Using Machine Learning Tools PG

Week 6 – Logistic Regression & Support Vector Machines

COMP SCI 7317

Trimester 2, 2024

THE UNIVERSITY of ADELAIDE

150 YEARS

From last week… Training models

1. Training Models

- Cost/error/loss function
- Gradient and stochastic gradient descent continued
- Learning rate continued
- Stopping criteria
- Training curve/learning curve

2. Regularisation

- L2 regularisation (ridge/Tikhonov)
- L1 regularisation (Lasso)
- Elastic net

This week

1. **Regularisation Continued**
   - L1, L2, Elastic Net
2. **Logistic Regression**
   - Logistic & Logit Regression
   - Training a logistic model
   - Softmax
3. **Support Vector Machines (SVM)**
   - Linear SVM - Hard Margin
   - Linear SVM - Soft Margin
   - Kernel trick

Regularisation Continued

Regularisation

Regularisation: a method that adds a term to the cost function to prevent overfitting. Controlled by an adjustable weight α.

$$Cost = data\_term + \alpha \; * regularisation\_term$$

Purpose:

- Help avoid overfitting by penalising large parameter values.
- Encourages ‘smooth’ outputs.
- Add desired properties to the model (a-priori knowledge).
- Balance between fitting data and simplicity (multi-objective optimisation).

L2 (Ridge/Tikhonov) Regularisation

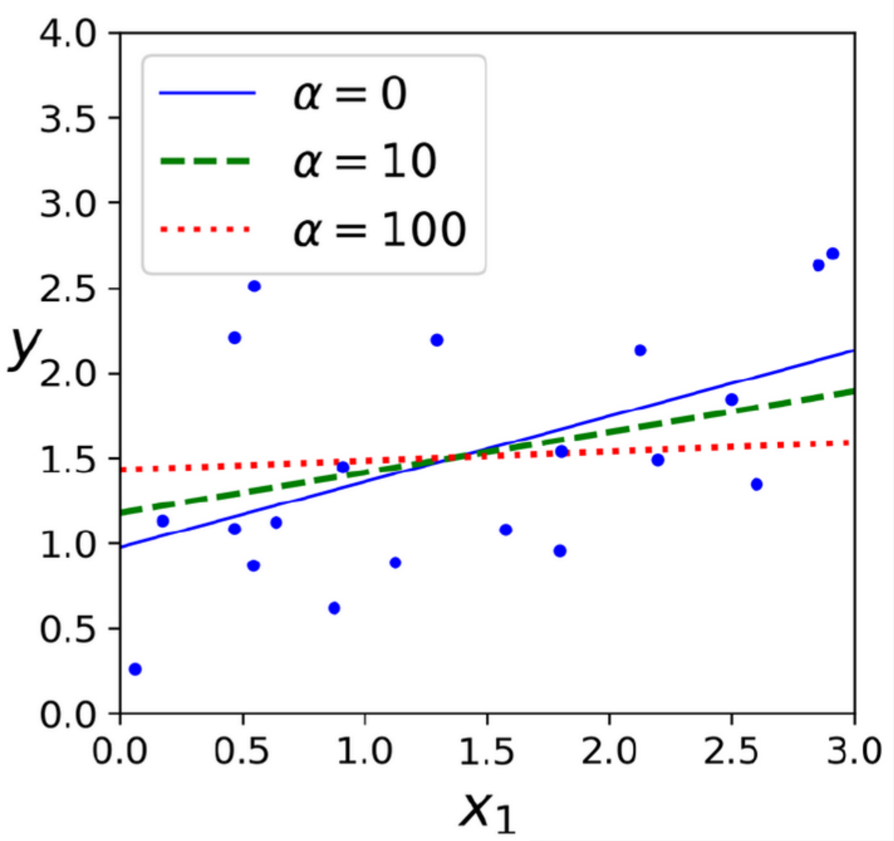
Effect: Penalises large model parameters to prevent overfitting.

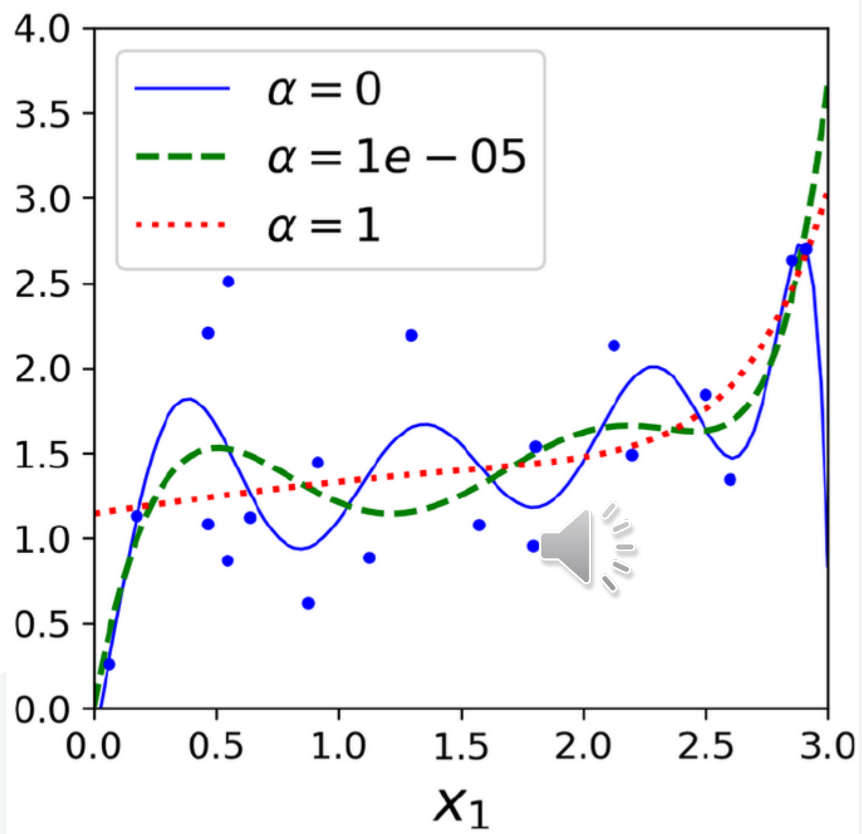
$$\overbrace{J(\boldsymbol{\theta})}^{\text{Cost}} = \overbrace{\mathrm{MSE}(\boldsymbol{\theta})}^{\text{Data term}} + \overbrace{\alpha\frac{1}{2}\Sigma_{i=1}^{n}\theta_i^{\,2}}^{\text{Regularisation term}}$$

L2 Norm

$\boldsymbol{\alpha}$: Regularisation strength or penalty term
$\boldsymbol{\theta}$: Model parameters, in this case excluding $\boldsymbol{\theta_0}$

- Scaling of data important for setting α
- Scikit-learn: `penalty` parameter "`l2`"





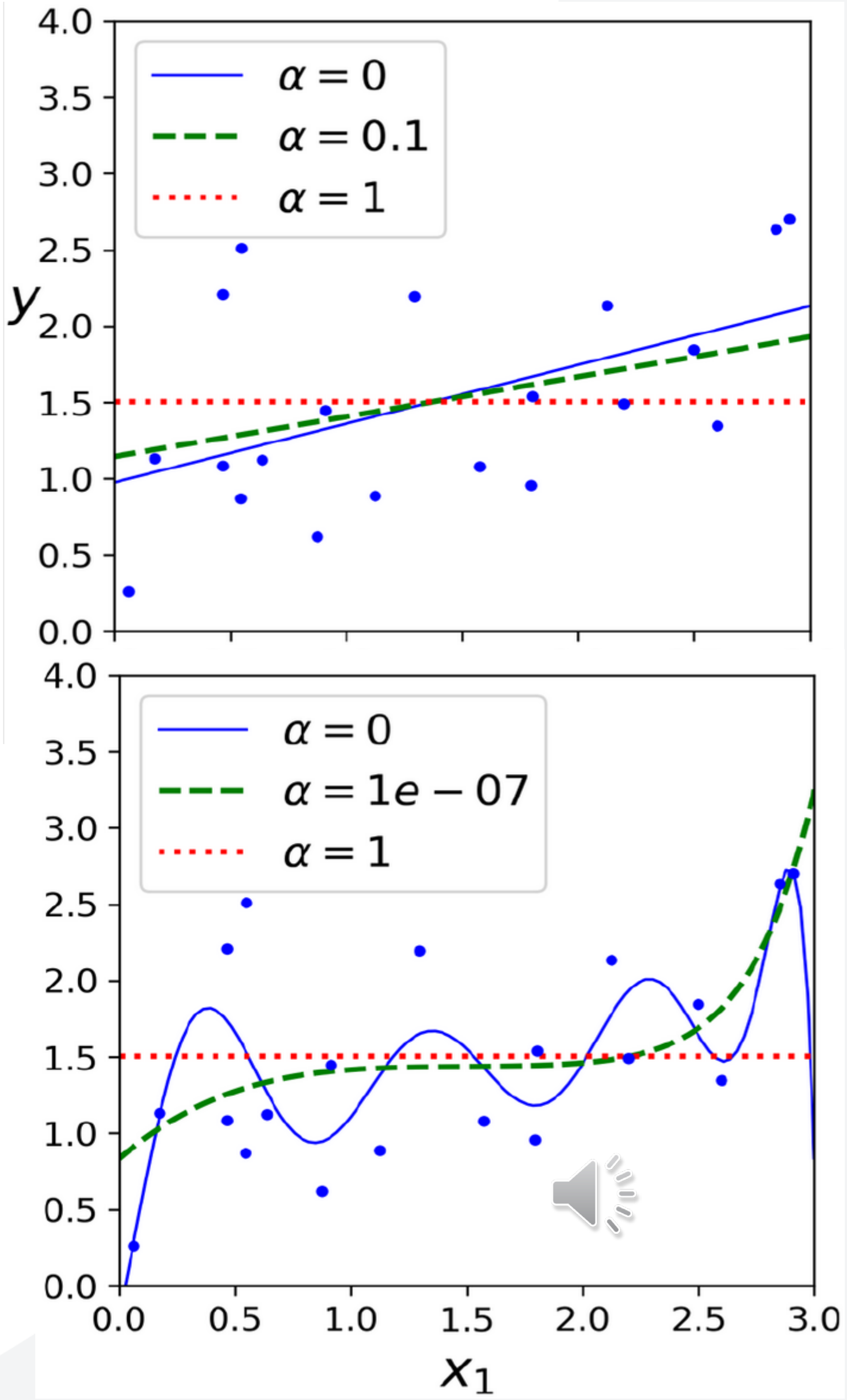
L1 (Lasso) Regularisation

- **LASSO** = **L**east **A**bsolute **S**hrinkage and **S**election **O**perator
- Effect: Penalises large model parameters to prevent overfitting. Tries to eliminate least important features (***θi = 0***)
- Sparsity: Encourages sparsity in the model by setting some coefficients = 0, aiding in feature selection/interpretability.

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \Sigma_{i=1}^{n} \left|\theta_i\right|$$

L1 Norm

- Scaling of Data: Important for setting α.
- Scikit-Learn: Use `penalty='l1'` parameter.
- Optimisation: Not differentiable at 0, but most optimisers can handle it.



Elastic Net

r = 0: L2 term dominates
r ~ 1: L1 term dominates

A mixture of L1 and L2 regularisation.

$$\underbrace{J(\boldsymbol{\theta})}_{\text{Cost}} = \underbrace{\mathrm{MSE}(\boldsymbol{\theta})}_{\text{Data term}} + \underbrace{\underbrace{r\alpha\Sigma_{i=1}^{n}|\theta_i|}_{\text{L1}} + \frac{1-r}{2}\underbrace{\alpha\Sigma_{i=1}^{n}\theta_i^{2}}_{\text{L2}}}_{\text{Regularisation term}}$$

Mixture ratio r

Feature Selection: Can eliminate less important features.

Handling Collinearity: Often better than Lasso (L1) alone.

Penalty Term: Combination of absolute and squared coefficients.

Model Complexity: Balances between sparsity and retaining all features.

Which answer is correct?

What does L1 regularization (Lasso) encourage in a model?

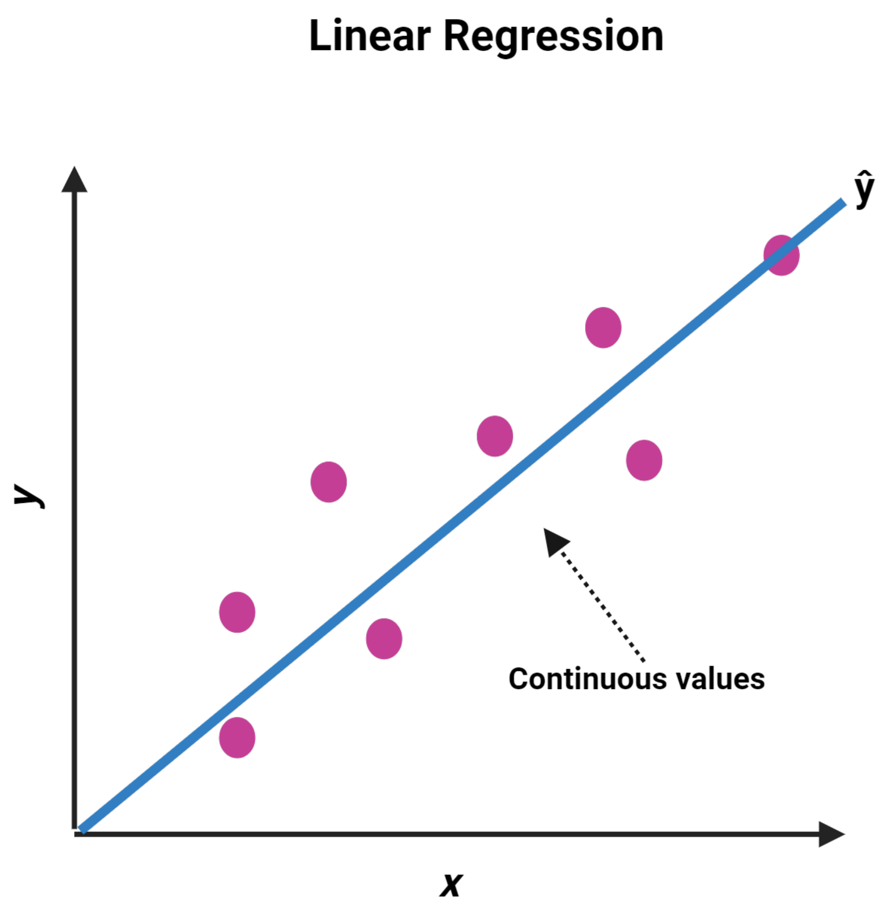
- Inclusion of more features
- Large coefficients
- Sparsity by setting some coefficients to exactly zero
- Higher training error

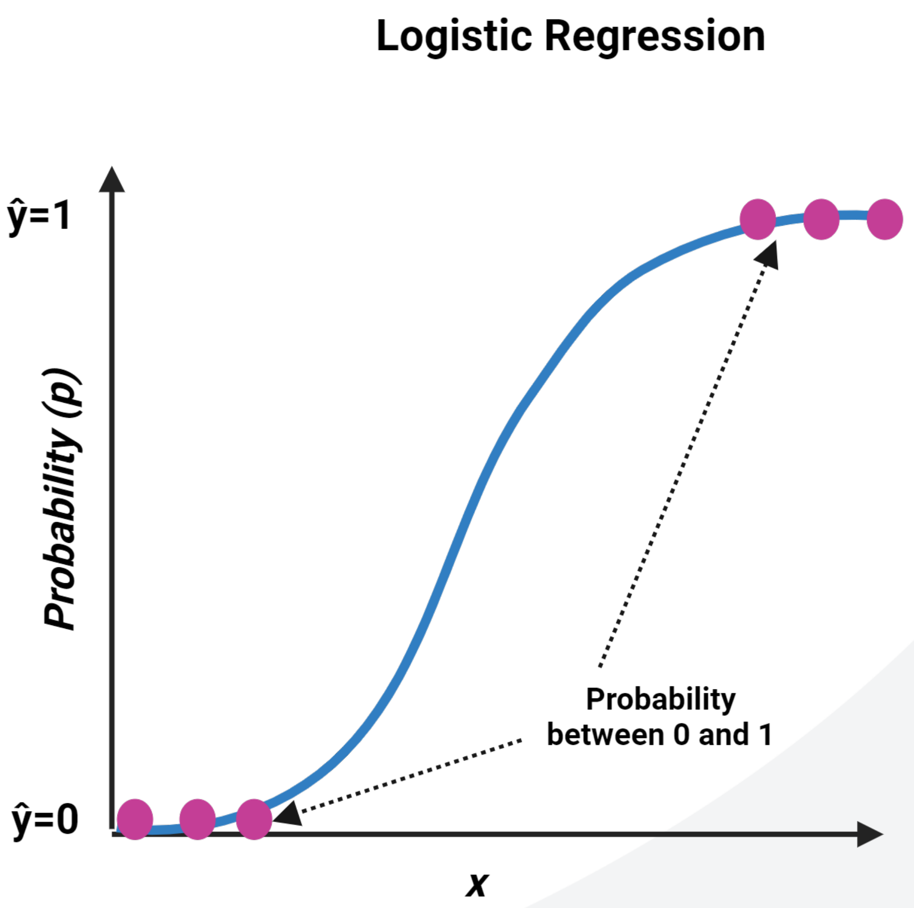
Logistic Regression

Logistic Regression

- **Logistic regression:** a statistical method used to estimate the probability of an outcome (i.e. an event occurring or not occurring) based on one or more independent variables.

- Unlike linear regression, which outputs continuous values, logistic regression outputs a **probability** value **between 0 and 1**.





THE UNIVERSITY of ADELAIDE
150 YEARS

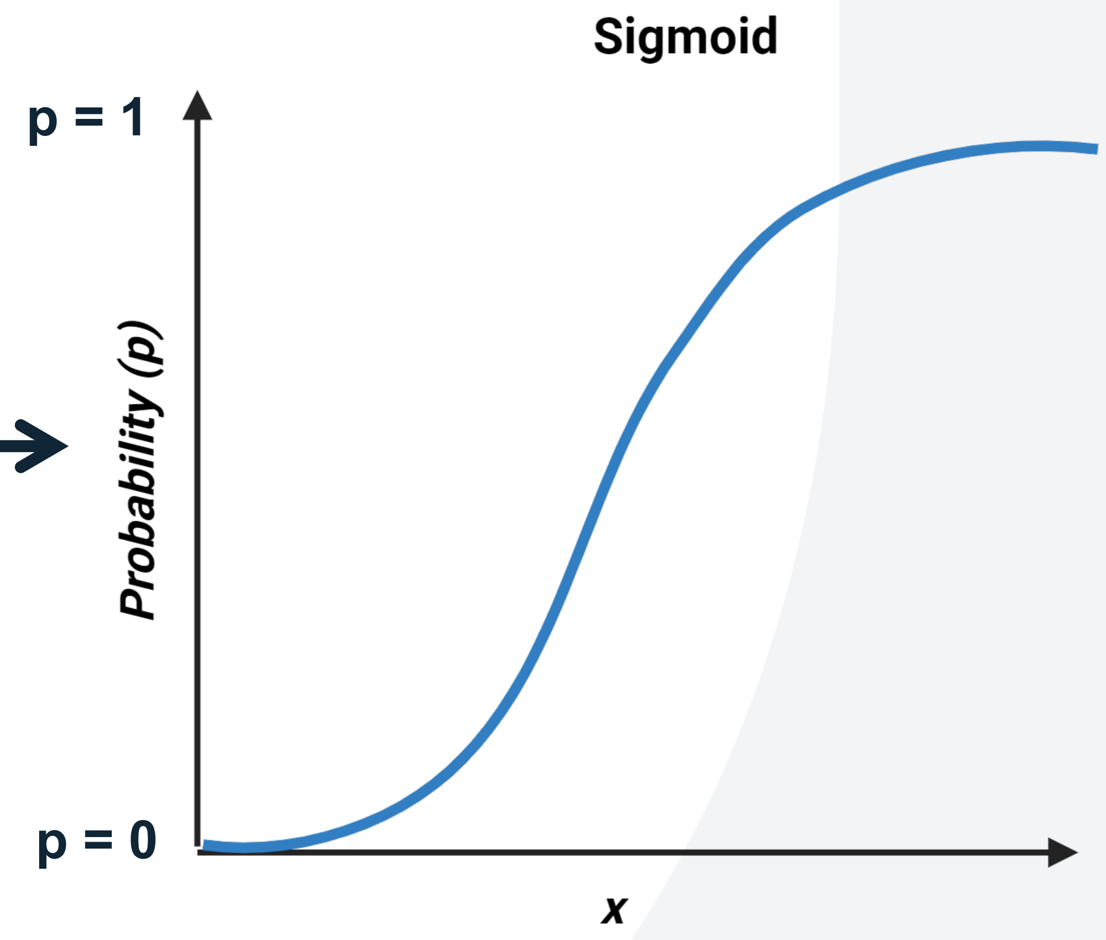
Logistic Regression

- Combines a linear model with a logistic function (sigmoid)

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \boldsymbol{x}^T \cdot \boldsymbol{\theta}$$ **Linear model**

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$ **Logistic function (sigmoid)**

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\mathbf{x}^\intercal \boldsymbol{\theta})$$ **Logistic model (regression)**



- Logistic function (sigmoid, $\sigma(t)$) maps the linear combination of input features ($x^T.\theta$; expressed as log-odds) to a probability value (p) between 0 and 1.

Logistic Regression

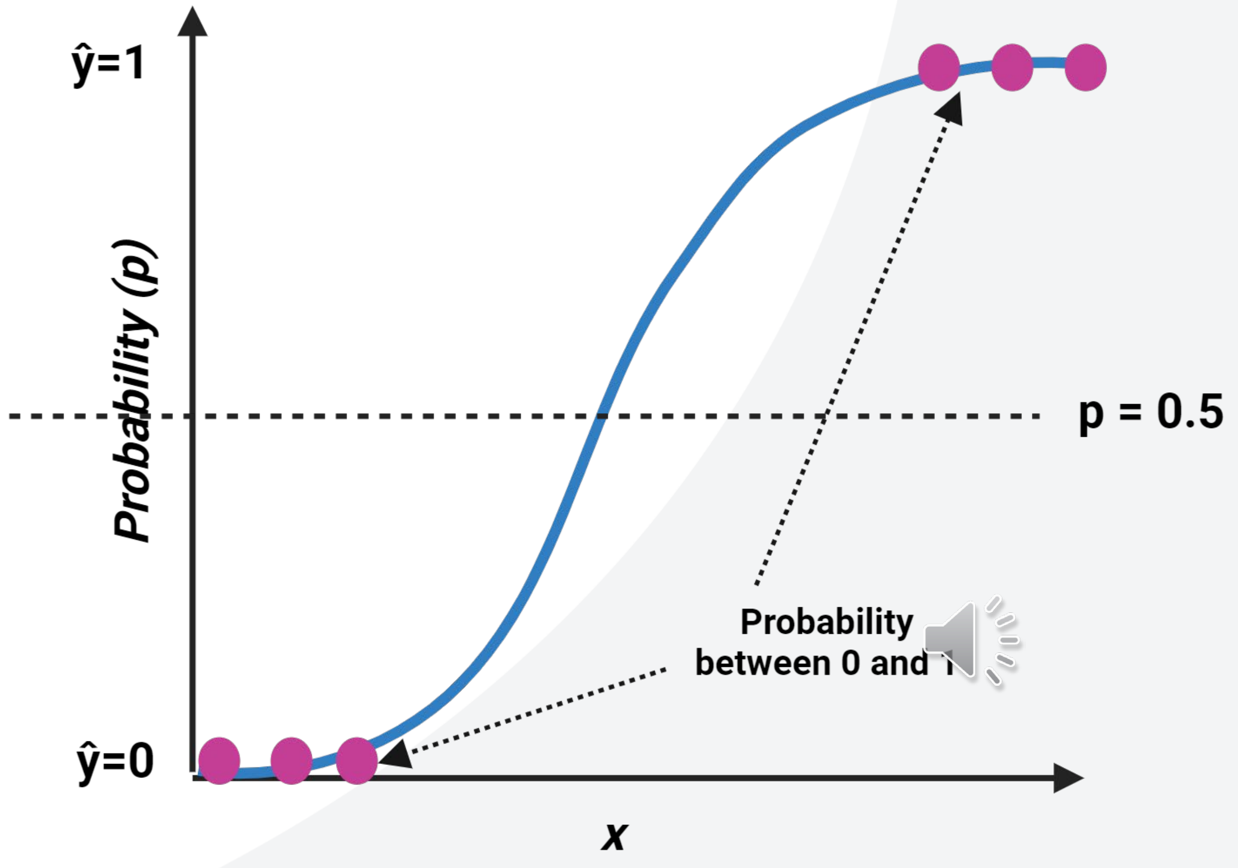
- Combines a linear model with a logistic function (sigmoid)

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \boldsymbol{x}^T \cdot \boldsymbol{\theta}$$ **Linear model**

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$ **Logistic function (sigmoid)**

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\mathbf{x}^\top \boldsymbol{\theta})$$ **Logistic model (regression)**

- Negative inputs = σ(t) < 0.5
- Positive inputs = σ(t) > 0.5
- Gradual probability change instead of a hard boundary.



- Used for binary classification ML problems where the output is a probability that the given input point belongs to a certain class:

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases}$$

Apply threshold = binary classifier

Logit Regression = Logistic Regression

- **Logit regression:** Another name for logistic regression. The term "logit" refers to the log-odds function, which is the **inverse** of the logistic function.

$$logit(p) = \log\left(\frac{p}{1-p}\right)$$ **Log-odds function**

- Transforms probabilities into log-odds (converse of logistic regression).

- Same as logistic regression = process of using the logistic function (sigmoid) to predict binary outcomes.

THE UNIVERSITY of ADELAIDE

150 YEARS

Training a logistic model

- To train a logistic model, we need to have an appropriate cost/loss function to optimise.
- Can be achieved by considering probabilities:
  - Try to maximise probability of observing targets y given data x:

Probability of observing binary data y if real probability is p:

$$P(y) = \begin{cases} p & \text{if} \quad y = 1 \\ 1 - p & \text{if} \quad y = 0 \end{cases} \quad \text{or} \quad P(y) = p^{\,y}\,(1 - p)^{1-y}$$

For multiple events (set of observations), multiply the probabilities for each event:

$$P(\mathbf{y}) = \prod_i \hat{p}_i^{\,y_i}\,(1 - \hat{p}_i)^{1-y_i}$$

Take the log to simplify:

$$\log\left(P(\mathbf{y})\right) = \sum_i y_i \log\left(\hat{p}_i\right) + (1 - y_i)\log\left(1 - \hat{p}_i\right)$$

Training a logistic model

$$\log\left(P(\mathbf{y})\right) = \sum_i y_i \log\left(\hat{p}_i\right) + (1 - y_i)\log\left(1 - \hat{p}_i\right)$$

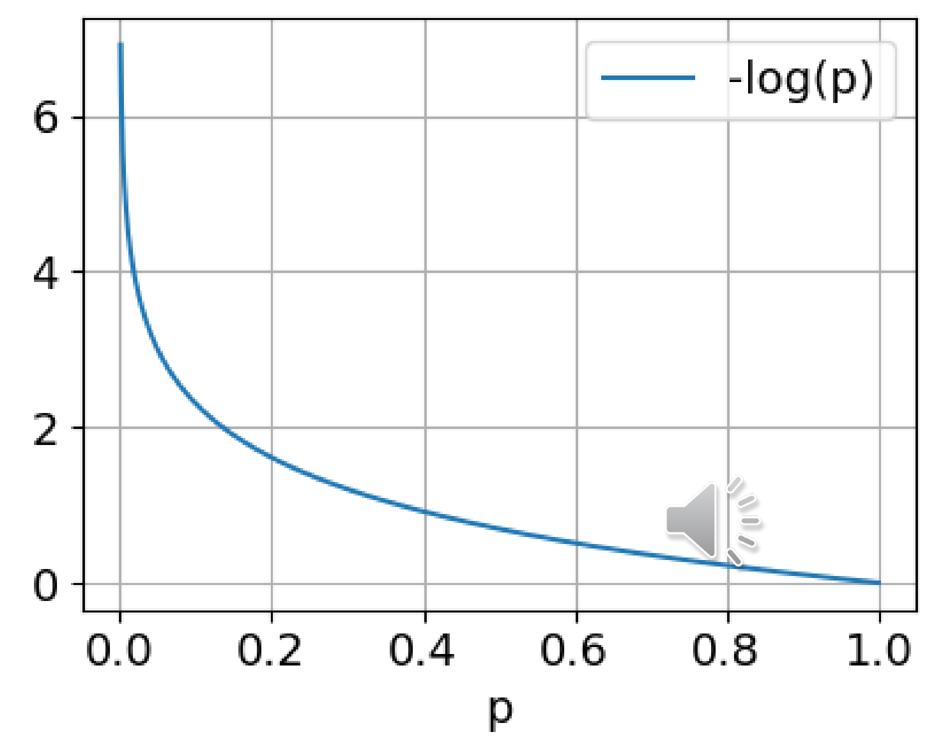
Our model predicts the estimated probability output by the regression

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma\left(\mathbf{x}^\top \boldsymbol{\theta}\right)$$

To train the model, we minimise the *negative* log probability (cost/loss function), also called log loss or binary cross-entropy:

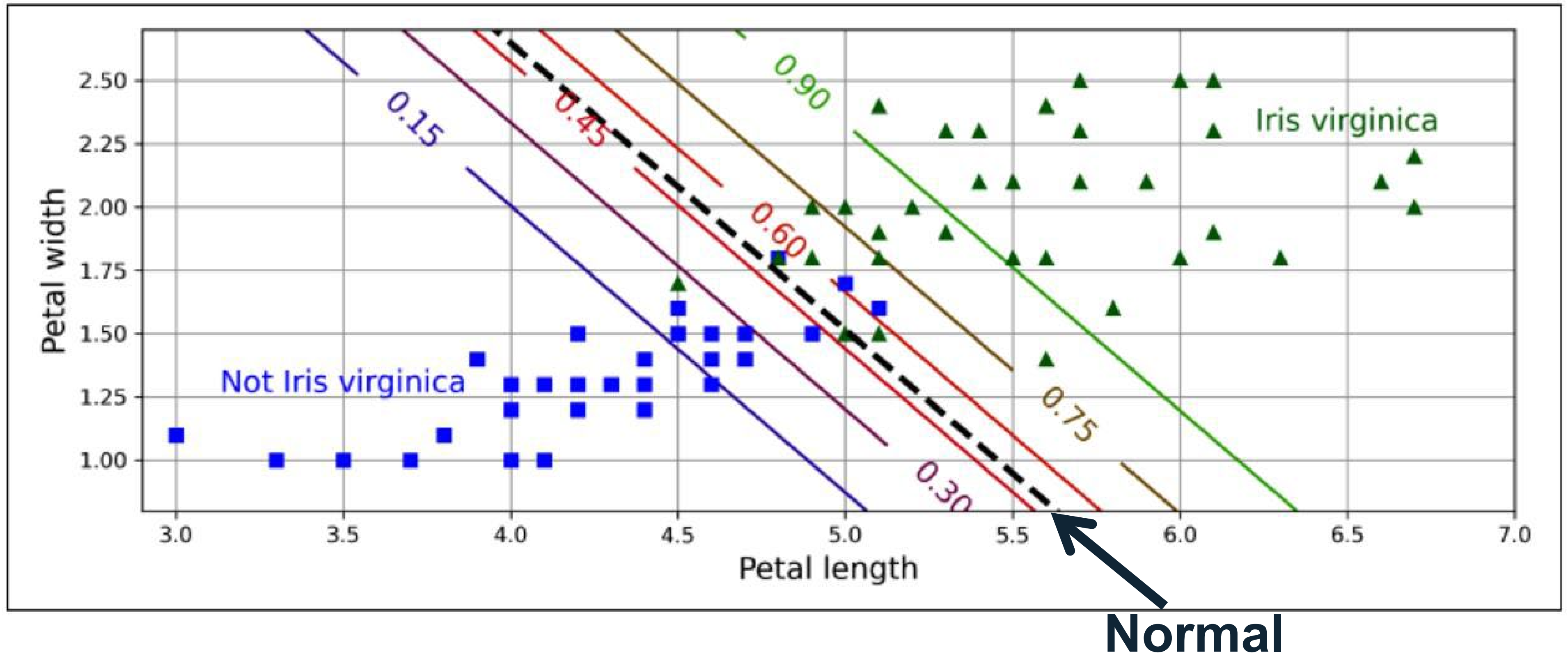
$$J(\boldsymbol{\theta}) = -\frac{1}{m}\Sigma_{i=1}^{m}\left[y^{(i)}log\left(\hat{p}^{(i)}\right) + \left(1 - y^{(i)}\right)log\left(1 - \hat{p}^{(i)}\right)\right]$$

- Advantage: Cost function is convex, which is good for optimisation as it has no local minima



Logistic regression: Decision boundary

- **Decision boundary:** Helps decide which class a new data point belongs to, based on its features.
  - In logistic regression, the decision boundary is linear/monotonic, meaning it creates a straight line (in 2D) or a flat plane (in 3D) to separate classes.
  - Normal to the decision boundary is direction in which the probability of being in a specific class increases most (due to s-shaped sigmoid).

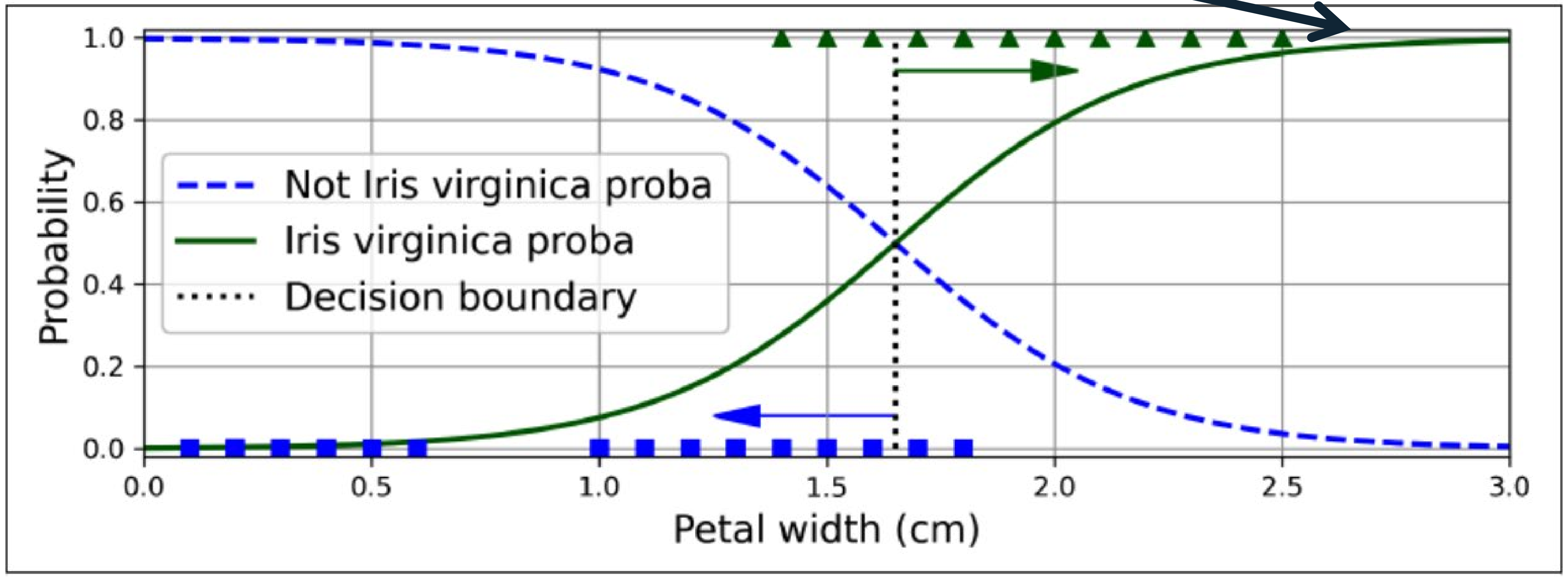


- **Near the boundary:** change in probability is more gradual.
- **Further from the boundary:** probability approaches 0 or 1 more quickly.

from GeronHands on Machine Learning

Logistic regression: Decision boundary

- **Probability prediction:** Graph below illustrates how the predicted probability changes with petal width.
- The intersection at p=0.5 is the decision boundary.
- This graph shows the rate of change near the decision boundary is the largest (probabilities plateau as you move further away).



from GeronHands on Machine Learning

Logistic regression with scikit-learn

Can have ways to predict both class and probabilities:

```
from sklearn.linear_model import LogisticRegression

log_model = LogisticRegression()
log_model.fit(X,y)
log_model.predict(X_new) # predict class
log_model.predict_proba(X_new) # predict probabilities
```

Softmax Regression

- Methodology so far explains the use of logistic regression for binary classification, but what if we have a multi-class classification problem?
- **Softmax:** Extends logistic regression framework to multiple classes. Also known as Multinomial Logistic Regression.

For each class k, we fit a linear model:

$$s_k(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\theta}^{(k)}$$

- $\boldsymbol{s_k(x)}$: 'logits'
- $\boldsymbol{x}$: Input feature vector
- $\boldsymbol{\theta^{(k)}}$: Model parameters for class $\boldsymbol{k}$

Softmax function: converts linear model outputs ('logits') for each class into probabilities:

$$\hat{p}_k = \sigma(\mathbf{s}(\mathbf{x}))_k = \frac{\exp\left(s_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(s_j(\mathbf{x})\right)}$$

- $\boldsymbol{exp}$: Exponentiation makes all logits positive
- $\boldsymbol{Normalisation}$: Sum of all probabilities $\sum_k \hat{p}_k = 1$
- $\boldsymbol{Range}$: Each probability $\hat{p}_k$ is between 0 and 1
- $\boldsymbol{Monotonic}$: Larger logits lead to higher $\hat{p}_k$

Softmax Regression

Cost function = cross entropy, measures how well the predicted probabilities match the actual labels:

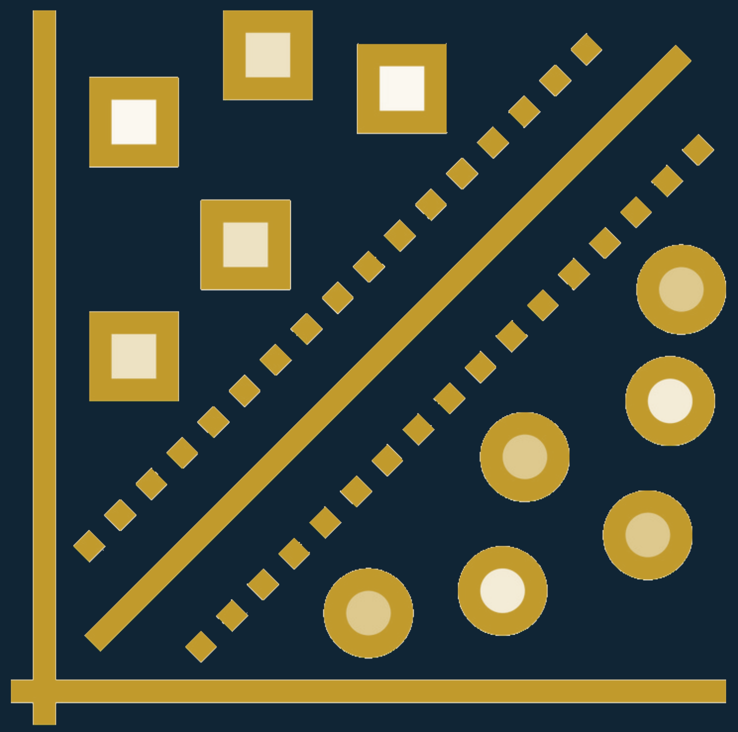
$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$$

Aim: Minimise this cost function ($J(\Theta)$).

- **Interpretation:** $y_k^{(i)}$ is 1 if the i-th sample belongs to class k, 0 otherwise.
- This cost function is differentiable, so we can use Gradient Descent as an optimisation scheme.

Comparison to logistic regression:

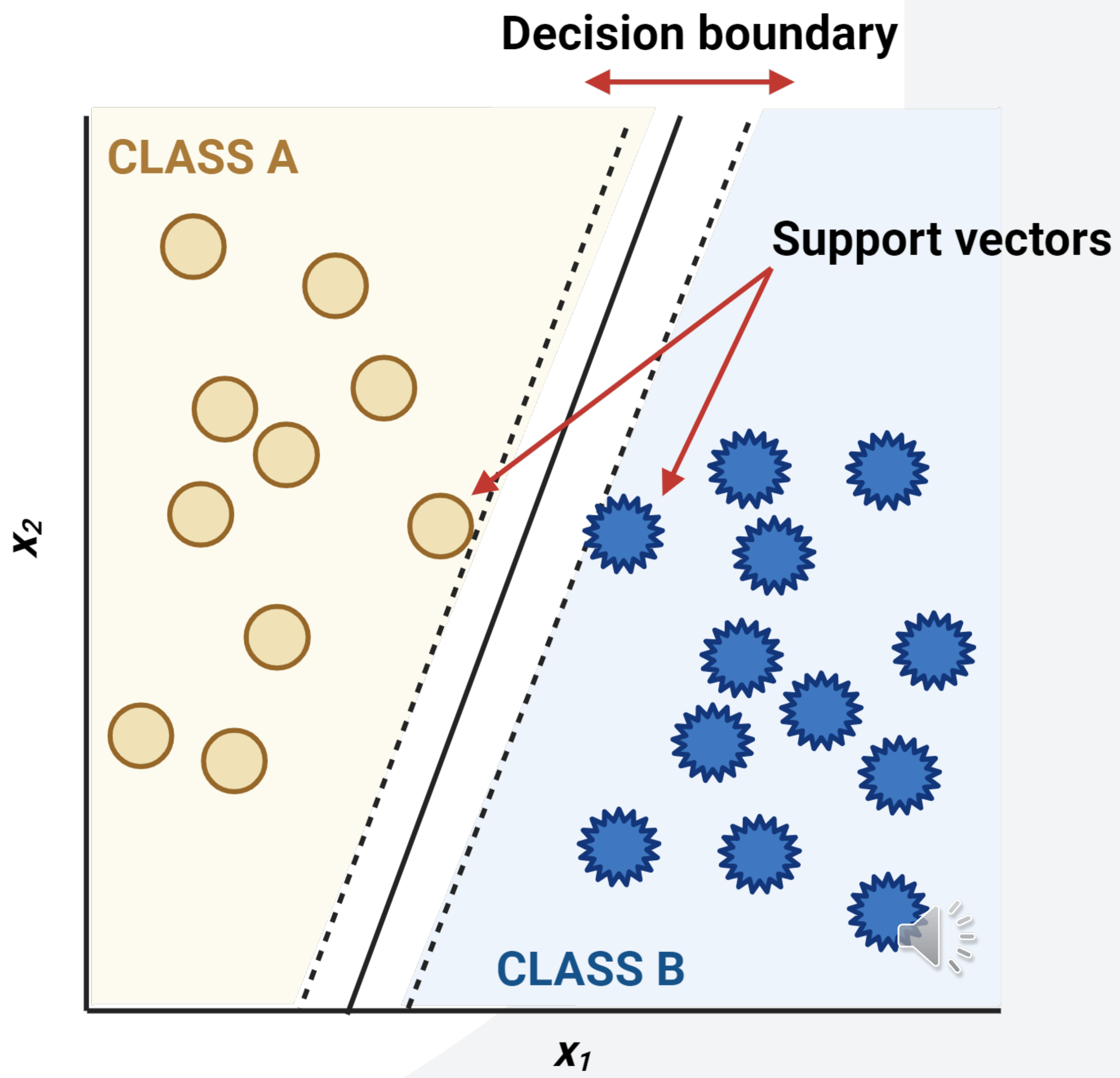
- Both fit a linear model to the input features an apply a non-linear transformation to produce probabilities (logistic regression - sigmoid function; Softmax – softmax function).
- Logistic Regression maps input features to a single probability value (binary classification); Softmax Regression maps input features to a probability distribution over multiple classes (multi-class classification).
- Both use some form of cross-entropy loss as cost function.



Support Vector Machines (SVM)

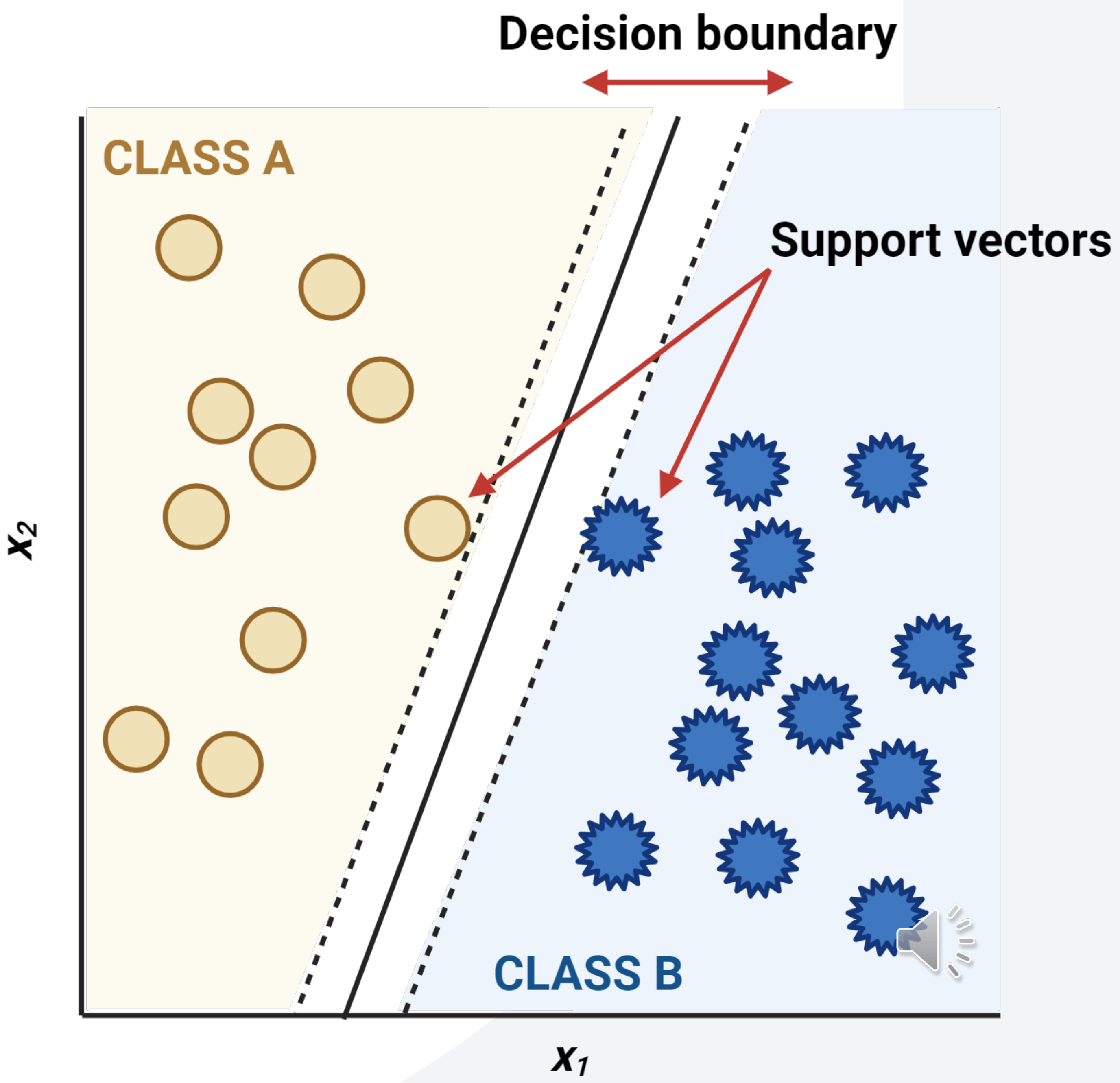
Support Vector Machines (SVMs)

- **Support Vector Machines (SVMs):** Type of supervised machine learning algorithm used for classification and regression tasks.
- Particularly effective for high-dimensional spaces and known for their ability to create clear boundaries between classes.
- **Key Idea:** Find the *widest possible decision boundary* between two classes while minimising errors (boundary violations).
- SVMs are based on distances between data points and the boundary.



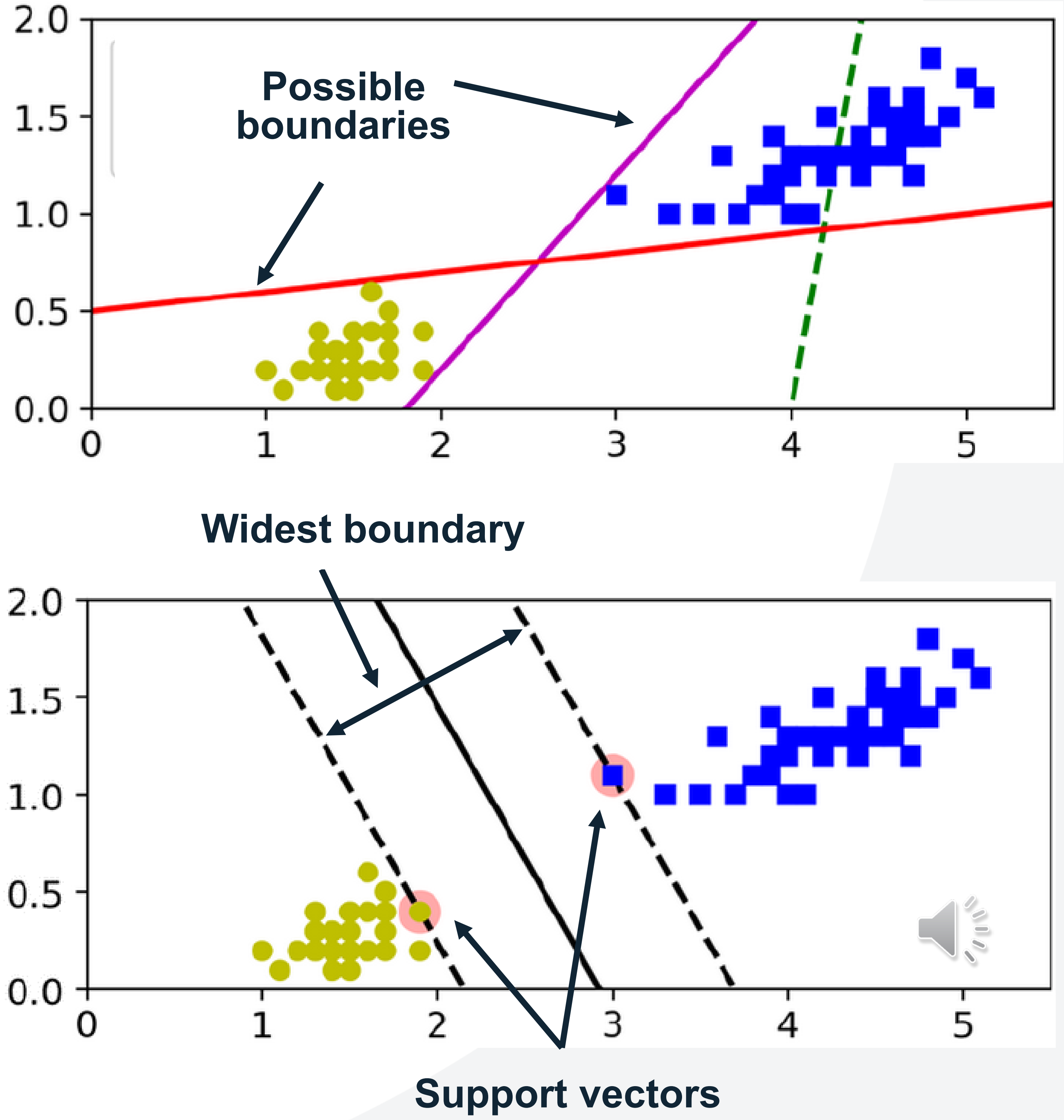
Support Vector Machines (SVM)

- **Support Vectors:** Boundary is defined by a few key data points called support vectors → Can identify boundary just from this small subset of samples.
- Good for complex boundaries, and small to medium datasets (large datasets can be slow).
- Boundary shape can be linear or non-linear.
- Uses a mathematical technique to called the ‘kernel trick’ handle non-linear boundaries efficiently.



Linear SVM – Hard Margin

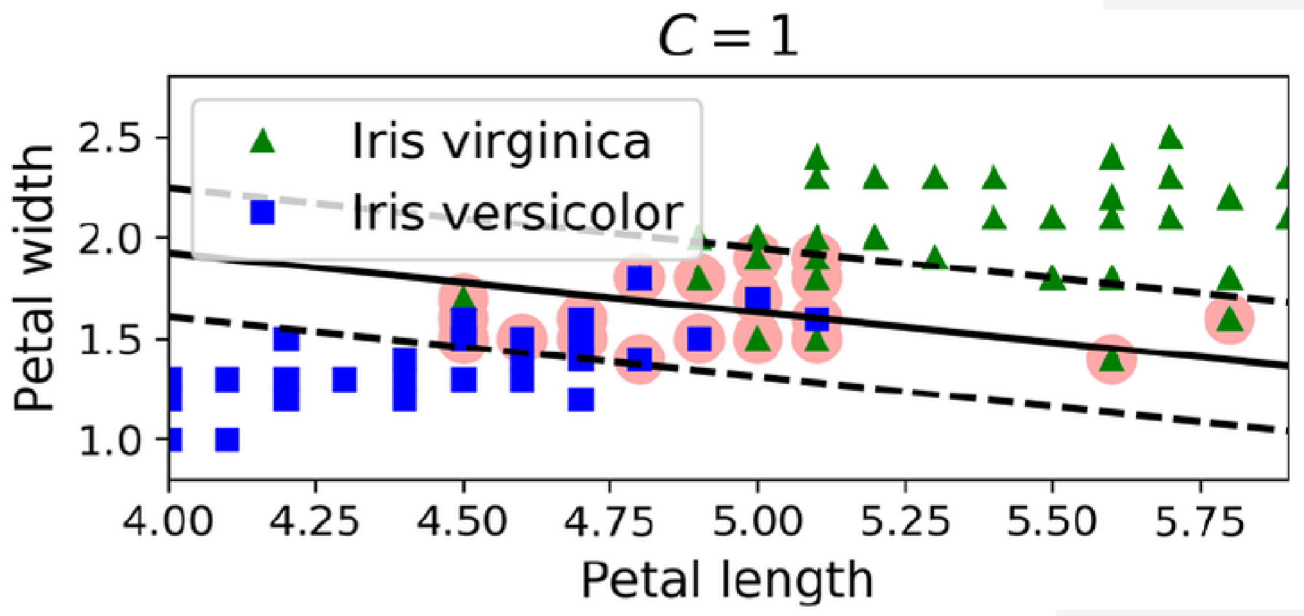
- Decision boundary is straight line (2D) or linear plane/hyperplane (higher dimensions).
- **Zero Boundary Violations:** This means no data points from one class are allowed to fall on the other side of the boundary.
- **Sensitive to Scaling:** Because SVMs use distances, it's crucial that all input features are on a similar scale. If features have different scales, the SVM might not work correctly.



Linear SVM – Soft Margin

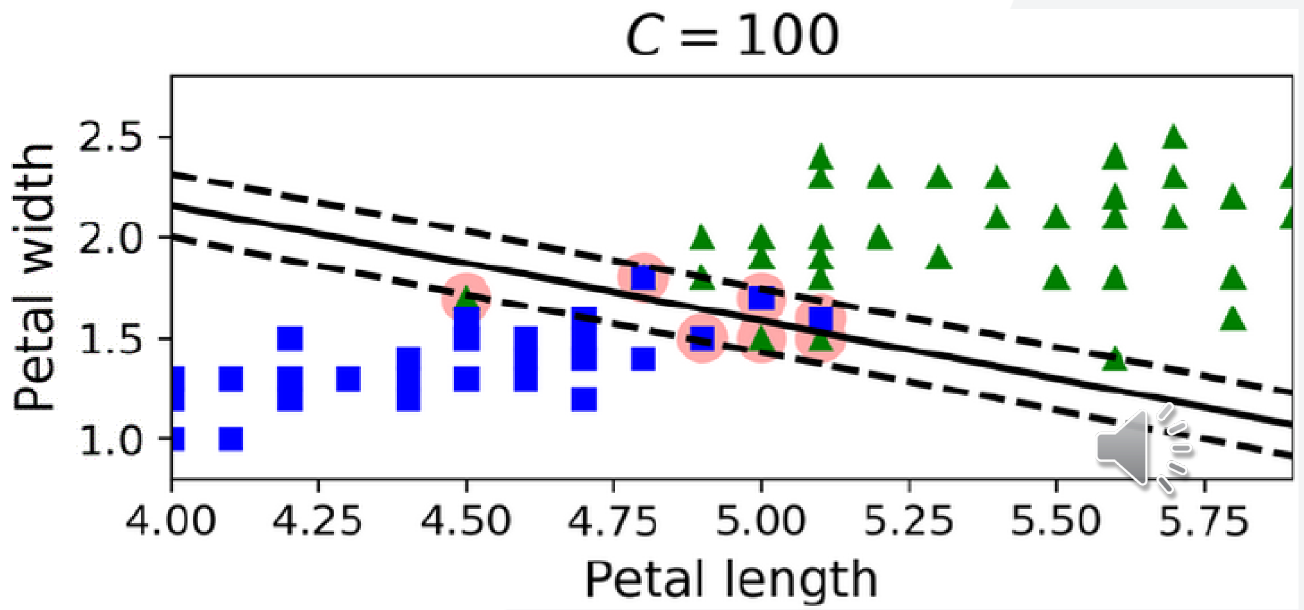
- **Allow Some Boundary Violations:** Allow some data points to be on the wrong side of the boundary to handle cases where classes are not perfectly separable.
- Balance between having a wider boundary and allowing some violations (misclassified points).
- The parameter C determines how much we allow boundary violations.

Wider boundary, more violations



- **Small C (Near Zero):** Focusing on maximising the boundary width with minimal importance given to violations.
- **Large C:** Achieves a narrower boundary and makes the model stricter about violations.

Less violations, narrower boundary

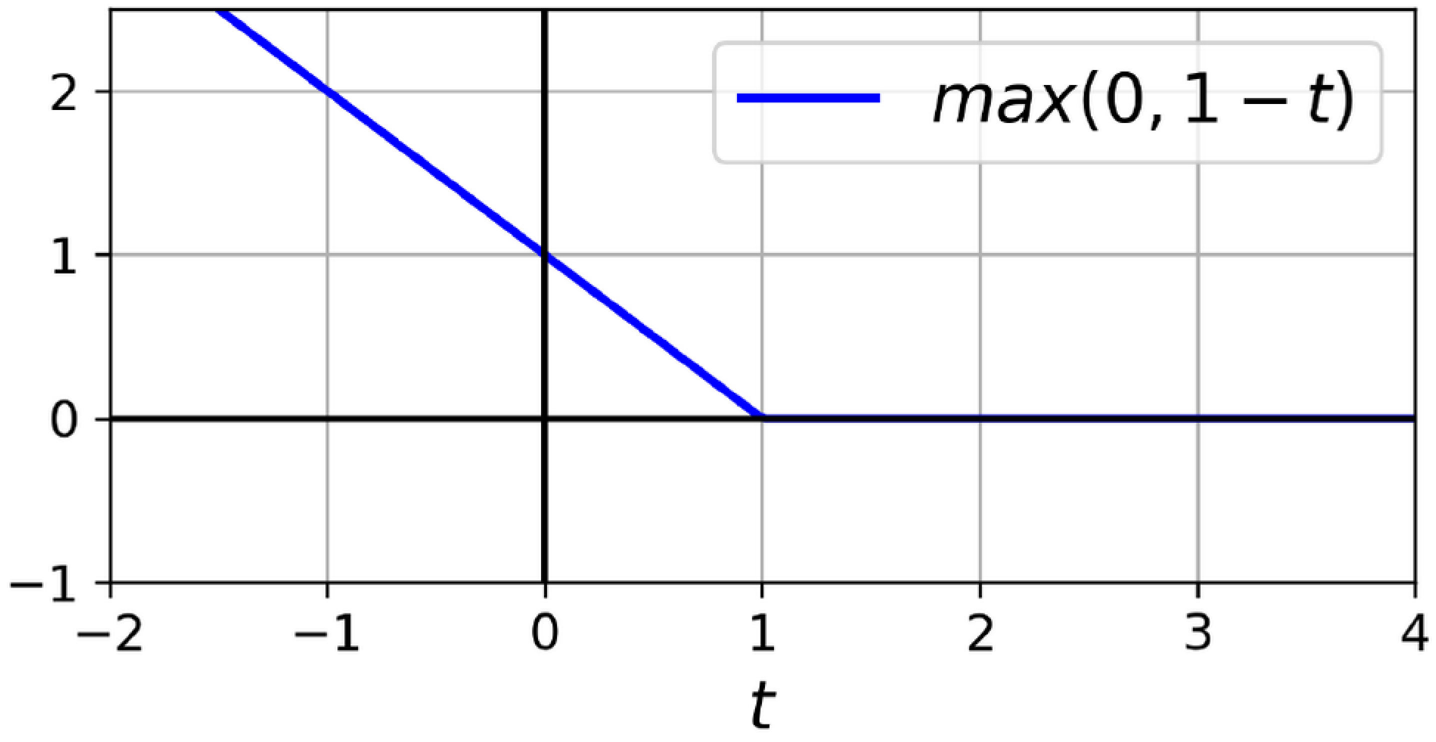


Hinge loss function & SVM

- Can write the SVM loss using the Hinge loss function:

$$\max(0,1-t)$$

where t is a value that combines the model's prediction and the actual class label.



When $t \geq 1$:
Model's prediction is correct and confident.
Hinge Loss = 0: No penalty.
When $0<t<1$:
Model's prediction is correct but not confident enough.
Hinge Loss > 0: There's a small penalty. Smaller penalty the closer t is to 1.
When $t \leq 0$:
Model's prediction is wrong.
Hinge Loss > 0: There's a significant penalty. Larger the penalty the further t is from 0.

- You will see this as an option in other machine learning methods.

Non-Linear Boundaries

- **Problem:** Many decision boundaries are not linear

- **Idea:** Add non-linear functions of the input data as extra features and use a linear boundary in this higher dimensional space
  - Longer computation time

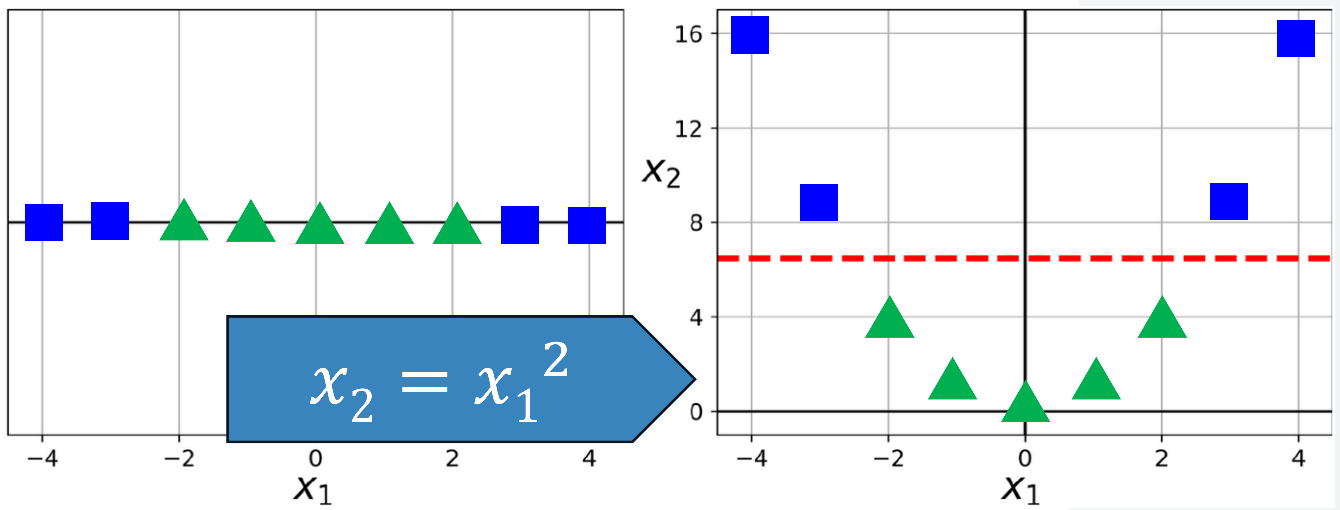
i.e. Gaussian Radial Basis Function (RBF)

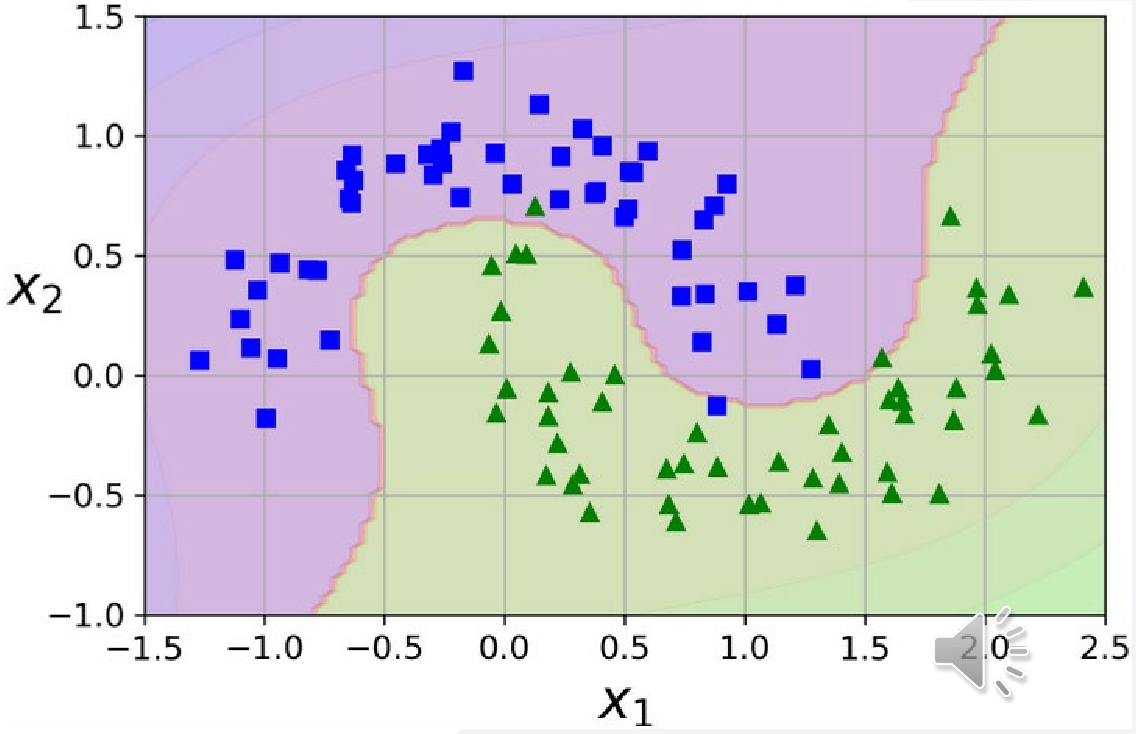
$$\phi_\gamma(\mathbf{x}, \ell) = \exp\left(-\gamma \overbrace{\| \mathbf{x} - \ell \|}^{\text{Distance between samples}}{}^2\right)$$

New non-linear feature

Hyperparameter Gamma

Not separable ☹ **Separable ☺**

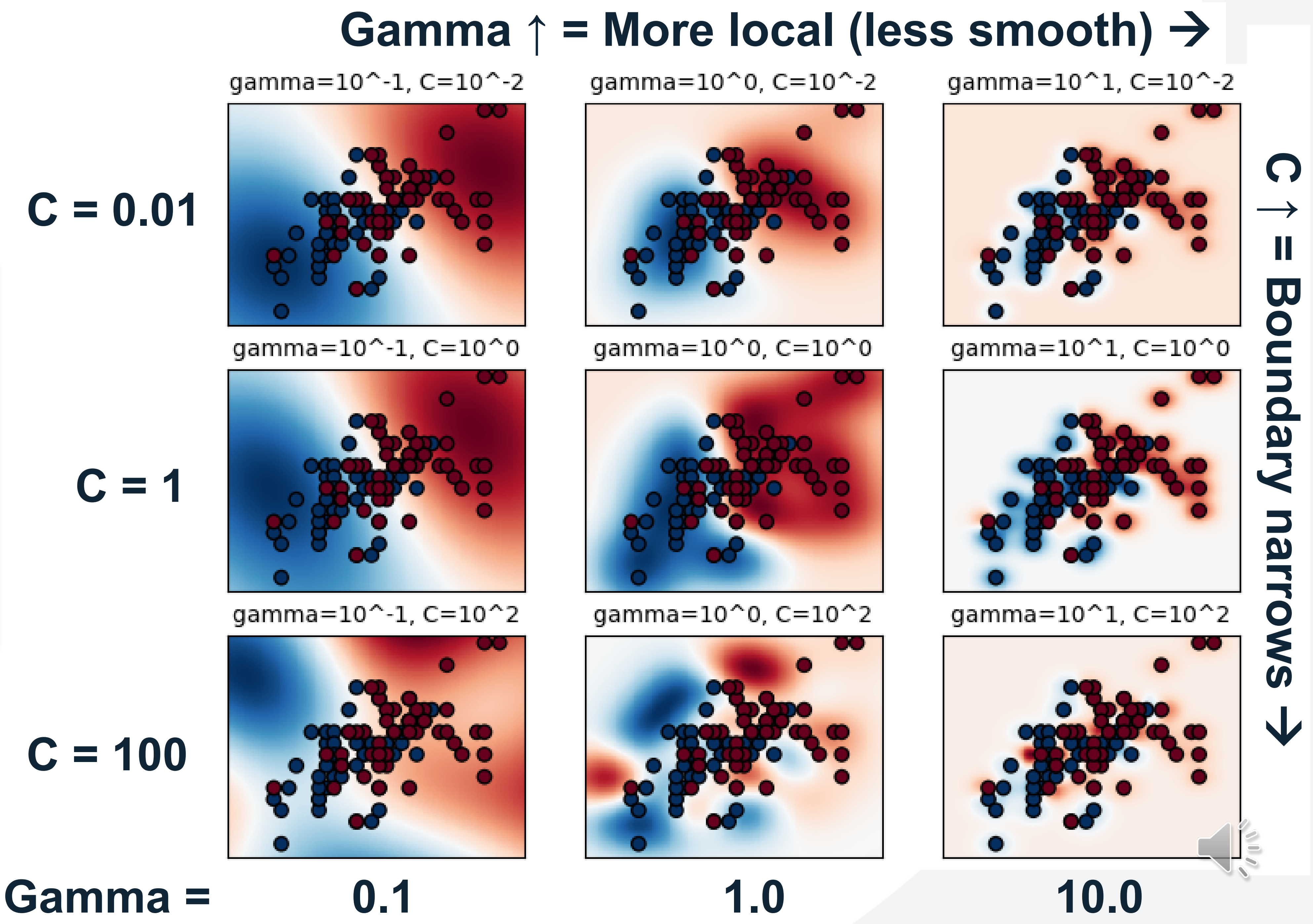




From Geron, Hands On ML

SVM Hyperparameters: C & Gamma

Example:
- 2 classes
- 50 samples each
- 2 (original) features
- Gaussian RBF kernel



Images: © 2007 -2019, scikit-learn developers (BSD License), rearranged

Kernel ‘trick’

- When we try to make our classifier more powerful, we sometimes add many extra features.
  - Can make model slow because of the large number of calculations needed.

- **Kernel trick:** Instead of calculating these extra features directly, we can use a clever shortcut - the SVM only needs to know the dot product between pairs of samples in the high-dimensional feature space, not the actual high-dimensional features themselves.

Kernel ‘trick’

- **How it Works:**
  - Dot Products: Measure similarity between pairs of samples in a high-dimensional space.
  - Kernel Function: $K(a,b)$ specifies the dot product calculation.
  - Efficiency: Store dot products in a $N_{\text{sample}} \times N_{\text{sample}}$ matrix. Avoids need for explicit extra feature computation.

Common kernels

Linear: $K(a,b) = a^T b$ ← **Linear classifier**

Polynomial: $K(a,b) = (\gamma a^T b + r)^d$ ← **Use polynomial feature combinations without having to compute them**

Gaussian RBF: $K(a,b) = \exp(-\gamma||a - b^2)$ ← **Use similarity to support vectors with Gaussian drop off**

Sigmoid: $K(a,b) = \tanh(\gamma a^T b + r)$

Summary

- **Regularisation continued**

- **Logistic regression**
  - Probability-based method
  - Uses sigmoid (logistic) function to get outputs in [0,1] range
  - Sigmoid & Softmax functions used in many deep learning models

- **Support Vector Machines**
  - Fit the widest possible boundary between two classes while limiting boundary violations
  - Create extra non-linear features → high dimensional space where they can be separated with a linear boundary in this space
  - Best to try simple kernels first, compare different kernels
  - Hyperparameters also need to be optimised
  - Data needs to be scaled uniformly (internally it is distance-based)

THE UNIVERSITY of ADELAIDE 150 YEARS

Questions?

dhani.dharmaprani@adelaide.edu.au