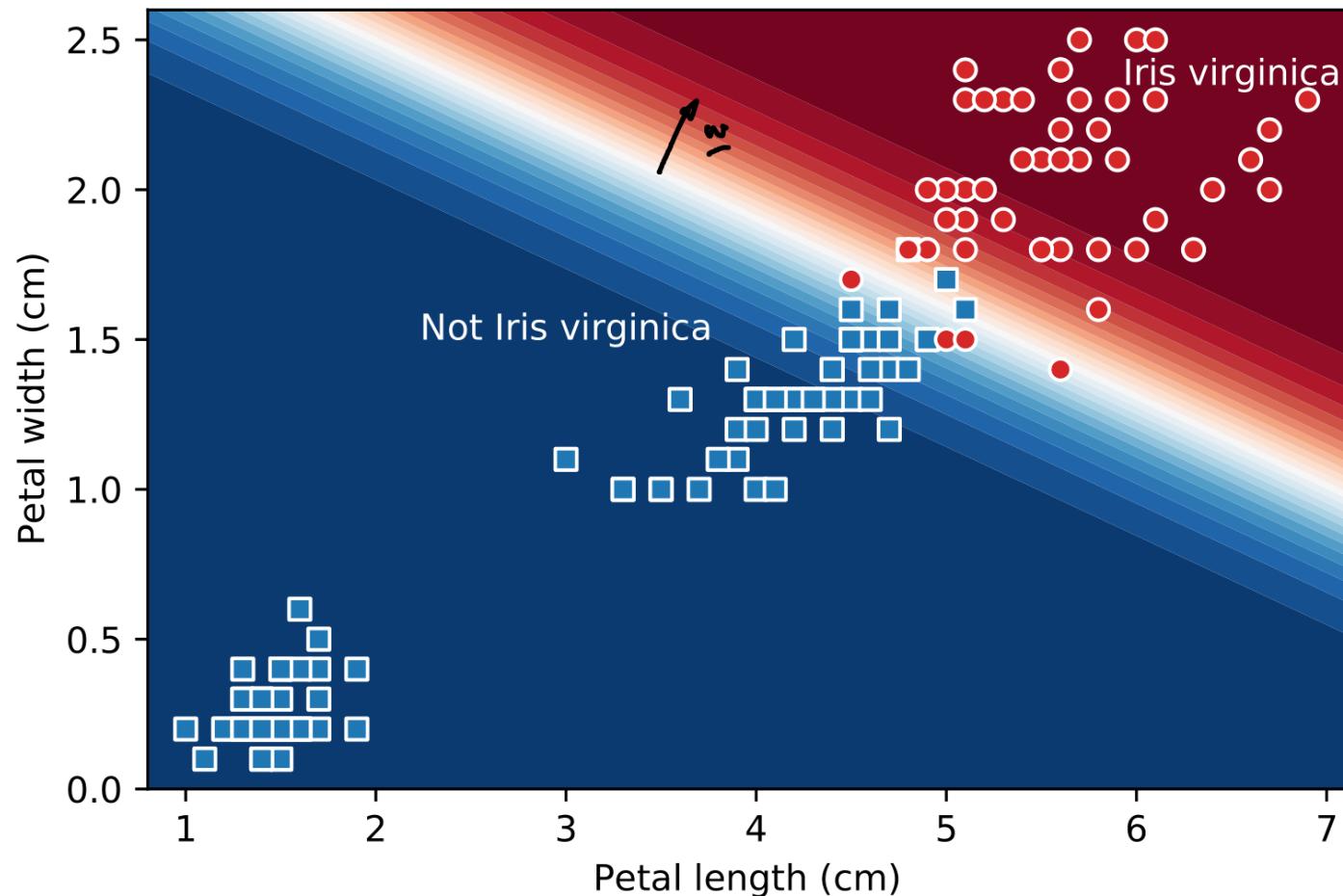
Iris dataset



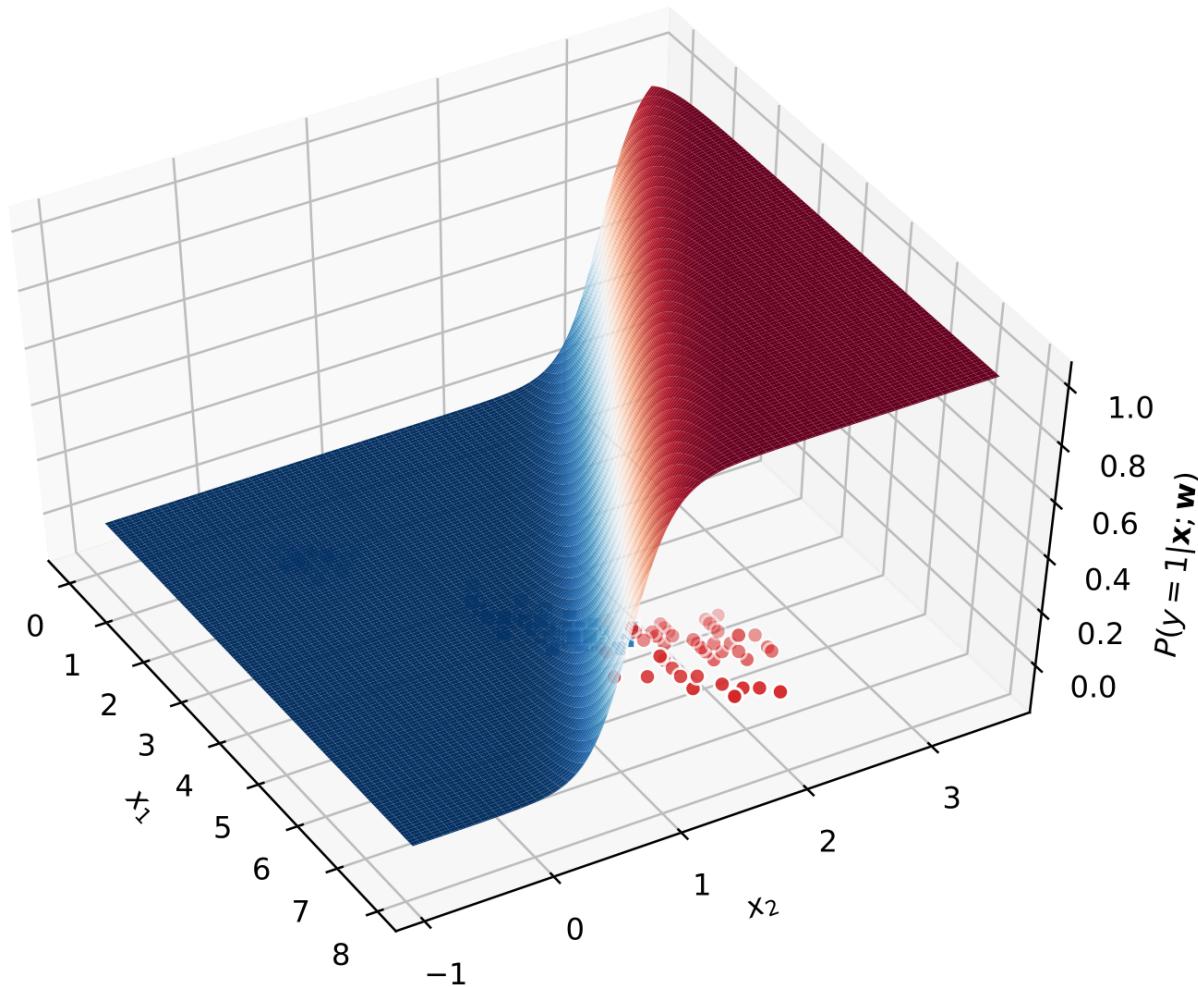
Iris dataset



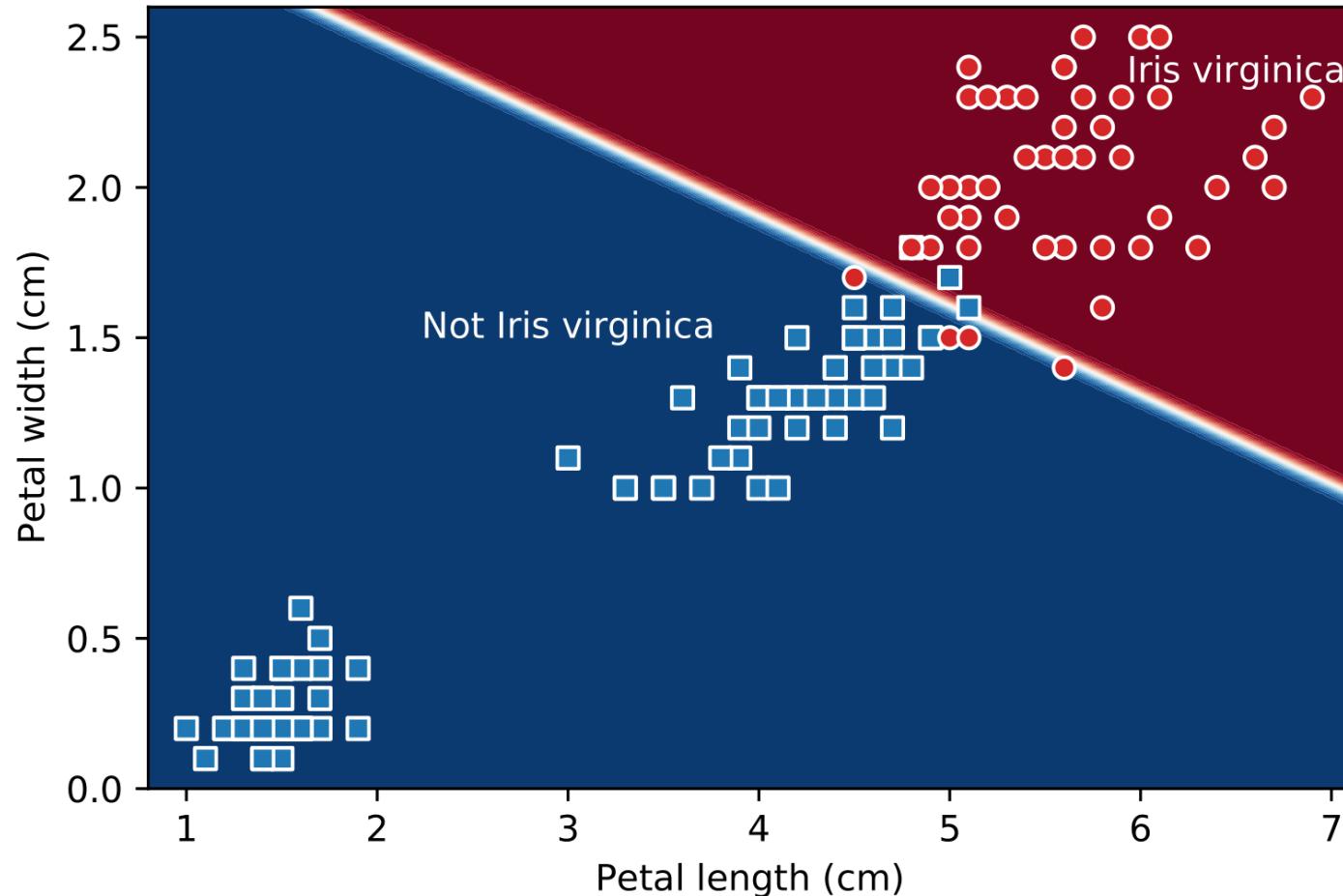
Visualising probabilities



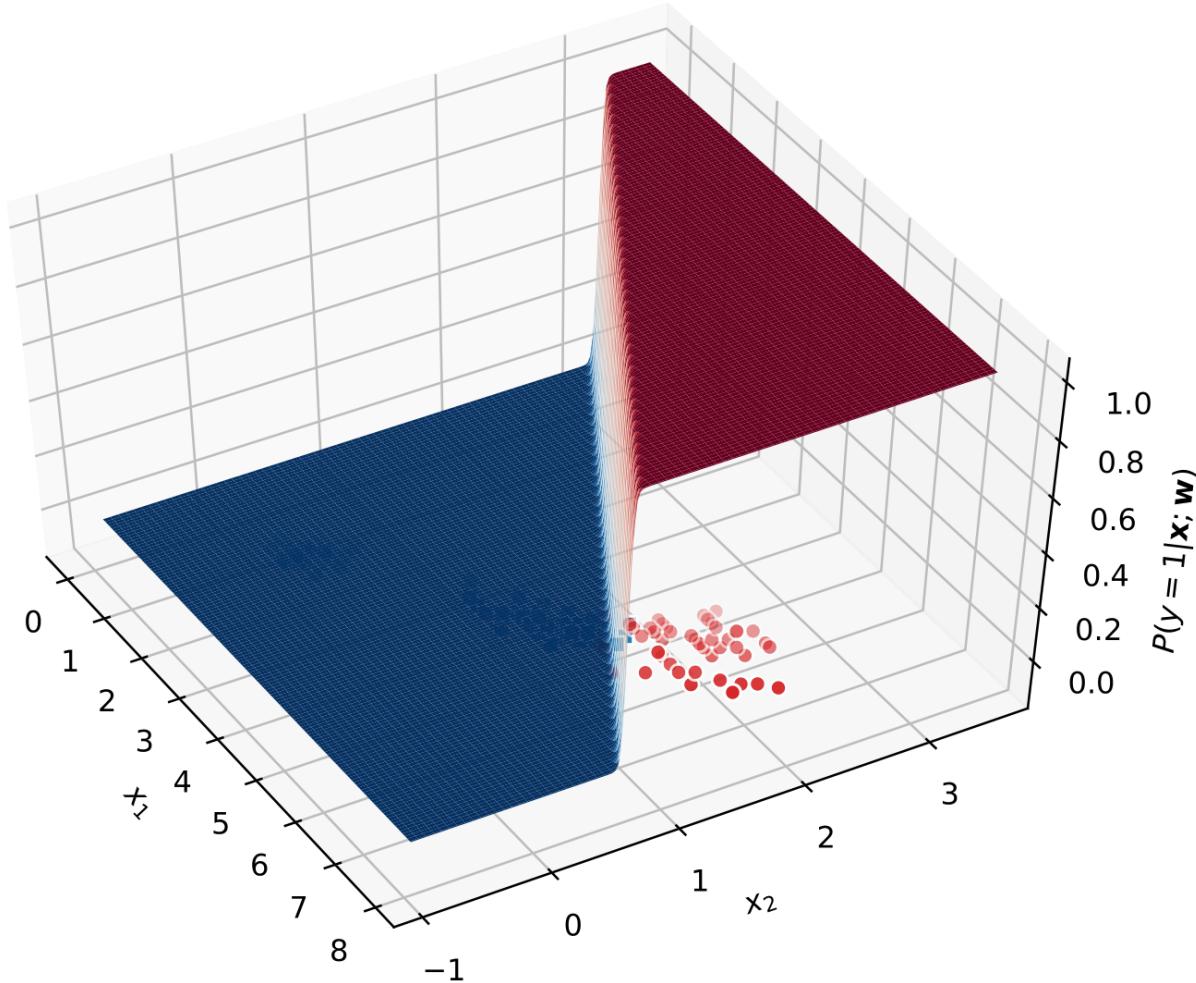
Probability surface



Probability surface with large $\|w\|$



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



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 will be assigned very high or low probabilities $P(y = 1|\mathbf{x}; \mathbf{w})$
- With a small $\|\mathbf{w}\|$, probability assignment will be more gradual

Binary logistic regression

Basis functions and regularisation

Herman Kamper

<http://www.kamperh.com/>

Basis functions and regularisation

Basis functions:

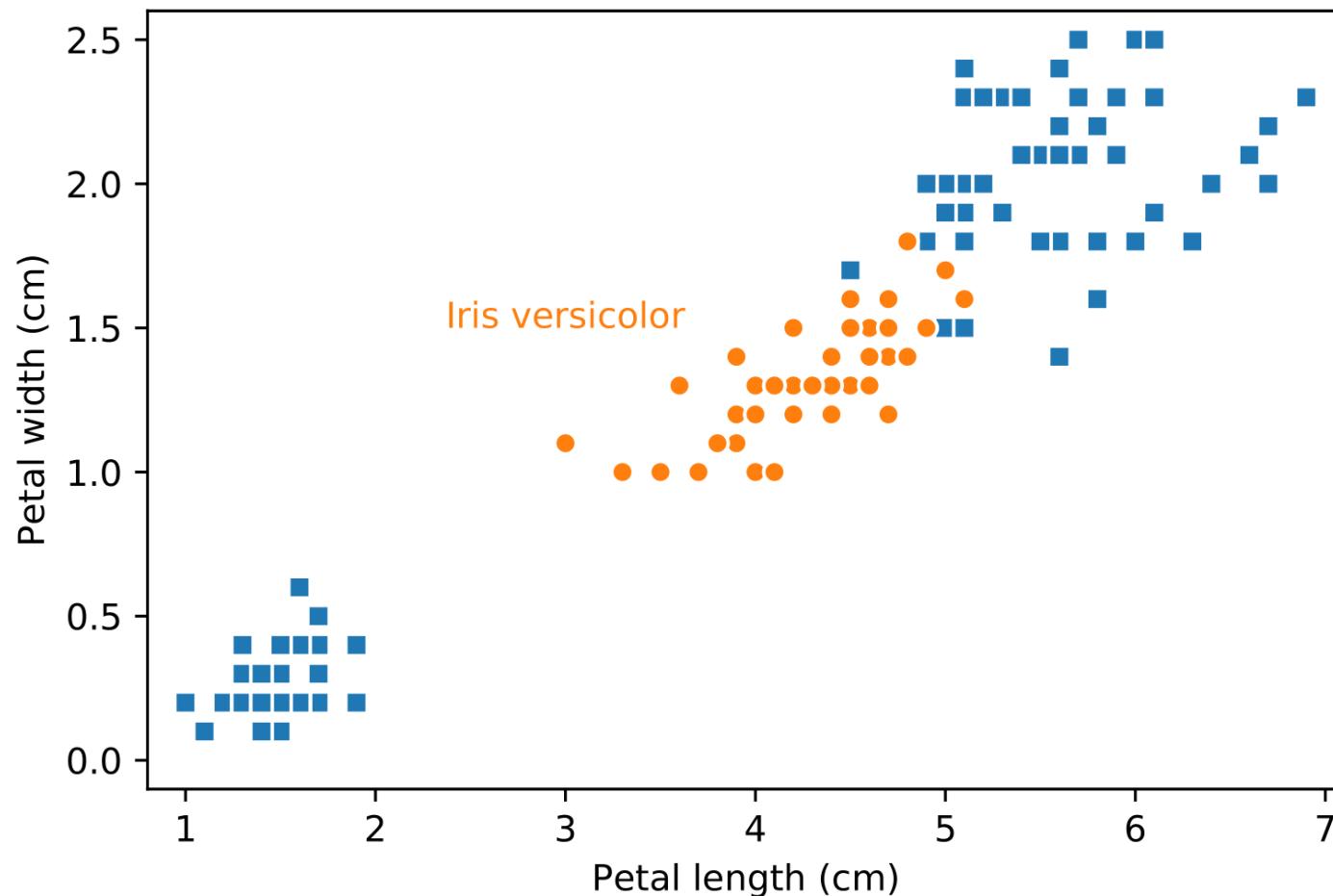
Anywhere we wrote an x in the previous videos, the feature vector x can be replaced with basis functions $\phi(x)$.

Regularisation:

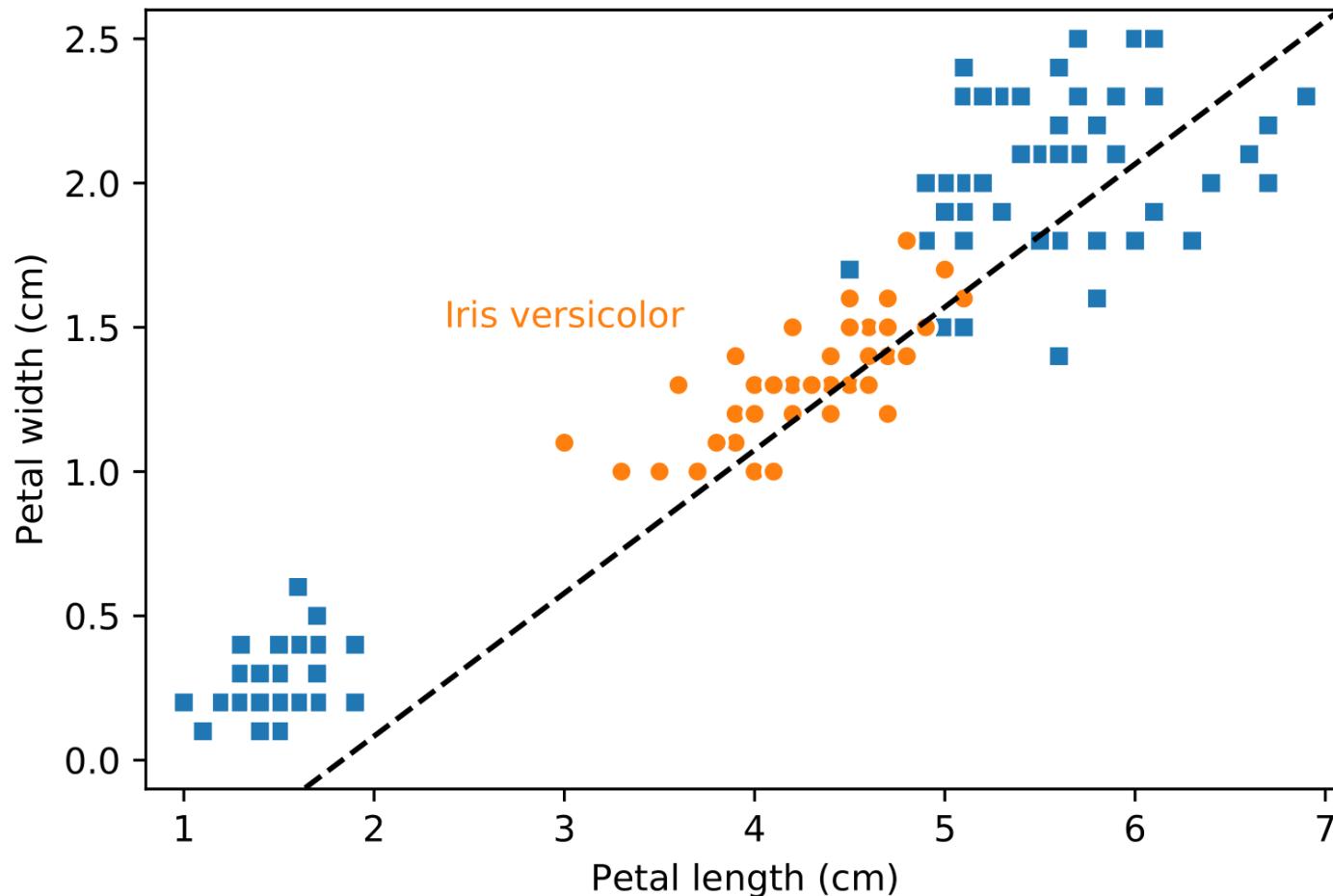
As in linear regression, we can perform regularised logistic regression by penalising the weights:

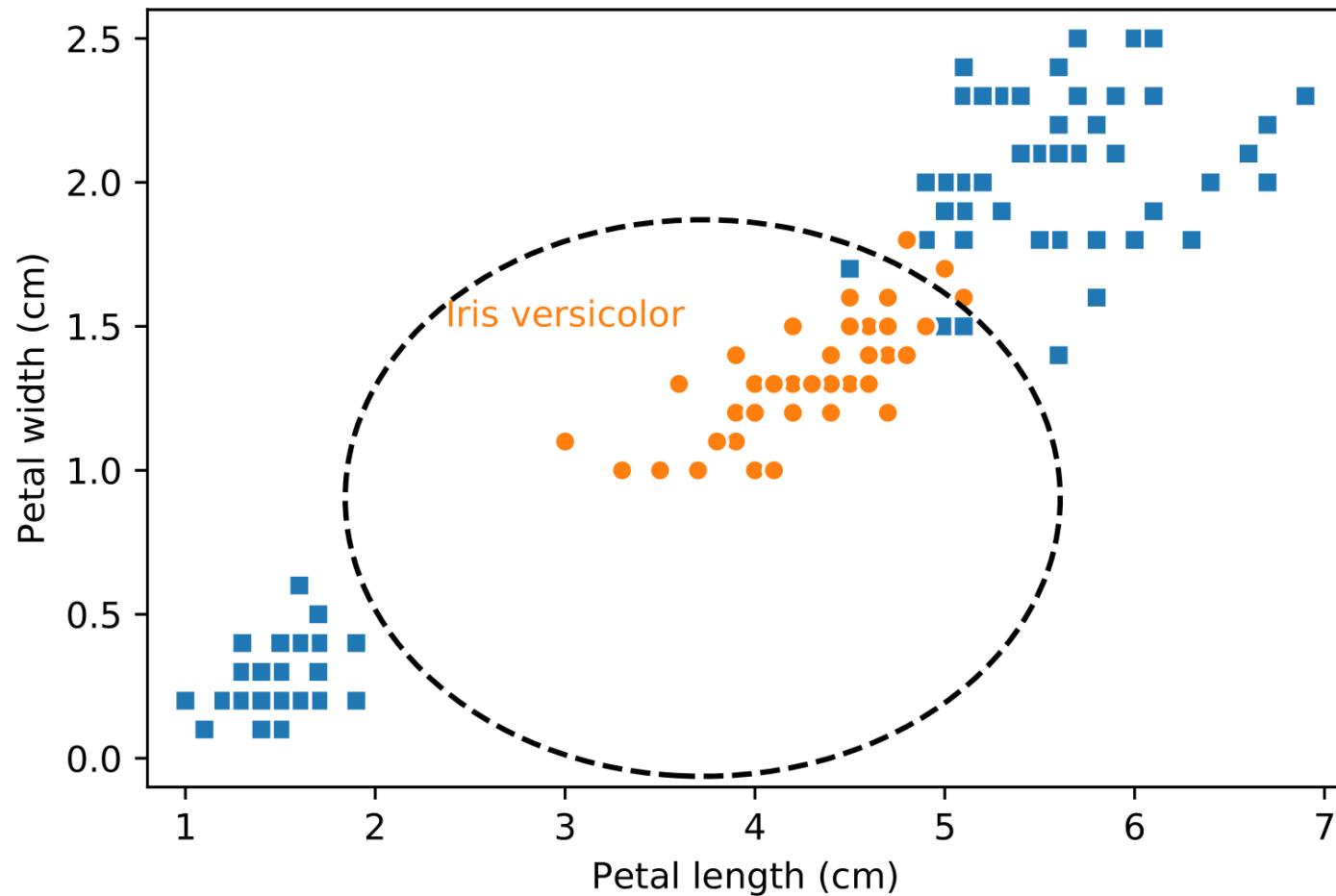
$$\begin{aligned} J(\underline{\omega}) &= -\log L(\underline{\omega}) + \lambda \sum_{d=1}^D \omega_d^2 \\ &= -\sum_{n=1}^N \left[y^{(n)} \log \sigma(\underline{\omega}^\top \underline{x}^{(n)}) + (1-y^{(n)}) \log (1-\sigma(\underline{\omega}^\top \underline{x}^{(n)})) \right] + \lambda \sum_{d=1}^D \omega_d^2 \end{aligned}$$

Logistic regression for non-separable classes

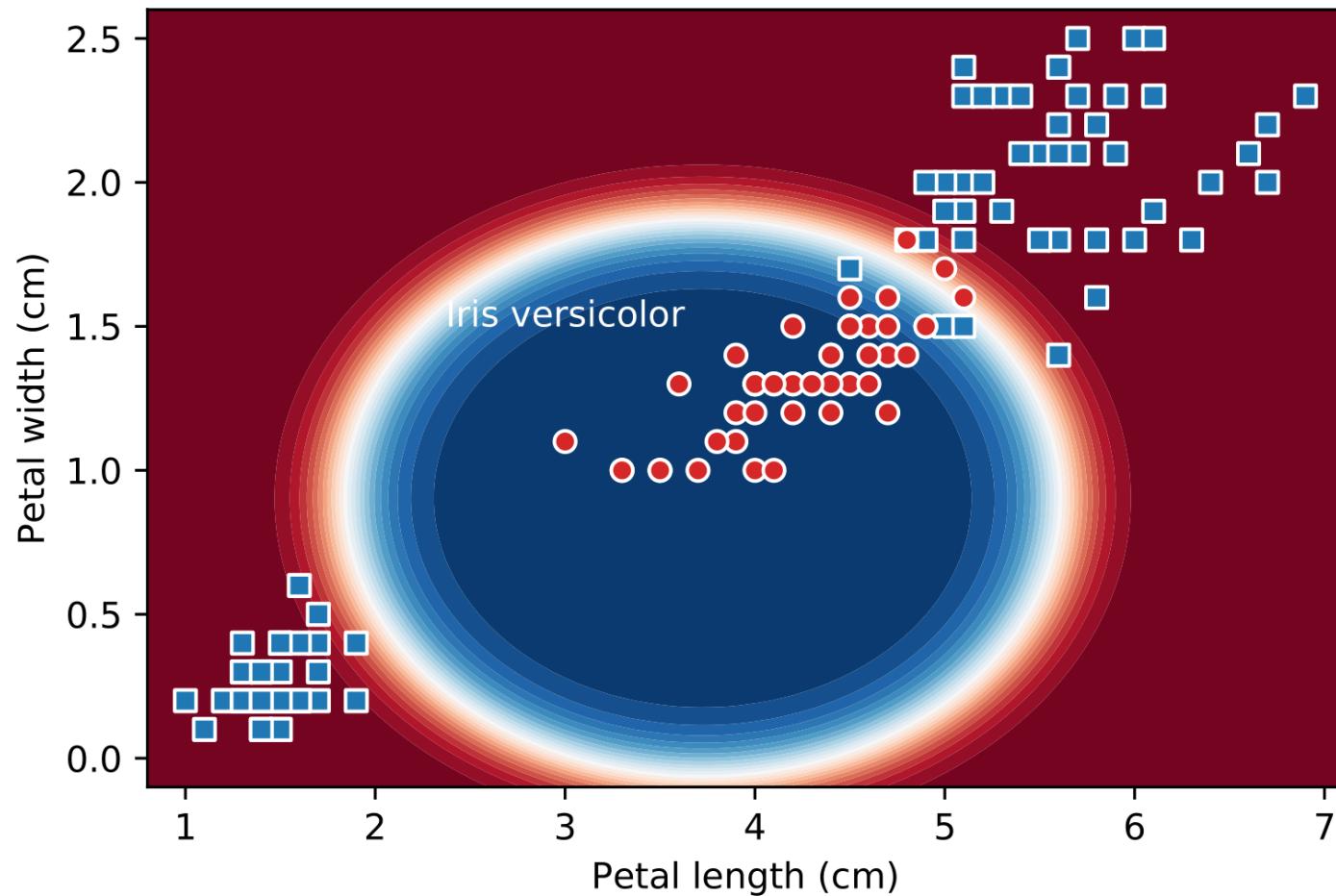


Logistic regression for non-separable classes





$$\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2]^\top$$



$$\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2]^\top$$