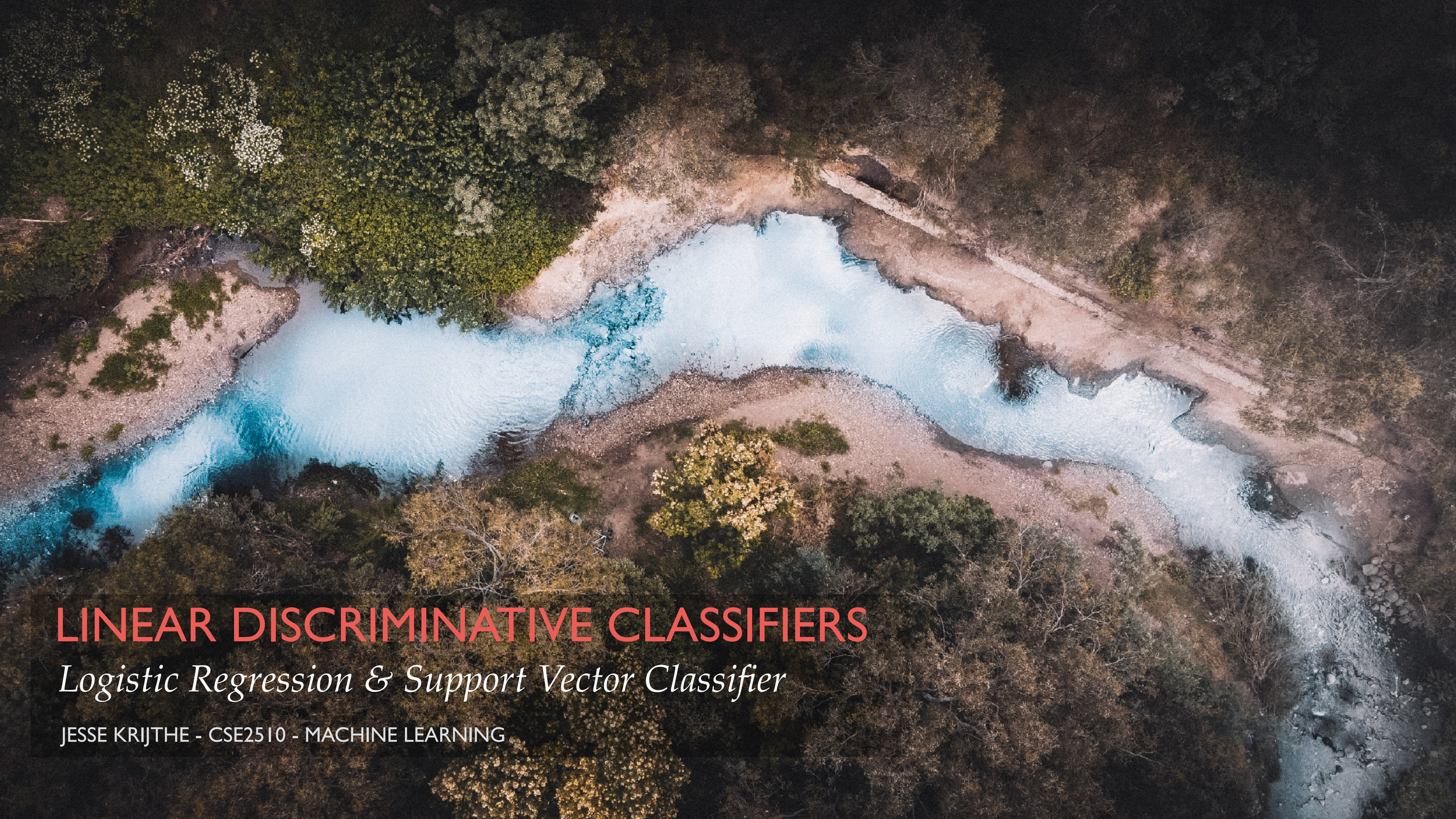


# LINEAR DISCRIMINATIVE CLASSIFIERS

*Logistic Regression & Support Vector Classifier*

JESSE KRIJTHE - CSE2510 - MACHINE LEARNING

# *Learning Goals & Reading*

## LEARNING GOALS

After this week you will be able to

- Distinguish between generative and discriminative models
- Reason about linear regression models and linear classifiers
- Explain what hypothesis and cost functions are
- Implement gradient descent to train a given linear model
- **Derive and implement logistic regression from its loss function**
- **Identify the principles behind support vector classifiers**
- **Describe some approaches to multi-class classification and their problems**

## LITERATURE

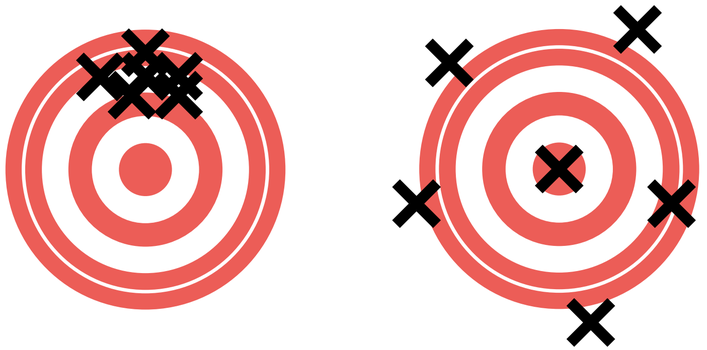
Bishop's "Pattern Recognition and Machine Learning"

- Chapter 3 up to and including 3.1.3
- 5.2.4
- **4.3 up to and including 4.3.2**
- **4.3.4**
- **7.1 up to and including equation (7.7)**
- **7.1.1 up to and including equation (7.22)**
- **4.1.2**

# *Bias, Bias, Bias*

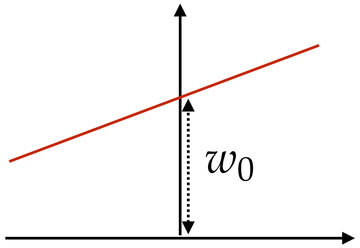
**Bias (of an estimator)**

when we train a model many times on different datasets that come from the same problem, how close is the method on average to the true solution.



**Bias (a parameter)**

name used to indicate a parameter in a model that adds a constant value to a function




**Bias (in behaviour or in data)**

(usually unwanted) inclination for or against something

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions
2. Define a cost (loss) function to measure the "quality" of each hypothesis
3. Measure the average cost on the training data
4. Find the function that minimises this cost

# *Linear Regression*

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)^2$$

*Cost function*: Squared loss

*Hypothesis class*: all linear models

*Minimization procedure:* gradient descent (or, in practice: closed form solution)

# *Gradient Descent Procedure*

Given an objective function *J*, learning rate (step size) $\alpha$, and number of iterations T.

1. Pick a starting value (for instance, random) for $\mathbf{w}$ and $w_0$
2. For T iterations, for all j = *0,1,…,D*, do:

$$w_j^{t+1} = w_j^t - \alpha \left. \frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} \right|_{\mathbf{w}, w_0 = \mathbf{w}^t, w_0^t}$$

*We looked at regression, but what about classification?*

# LOGISTIC REGRESSION

*Regression???*

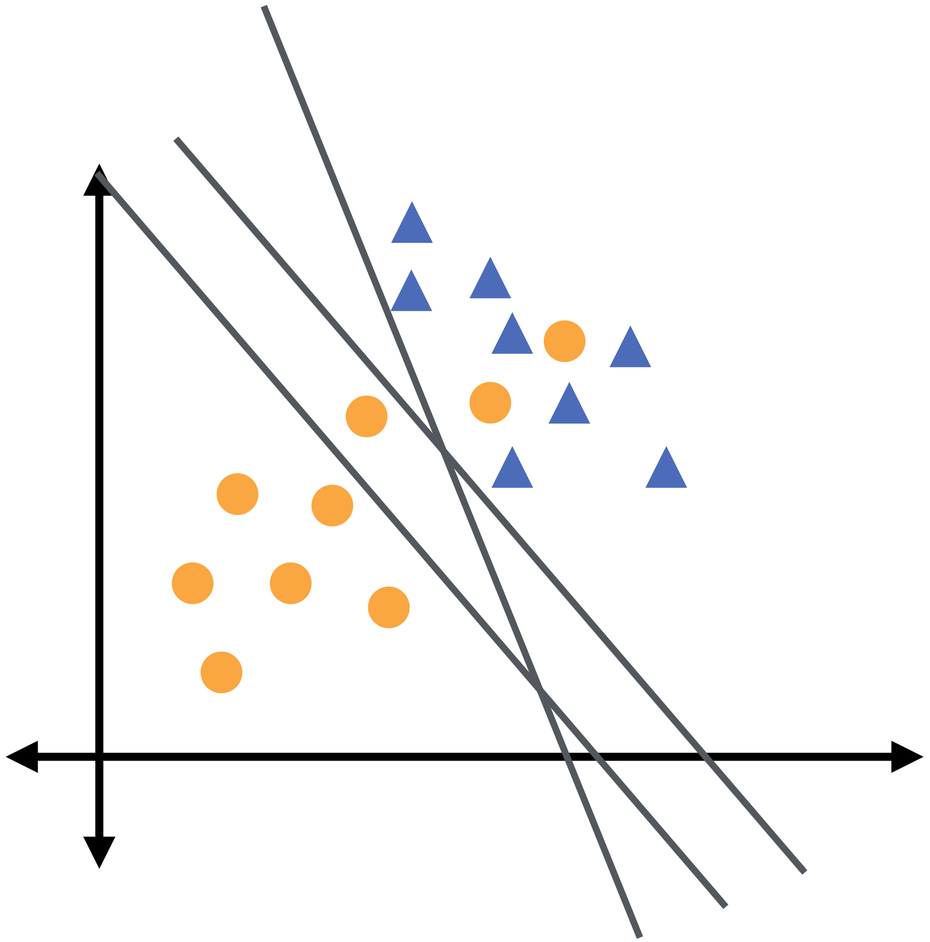
# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions
2. Define a cost (loss) function to measure the "quality" of each hypothesis
3. Measure the average cost on the training data
4. Find the function that minimises this cost

# *Classification Losses*

*It's obvious! Just count the number of mistakes!*



Unfortunately, it's not so simple:

- Accuracy is difficult to optimize (imagine standing on a flat surface, where should you go?)
- Multiple good solutions, which one do we pick?

Two reasons to use some other loss function:

1. There are other objectives in classification: different weight for different mistakes, accuracy of probabilities, correct "ordering" of the objects, etc.
2. Find a good solution for accuracy, by using some other "surrogate" loss

# *Classification Losses: Logistic Regression*

**Different Goal:**

Find a function that accurately approximates the probability of a label:

$$h(X) \approx P(Y = 1|X)$$

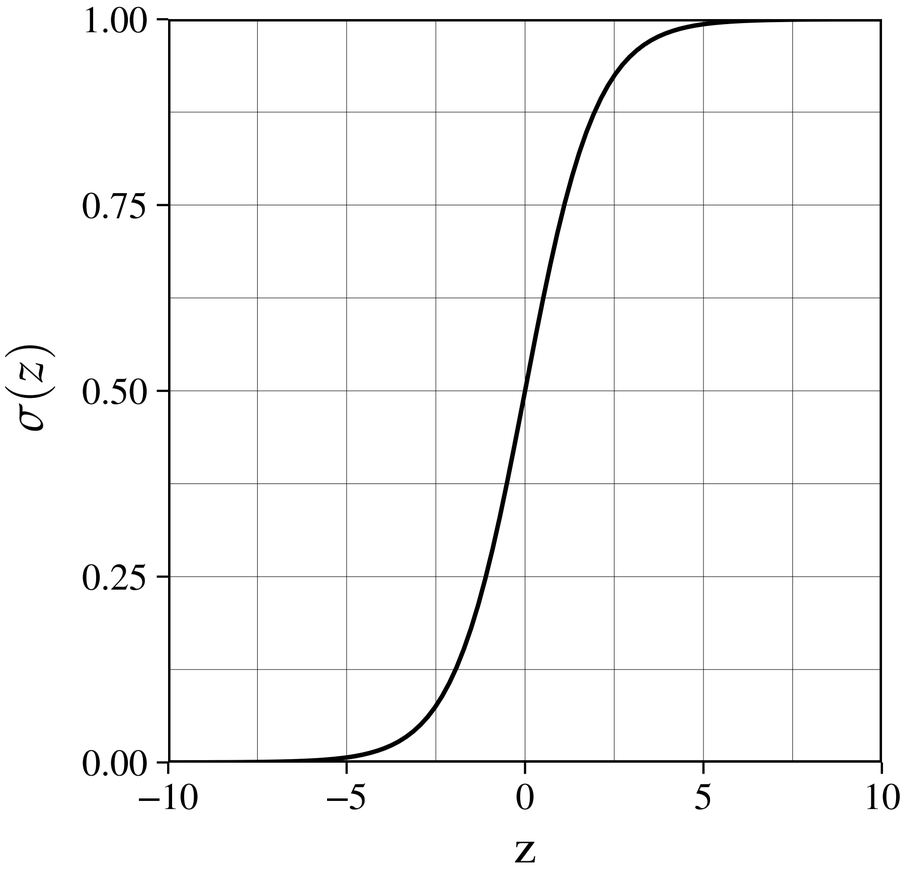
Probabilities have to be between [0,1]

Our hypothesis function *h(X)* can return any continuous value

Let's use some function to map *h(X)* to [0,1]

# *Logistic Function*

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$




This is known as the logistic function, an example of a sigmoid function

# *Classification Losses: Logistic Regression*

Assume $Y$ is either 0 or 1.

Model the class posterior probability $P(Y=1 \mid X)$, using $\sigma(h(X))$

Measure of quality of the model: given a choice of $h(X)$, what is the probability we would have observed the labels we observed in the data.

N Bernoulli trials. The estimated probability of observing label $y_i$ for object $i$:

$$\begin{cases} \sigma(h(\mathbf{x}_i)), & \text{if } y_i = 1 \\ 1 - \sigma(h(\mathbf{x}_i)), & \text{if } y_i = 0 \end{cases}$$

Combining this, the likelihood is

$$L(h) = \prod_{i=1}^{N} \sigma(h(\mathbf{x}_i))^{y_i} (1 - \sigma(h(\mathbf{x}_i)))^{1-y_i}$$

# *Logistic Regression: Objective Function*

$$L(h) = \prod_{i=1}^{N} \sigma(h(\mathbf{x}_i))^{y_i} (1 - \sigma(h(\mathbf{x}_i)))^{1-y_i}$$

The likelihood is a measure of how well the estimated probabilities explain the observed labels, so we want to maximise this likelihood

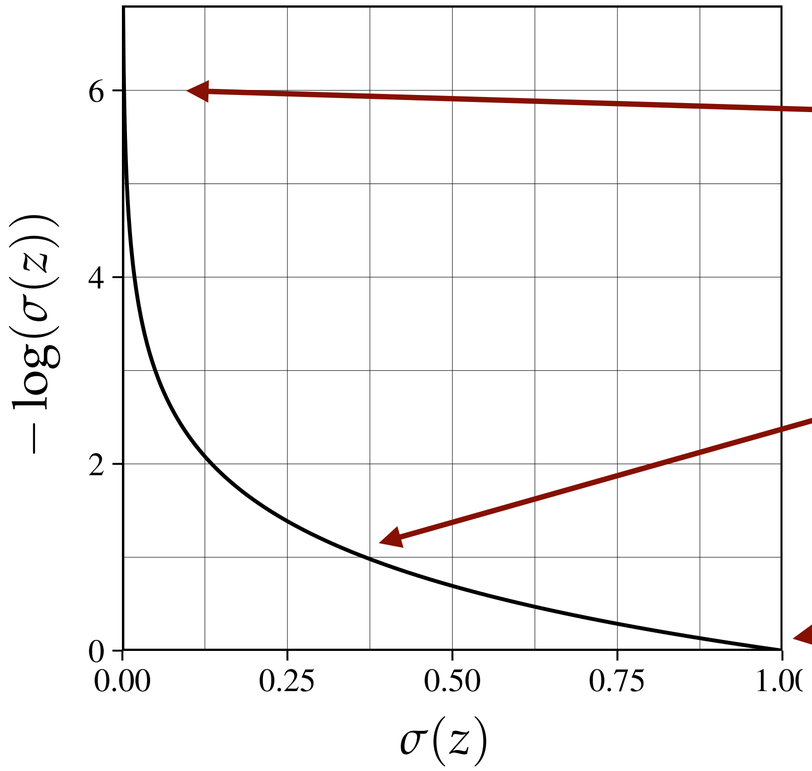
This gives the same optimum as minimizing the negative log likelihood:

$$J(h) = -\sum_{i=1}^{N} y_i \log[\sigma(h(\mathbf{x}_i))] + (1 - y_i) \log[1 - \sigma(h(\mathbf{x}_i))]$$

# *Logistic Loss, Visualized*

$$J(h) = -\sum_{i=1}^{N} y_i \log[\sigma(h(\mathbf{x}_i))] + (1 - y_i) \log[1 - \sigma(h(\mathbf{x}_i))]$$

*For $y_i = 1$*

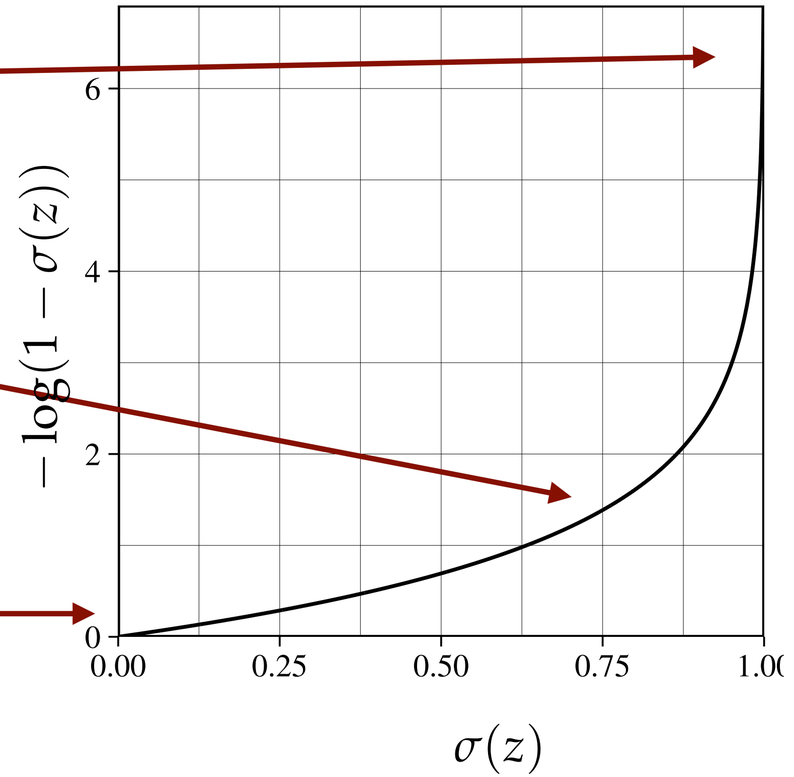



*Infinite loss if prediction is certain but wrong*

*The higher the certainty of the correct label, the lower the loss*

*Zero loss if prediction is certain but correct*

*For $y_i = 0$*



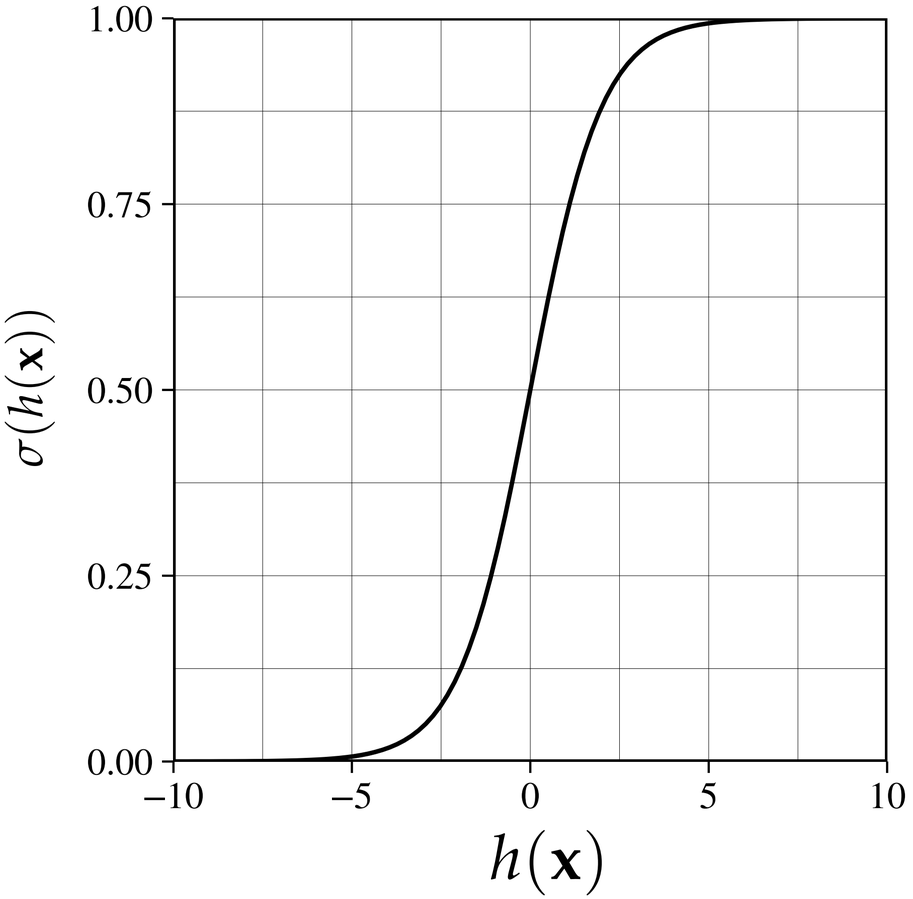
# *Logistic Regression: Hypothesis Class*

For linear logistic regression,
again consider all **w,** $w_0$ for $h(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + w_0$

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} -\sum_{i=1}^{N} y_i \log[\sigma(\mathbf{x}_i^\top \mathbf{w} + w_0)] + (1 - y_i) \log[1 - \sigma(\mathbf{x}_i^\top \mathbf{w} + w_0)]$$

# *Logistic Regression*

- If we want class predictions, we have to use a cut-off at a probability
- Corresponds to a threshold for $h(\mathbf{x})$



# *Logistic Regression: Example*

https://jkrijthe.shinyapps.io/lr-optimization/

# *Practice Question: Logistic Regression*

$$\sigma(\mathbf{x}_i^\top \mathbf{w} + w_0)$$

Suppose we have a linear logistic regression model, with one feature $x$, where $w_1 = 1$ and $w_0 = 0$. Answer the following questions:

**True or False:** because we have a linear model, the estimated probability of the positive class for $2x$ is twice as high as for $x$, for all choices of $x$.

**True or False:** because the logistic function is a non-linear function, logistic regression is not a linear classifier.

# *Classifier Construction using Empirical Risk Minimisation*

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions
2. Define a cost (loss) function to measure the "quality" of each hypothesis
3. Measure the average cost on the training data
4. Find the function that minimises this cost

# *Logistic Regression*

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} -\sum_{i=1}^{N} y_i \log[\sigma(\mathbf{x}_i^\top \mathbf{w} + w_0)] + (1 - y_i) \log[1 - \sigma(\mathbf{x}_i^\top \mathbf{w} + w_0)]$$

# *Gradient Descent Procedure*

Given an objective function *J*, learning rate (step size) $\alpha$, and number of iterations T.

1. Pick a starting value (for instance, random) for $\mathbf{w}$ and $w_0$
2. For T iterations, for all j = *0,1,…,D*, do:

$$w_j^{t+1} = w_j^t - \alpha \left. \frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} \right|_{\mathbf{w}, w_0 = \mathbf{w}^t, w_0^t}$$

# *Logistic Regression Gradient*

$$J(\mathbf{w}) = -\sum_{i=1}^{N} y_i \log[\sigma(\mathbf{x}_i^\top \mathbf{w})] + (1 - y_i) \log[1 - \sigma(\mathbf{x}_i^\top \mathbf{w})]$$

Find the gradient, by taking the derivative and using $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z))$

*Homework: prove this!*

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = -\sum_{i=1}^{N} y_i \frac{1}{\sigma(\mathbf{x}_i^\top \mathbf{w})} \sigma(\mathbf{x}_i^\top \mathbf{w})(1 - \sigma(\mathbf{x}_i^\top \mathbf{w}))\mathbf{x}_i + (1 - y_i) \frac{1}{1 - \sigma(\mathbf{x}_i^\top \mathbf{w})} \cdot -\sigma(\mathbf{x}_i^\top \mathbf{w})(1 - \sigma(\mathbf{x}_i^\top \mathbf{w}))\mathbf{x}_i$$

Some terms cancel:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = -\sum_{i=1}^{N} y_i (1 - \sigma(\mathbf{x}_i^\top \mathbf{w}))\mathbf{x}_i + (1 - y_i) \cdot -\sigma(\mathbf{x}_i^\top \mathbf{w})\mathbf{x}_i$$

Note that we can write this as:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = -\sum_{i=1}^{N} (y_i - \sigma(\mathbf{x}_i^\top \mathbf{w}))\mathbf{x}_i$$

# *Logistic Regression: Gradient Descent Example*

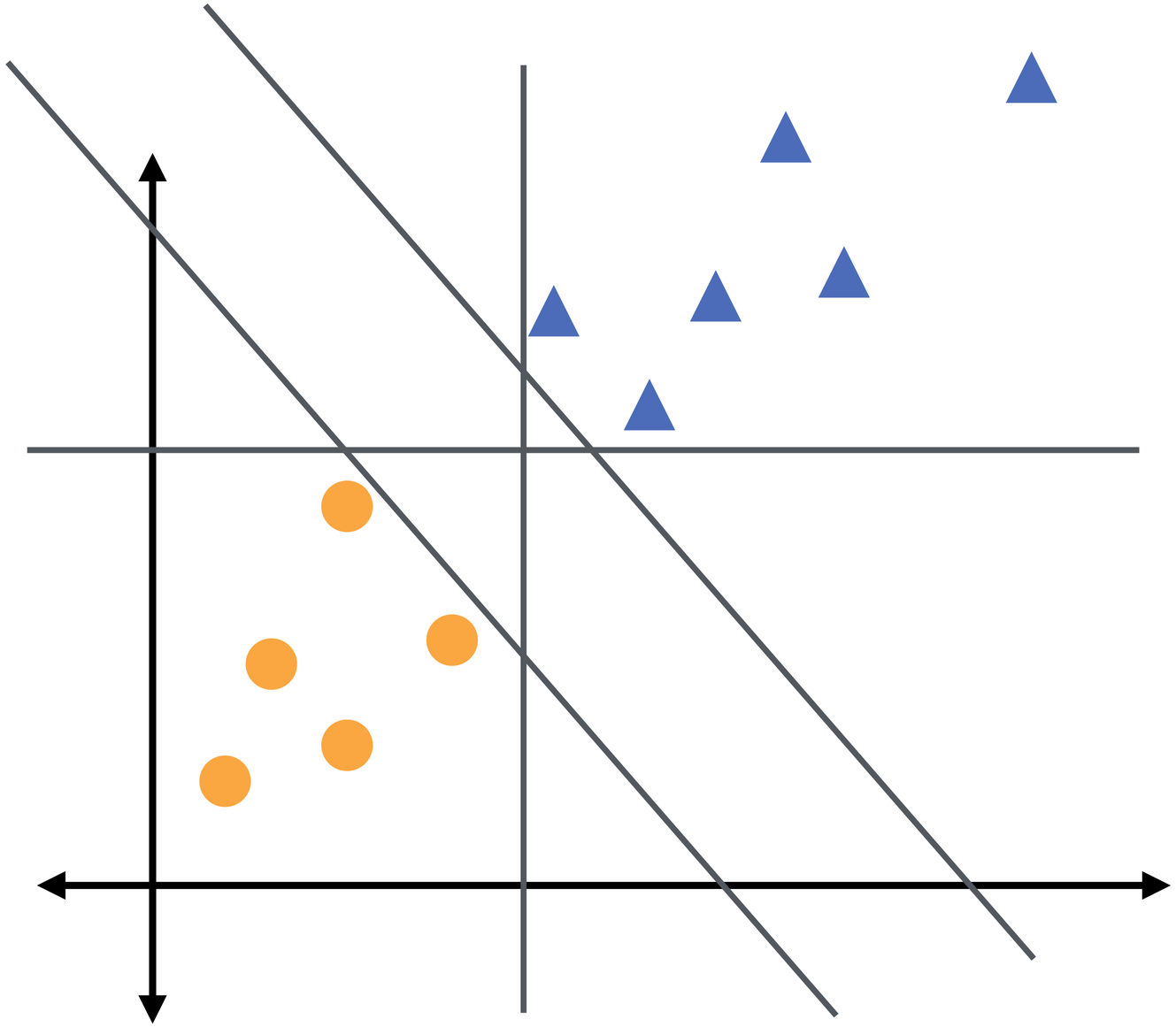
https://jkrijthe.shinyapps.io/lr-optimization/

# *Logistic Regression Recap*

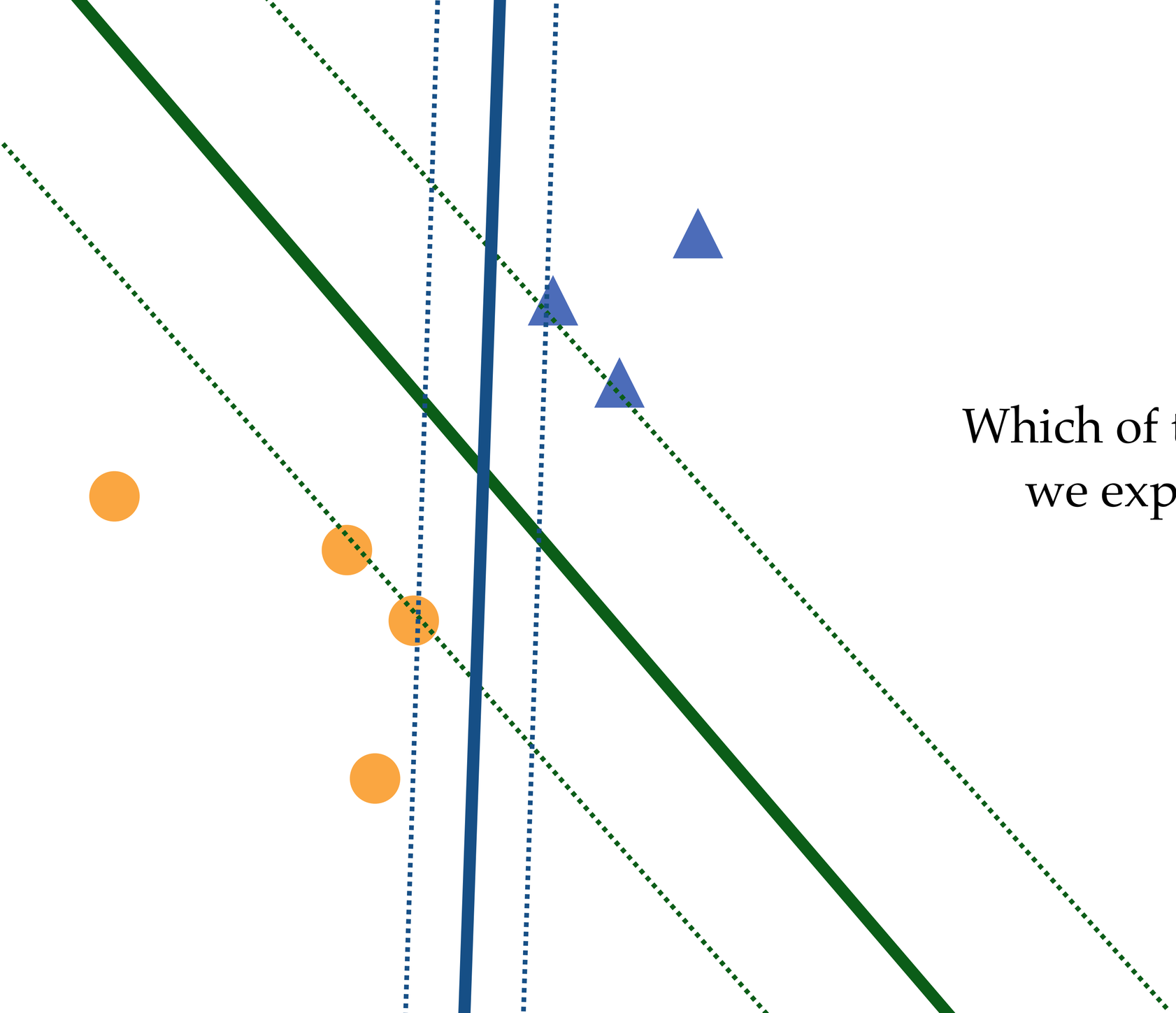
- Linear classifier
- Maximize the (log) likelihood of the observed posterior probabilities
- Or: minimize the negative (log) likelihood
- No analytical solution, so we use iterative procedures, like gradient descent (more efficient alternatives exist)
- If no class overlap our quality measure is not sufficient: many equally good solutions.

# SUPPORT VECTOR MACHINES

*Note: the material in the book covers some additional details about the optimization, here we will focus on the intuition*

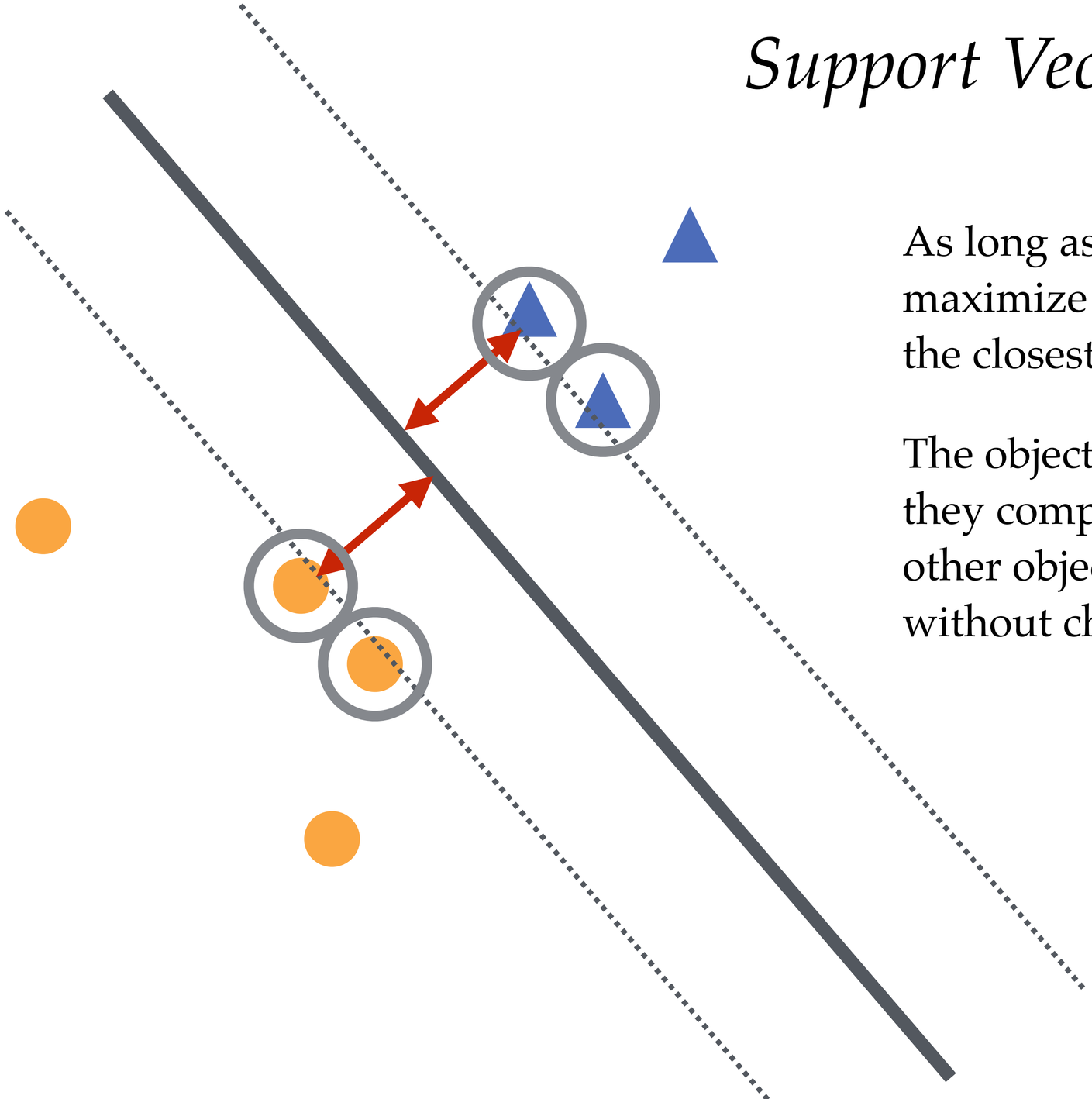


*Many possible decision boundaries with zero error.*
*How do we choose?*



Which of these two classifiers do we expect to perform best?

# *Support Vector Machines*



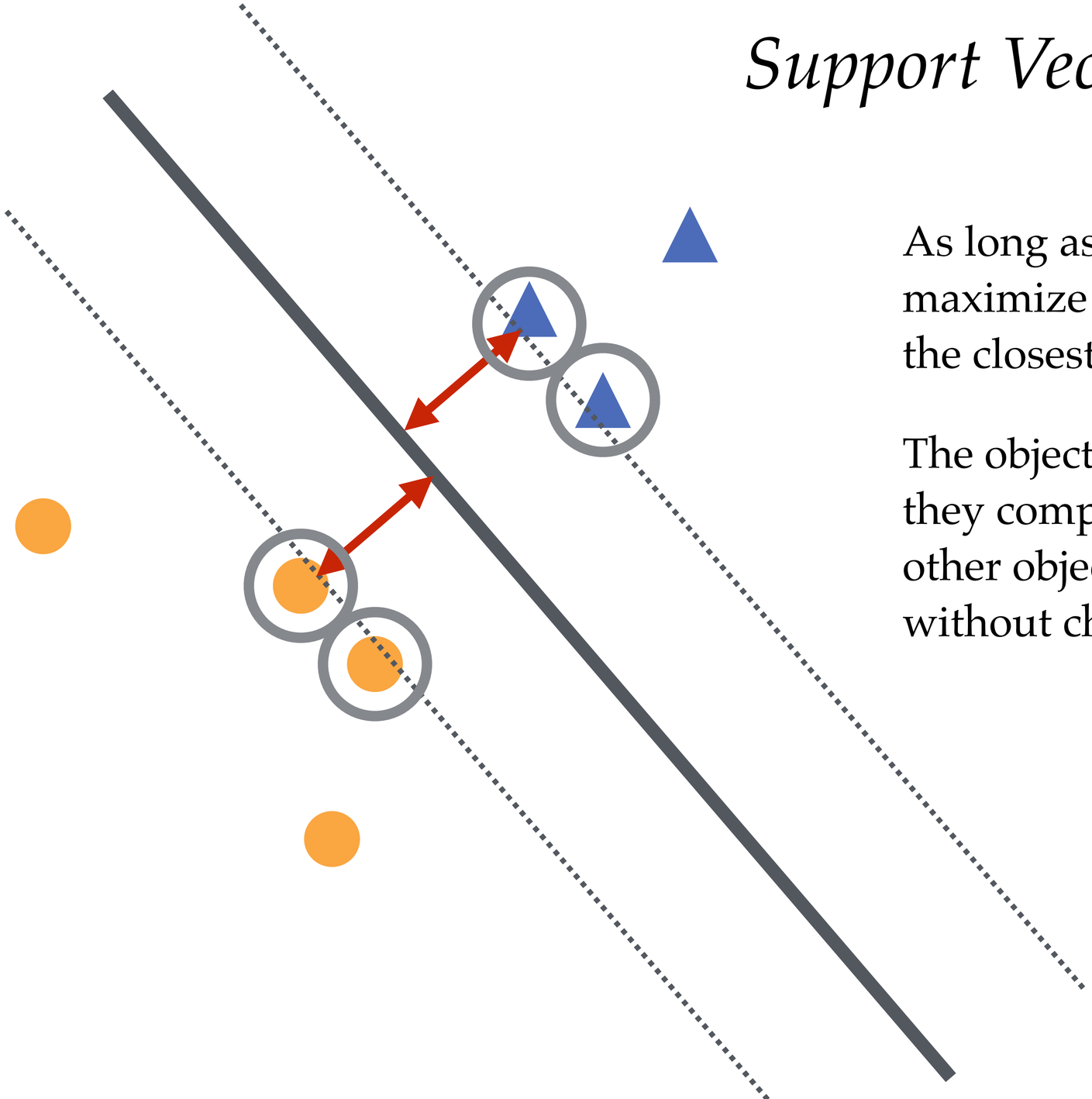
As long as all objects are correctly classified, maximize the size of the "**margin**": the distance of the closest points to the decision boundary

The objects on the margin are the **support vectors**: they completely define the decision boundary. The other objects can be moved (outside the margin) without changing the classifier.

# *Support Vector Machines*

- Goal: intuition and principles behind SVMs. There are lots of extensions and theory that will have to wait.
- At first, assume the classes are linearly separable (we shortly cover more general setting, but the details will be in later courses)
- Intuition: stable solution (slight changes in the data would not have led to different decisions)
- Formally: you can prove nice generalization behaviour

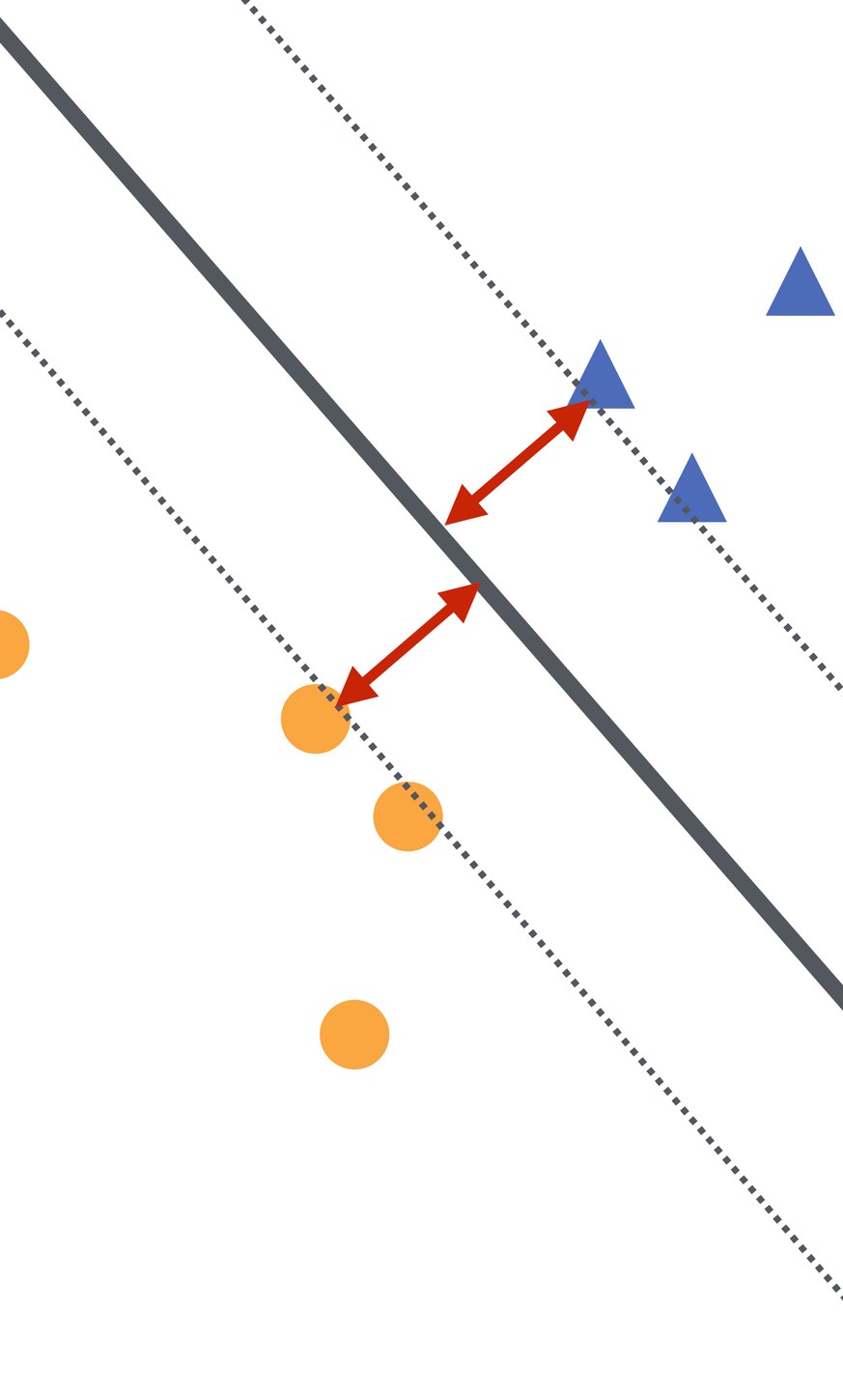
# *Support Vector Machines*



As long as all objects are correctly classified, maximize the size of the "**margin**": the distance of the closest points to the decision boundary

The objects on the margin are the **support vectors**: they completely define the decision boundary. The other objects can be moved (outside the margin) without changing the classifier.

https://jkrijthe.shinyapps.io/ClassroomSVM/



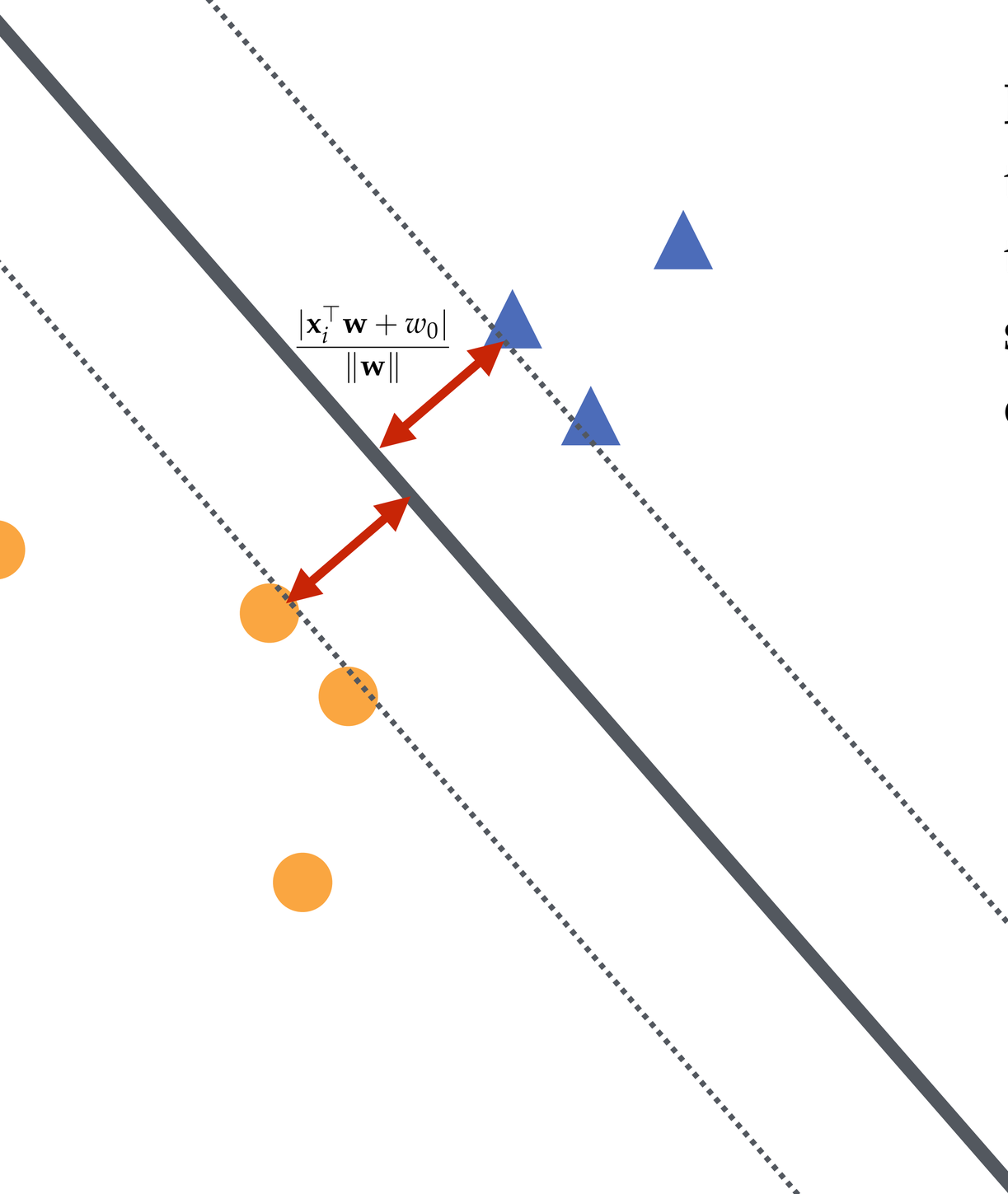
# *Defining the SVC*

We want to classify the object as either the positive class or the negative class, which we decide based on the sign of

$$\mathbf{x}_i^\top \mathbf{w} + w_0$$

We want the closest objects to be "far away" from the decision boundary, so let's say we want a decision value of at least M:

$$\mathbf{x}_i^\top \mathbf{w} + w_0 \geq M \text{ if } y_i = +1$$
$$\mathbf{x}_i^\top \mathbf{w} + w_0 \leq -M \text{ if } y_i = -1$$




**Problem**: we always have the freedom to scale $\mathbf{w}$ to reach $M$, without changing the decision boundary. Let's choose the scale such that the closest object has decision value -1 or 1.

$$\mathbf{x}_i^\top \mathbf{w} + w_0 \geq 1 \text{ if } y_i = +1$$
$$\mathbf{x}_i^\top \mathbf{w} + w_0 \leq -1 \text{ if } y_i = -1$$

What is the distance of the decision boundary to the closest object?

$$\frac{|\mathbf{x}_i^\top \mathbf{w} + w_0|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

# *Defining the Support Vector Classifier*

For the points on the margin, we thus know that their distance to the decision boundary is:

$$\frac{|\mathbf{x}_i^\top \mathbf{w} + w_0|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

So the margin is: $\frac{2}{\|\mathbf{w}\|}$

We wanted to maximise this margin: $\max_{\mathbf{w}, w_0} \frac{2}{\|\mathbf{w}\|}$

Which gives the same solution as minimising $\min_{\mathbf{w}, w_0} \frac{1}{2}\|\mathbf{w}\|^2$

Subject to

$$\mathbf{x}_i^\top \mathbf{w} + w_0 \geq 1 \text{ if } y_i = +1$$

$$\mathbf{x}_i^\top \mathbf{w} + w_0 \leq -1 \text{ if } y_i = -1$$

$$\min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|^2$$

Subject to

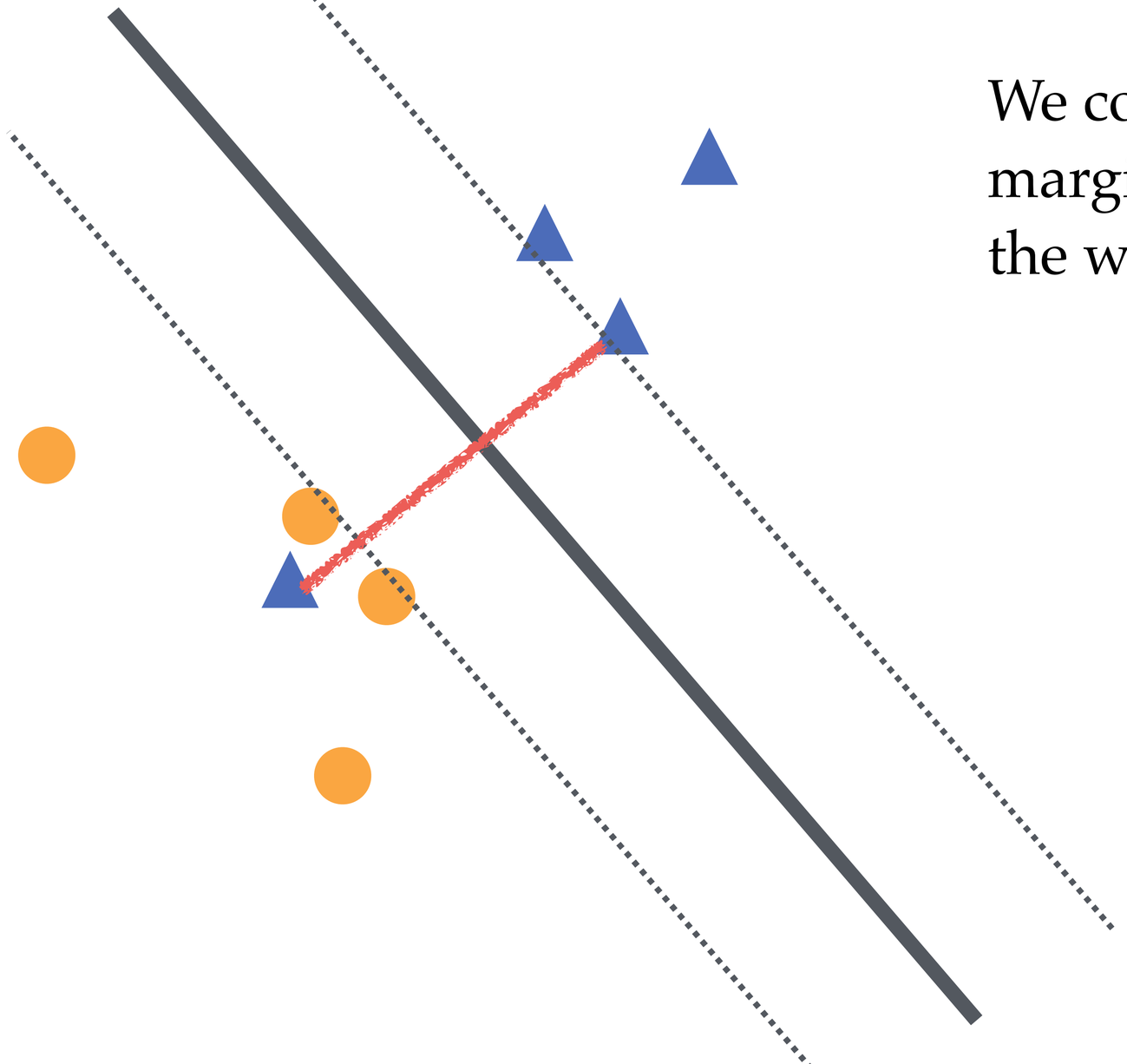
$$\mathbf{x}_i^\top \mathbf{w} + w_0 \geq 1 \text{ if } y_i = +1$$

$$\mathbf{x}_i^\top \mathbf{w} + w_0 \leq -1 \text{ if } y_i = -1$$

# *Practice Question: Support Vector Machine*

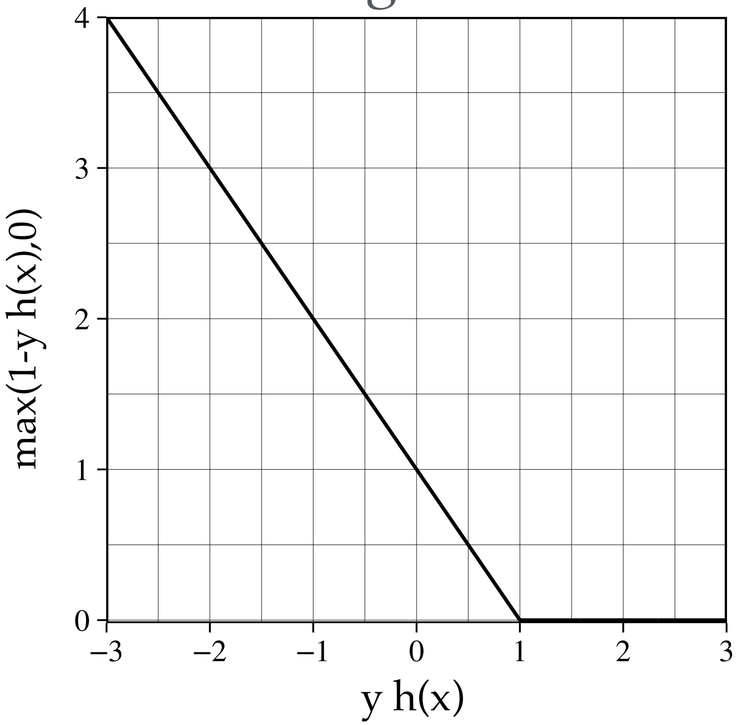
Consider a 1D linearly separable classification problem, with N unique objects. If I train a Support Vector Classifier, how many support vectors will there be?

# *What if the data is not linearly separable?*



We could allow for “violations” of the margin: penalize how far the object is on the wrong side of the margin.

*Hinge Loss*



# *Soft-Margin SVM*

$$\min_{\mathbf{w},w_0} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}[1 - y_i(\mathbf{x}_i^\top \mathbf{w} + w_0)]_+$$

Where $[.]_+$ indicates the value of the function if the value is positive, and 0 otherwise.

# *Classifier Construction using Empirical Risk Minimisation*

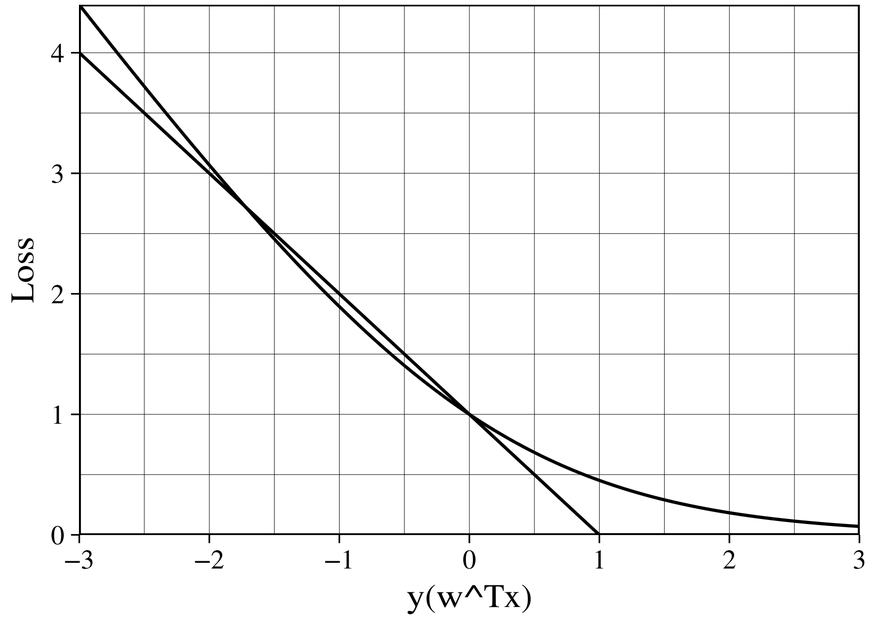
$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions
2. Define a cost (loss) function to measure the "quality" of each hypothesis
3. Measure the average cost on the training data
4. Find the function that minimises this cost

# *SVM as Empirical Risk Minimization*

$$\min_{\mathbf{w},w_0} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}[1 - y_i(\mathbf{x}_i^\top \mathbf{w} + w_0)]_+$$

*A Regularization Term?*

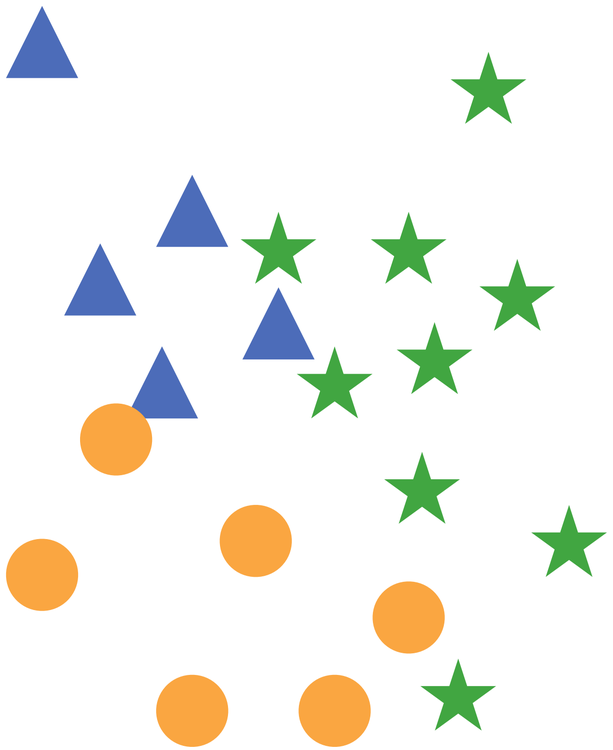


# *Support Vector Machines Summary*

- Linearly separable data: maximise the margin
- The objects on the margin completely determine what the decision boundary looks like: **support vectors**
- Removing the other objects will not change the decision boundary
- SVMs can also be formulated as a risk minimisation problem
- Extensions: non-linear functions

# MULTI-CLASS CLASSIFICATION

# *Multiclass Classification*



- More than 2 classes
- For generative models: model the extra $P(X \mid Y=k)$'s
- Discriminative models: two approaches:
  - Algorithm specific adaptations: directly construct a multi-class classifier
  - General techniques: combining multiple 2-class classifiers

# *Algorithm Specific Adaptions: Example*

Change the algorithm to return posteriors / decision values for K-classes, rather than 1.

**Example**: In logistic regression, instead of modelling the probability of positive vs. negative, we can also model the probability of multiple classes:

*Two Class*

$$p(Y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{x}^\top \mathbf{w})}$$

$$p(Y = 0|\mathbf{x}) = \frac{\exp(-\mathbf{x}^\top \mathbf{w})}{1 + \exp(-\mathbf{x}^\top \mathbf{w})}$$

*Multiclass*

$$p(Y = k|\mathbf{x}) = \frac{\exp(-\mathbf{x}^\top \mathbf{w}_k)}{\sum_{k=1}^{K} \exp(-\mathbf{x}^\top \mathbf{w}_k)}$$

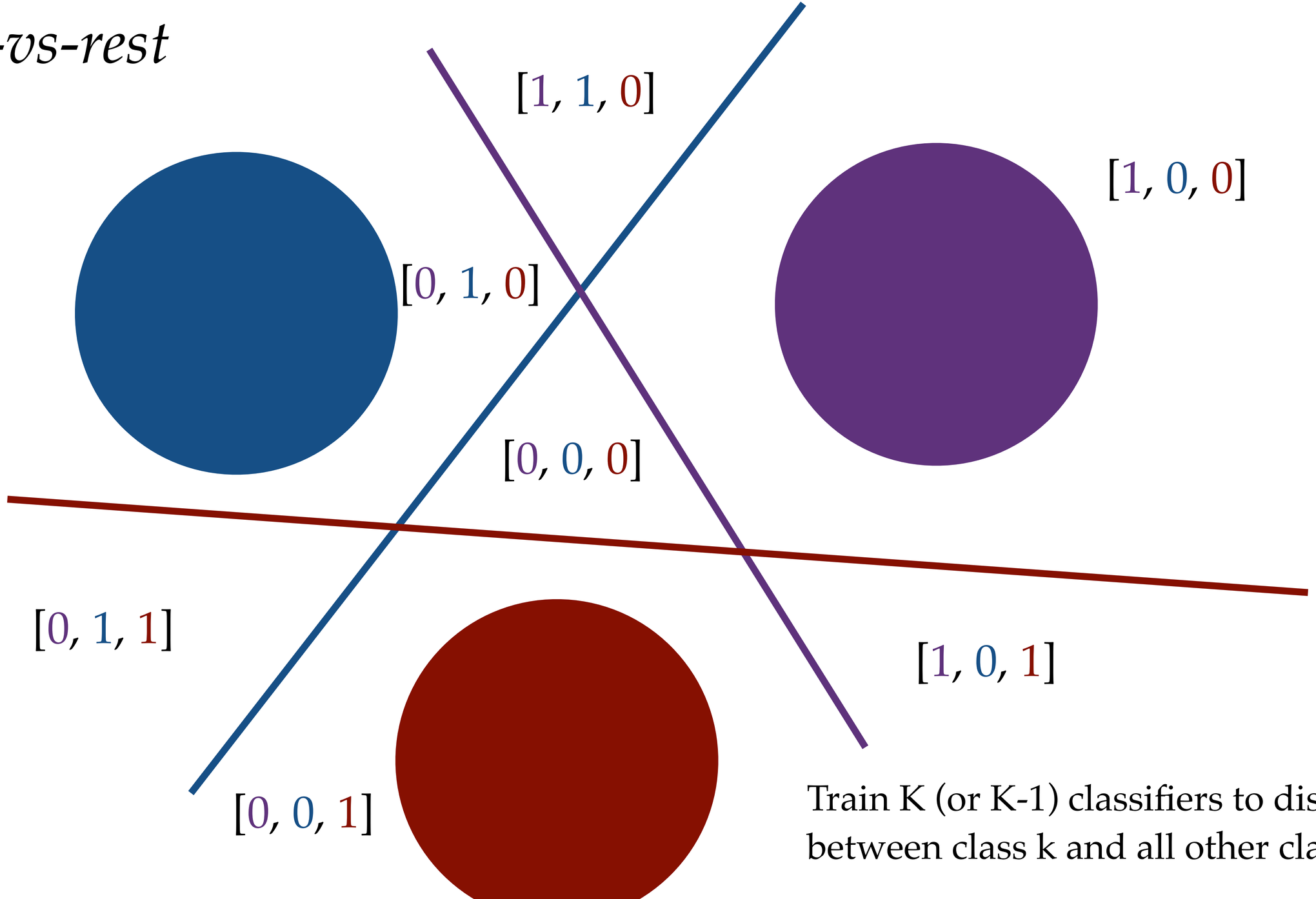
# *General Approaches*

Sometimes, the algorithm might be inherently binary, and hard to adapt to multiple classes. Can we still use these binary classifiers to do multiclass classification?

Consider two strategies:

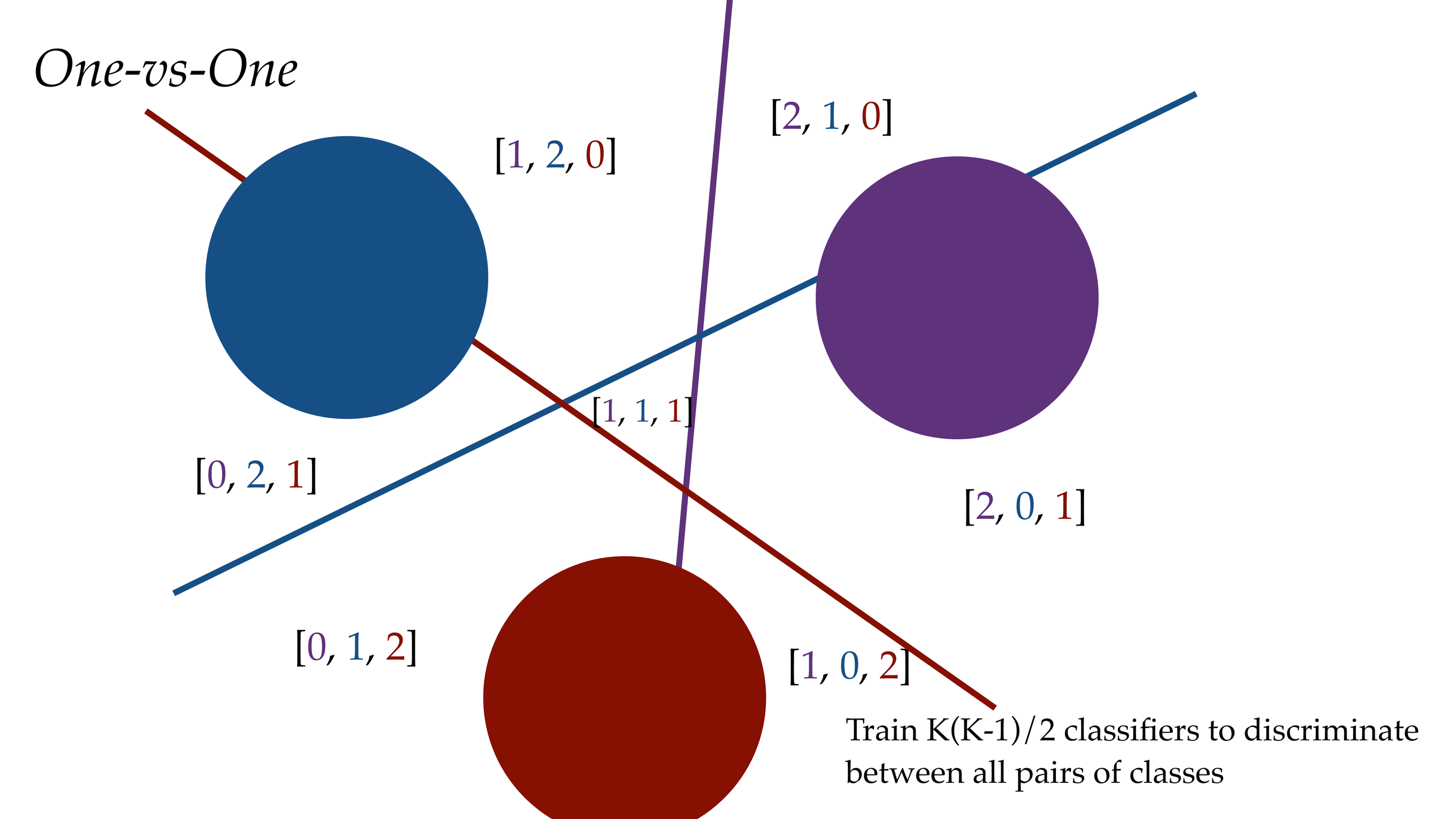
- One-vs-rest (aka one-vs-all)
- One-vs-one

# *One-vs-rest*




Train K (or K-1) classifiers to discriminate between class k and all other classes

# *One-vs-One*




Train K(K-1)/2 classifiers to discriminate between all pairs of classes

# *Solving the Ambiguity*

- Possible solution to the ambiguity: use the decision values rather than predicted labels of the classifiers
- Ideally, we train these classifiers jointly, to optimize performance on the multiclass problem
- Often, we can adapt the original model from 2 class to K-class, as in the logistic regression example

# *Summary*

- Linear Logistic Regression with logistic loss and gradient descent
- Linear Support Vector Machines: maximising the “margin”
- Multi-class discriminative classifiers:
  - Algorithm specific adaptation
  - Generic techniques: one-vs-all, one-vs-rest (but we have to deal with ambiguities)

# THANKS