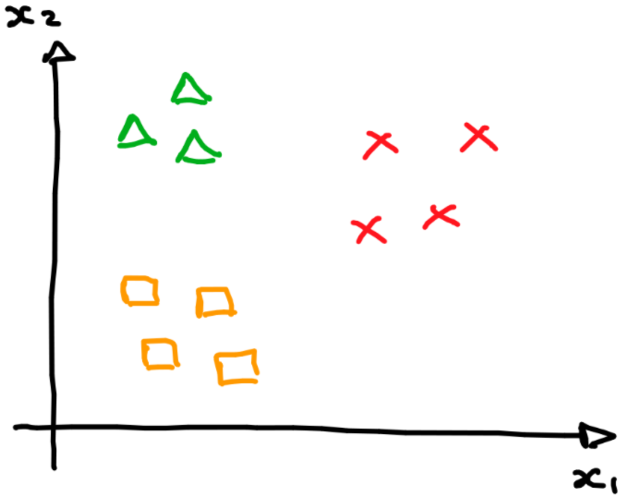
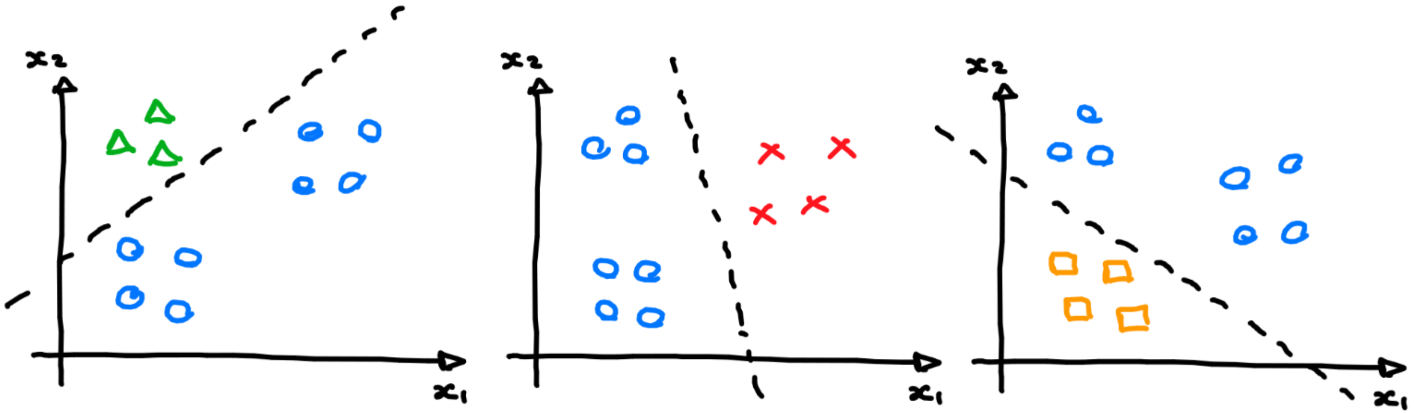
Multiclass logistic regression

Herman Kamper

http://www.kamperh.com/

One-vs-rest classification





Strategy: Train three classifiers with $y \in \{0, 1\}$, where each classifier considers another class as the positive class.

We then get three classification models:

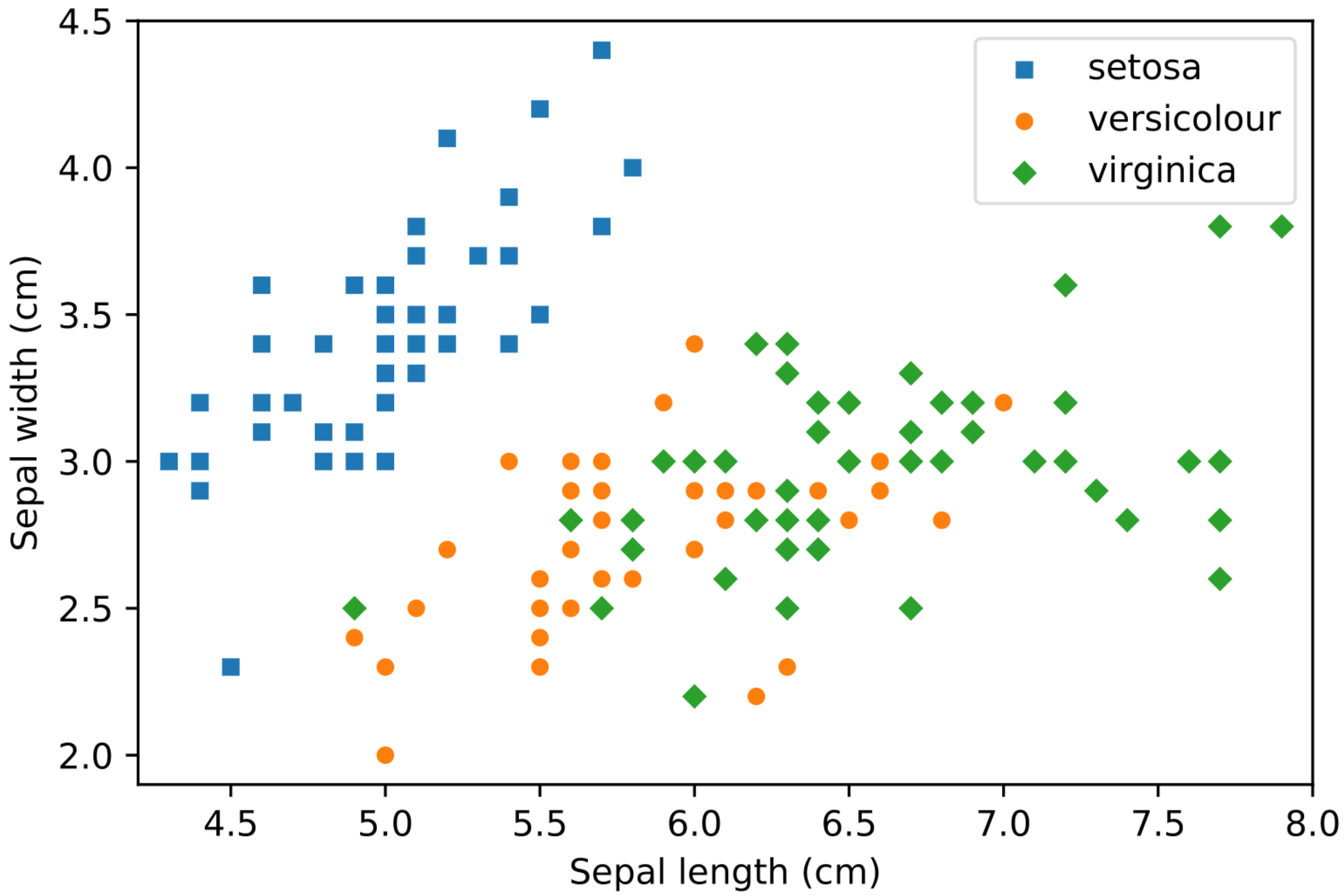
$f_1(\underline{x}; \underline{w}_1)$ 1: △

$f_2(\underline{x}; \underline{w}_2)$ 2: ×

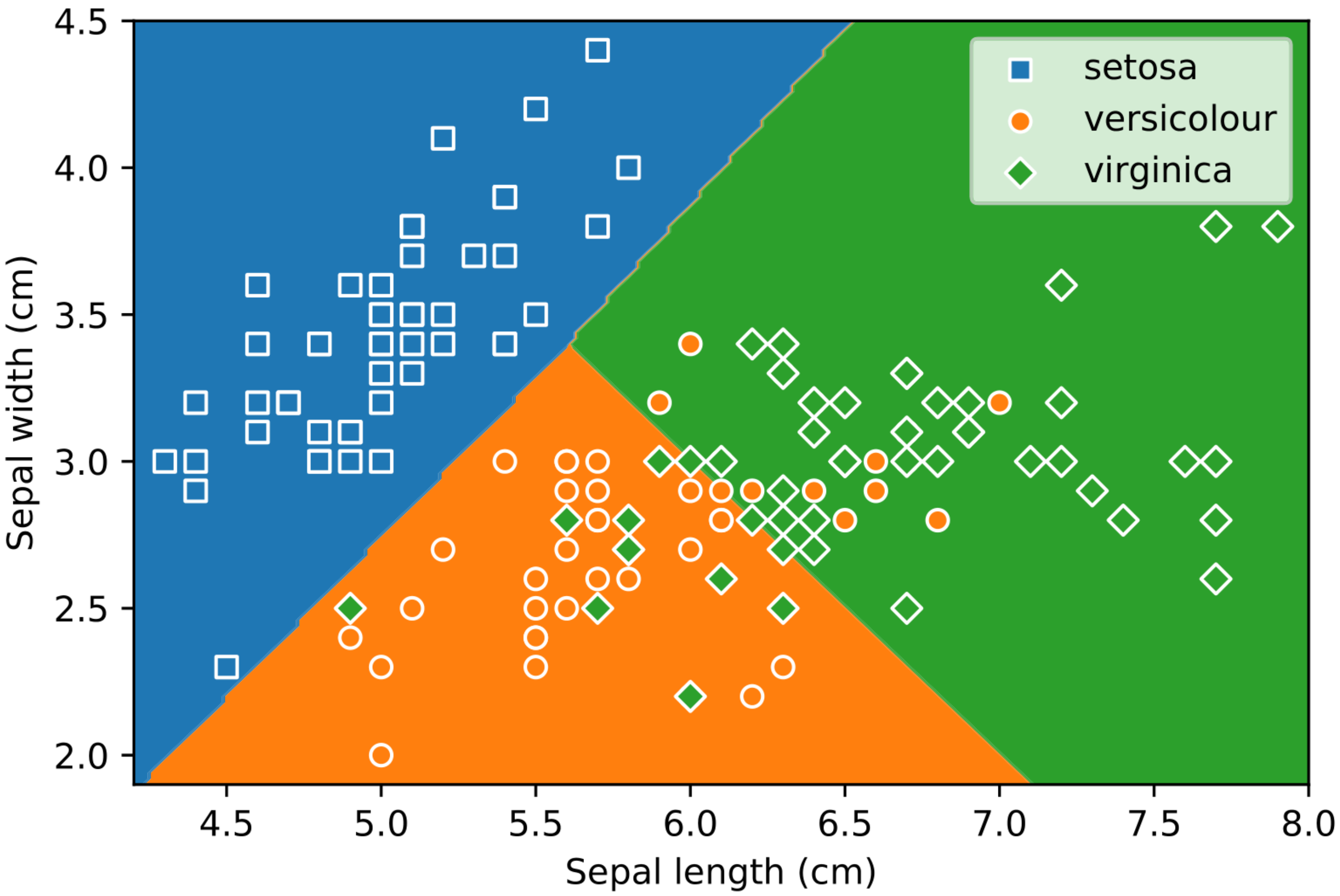
$f_3(\underline{x}; \underline{w}_3)$ 3: □

Final predictions: $\arg\max_k f_k(\underline{x}; \underline{w}_k)$

Iris dataset



One-vs-rest decision boundary



Softmax regression

- For binary logistic regression we had $f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \dfrac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$ with $y \in \{0, 1\}$.
- We interpreted the output as $P(y = 1 | \mathbf{x}; \mathbf{w})$, implying $P(y = 0 | \mathbf{x}; \mathbf{w}) = 1 - f(\mathbf{x}; \mathbf{w})$.
- For the multiclass setting we now have $y \in \{1, 2, \ldots, K\}$.
- **Idea:** Instead of just outputting a single value for the positive class, let's output a vector of probabilities for each class:

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

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

Softmax regression

- Each element in $\boldsymbol{f}(\mathbf{x};\mathbf{W})$ 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 $\mathbf{w}_k^\top \mathbf{x}$.
- But probabilities need to be positive. So let's take the exponential: $e^{\mathbf{w}_k^\top \mathbf{x}}$.
- But probabilities need to sum to one. So let's normalise:

$$P(y = k|\mathbf{x};\mathbf{W}) = \frac{e^{\mathbf{w}_k^\top \mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}}}$$

- This gives us the softmax regression model:

$$\underline{f}(\underline{x};\underline{W}) = \frac{1}{\sum_{j=1}^{k} e^{\underline{w}_j^T \underline{x}}} \begin{bmatrix} e^{\underline{w}_1^T \underline{x}} \\ e^{\underline{w}_2^T \underline{x}} \\ \vdots \\ e^{\underline{w}_k^T \underline{x}} \end{bmatrix} = \text{softmax}(\underline{W}\,\underline{x})$$

Parameters:

Vectors $\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_K$

Parameter matrix:

$$\underline{W} = \begin{bmatrix} — (\underline{w}_1)^T — \\ — (\underline{w}_2)^T — \\ \vdots \\ — (\underline{w}_K)^T — \end{bmatrix}$$

Optimisation

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

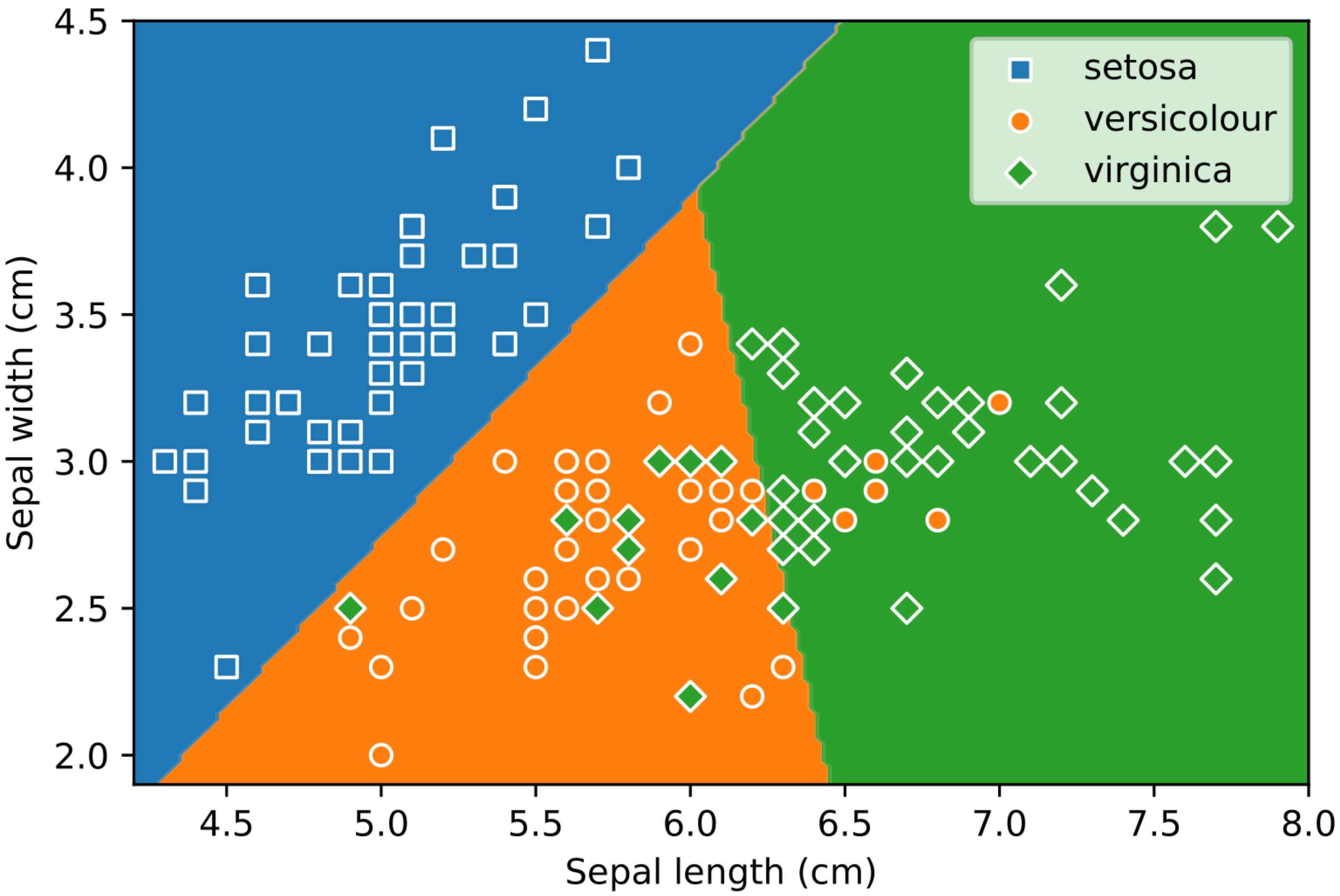
$$\begin{aligned} J(\mathbf{W}) &= -\log L(\mathbf{W}) \\ &= -\sum_{n=1}^{N} \log P(y^{(n)} | \mathbf{x}^{(n)}; \mathbf{W}) \\ &= -\sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}\{y^{(n)} = k\} \log \frac{e^{\mathbf{w}_k^\top \mathbf{x}^{(n)}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}^{(n)}}} \end{aligned}$$

- Derivatives:

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

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

Softmax regression decision boundary



Output representation

Sometimes it is convenient to represent the target output as a *one-hot vector*:

$$\mathbf{y}^{(n)} = \begin{bmatrix} \overset{1}{0} & \overset{2}{0} & \dots & 0 & \overset{k}{1} & 0 & \dots & \overset{K}{0} \end{bmatrix}^\top$$

This one-hot vector has a one in the position $y_k^{(n)}$ if $\mathbf{x}^{(n)}$ is of class k, with zeros everywhere else. This is a convenient representation for the target output, since it allows us to vectorise algorithms. We can then write the loss and gradient as:

$$J(\mathbf{W}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_k^{(n)} \log \frac{e^{\mathbf{w}_k^\top \mathbf{x}^{(n)}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}^{(n)}}}$$

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

Relationship between softmax and binary logistic regression

- For the special case that $K = 2$, you can show that softmax regression reduces to:

$$\boldsymbol{f}(\mathbf{x}; \mathbf{W}) = \begin{bmatrix} \frac{1}{1+\exp\{(\mathbf{w}_1-\mathbf{w}_2)^\top \mathbf{x}\}} \\ 1 - \frac{1}{1+\exp\{(\mathbf{w}_1-\mathbf{w}_2)^\top \mathbf{x}\}} \end{bmatrix}$$

- So the model only depends on $\mathbf{w}_2 - \mathbf{w}_1$, a single vector.
- We can replace this vector with $\mathbf{w}' = \mathbf{w}_2 - \mathbf{w}_1$, and only need to fit $\mathbf{w}'$.
- This is equivalent to binary logistic regression.