Binary logistic regression

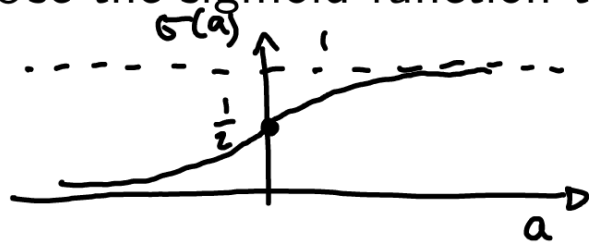
Model and loss

Herman Kamper

http://www.kamperh.com/

Model

- In general: $P(y=k \mid \underline{x}; \underline{w})$
- 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:



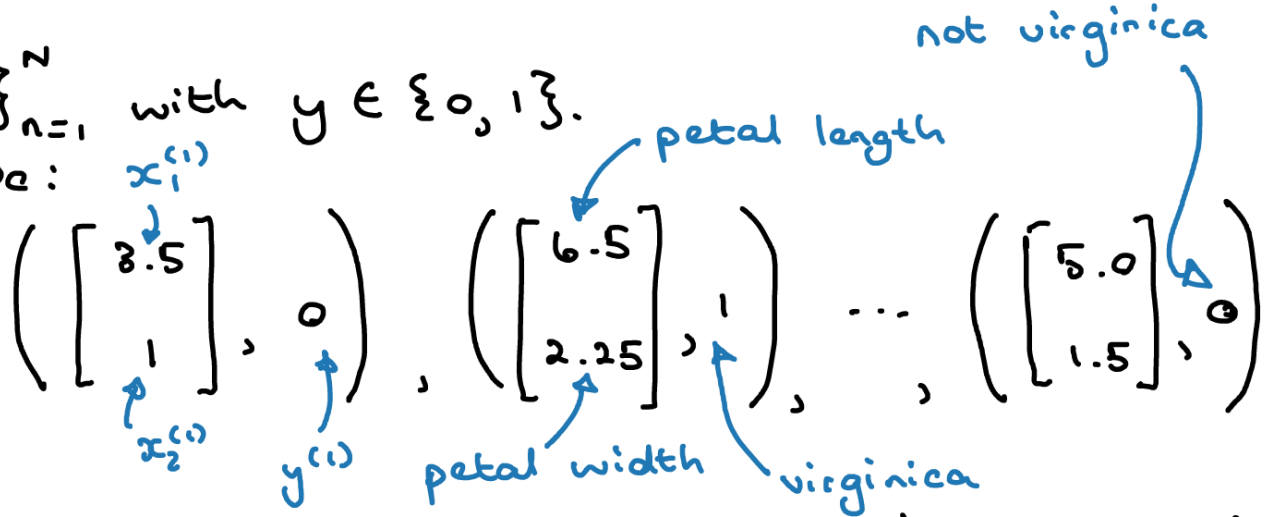
$$f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$$

- 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 $\{(\underline{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ with $y \in \{0, 1\}$.

E.g. for Iris data might have:



Use maximum likelihood:

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

Or minimise the negative log likelihood:

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

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

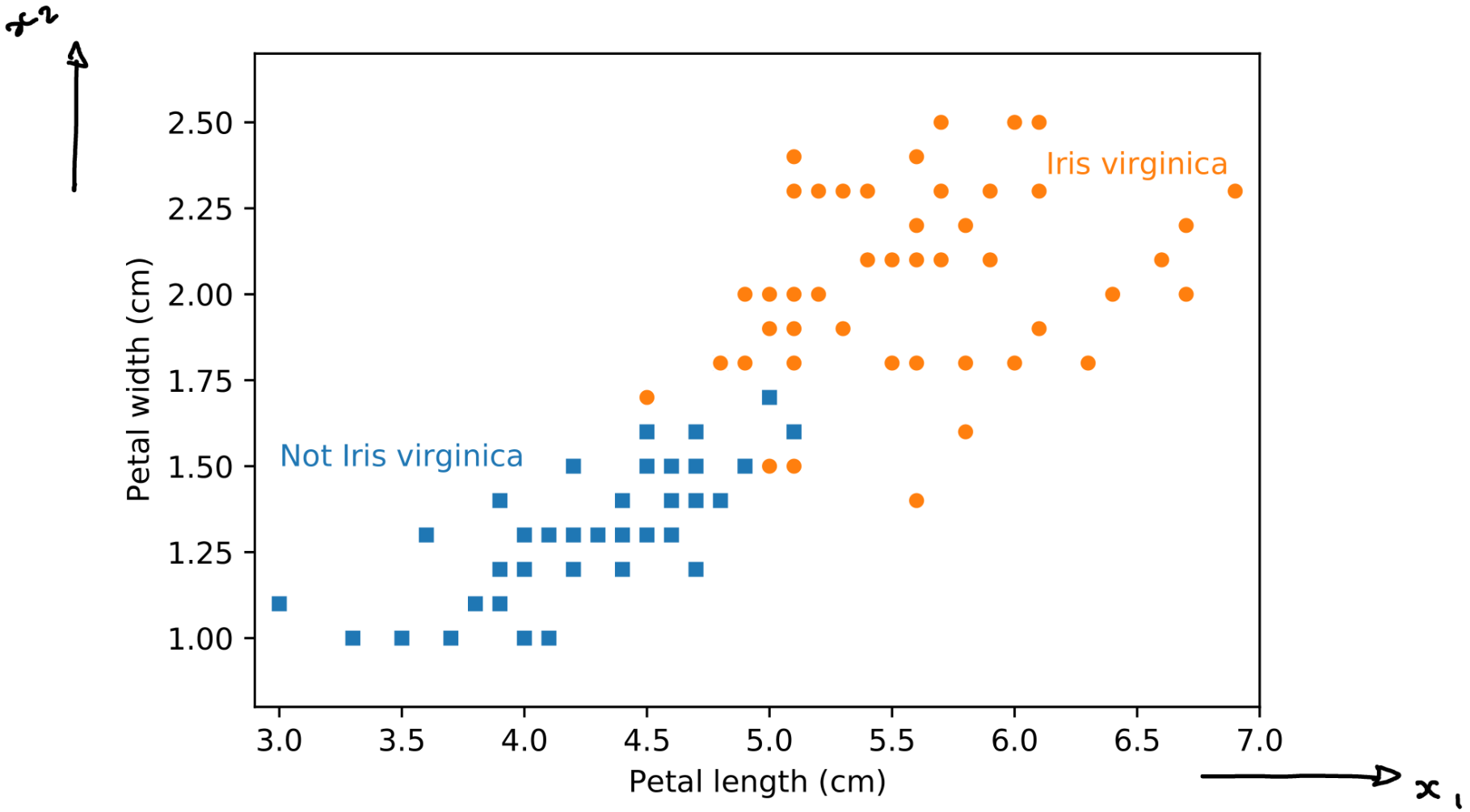
Loss function

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

$$J(\underline{w}) = -\sum_{n=1}^{N} \log \left[\sigma(\underline{w}^T \underline{x}^{(n)})^{y^{(n)}} \; (1 - \sigma(\underline{w}^T \underline{x}^{(n)}))^{(1-y^{(n)})} \right]$$

$$= -\sum_{n=1}^{N} \left[y^{(n)} \log \sigma(\underline{w}^T \underline{x}^{(n)}) + (1 - y^{(n)}) \cdot \log (1 - \sigma(\underline{w}^T \underline{x}^{(n)})) \right]$$

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:

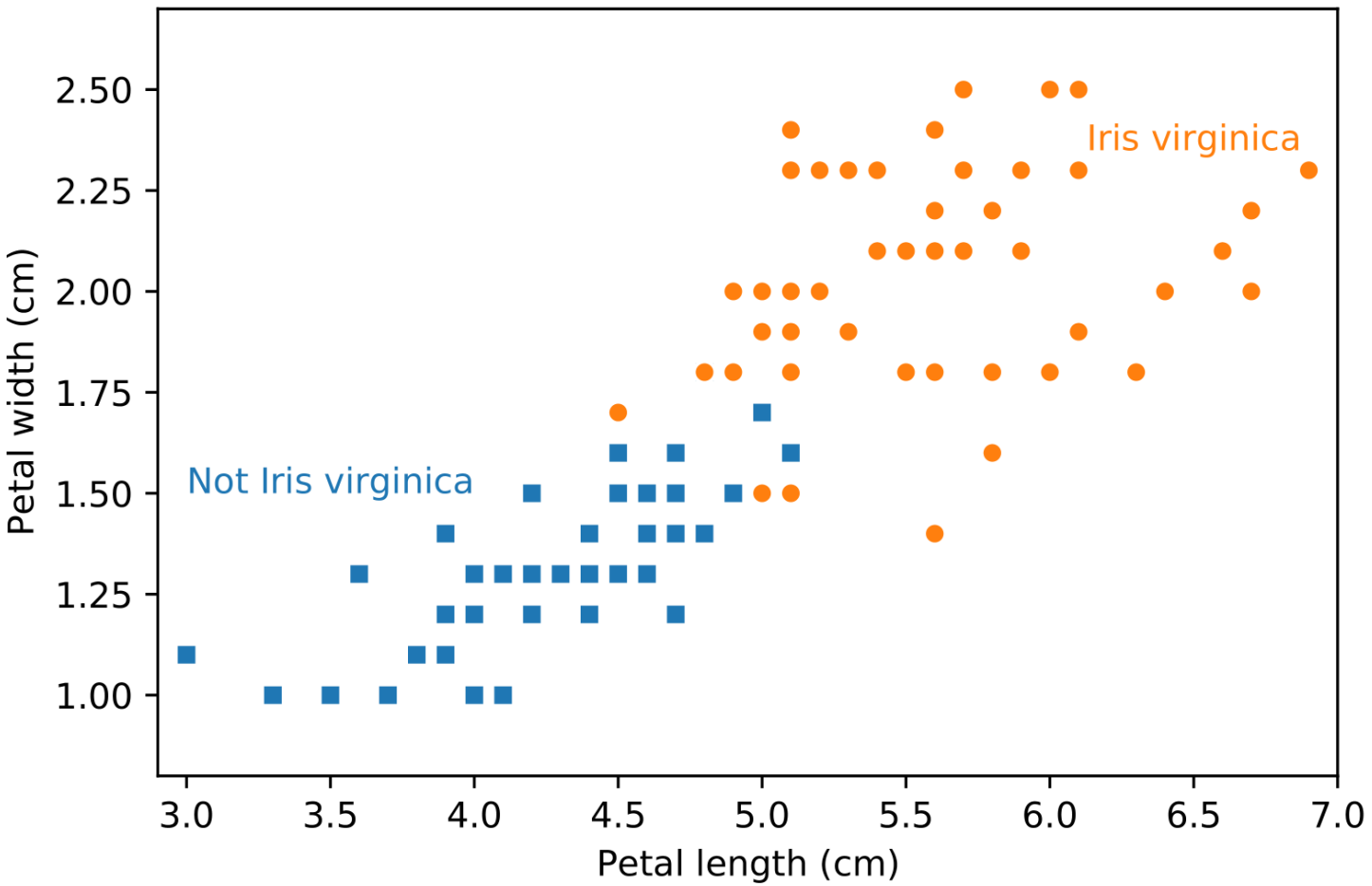
$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{n=1}^{N} (y^{(n)} - f(\mathbf{x}^{(n)}; \mathbf{w}))\mathbf{x}^{(n)}$$

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:

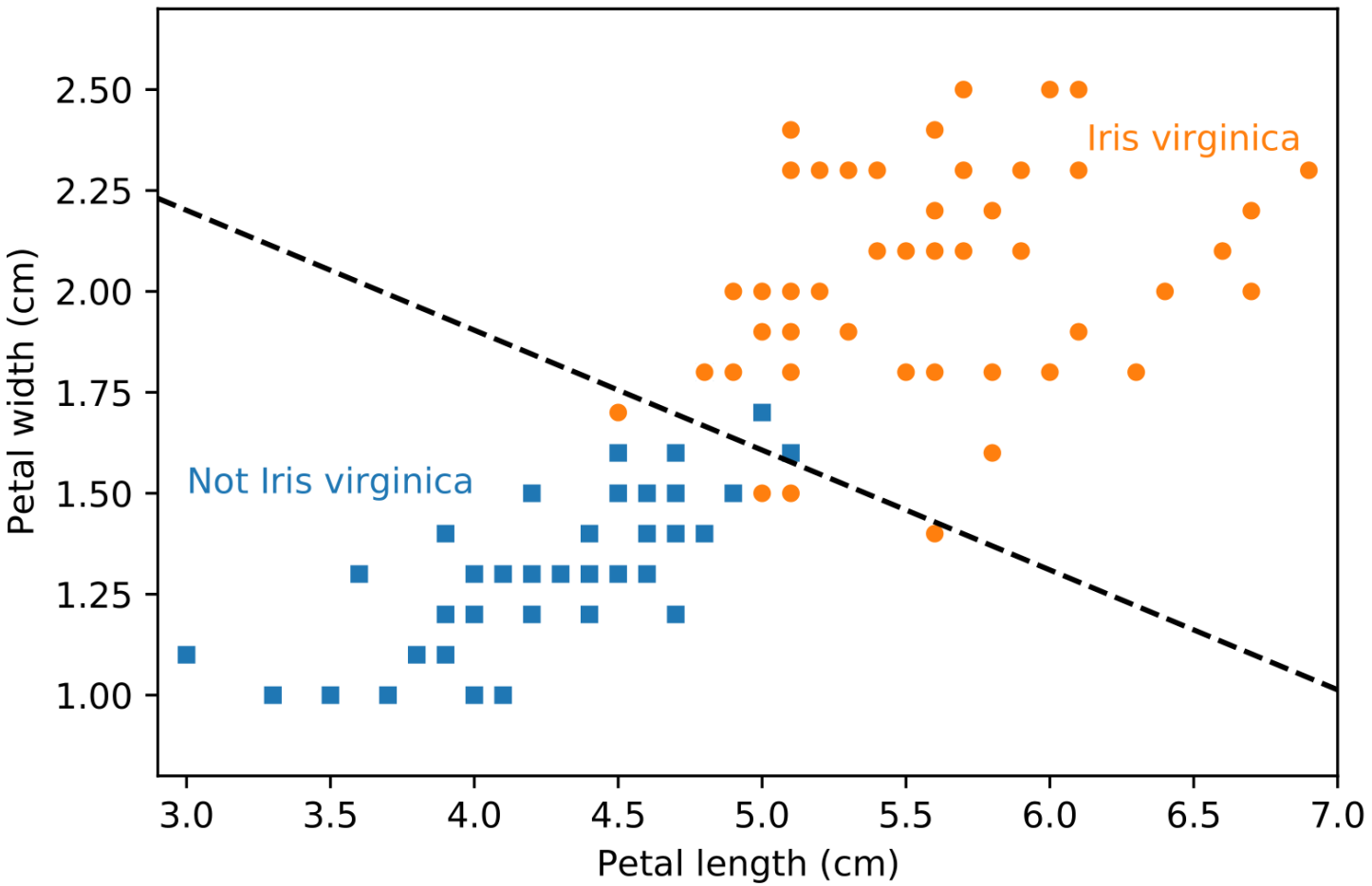
$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

[Watch the video on gradient descent]

Binary logistic regression on the Iris dataset

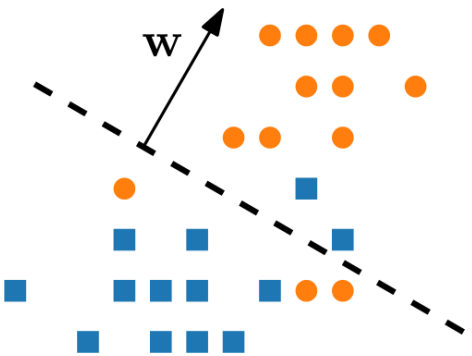


Binary logistic regression on the Iris dataset



Binary logistic regression summary

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



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

$$= -\sum_{n=1}^{N} \left[y^{(n)} \log f(\mathbf{x}^{(n)}; \mathbf{w}) + (1 - y^{(n)}) \log(1 - f(\mathbf{x}^{(n)}; \mathbf{w})) \right]$$

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

Deriving the gradients

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

$$z^{(i)} \in \{-1, 1\}$$
$$z^{(i)} = 2y^{(i)} - 1$$

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

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

Useful identities:

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

$$\frac{\partial g(\underline{w})}{\partial \underline{w}} = \frac{\partial \underline{x}}{\partial \underline{w}} \frac{\partial g(\underline{w})}{\partial \underline{x}}$$

$$\begin{aligned}
\frac{\partial J(\underline{w})}{\partial \underline{w}} &= -\sum_{i=1}^{n} \frac{\partial}{\partial \underline{w}} \log \sigma(z^{(i)} \underline{w}^T \underline{x}^{(i)}) \\
&= -\sum_{i=1}^{n} \left[\frac{\partial}{\partial \underline{w}} \sigma(z^{(i)} \cdot \underline{w}^T \underline{x}^{(i)})\right] \cdot \frac{1}{\sigma(\cdots)} \\
&= -\sum_{i=1}^{n} \left[\frac{\partial}{\partial \underline{w}} (z^{(i)} \underline{w}^T \underline{x}^{(i)})\right] \cancel{\sigma(\cdots)} (1 - \sigma(\cdots)) \frac{1}{\cancel{\sigma(\cdots)}} \\
&= -\sum_{i=1}^{n} z^{(i)} \underline{x}^{(i)} (1 - \sigma(z^{(i)} \underline{w}^T \underline{x}^{(i)})) \quad \cdots ②
\end{aligned}$$

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

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

Now ② can be written as:

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

Binary logistic regression

The decision boundary and weight vector

Herman Kamper

http://www.kamperh.com/

Decision boundary

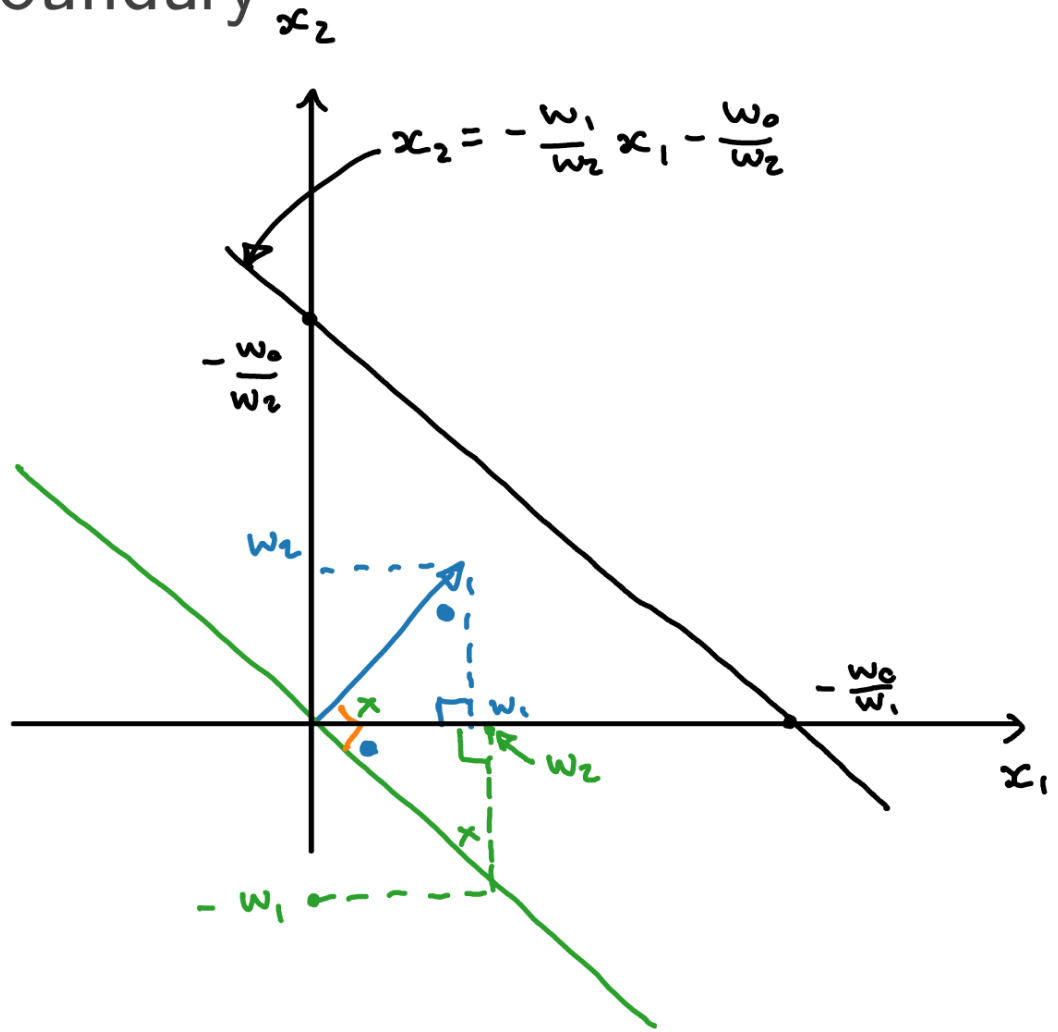
The decision boundary is the values of $\mathbf{x}$ for which $f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}) = 0.5$, i.e. $\mathbf{w}^T \mathbf{x} = 0$.

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 $\mathbf{w} = [w_1, w_2]^\top$ 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:

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

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

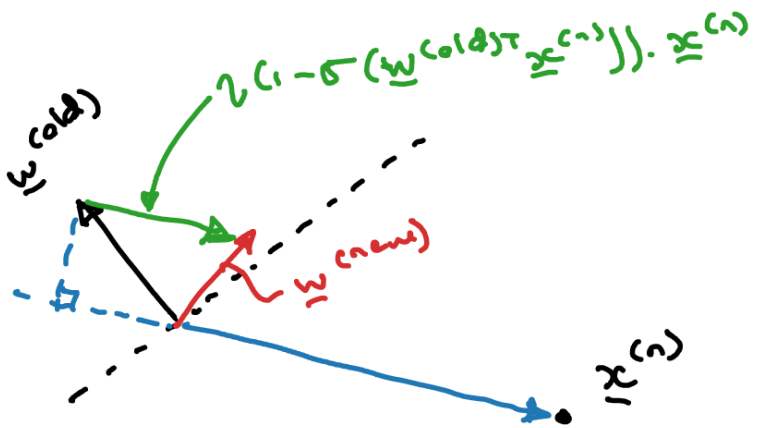
Interpreting gradient descent

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

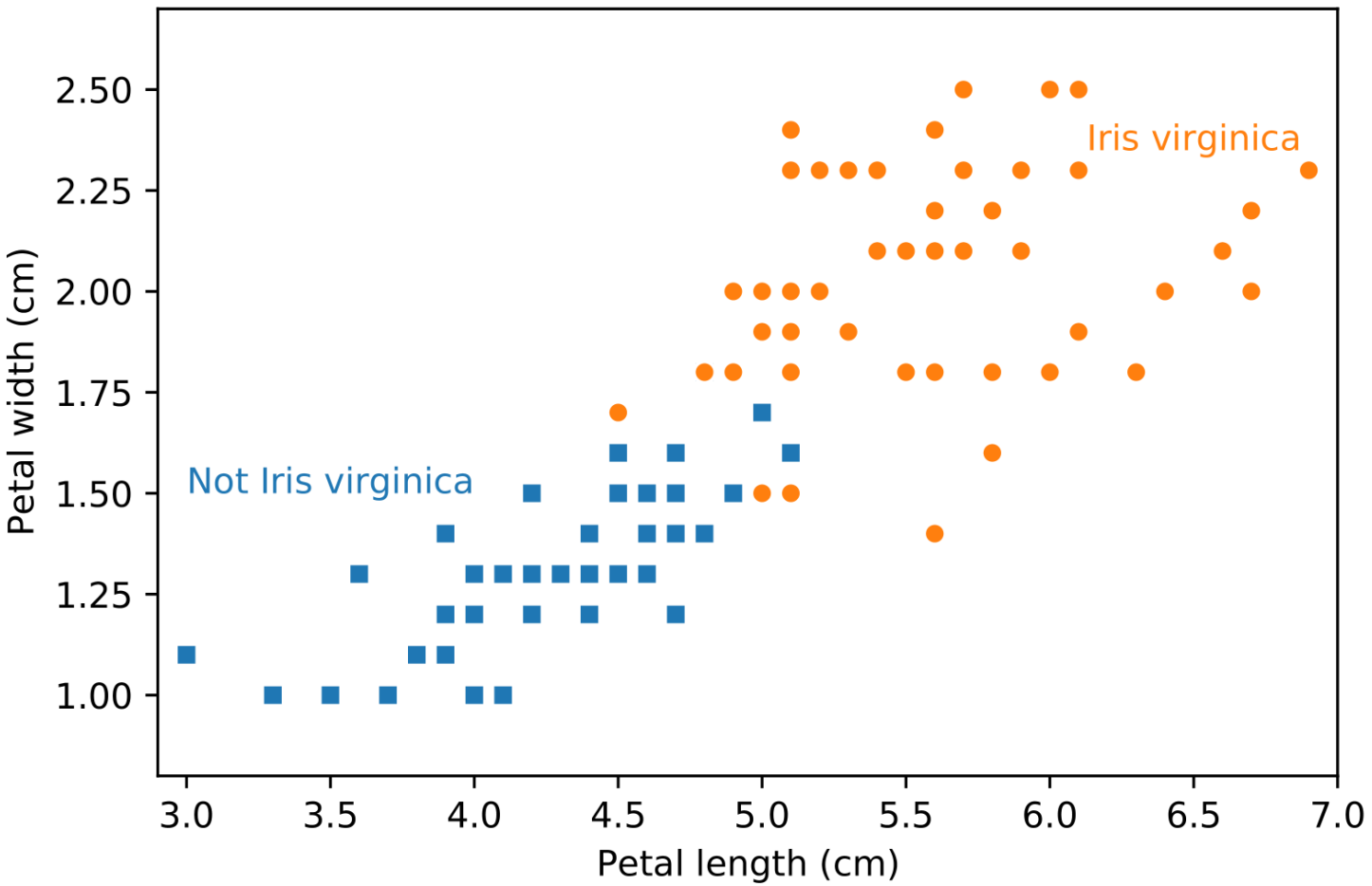
$$\underline{w}^{(new)} \leftarrow \underline{w}^{(old)} - \eta \frac{\partial J}{\partial \underline{w}}$$

$$\underline{w}^{(new)} \leftarrow \underline{w}^{(old)} + \eta \left(1 - \sigma(\underline{w}^{(old)T} \cdot \underline{x}^{(n)})\right) \underline{x}^{(n)}$$

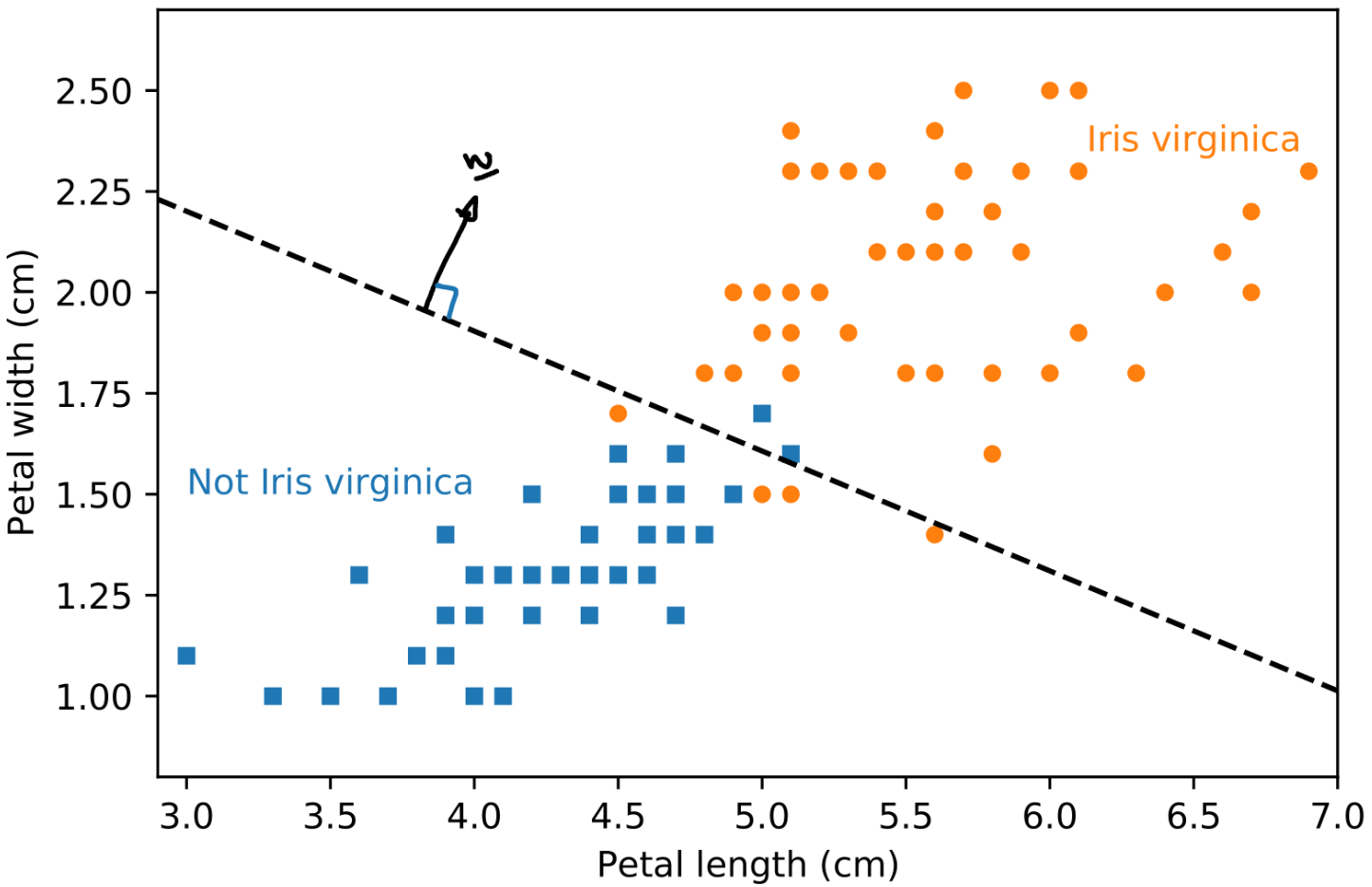
- $\mathbf{w}$ is a vector orthogonal to the decision boundary
- Let's pretend we have a single training example with a positive label $y^{(n)} = 1$
- How does this single example affect the decision boundary in the gradient descent update step?
- We also pretend we don't have a bias term w_0



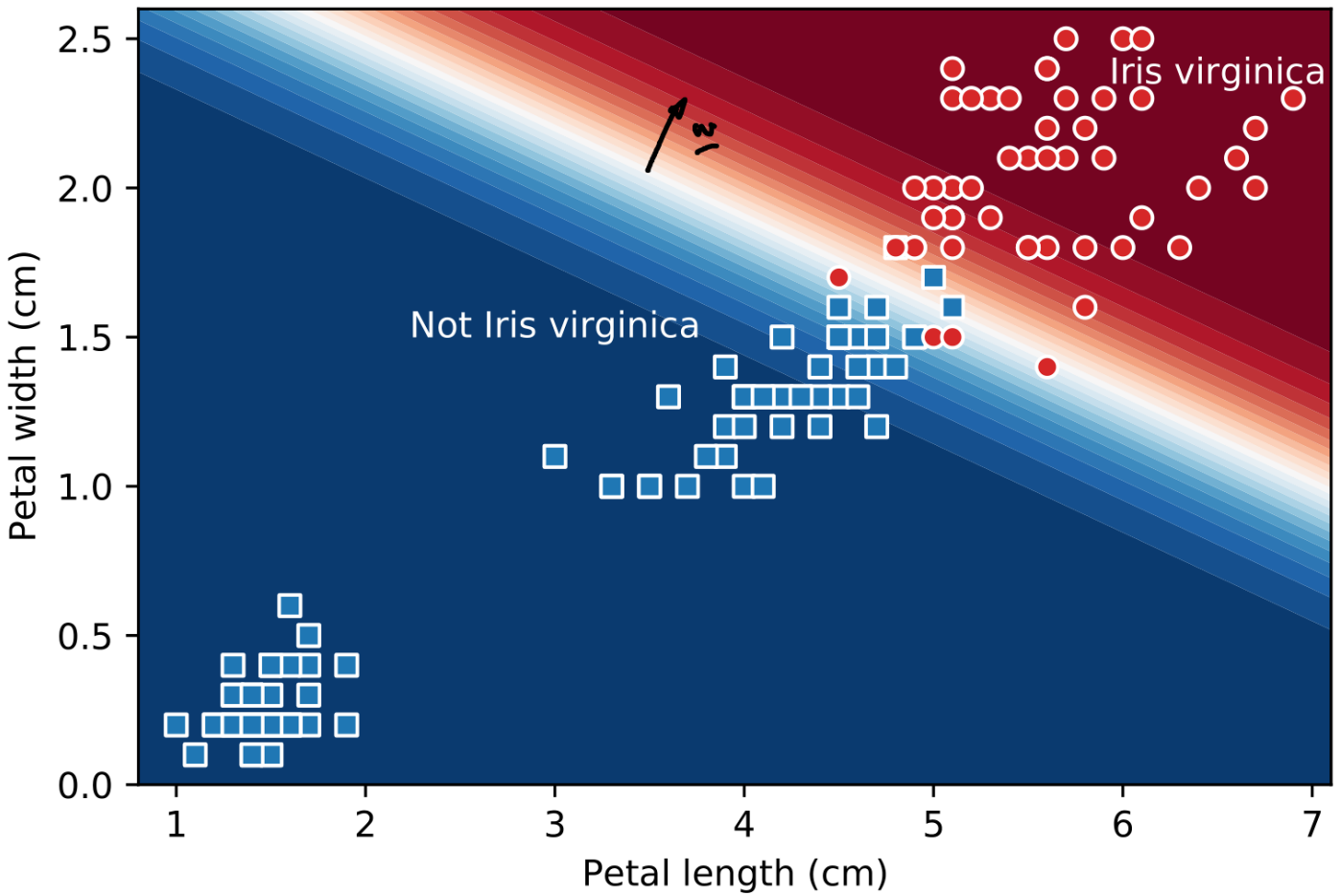
Iris dataset



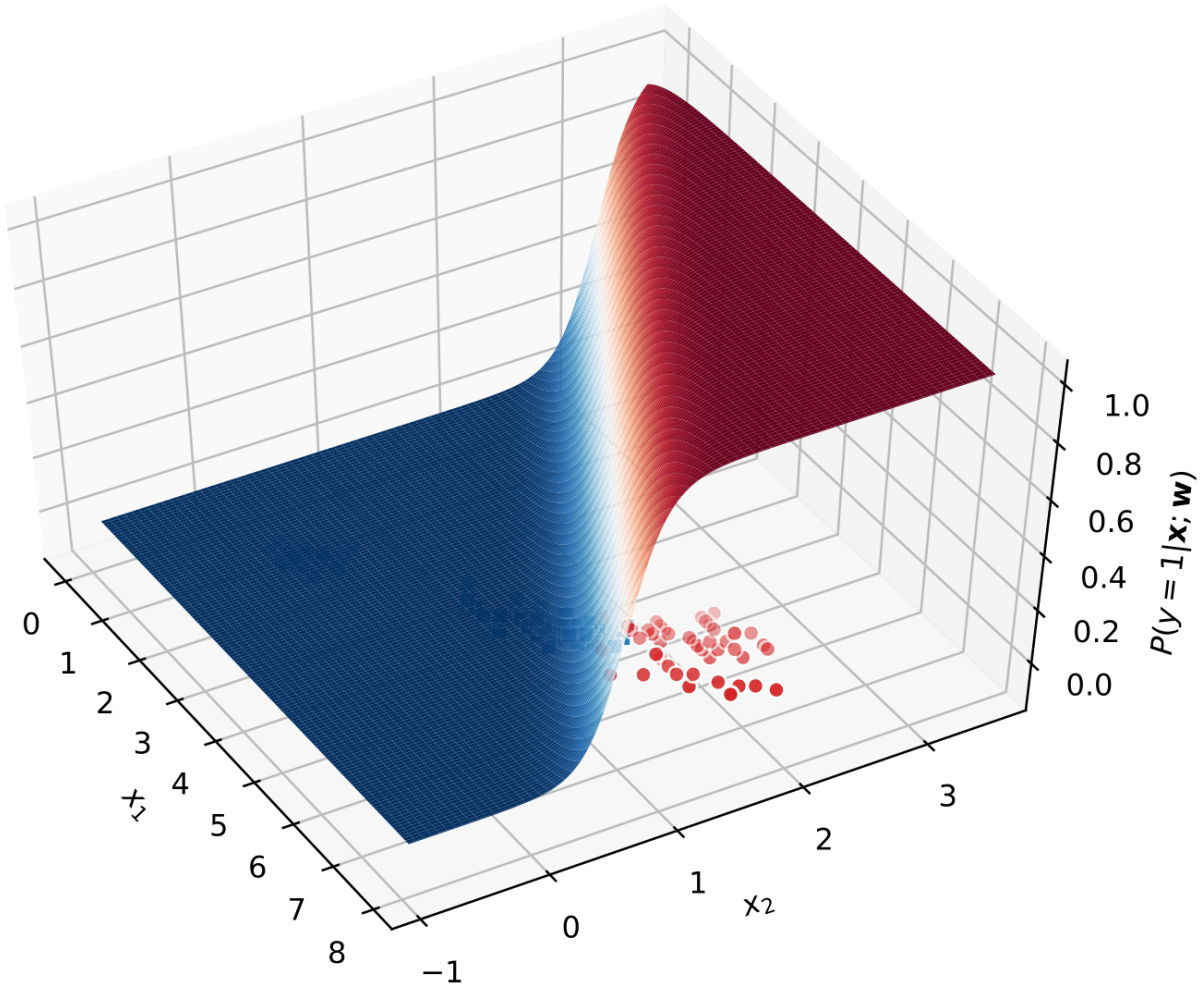
Iris dataset



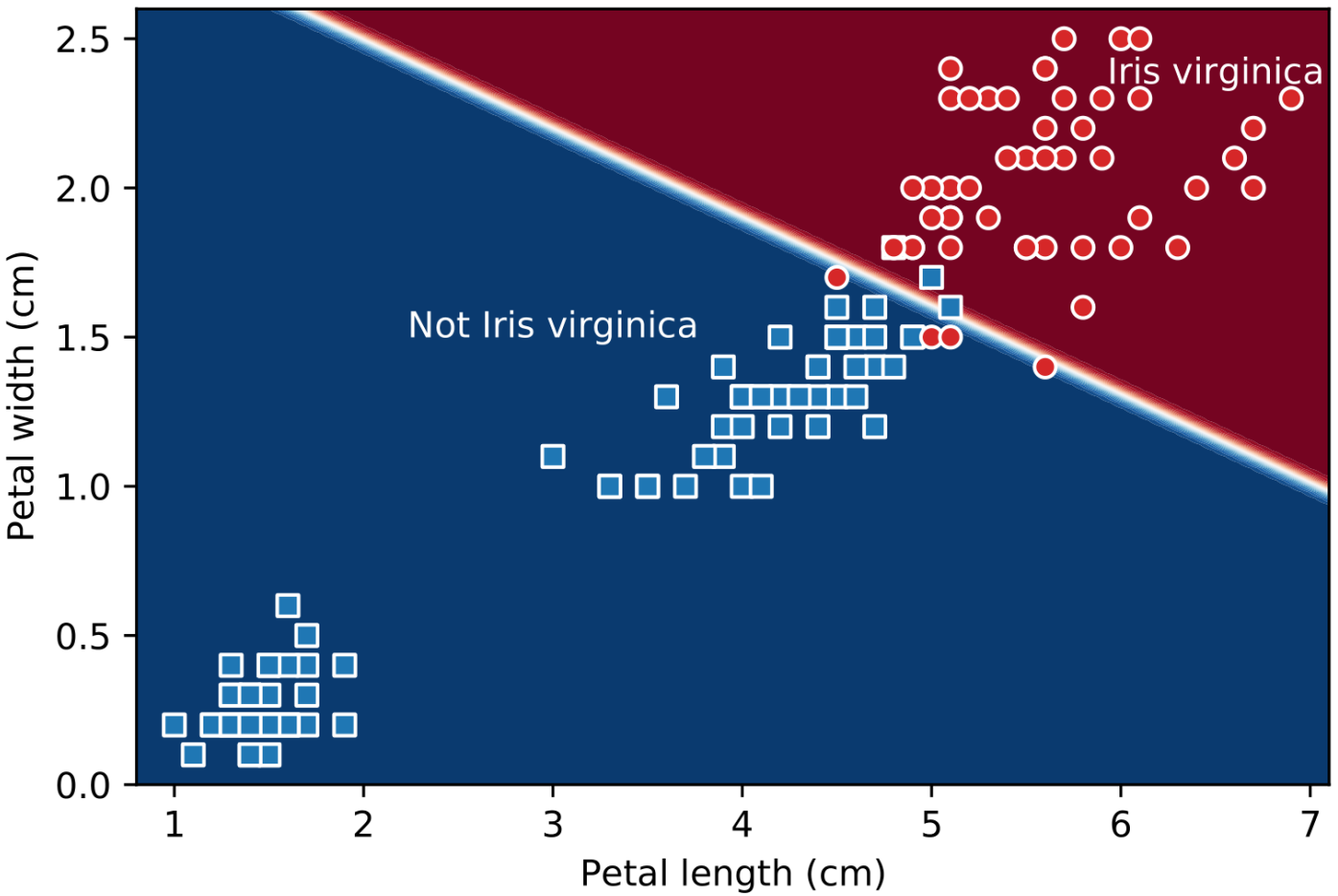
Visualising probabilities



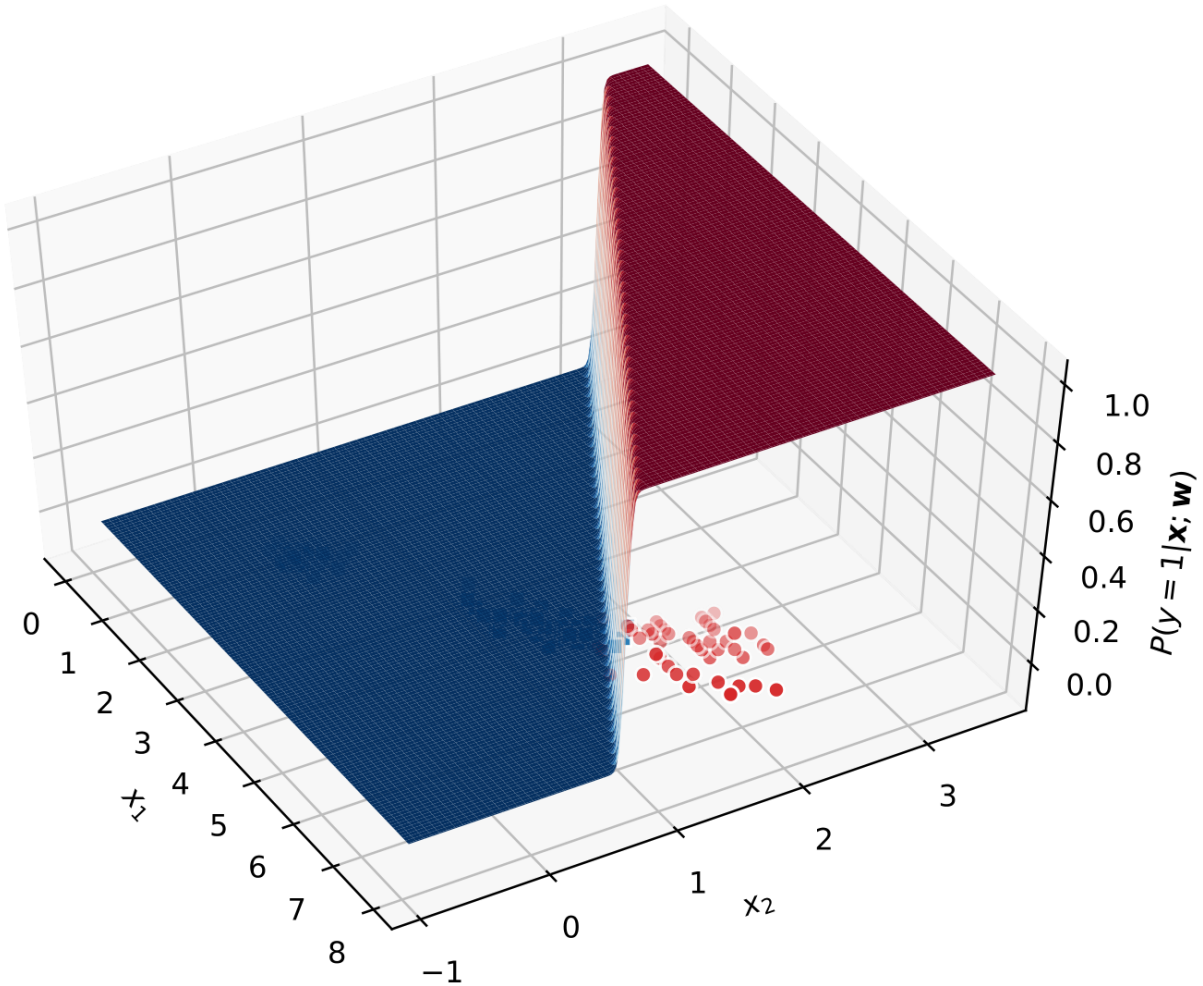
Probability surface



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



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

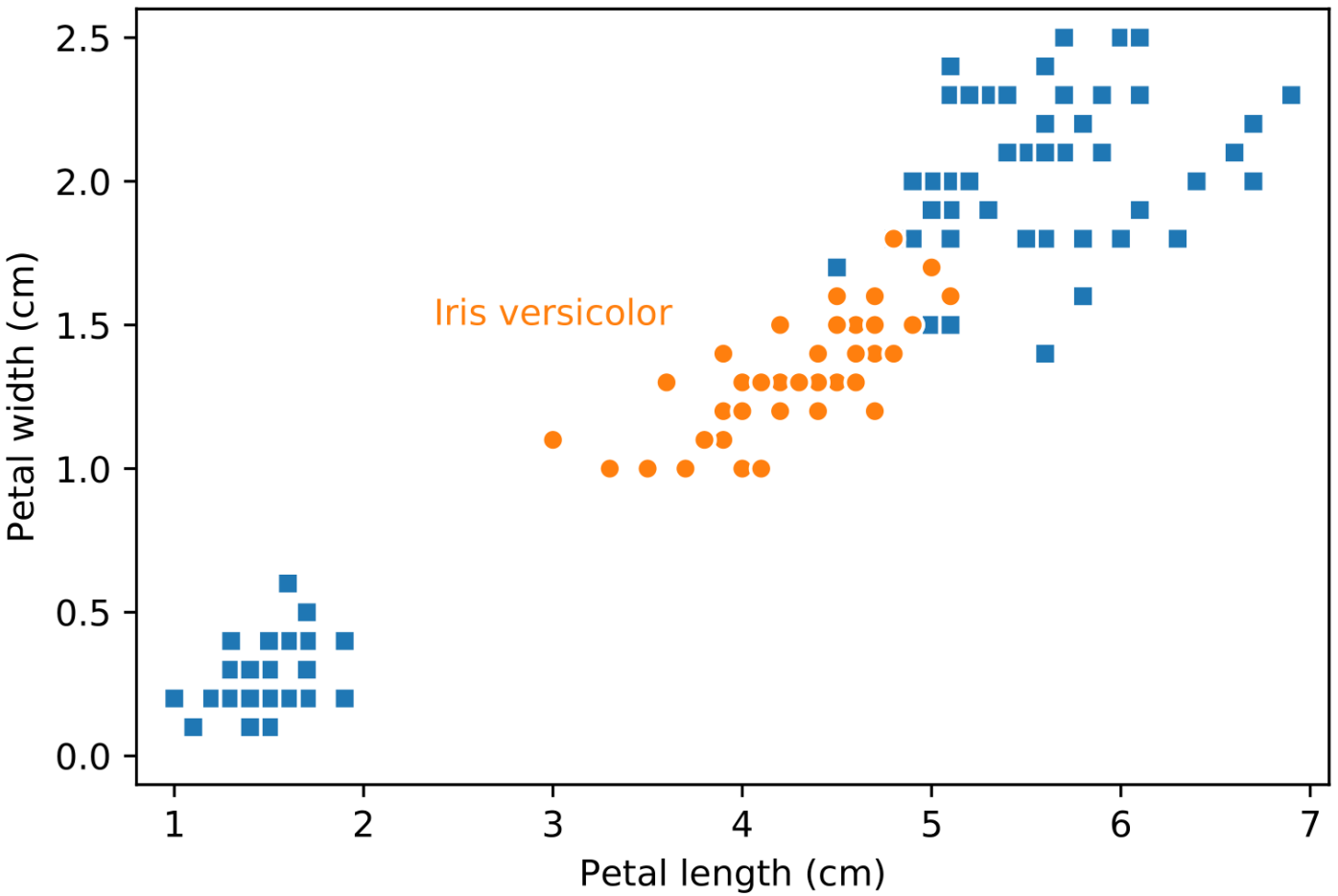
Regularisation:

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

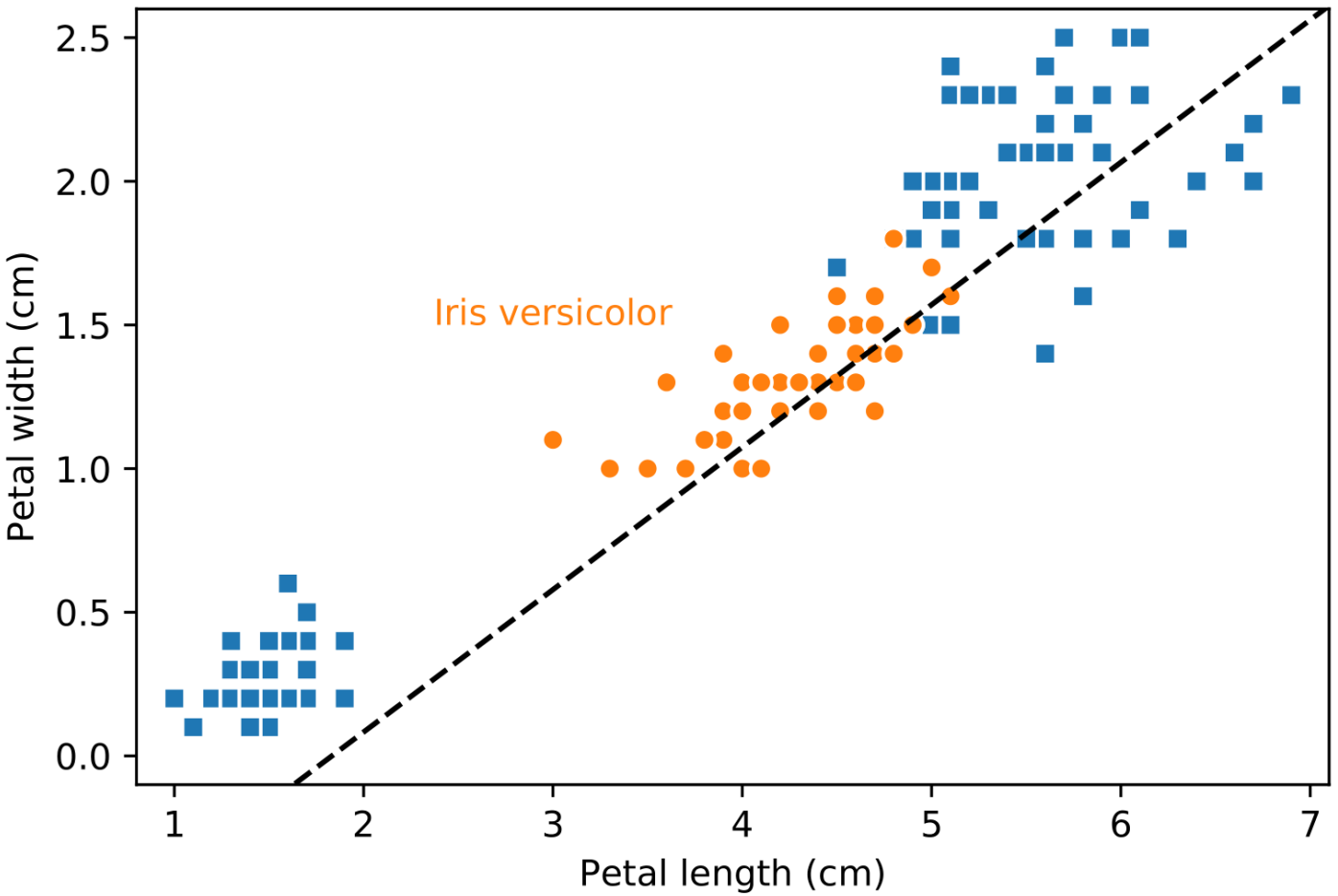
$$J(\underline{w}) = -\log L(\underline{w}) + \lambda \sum_{d=1}^{D} w_d^2$$

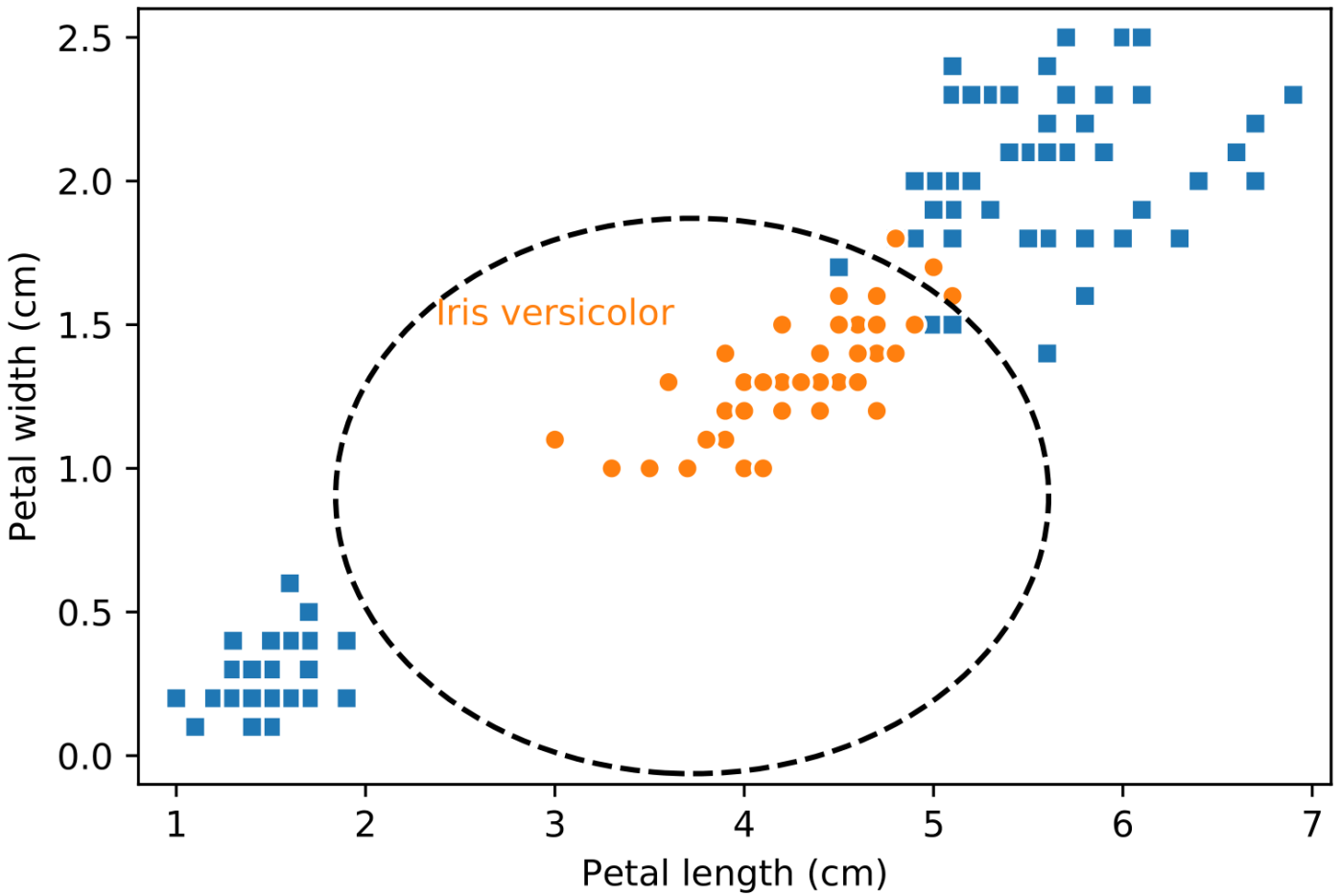
$$= -\sum_{n=1}^{N} \left[y^{(n)} \log \sigma(\underline{w}^T \underline{x}^{(n)}) + (1-y^{(n)}) \log(1-\sigma(\underline{w}^T \underline{x}^{(n)})) \right] + \lambda \sum_{d=1}^{D} w_d^2$$

Logistic regression for non-separable classes

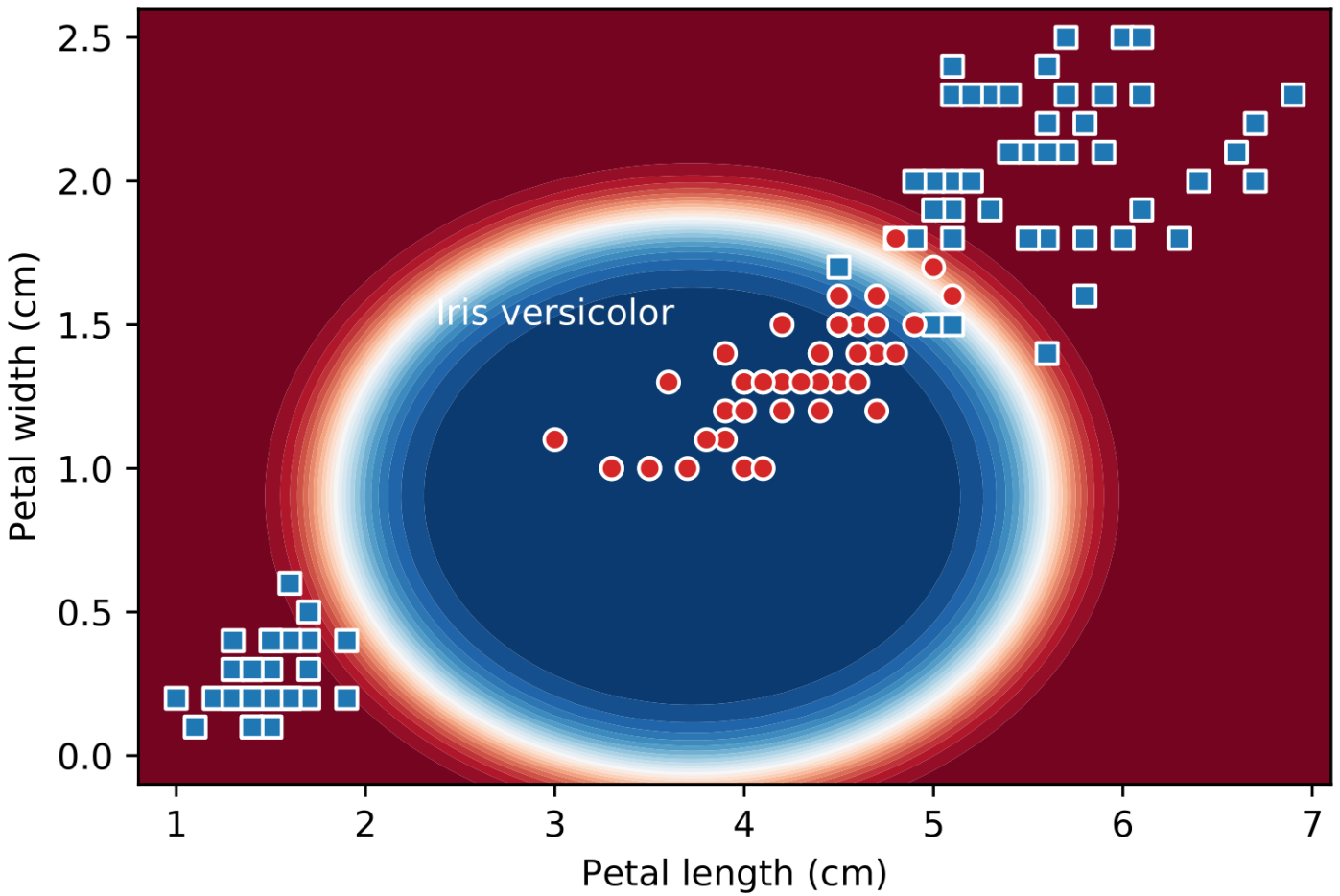


Logistic regression for non-separable classes





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



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