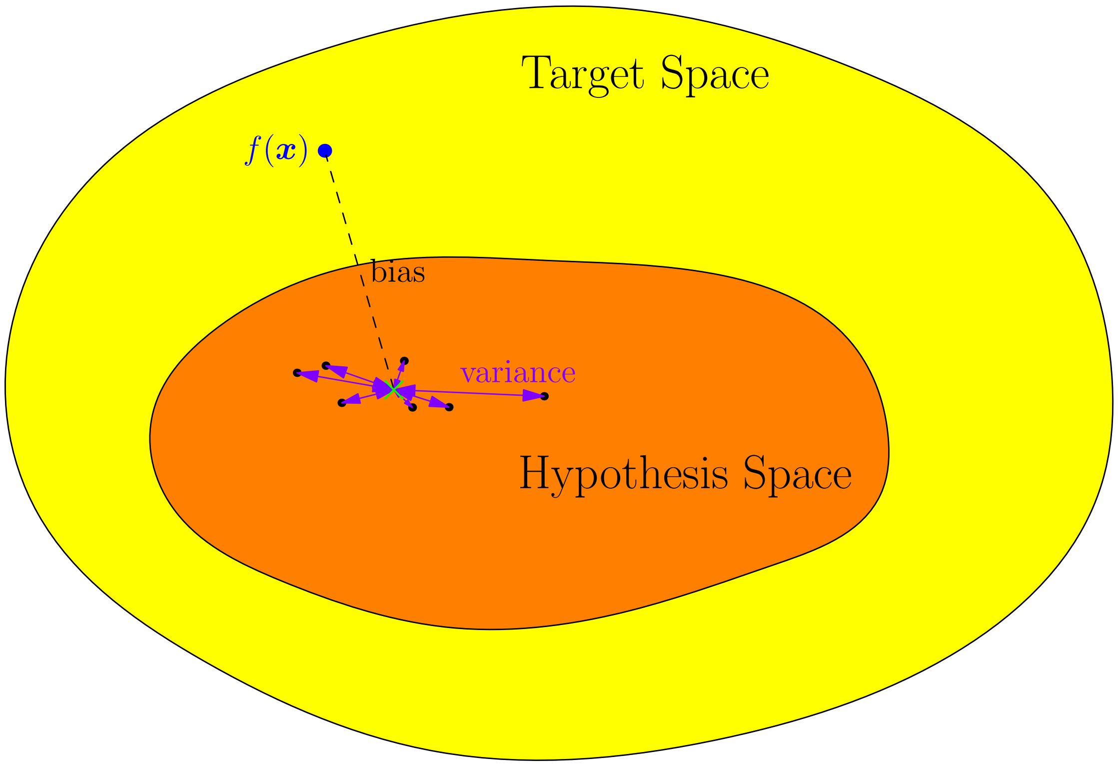
NGCM ML Workshop

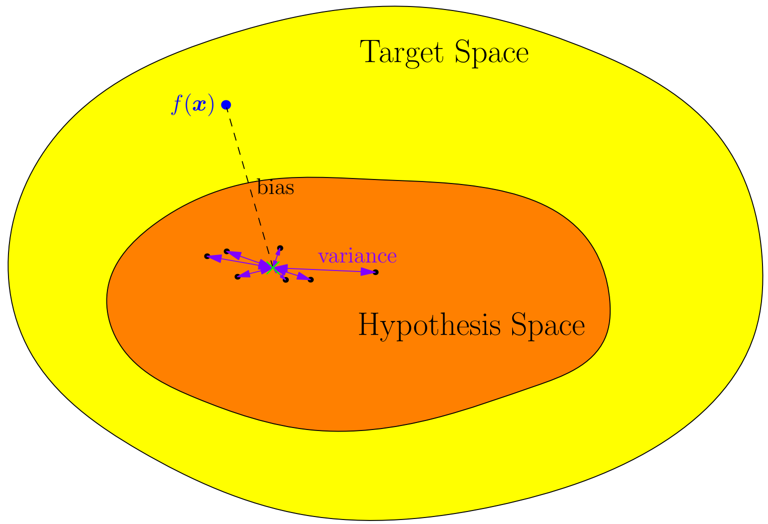
Advanced Machine Learning



When ML Works, SVMs, Decision Trees, Ensemble Methods, Bayesian Inference

Outline

1. **What Makes a Good Learning Machine?**
2. SVMs
3. Ensemble Methods
4. Bayesian Inference



What Makes a Good Learning Machine?

- We are going to cover some advanced machine learning techniques
- To understand why these works we need to understand what makes a good learning machine
- For this we have to get conceptual and think about **generalisation** performance

  generalisation: *how well do we do on unseen data as opposed to the training data*

What Makes a Good Learning Machine?

- We are going to cover some advanced machine learning techniques
- To understand why these works we need to understand what makes a good learning machine
- For this we have to get conceptual and think about **generalisation** performance

  generalisation: *how well do we do on unseen data as opposed to the training data*

What Makes a Good Learning Machine?

- We are going to cover some advanced machine learning techniques
- To understand why these works we need to understand what makes a good learning machine
- For this we have to get conceptual and think about **generalisation** performance

  generalisation*: how well do we do on unseen data as opposed to the training data*

What Makes a Good Learning Machine?

- We are going to cover some advanced machine learning techniques
- To understand why these works we need to understand what makes a good learning machine
- For this we have to get conceptual and think about **generalisation** performance

  ***generalisation**: how well do we do on unseen data as opposed to the training data*

What Makes Machine Learning Hard?

- Typically work in high dimensions (i.e. have many features)
- The problem can be over-constrained (i.e. we have conflicting data to deal with)
- The problem can be under-constrained (i.e. there are many possible solutions that are consistent with the data)

- Typically in machine learning the data will be over-constrained in some dimensions and under-constrained in others
- We can't visualise the data to know what is going on

What Makes Machine Learning Hard?

- Typically work in high dimensions (i.e. have many features)
- The problem can be over-constrained (i.e. we have conflicting data to deal with)
- The problem can be under-constrained (i.e. there are many possible solutions that are consistent with the data)

- Typically in machine learning the data will be over-constrained in some dimensions and under-constrained in others
- We can't visualise the data to know what is going on

What Makes Machine Learning Hard?

- Typically work in high dimensions (i.e. have many features)
- The problem can be over-constrained (i.e. we have conflicting data to deal with)—can minimise an error function
- The problem can be under-constrained (i.e. there are many possible solutions that are consistent with the data)

- Typically in machine learning the data will be over-constrained in some dimensions and under-constrained in others
- We can't visualise the data to know what is going on

What Makes Machine Learning Hard?

- Typically work in high dimensions (i.e. have many features)
- The problem can be over-constrained (i.e. we have conflicting data to deal with)—can minimise an error function
- The problem can be under-constrained (i.e. there are many possible solutions that are consistent with the data)
- Typically in machine learning the data will be over-constrained in some dimensions and under-constrained in others
- We can't visualise the data to know what is going on

What Makes Machine Learning Hard?

- Typically work in high dimensions (i.e. have many features)
- The problem can be over-constrained (i.e. we have conflicting data to deal with)—can minimise an error function
- The problem can be under-constrained (i.e. there are many possible solutions that are consistent with the data)—need to choose a plausible solution
- Typically in machine learning the data will be over-constrained in some dimensions and under-constrained in others
- We can't visualise the data to know what is going on

What Makes Machine Learning Hard?

- Typically work in high dimensions (i.e. have many features)
- The problem can be over-constrained (i.e. we have conflicting data to deal with)—can minimise an error function
- The problem can be under-constrained (i.e. there are many possible solutions that are consistent with the data)—need to choose a plausible solution
- Typically in machine learning the data will be over-constrained in some dimensions and under-constrained in others
- We can't visualise the data to know what is going on

What Makes Machine Learning Hard?

- Typically work in high dimensions (i.e. have many features)
- The problem can be over-constrained (i.e. we have conflicting data to deal with)—can minimise an error function
- The problem can be under-constrained (i.e. there are many possible solutions that are consistent with the data)—need to choose a plausible solution
- Typically in machine learning the data will be over-constrained in some dimensions and under-constrained in others
- We can't visualise the data to know what is going on

Least Squared Errors

- Suppose we want to learn some function $f(\boldsymbol{x})$
- We construct a learning machine that makes a prediction $\hat{f}(\boldsymbol{x}|\boldsymbol{w})$, where $\boldsymbol{w}$ are weights we want to learn
- We typically choose the weights to minimise a *training error*

$$E_T(\boldsymbol{w}) = \sum_{\boldsymbol{x} \in \mathcal{D}} \left(\hat{f}(\boldsymbol{x}|\boldsymbol{w}) - f(\boldsymbol{x}) \right)^2$$

where $\mathcal{D}$ is a finite data set of size N, sampled from the set of all inputs, $\mathcal{X}$, according to a probability distribution $p(\boldsymbol{x})$ describing where our data is

Least Squared Errors

- Suppose we want to learn some function $f(\boldsymbol{x})$
- We construct a learning machine that makes a prediction $\hat{f}(\boldsymbol{x}|\boldsymbol{w})$, where $\boldsymbol{w}$ are weights we want to learn
- We typically choose the weights to minimise a *training error*

$$E_T(\boldsymbol{w}) = \sum_{\boldsymbol{x} \in \mathcal{D}} \left(\hat{f}(\boldsymbol{x}|\boldsymbol{w}) - f(\boldsymbol{x}) \right)^2$$

where $\mathcal{D}$ is a finite data set of size N, sampled from the set of all inputs, $\mathcal{X}$, according to a probability distribution $p(\boldsymbol{x})$ describing where our data is

Least Squared Errors

- Suppose we want to learn some function $f(\boldsymbol{x})$
- We construct a learning machine that makes a prediction $\hat{f}(\boldsymbol{x}|\boldsymbol{w})$, where $\boldsymbol{w}$ are weights we want to learn
- We typically choose the weights to minimise a *training error*

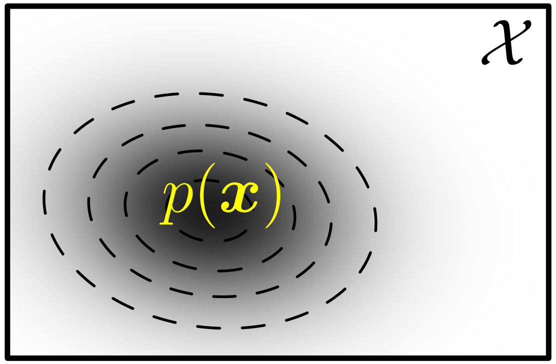
$$E_T(\boldsymbol{w}) = \sum_{\boldsymbol{x} \in \mathcal{D}} \left(\hat{f}(\boldsymbol{x}|\boldsymbol{w}) - f(\boldsymbol{x}) \right)^2$$

where $\mathcal{D}$ is a finite data set of size N, sampled from the set of all inputs, $\mathcal{X}$, according to a probability distribution $p(\boldsymbol{x})$ describing where our data is

Generalisation Error

- We want to minimise the *generalisation error* which in this case we can measure as

$$E_G(\boldsymbol{w}) = \sum_{\boldsymbol{x} \in \mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\boldsymbol{w}) - f(\boldsymbol{x}) \right)^2$$



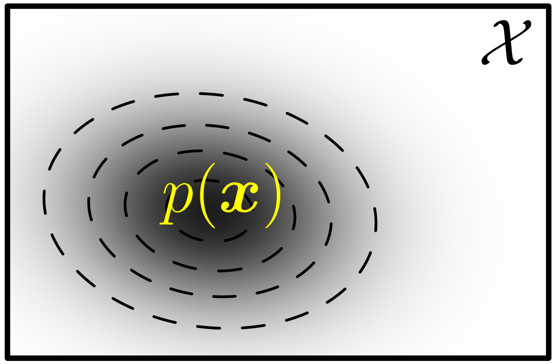
(we can estimate this if we have some examples with known labels $y_i = f(\boldsymbol{x}_i)$ which we have not trained on)

- We want to minimise $E_G(\boldsymbol{w})$ but in practice we are minimising $E_T(\boldsymbol{w})$

Generalisation Error

- We want to minimise the *generalisation error* which in this case we can measure as

$$E_G(\boldsymbol{w}) = \sum_{\boldsymbol{x} \in \mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\boldsymbol{w}) - f(\boldsymbol{x}) \right)^2$$



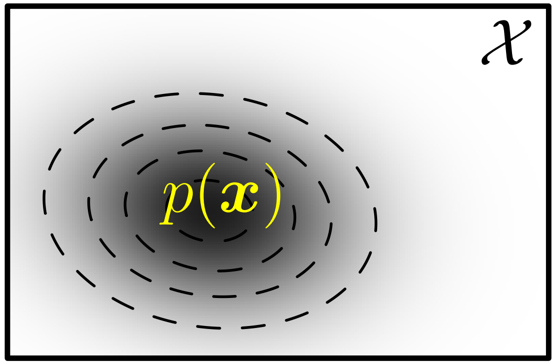
(we can estimate this if we have some examples with known labels $y_i = f(\boldsymbol{x}_i)$ which we have not trained on)

- We want to minimise $E_G(\boldsymbol{w})$ but in practice we are minimising $E_T(\boldsymbol{w})$

Generalisation Error

- We want to minimise the *generalisation error* which in this case we can measure as

$$E_G(\boldsymbol{w}) = \sum_{\boldsymbol{x} \in \mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\boldsymbol{w}) - f(\boldsymbol{x}) \right)^2$$



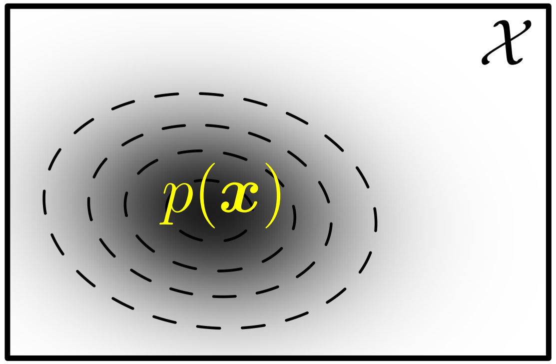
(we can estimate this if we have some examples with known labels $y_i = f(\boldsymbol{x}_i)$ which we have not trained on)

- We want to minimise $E_G(\boldsymbol{w})$ but in practice we are minimising $E_T(\boldsymbol{w})$

Generalisation Error

- We want to minimise the *generalisation error* which in this case we can measure as

$$E_G(\boldsymbol{w}) = \sum_{\boldsymbol{x} \in \mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\boldsymbol{w}) - f(\boldsymbol{x}) \right)^2$$

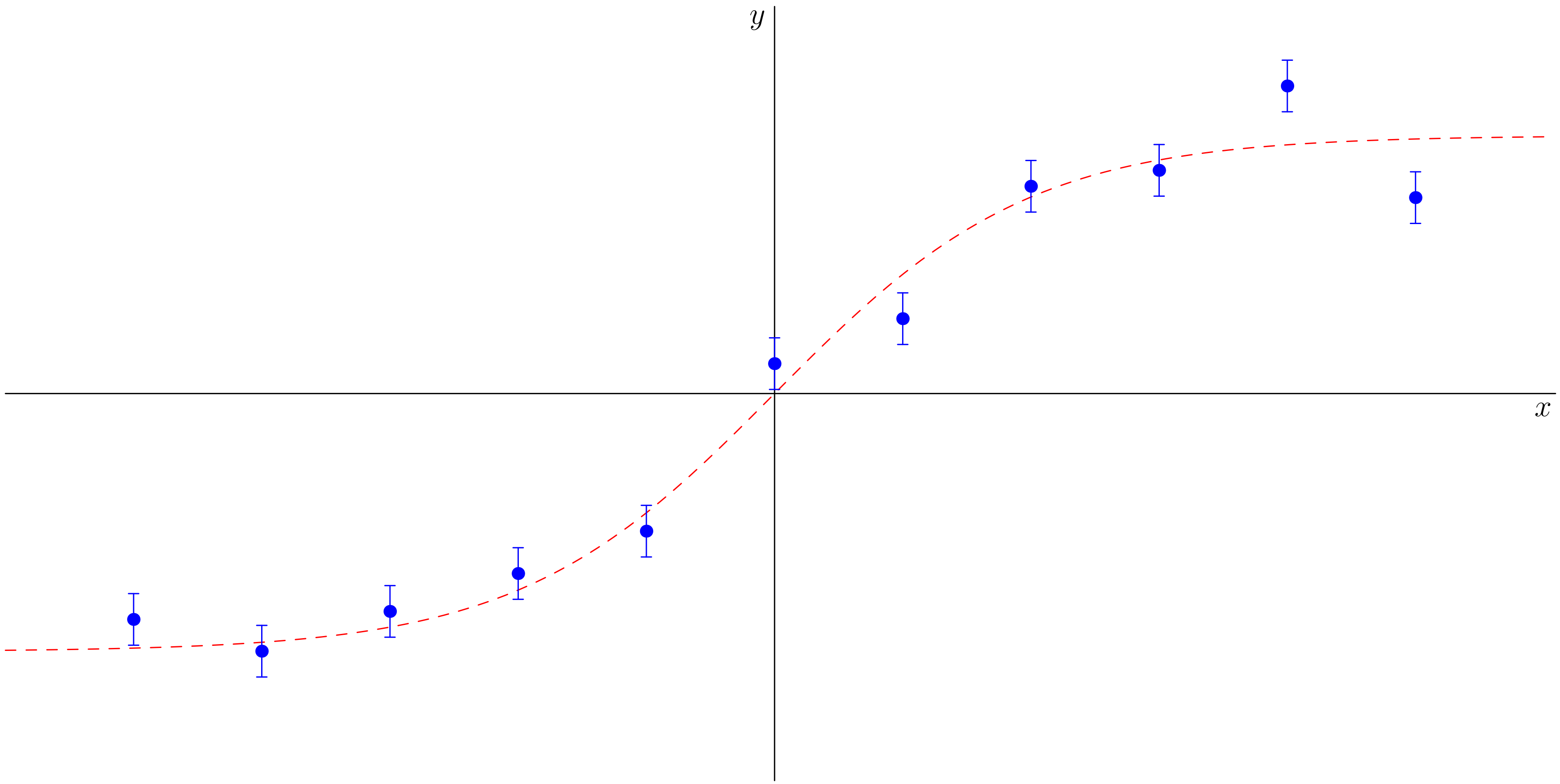


(we can estimate this if we have some examples with known labels $y_i = f(\boldsymbol{x}_i)$ which we have not trained on)

- We want to minimise $E_G(\boldsymbol{w})$ but in practice we are minimising $E_T(\boldsymbol{w})$, *what could possibly go wrong?*

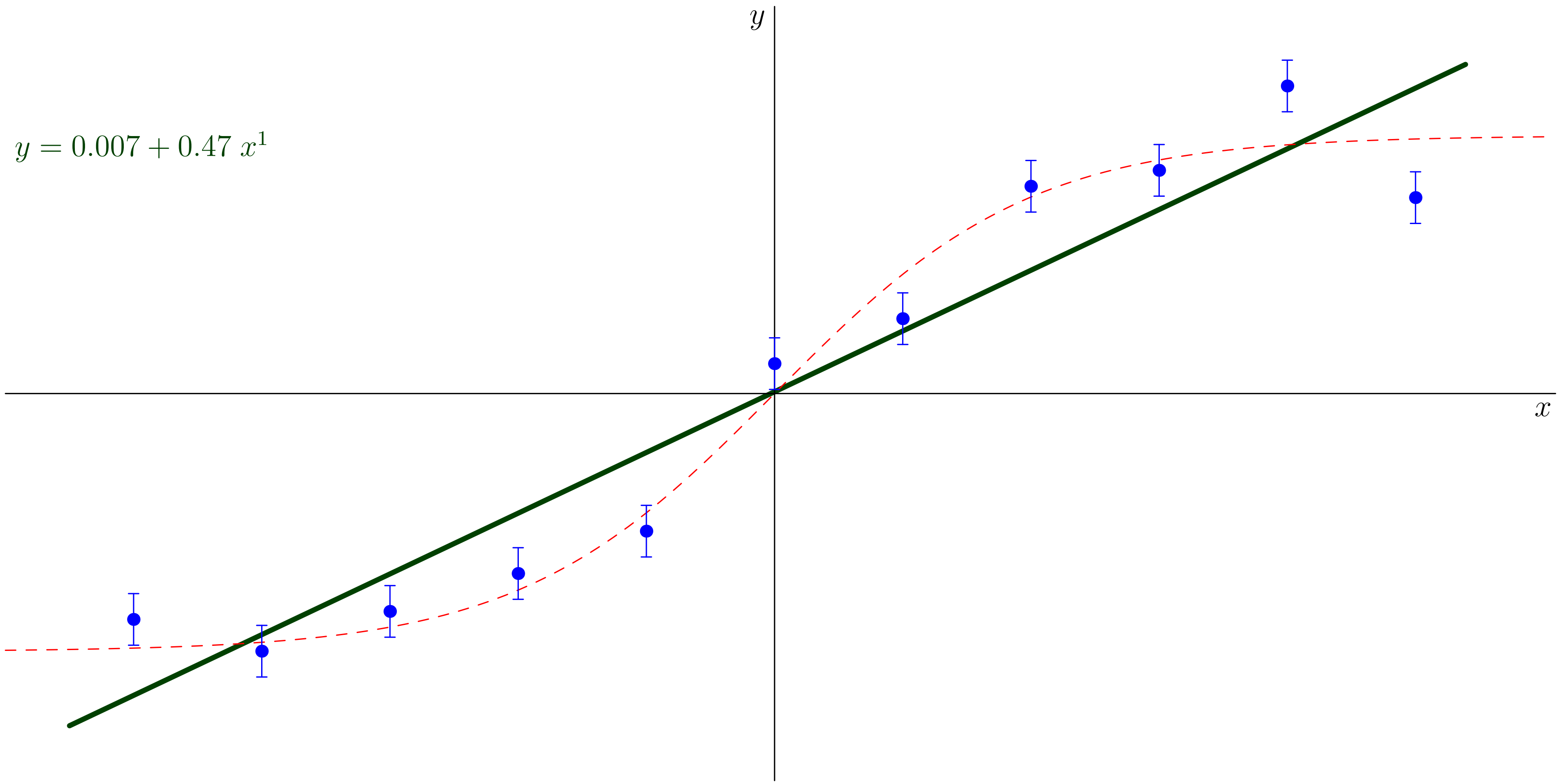
Too Simple or Too Complex?

- Fit $\hat{f}(x, \boldsymbol{w})$ to data



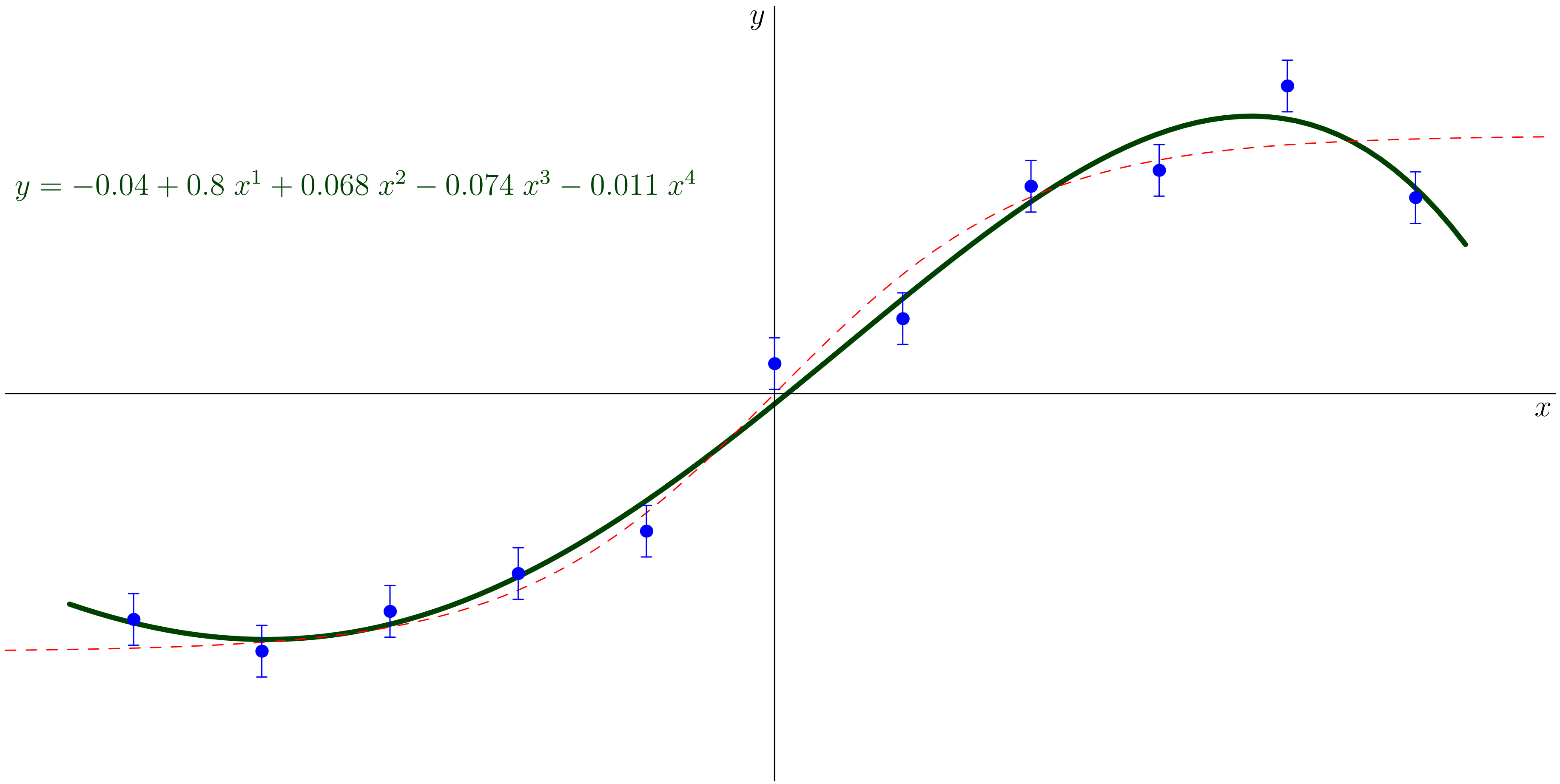
Too Simple or Too Complex?

- Fit $\hat{f}(x, \boldsymbol{w})$ to data



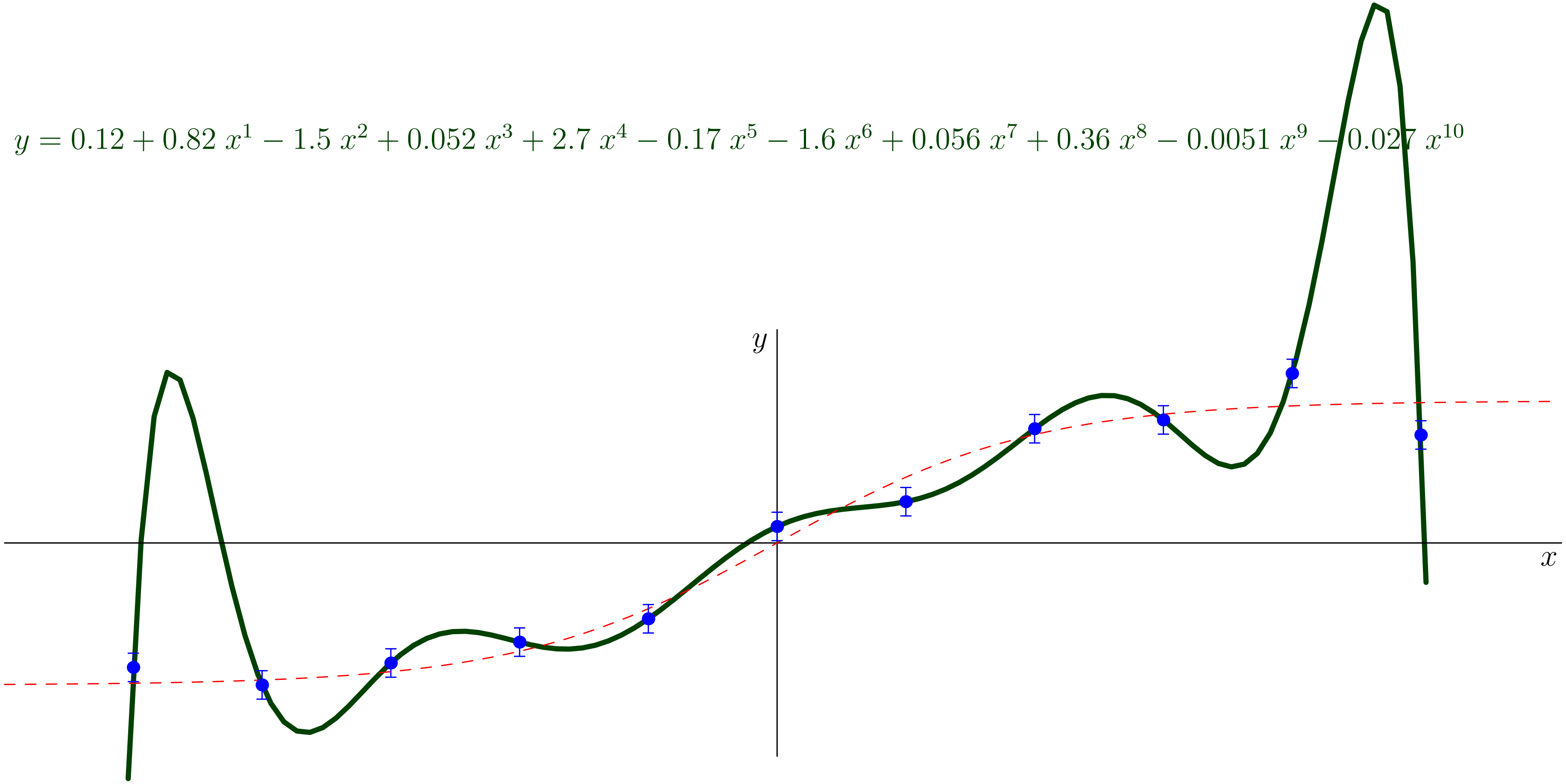
Too Simple or Too Complex?

- Fit $\hat{f}(x, \boldsymbol{w})$ to data



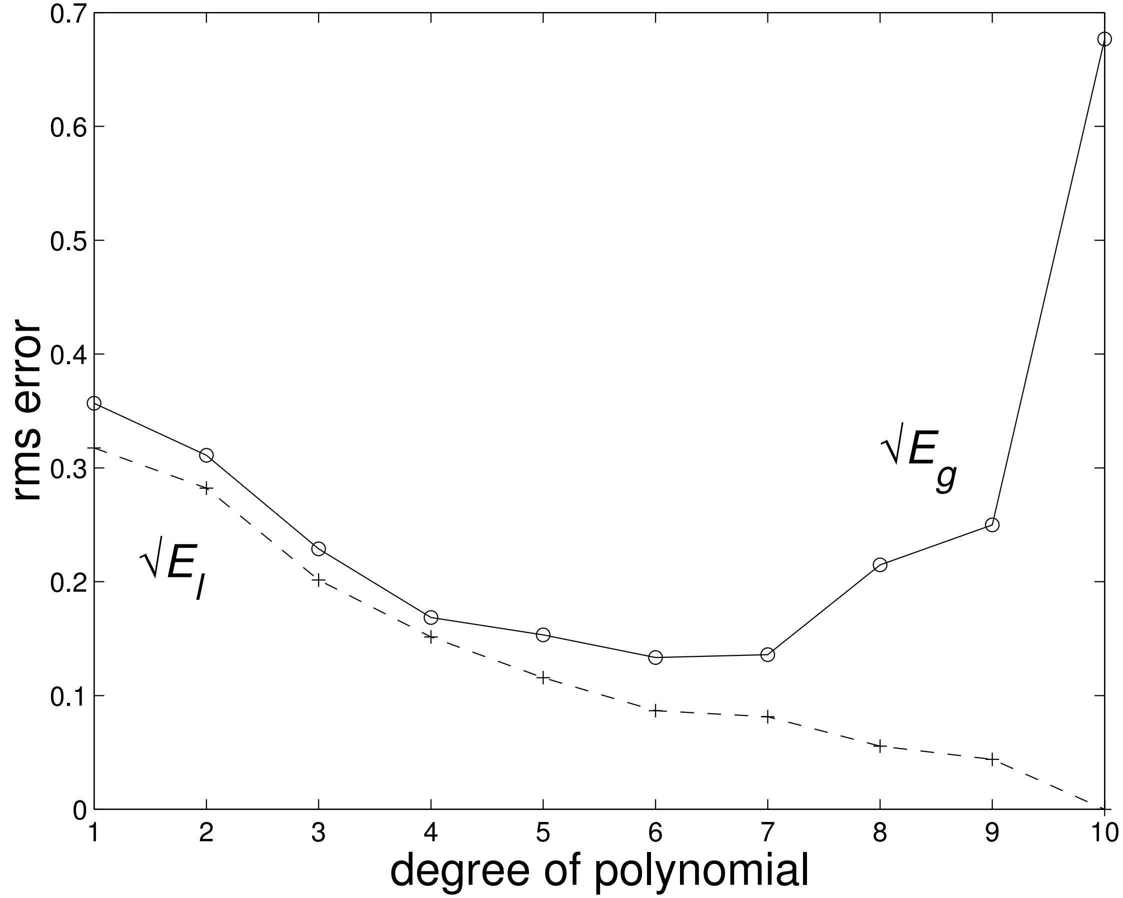
Too Simple or Too Complex?

- Fit $\hat{f}(x, \boldsymbol{w})$ to data



Measuring Generalisation Error for Regression

- Consider the regression example. The root mean squared error is



Expected Generalisation Performance

- Our generalisation performance will depend on our training set, $\mathcal{D}$
- To reason about generalisation we can ask what is the *expected generalisation*, that is, when we average over all different data sets of size m drawn independently from $p(\boldsymbol{x})$
- For each data set, $\mathcal{D}$, we would learn a different approximator $\hat{f}(\boldsymbol{x}|\mathcal{D})$ (usually through weights $\boldsymbol{w}_{\mathcal{D}}$)
- Note that in practice we only get one data set. We might be lucky and do better than the expected generalisation or we might be unlucky and do worse

Expected Generalisation Performance

- Our generalisation performance will depend on our training set, $\mathcal{D}$
- To reason about generalisation we can ask what is the *expected generalisation*, that is, when we average over all different data sets of size m drawn independently from $p(\boldsymbol{x})$
- For each data set, $\mathcal{D}$, we would learn a different approximator $\hat{f}(\boldsymbol{x}|\mathcal{D})$ (usually through weights $\boldsymbol{w}_{\mathcal{D}}$)
- Note that in practice we only get one data set. We might be lucky and do better than the expected generalisation or we might be unlucky and do worse

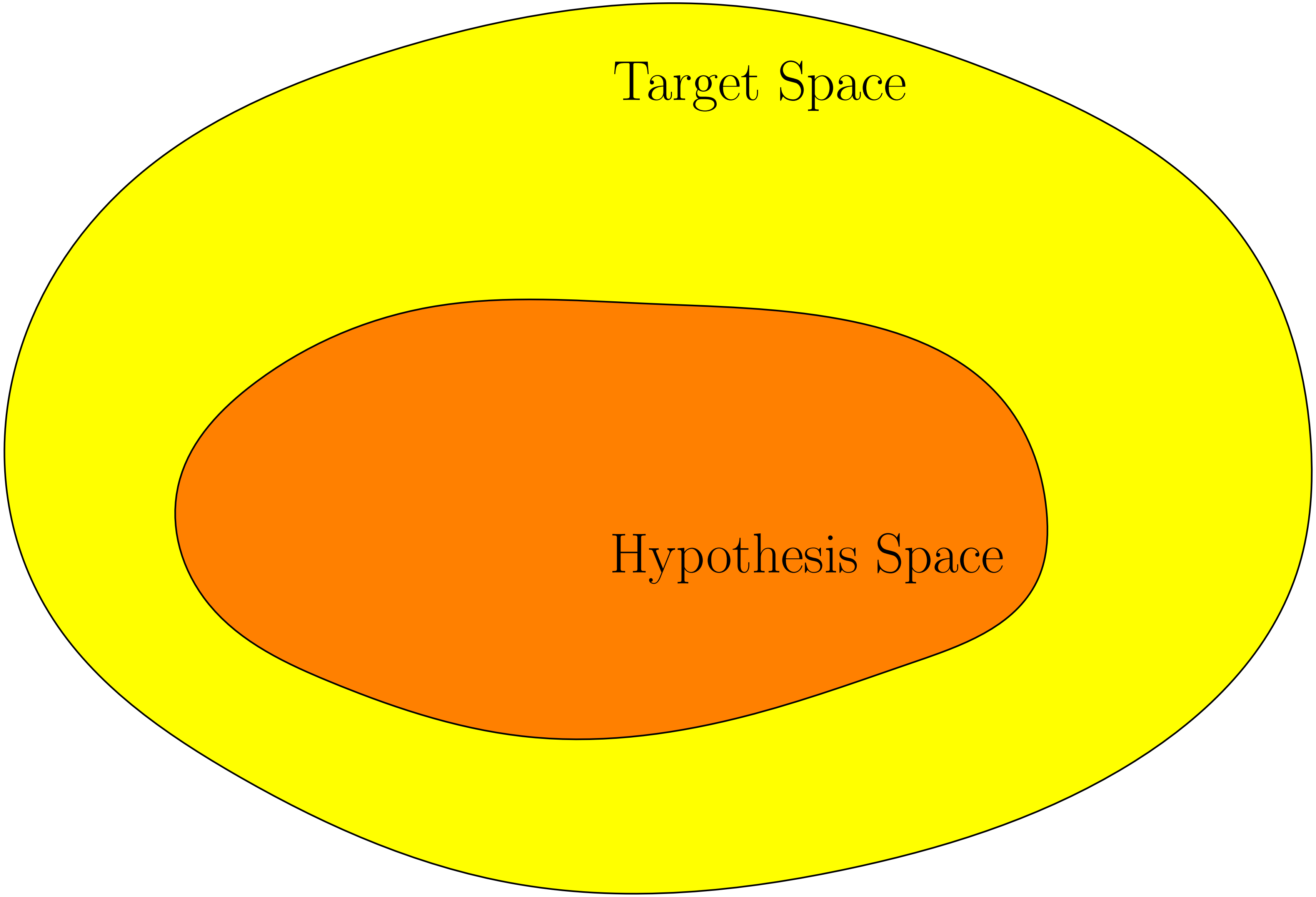
Expected Generalisation Performance

- Our generalisation performance will depend on our training set, $\mathcal{D}$
- To reason about generalisation we can ask what is the *expected generalisation*, that is, when we average over all different data sets of size m drawn independently from $p(\boldsymbol{x})$
- For each data set, $\mathcal{D}$, we would learn a different approximator $\hat{f}(\boldsymbol{x}|\mathcal{D})$ (usually through weights $\boldsymbol{w}_{\mathcal{D}}$)
- Note that in practice we only get one data set. We might be lucky and do better than the expected generalisation or we might be unlucky and do worse

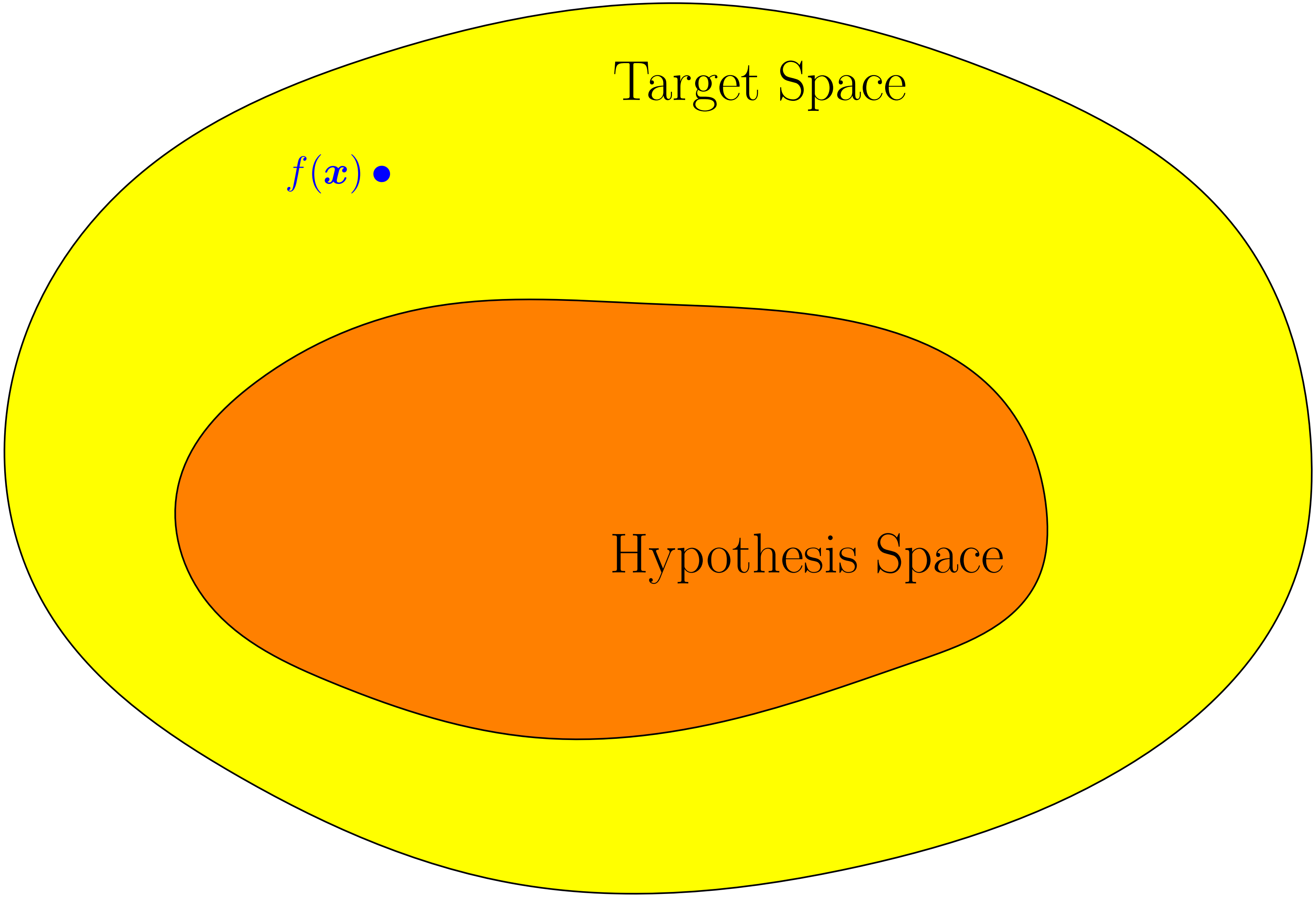
Expected Generalisation Performance

- Our generalisation performance will depend on our training set, $\mathcal{D}$
- To reason about generalisation we can ask what is the *expected generalisation*, that is, when we average over all different data sets of size m drawn independently from $p(\boldsymbol{x})$
- For each data set, $\mathcal{D}$, we would learn a different approximator $\hat{f}(\boldsymbol{x}|\mathcal{D})$ (usually through weights $\boldsymbol{w}_{\mathcal{D}}$)
- Note that in practice we only get one data set. We might be lucky and do better than the expected generalisation or we might be unlucky and do worse

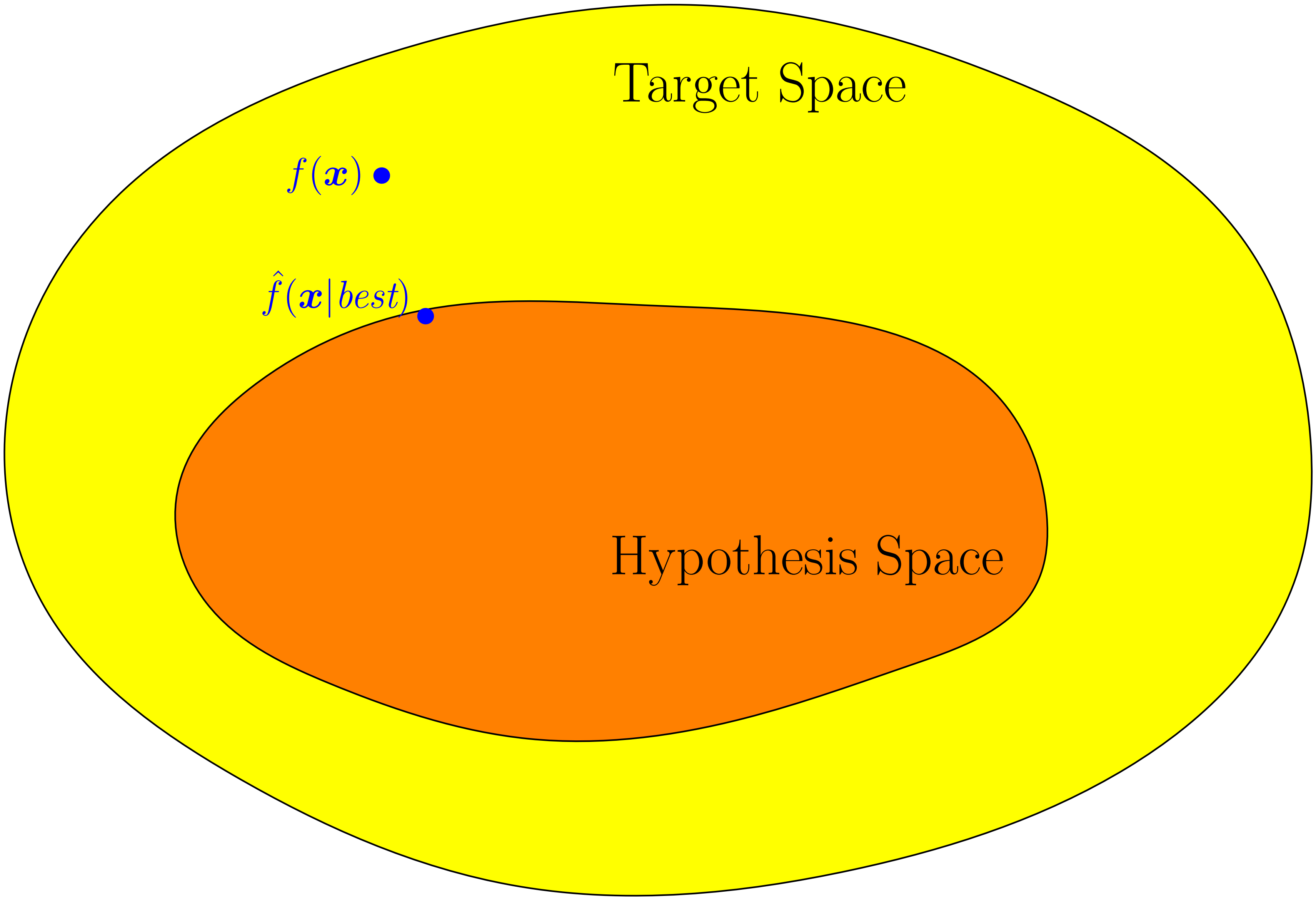
Approximation and Estimation Errors



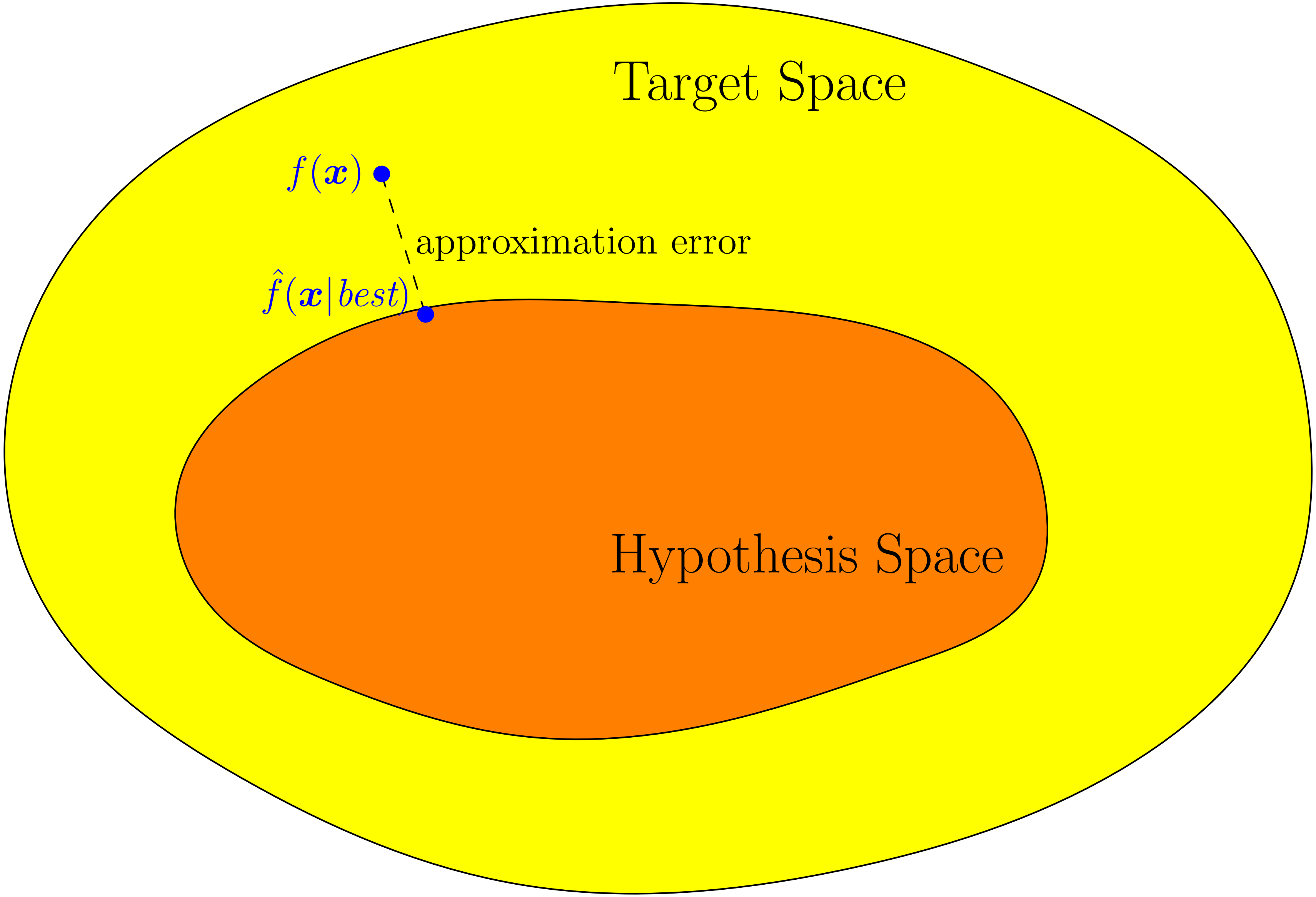
Approximation and Estimation Errors



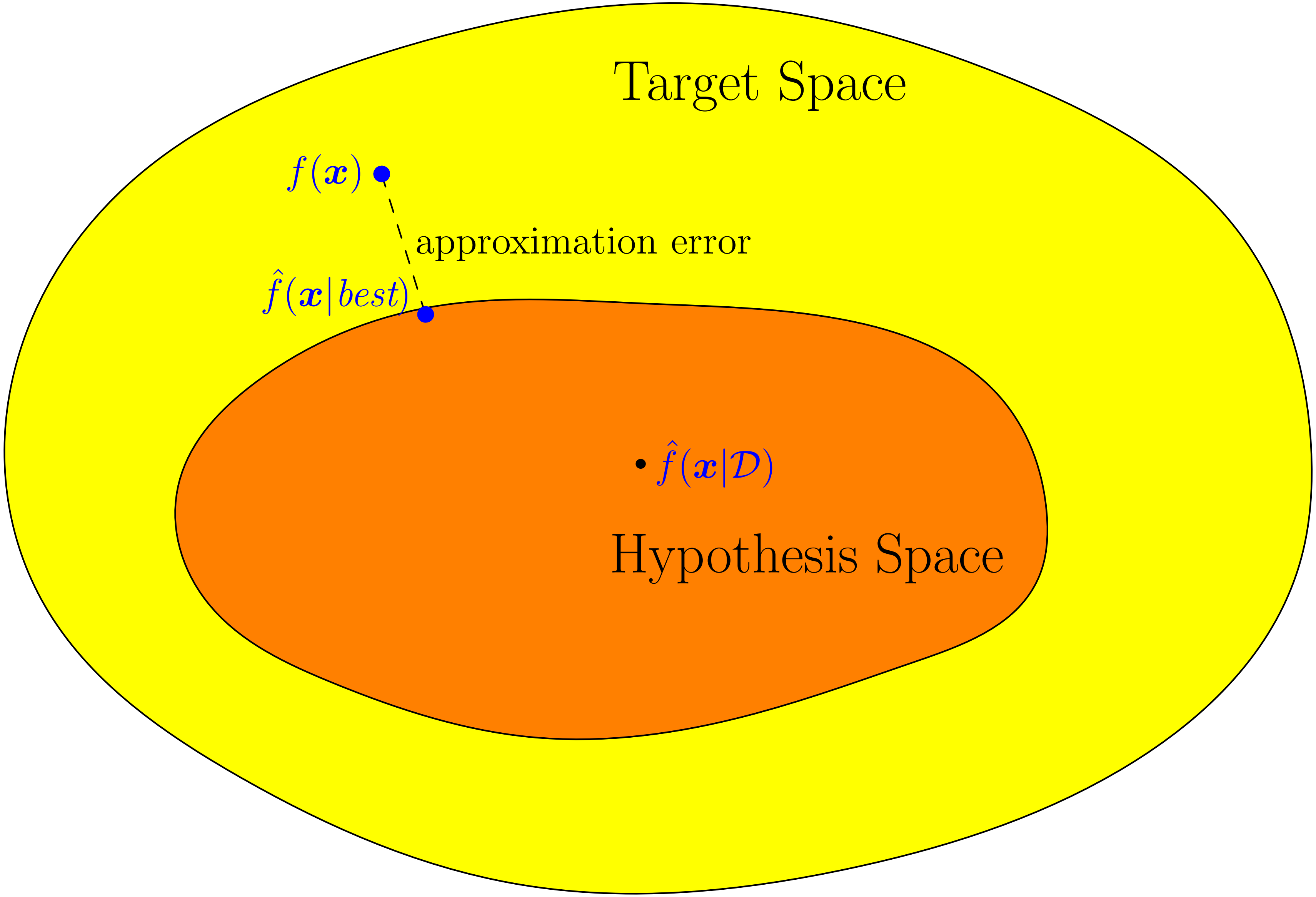
Approximation and Estimation Errors



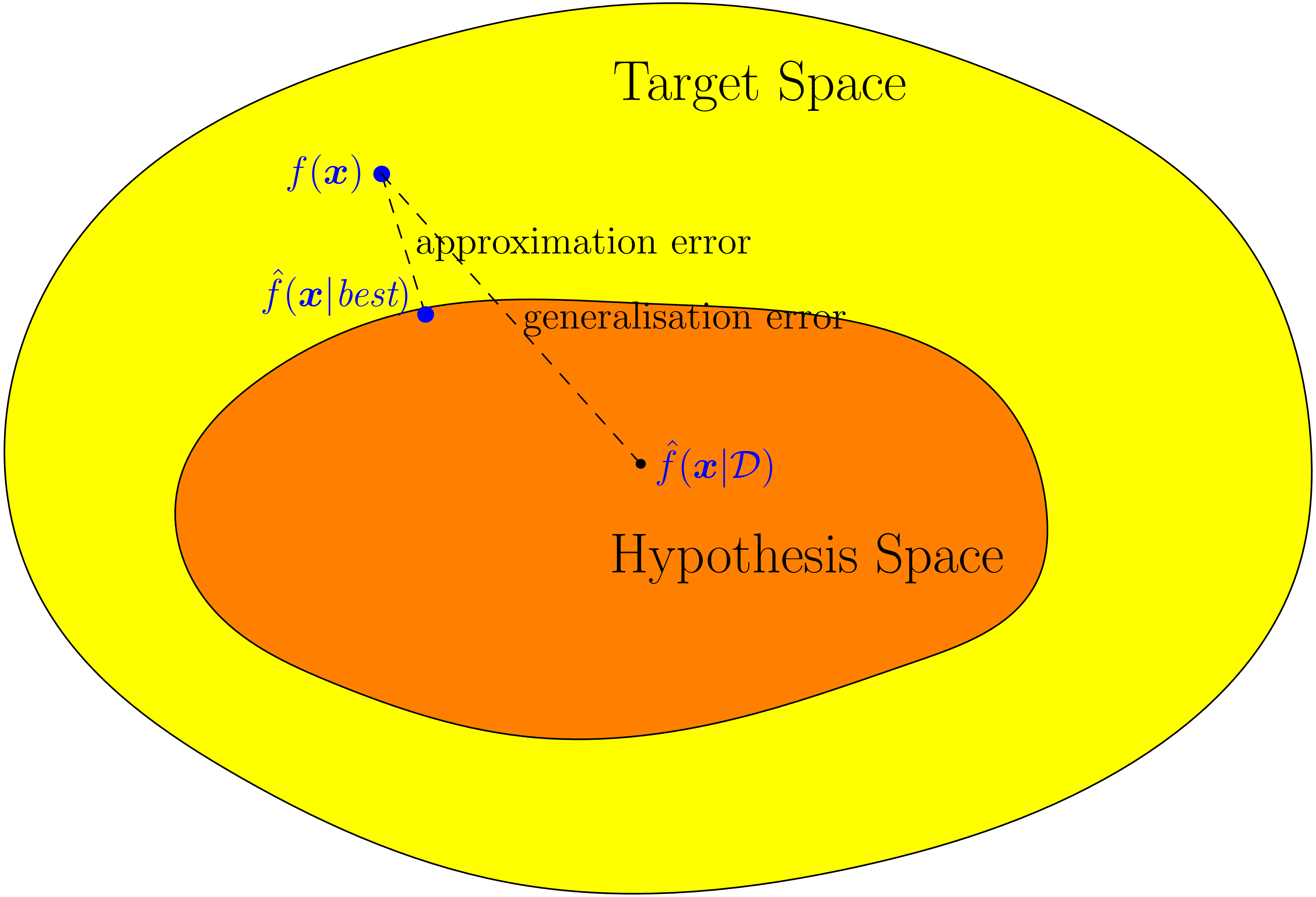
Approximation and Estimation Errors



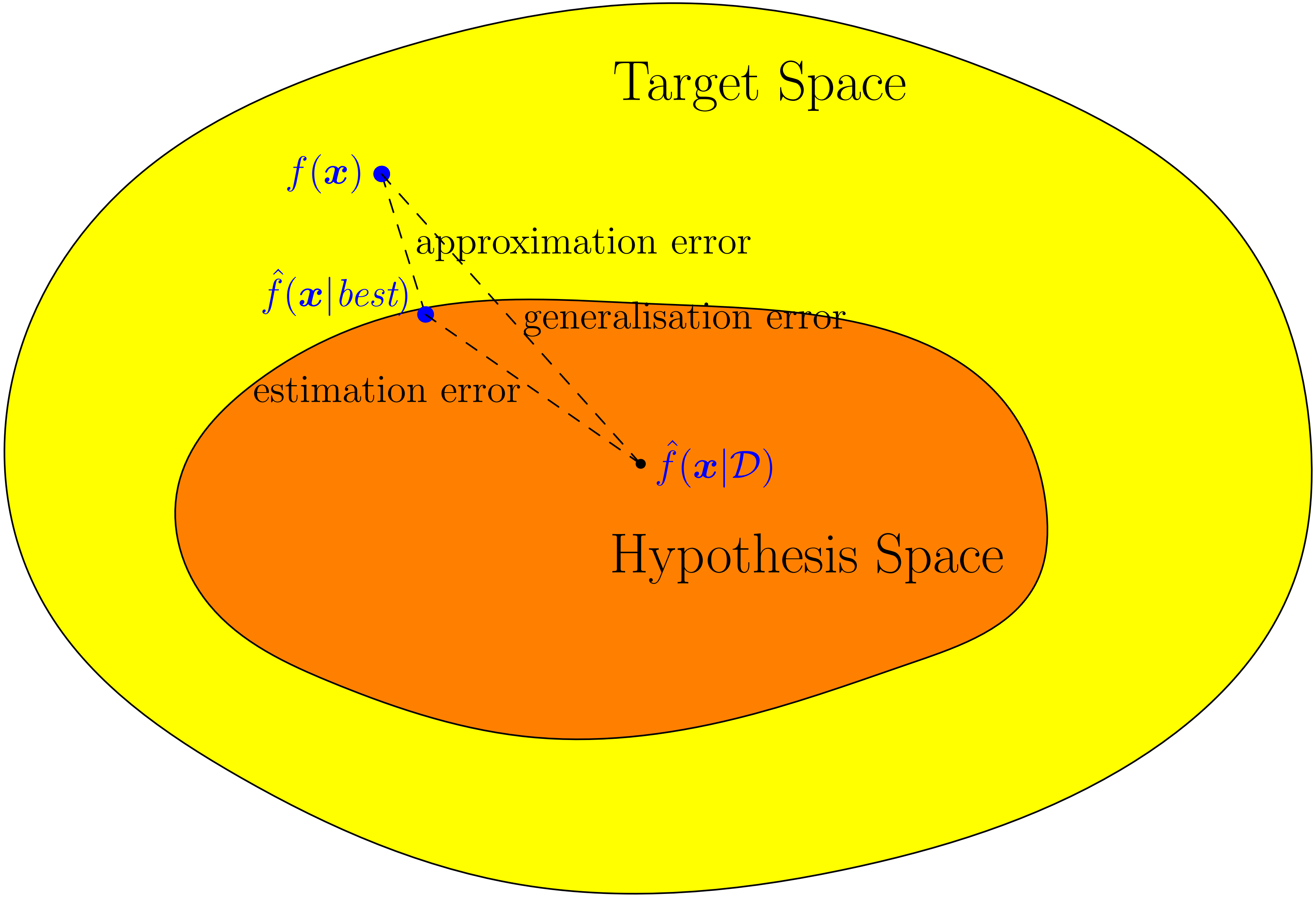
Approximation and Estimation Errors



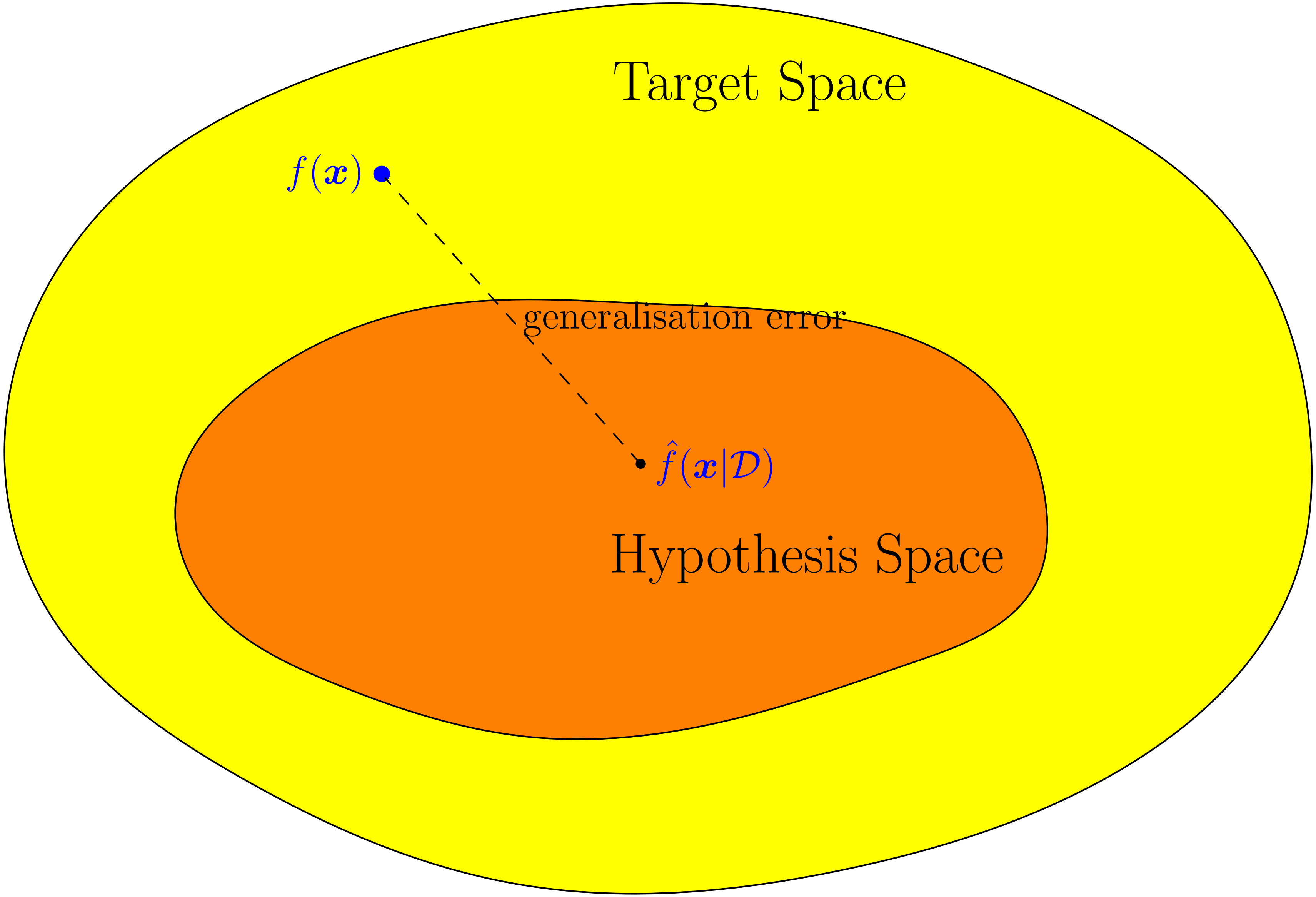
Approximation and Estimation Errors



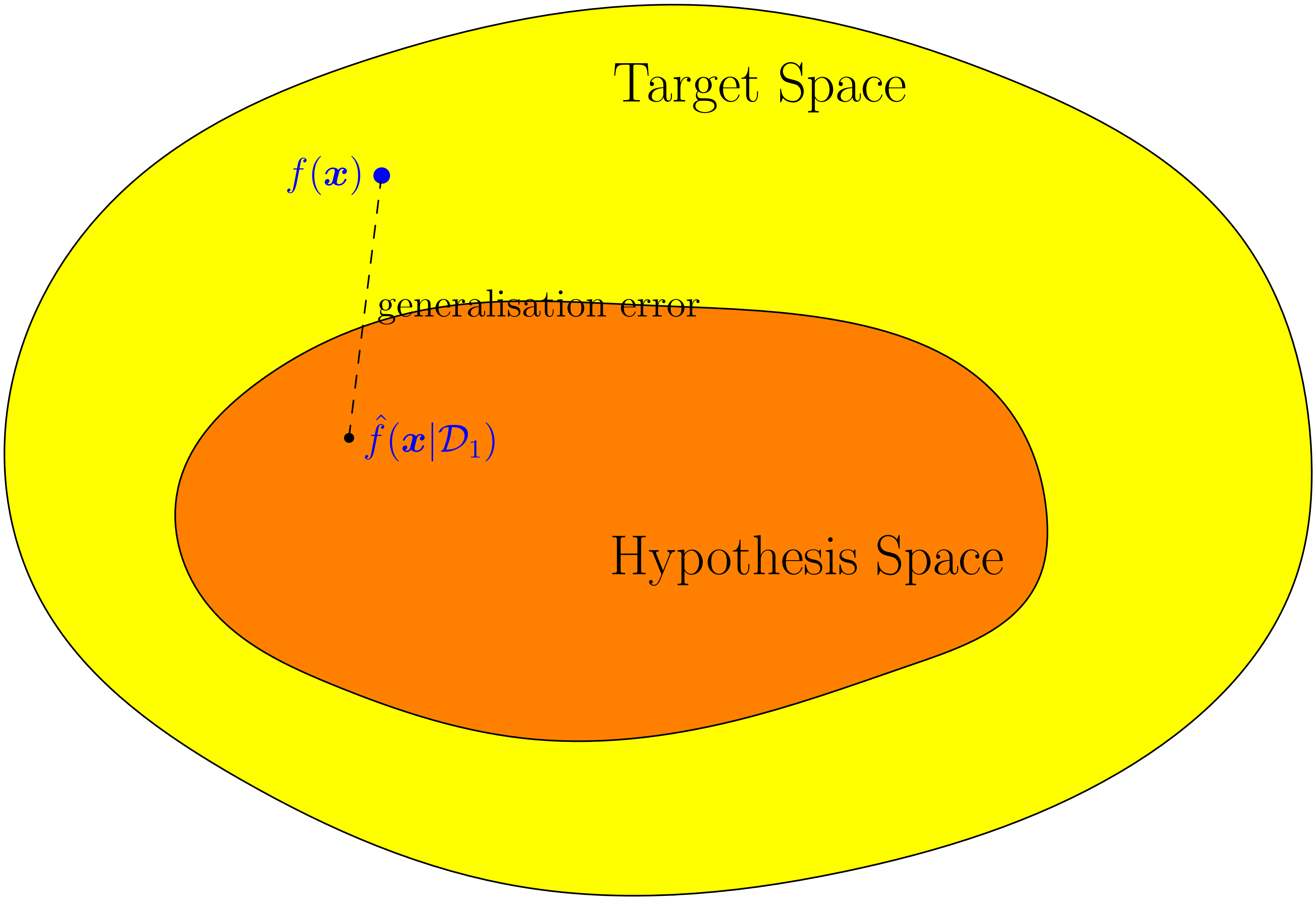
Approximation and Estimation Errors



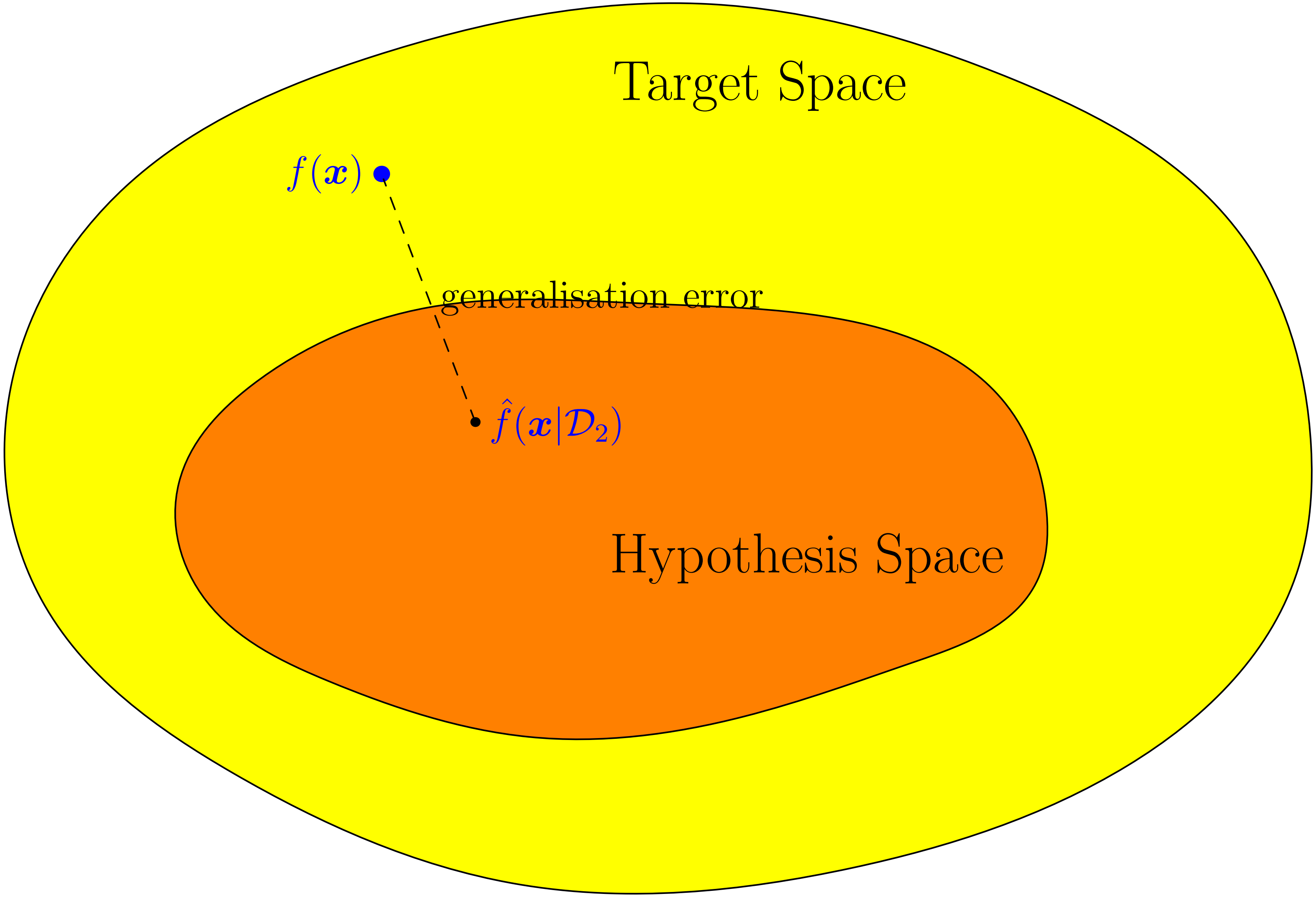
Approximation and Estimation Errors



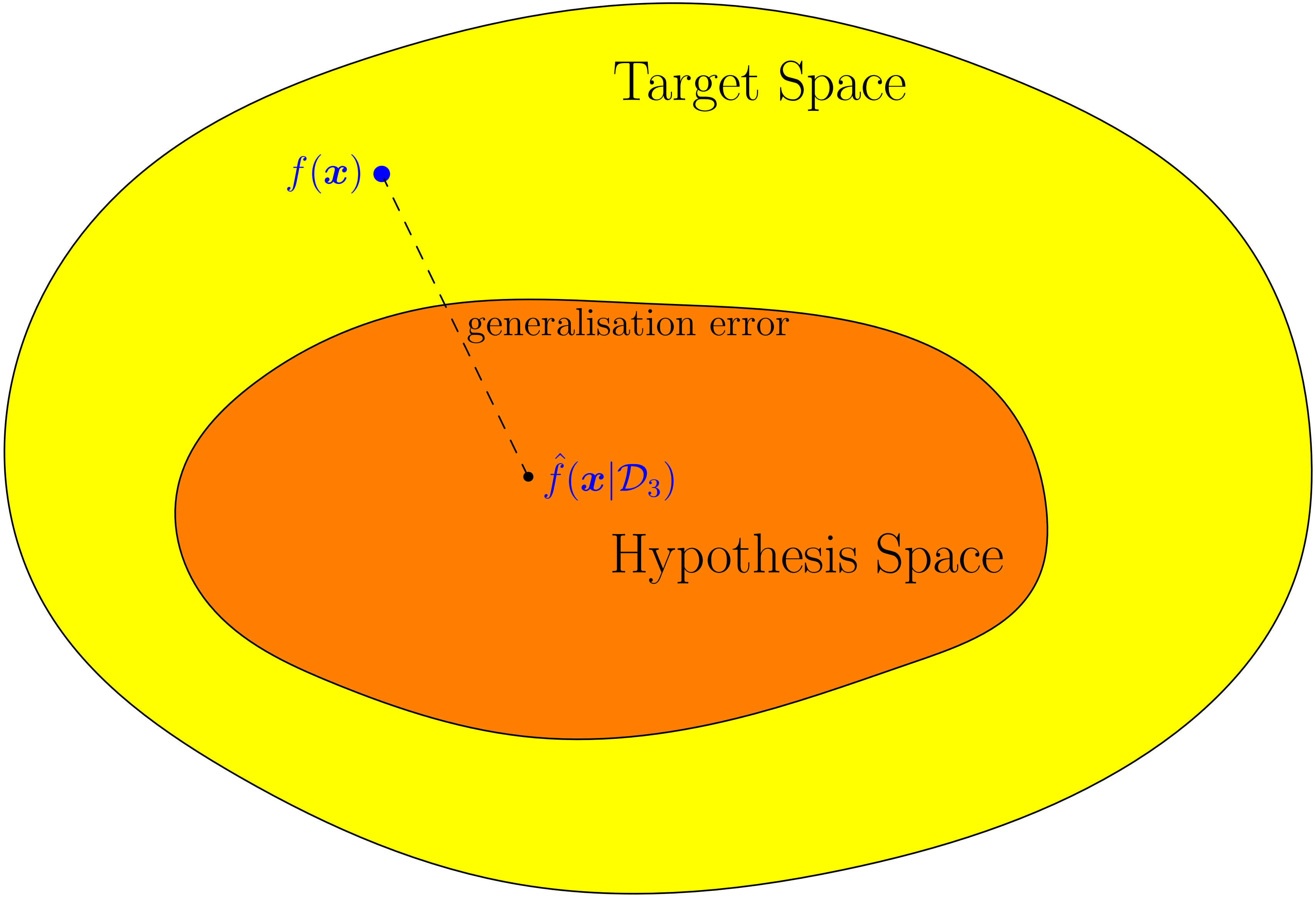
Approximation and Estimation Errors



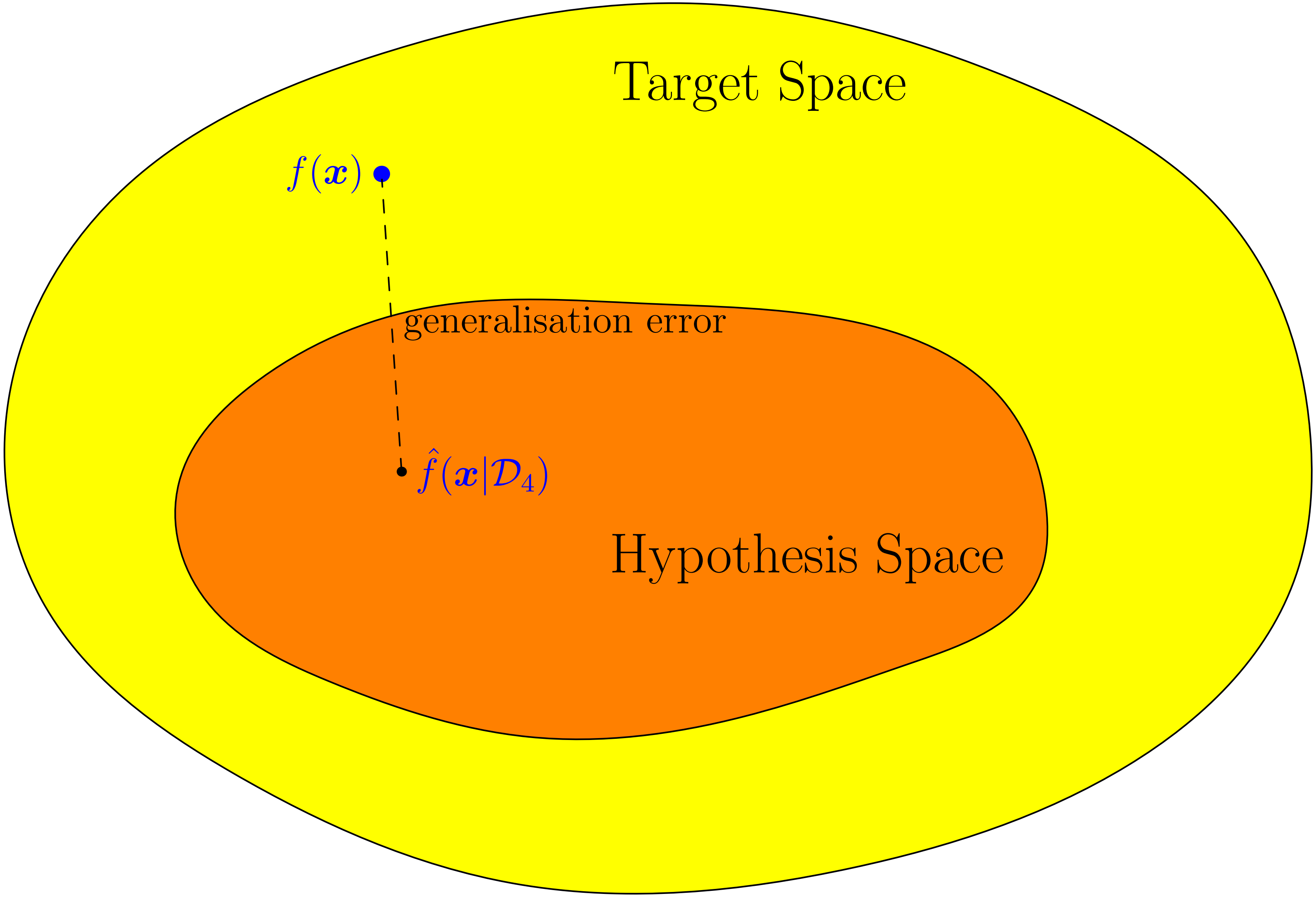
Approximation and Estimation Errors



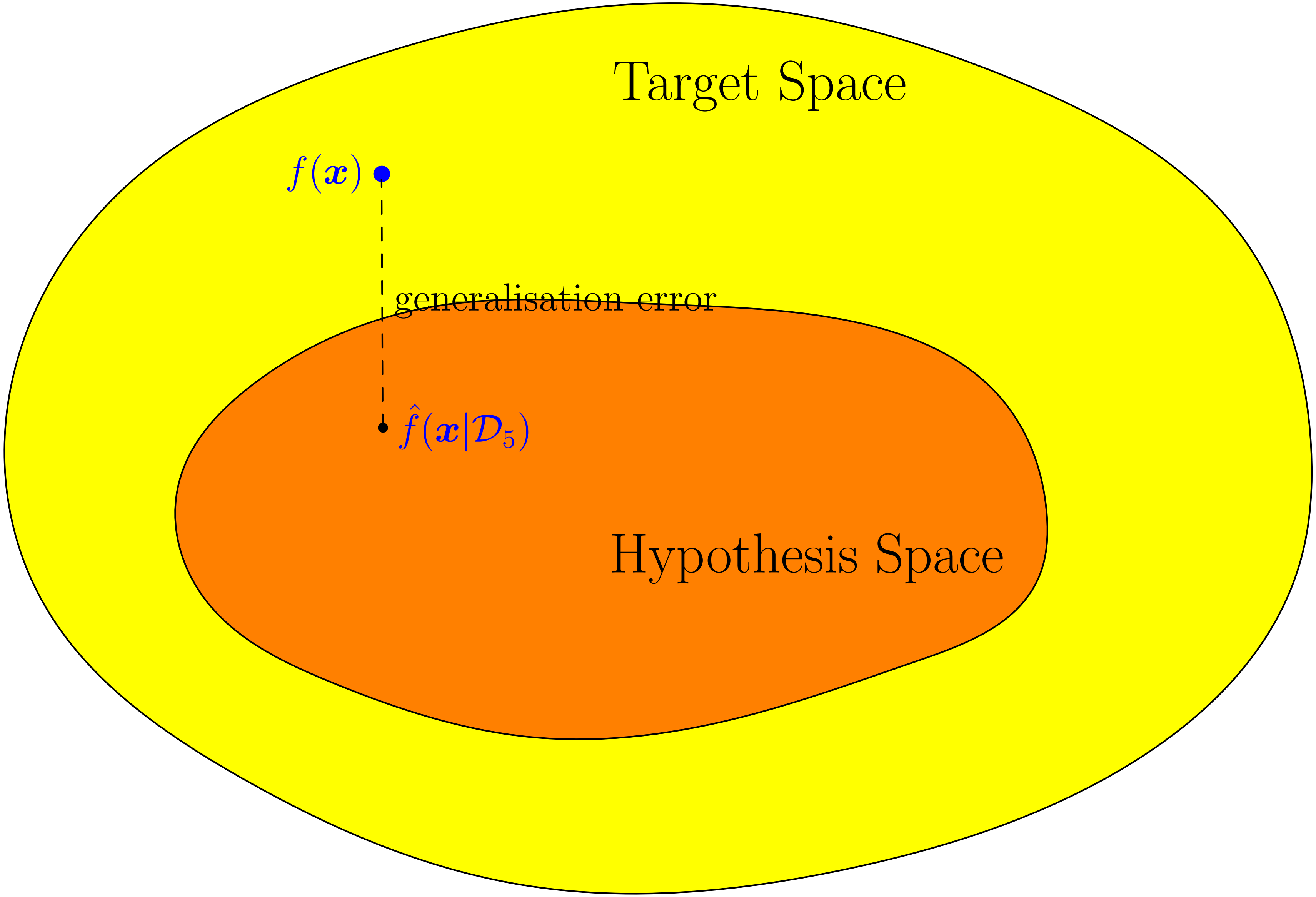
Approximation and Estimation Errors



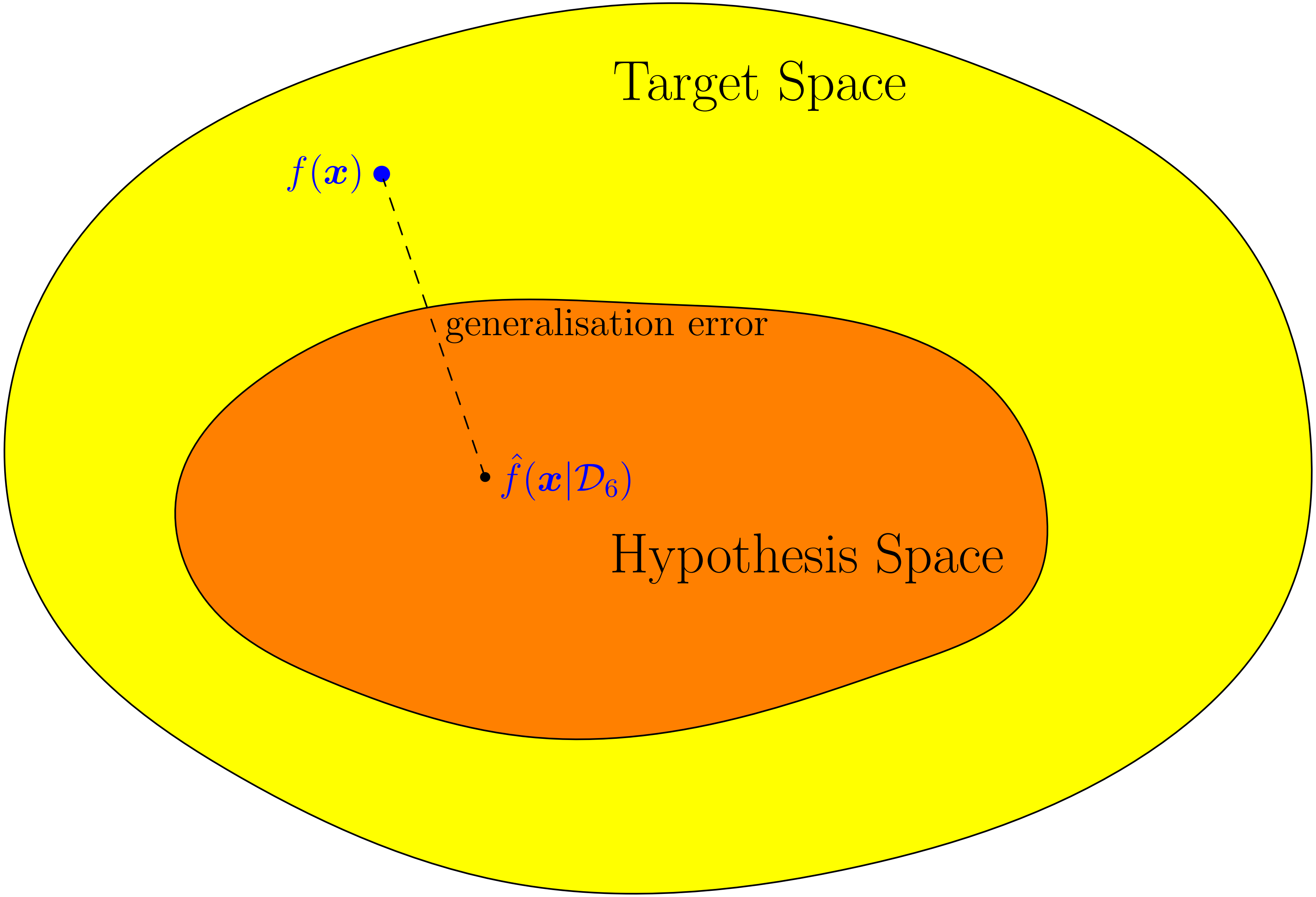
Approximation and Estimation Errors



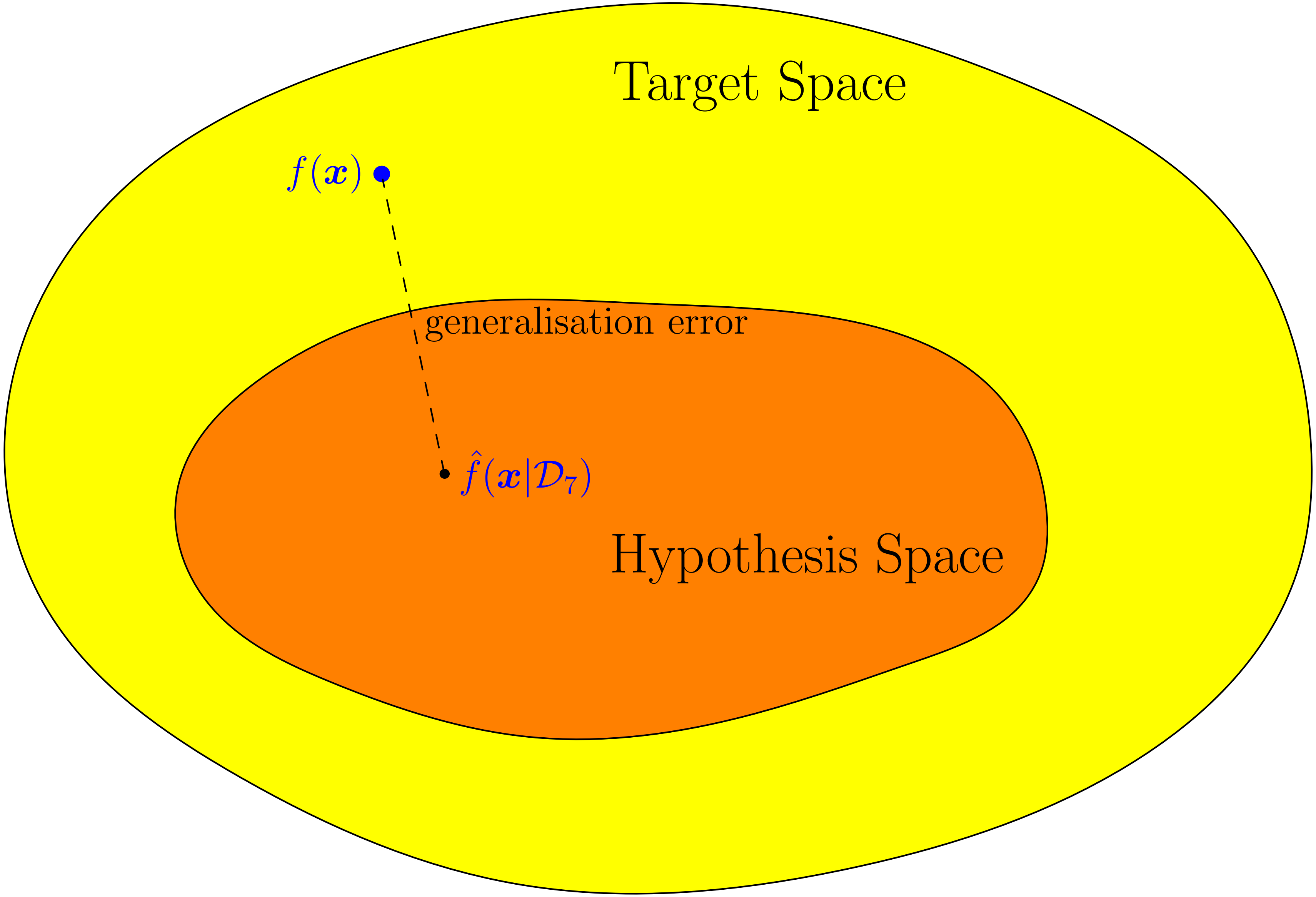
Approximation and Estimation Errors



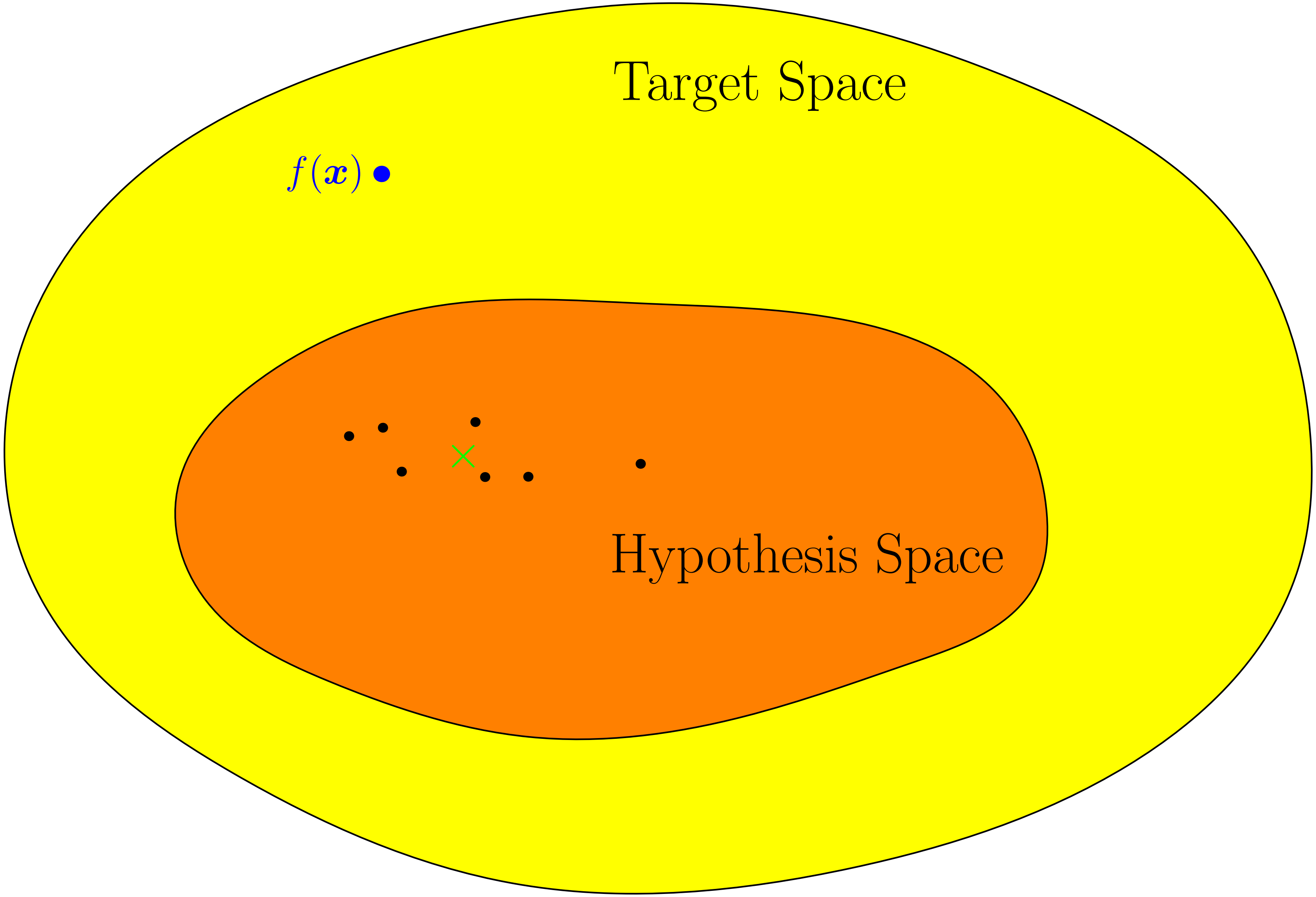
Approximation and Estimation Errors



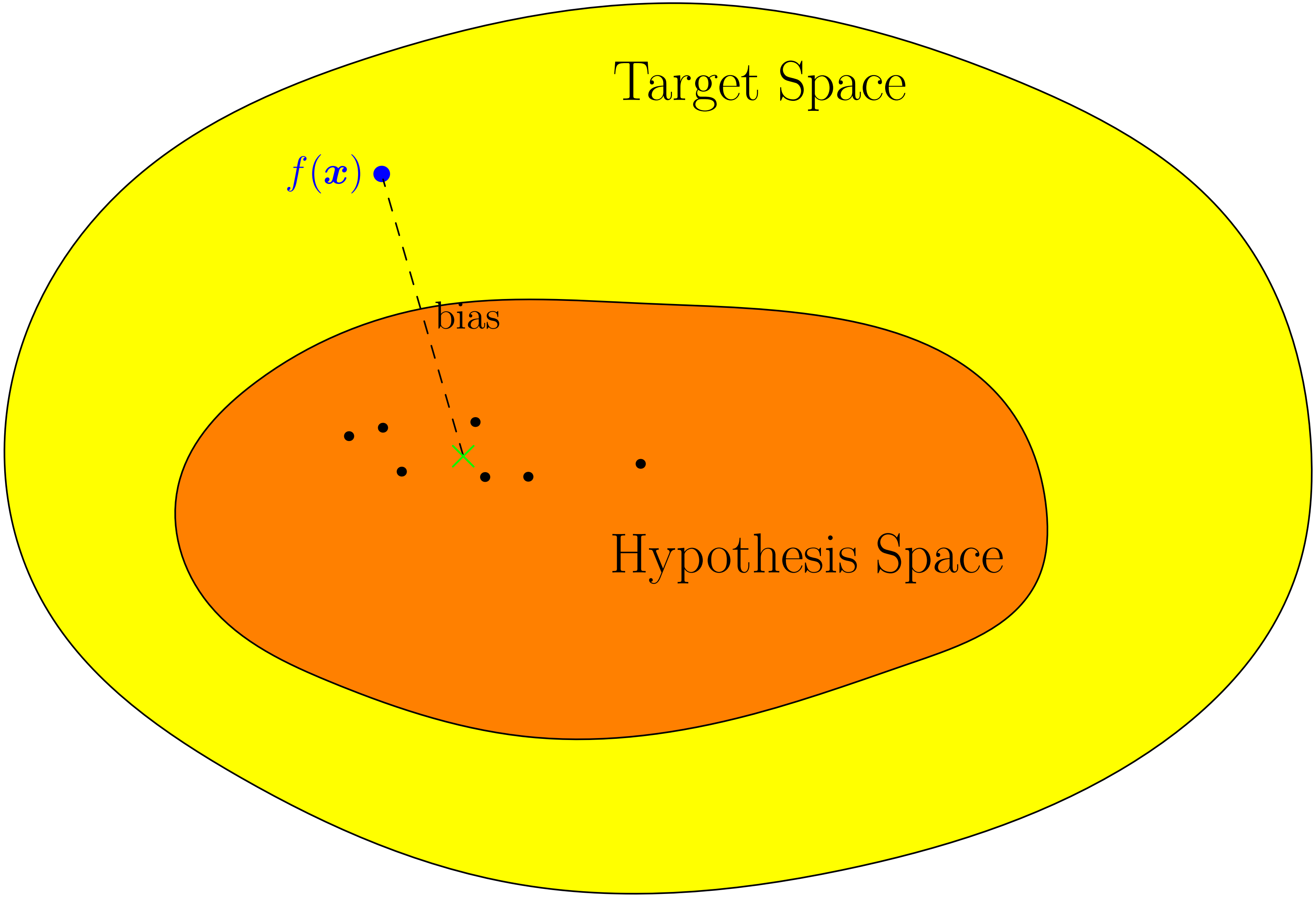
Approximation and Estimation Errors



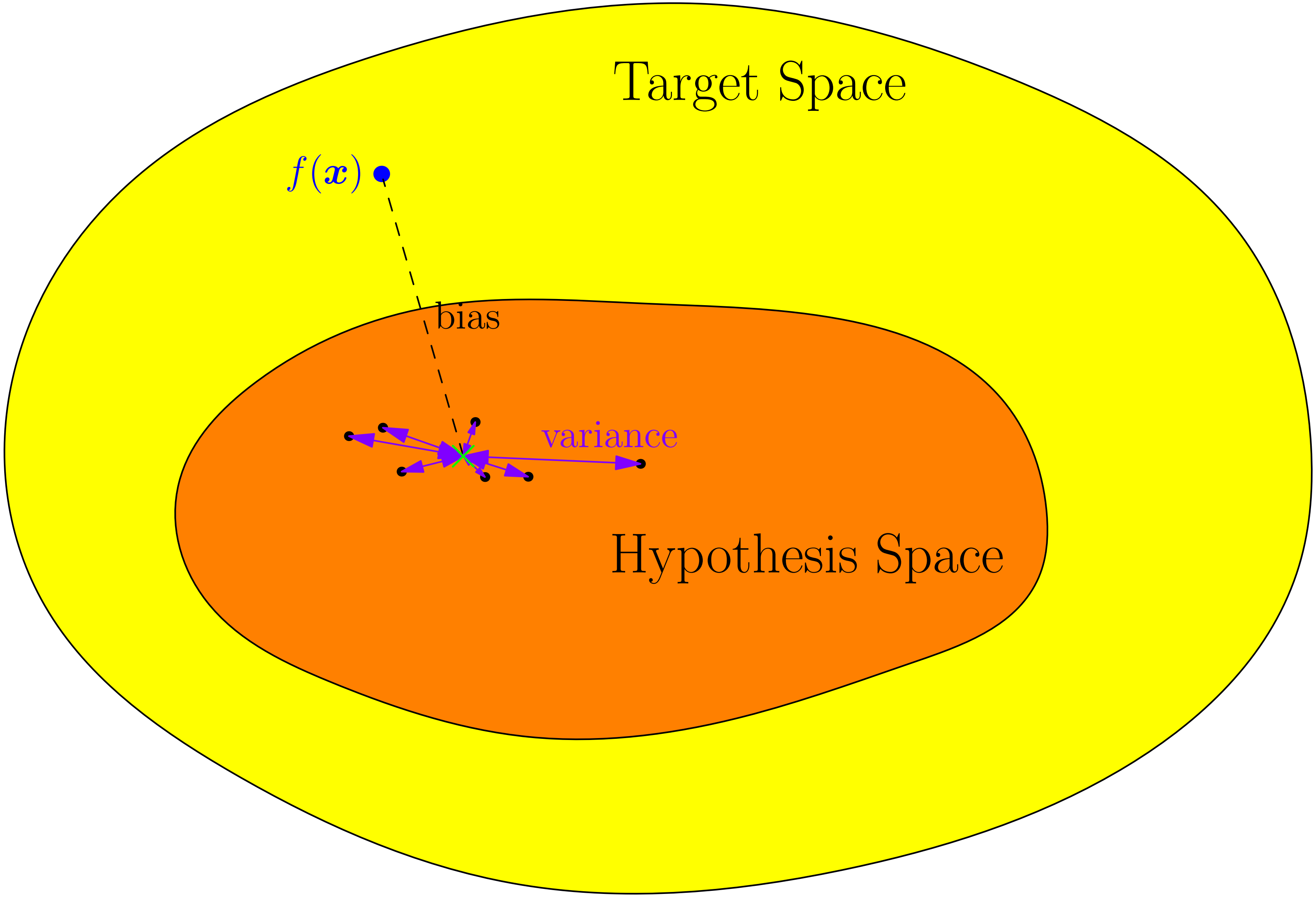
Approximation and Estimation Errors



Approximation and Estimation Errors



Approximation and Estimation Errors



Mean Machine

- To help understand generalisation we can consider the mean prediction with respect to machines trained with all data sets of size m

$$\hat{f}_m(\boldsymbol{x}) = \mathbb{E}_{\mathcal{D}}\left[\hat{f}\left(\boldsymbol{x}|\mathcal{D}\right)\right]$$

- We can define the **bias** to be generalisation performance of the mean machine

$$B = \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}_m(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2$$

Mean Machine

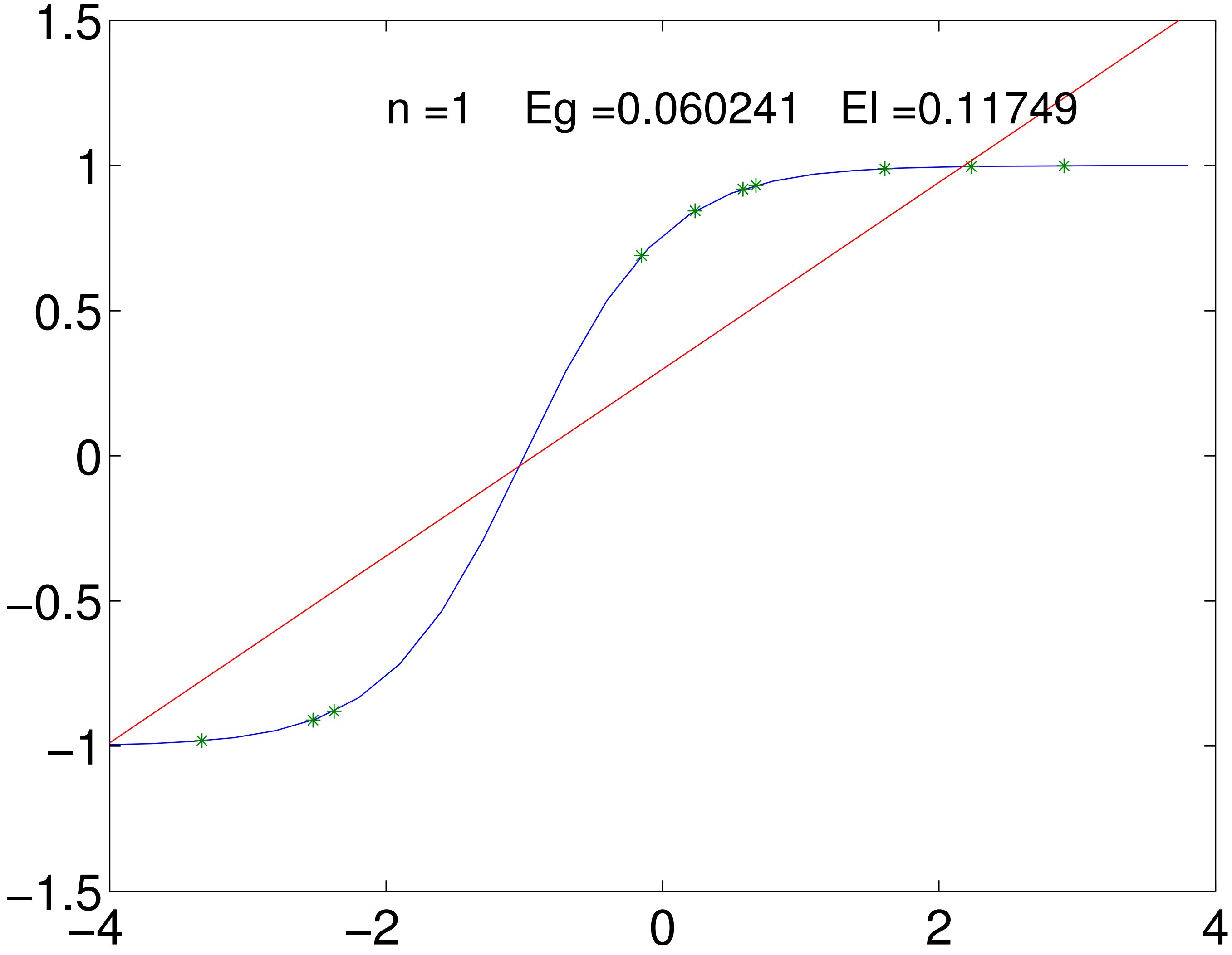
- To help understand generalisation we can consider the mean prediction with respect to machines trained with all data sets of size m

$$\hat{f}_m(\boldsymbol{x}) = \mathbb{E}_{\mathcal{D}}\left[\hat{f}\left(\boldsymbol{x}|\mathcal{D}\right)\right]$$

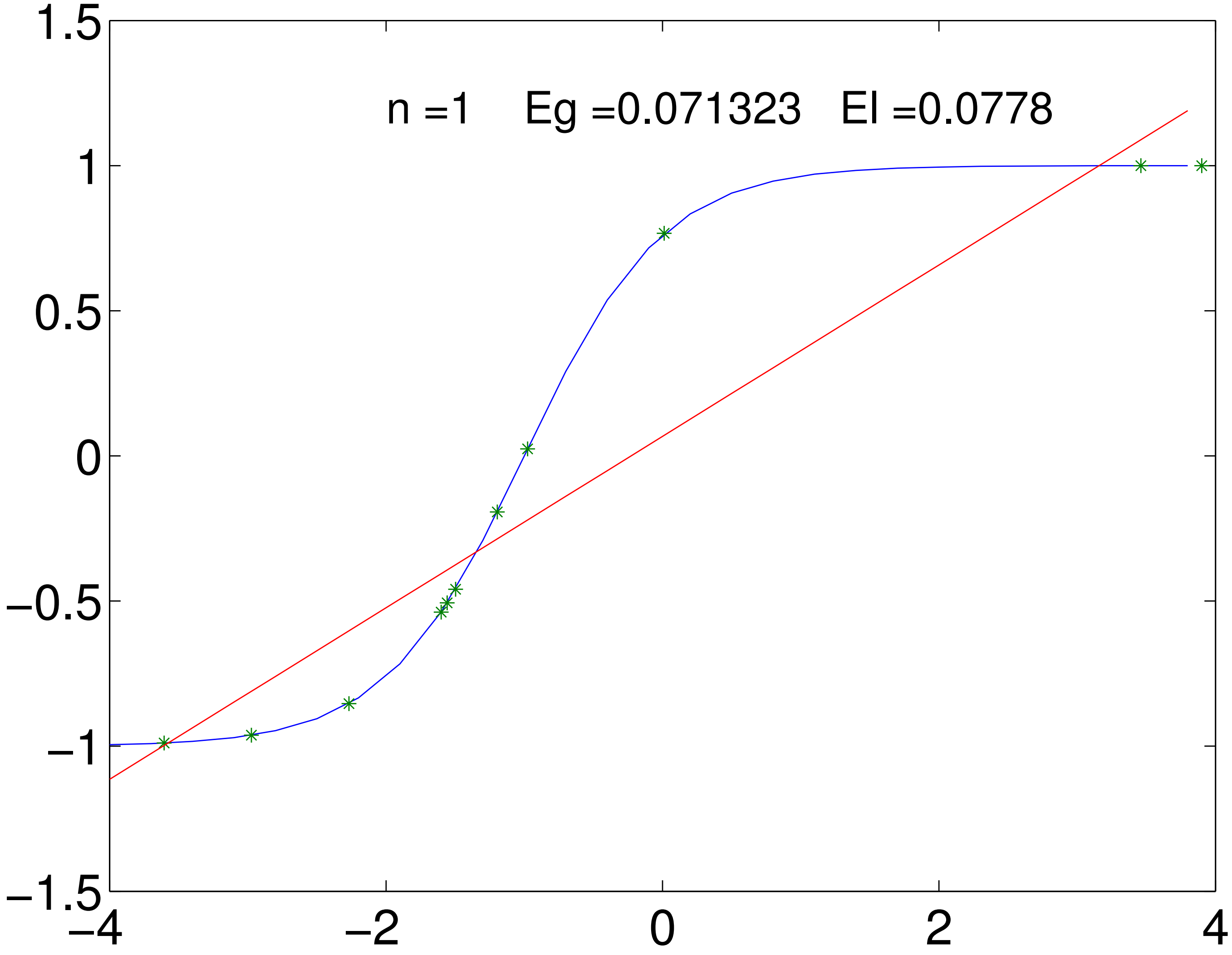
- We can define the **bias** to be generalisation performance of the mean machine

$$B = \sum_{\boldsymbol{x} \in \mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}_m(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2$$

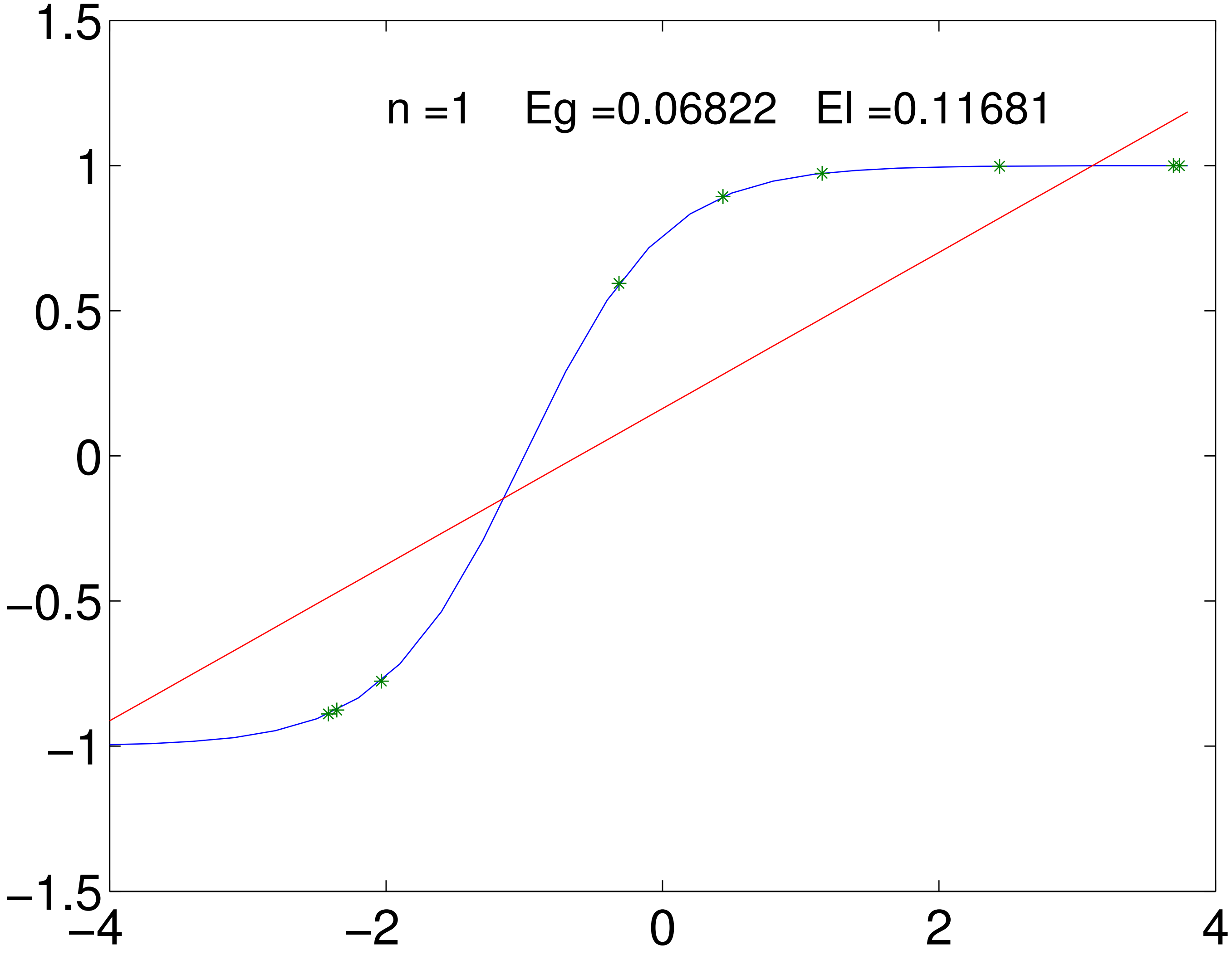
Regression Example $n = 1$



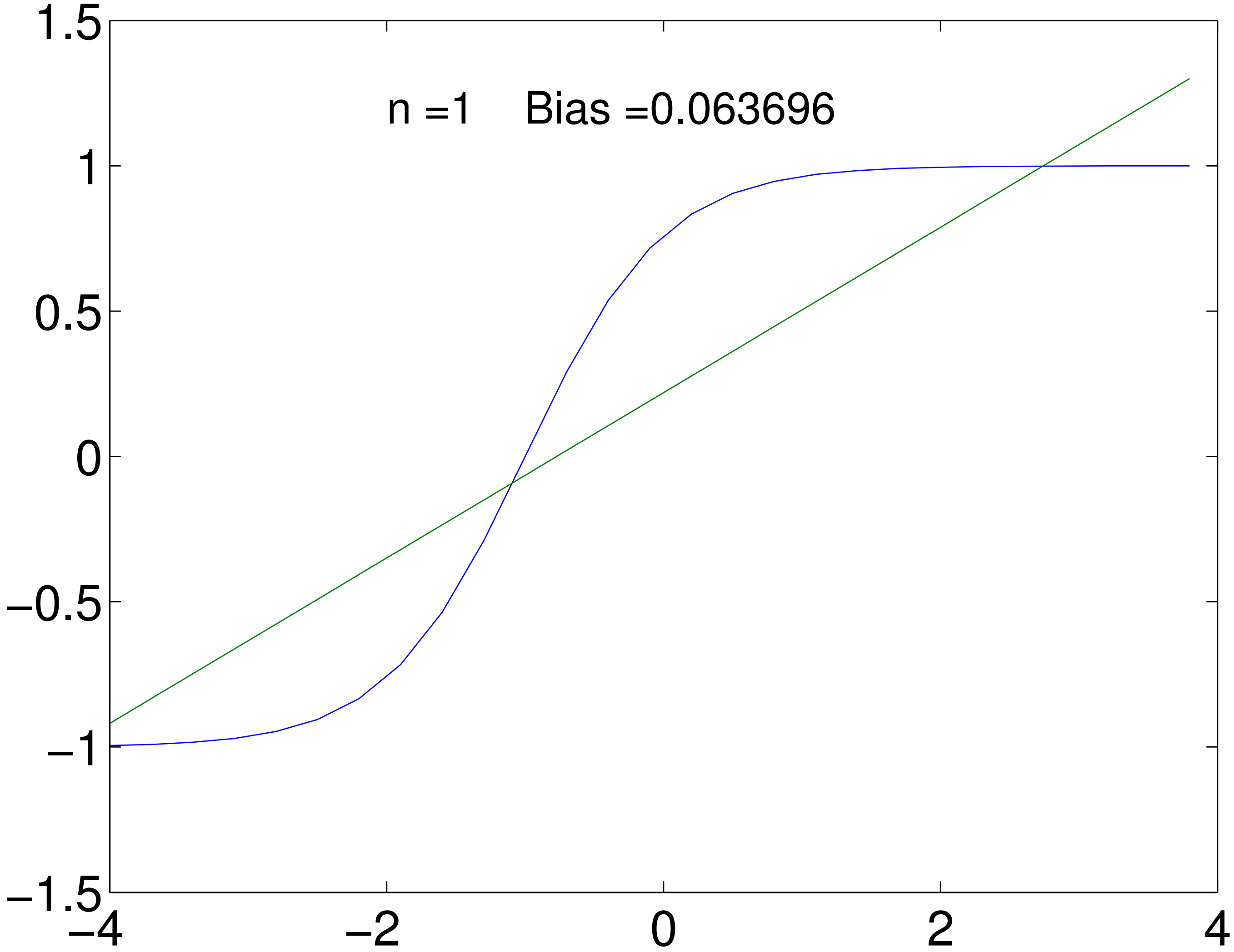
Regression Example $n = 1$



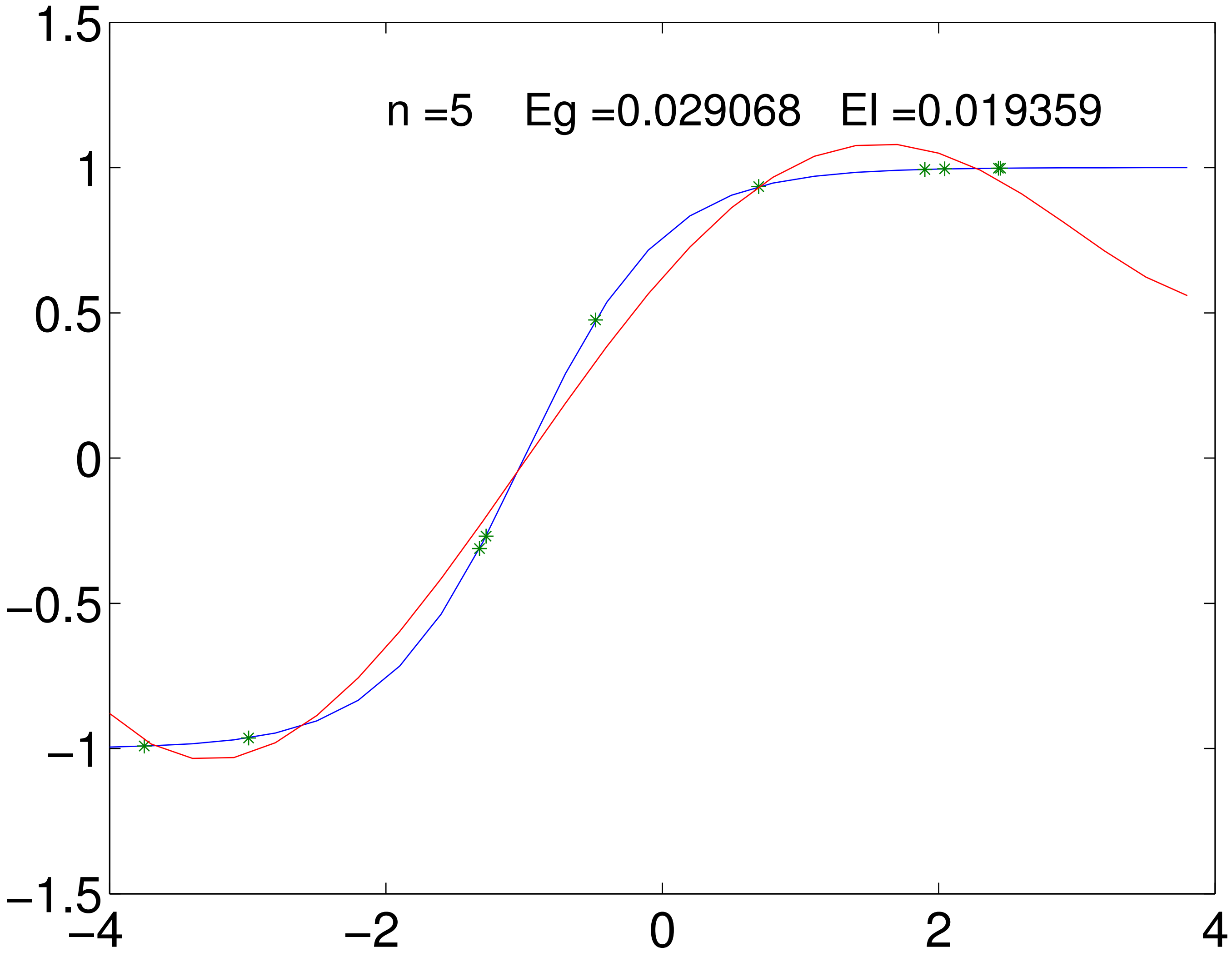
Regression Example $n = 1$



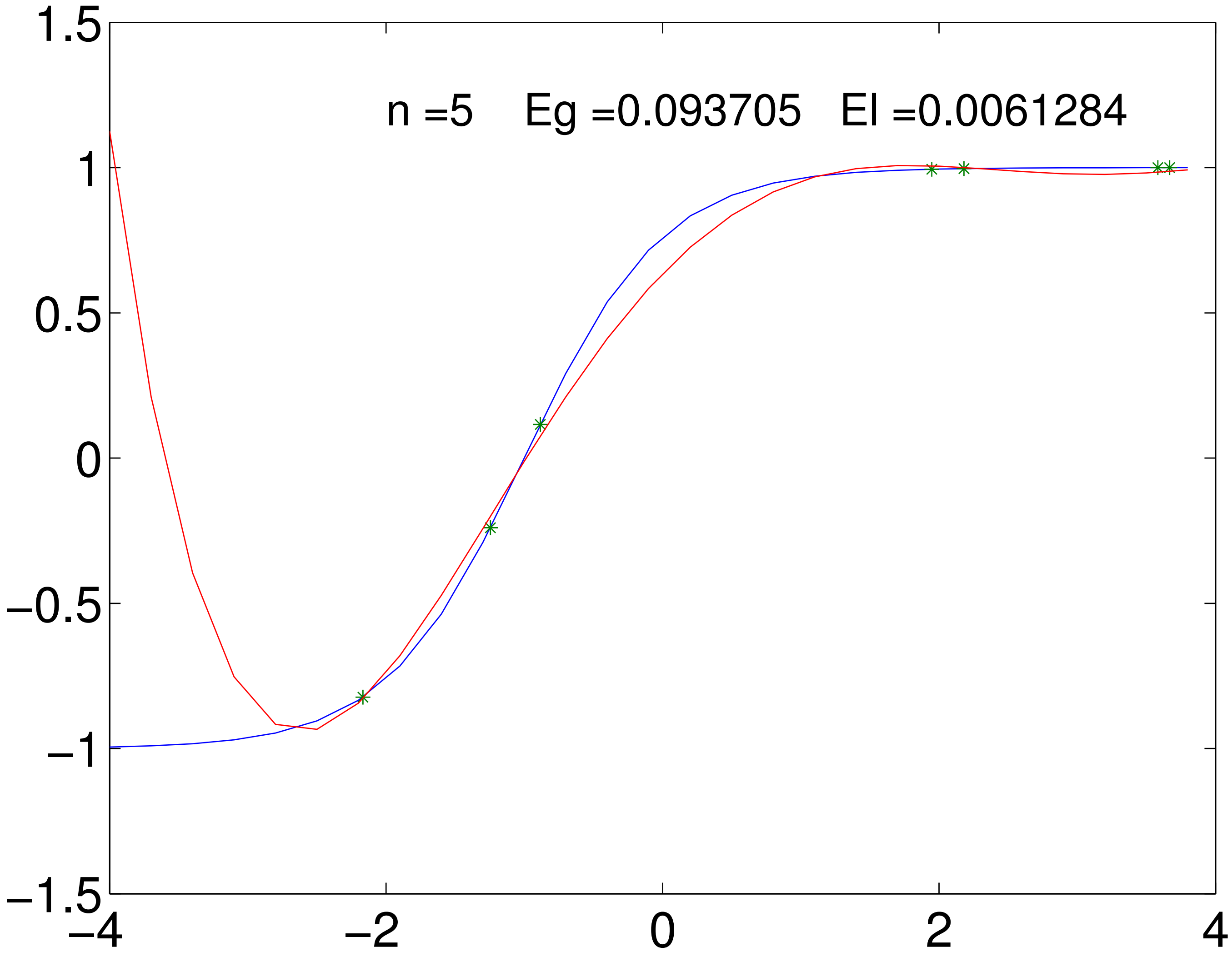
Regression Example $n = 1$



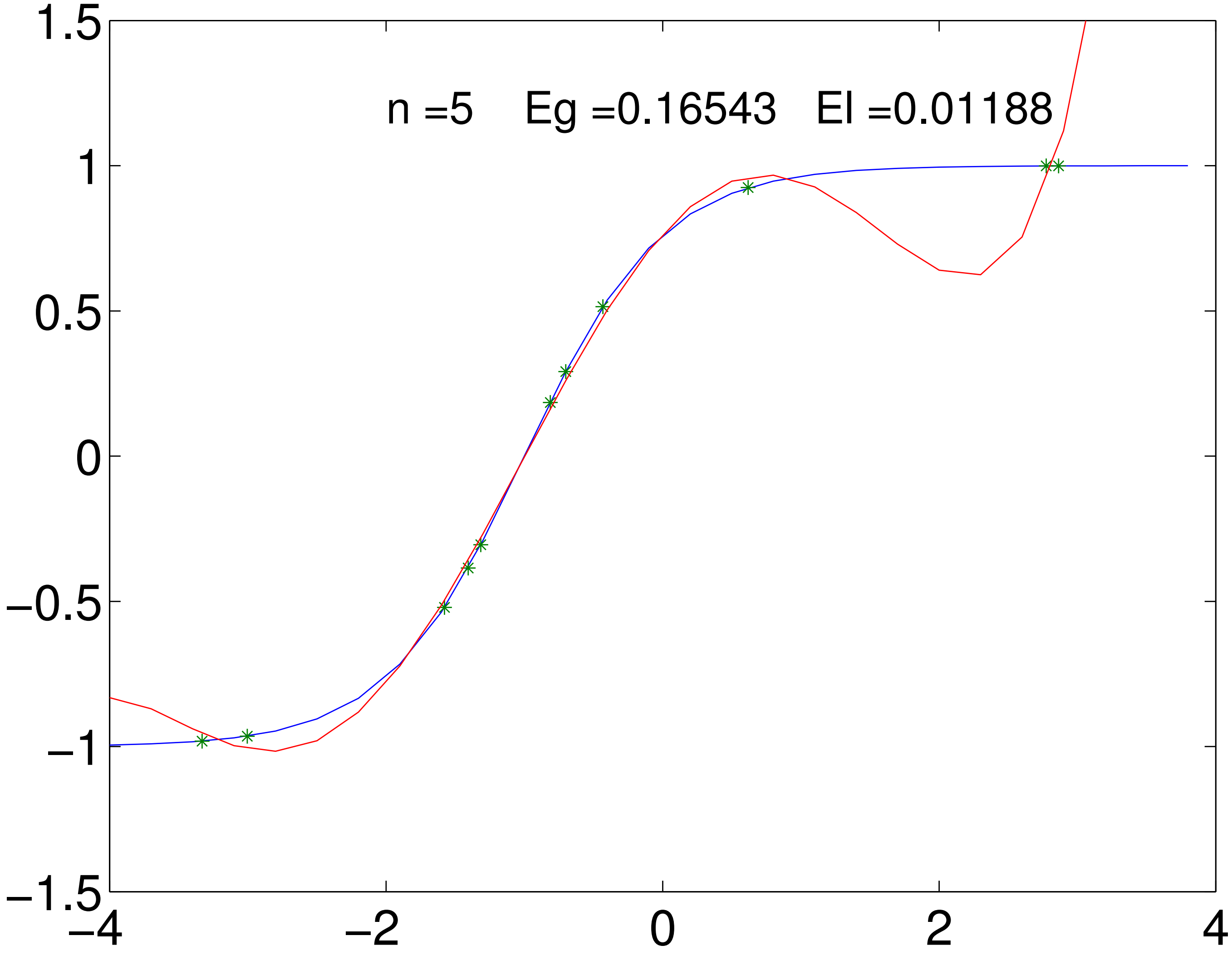
Regression Example $n = 5$



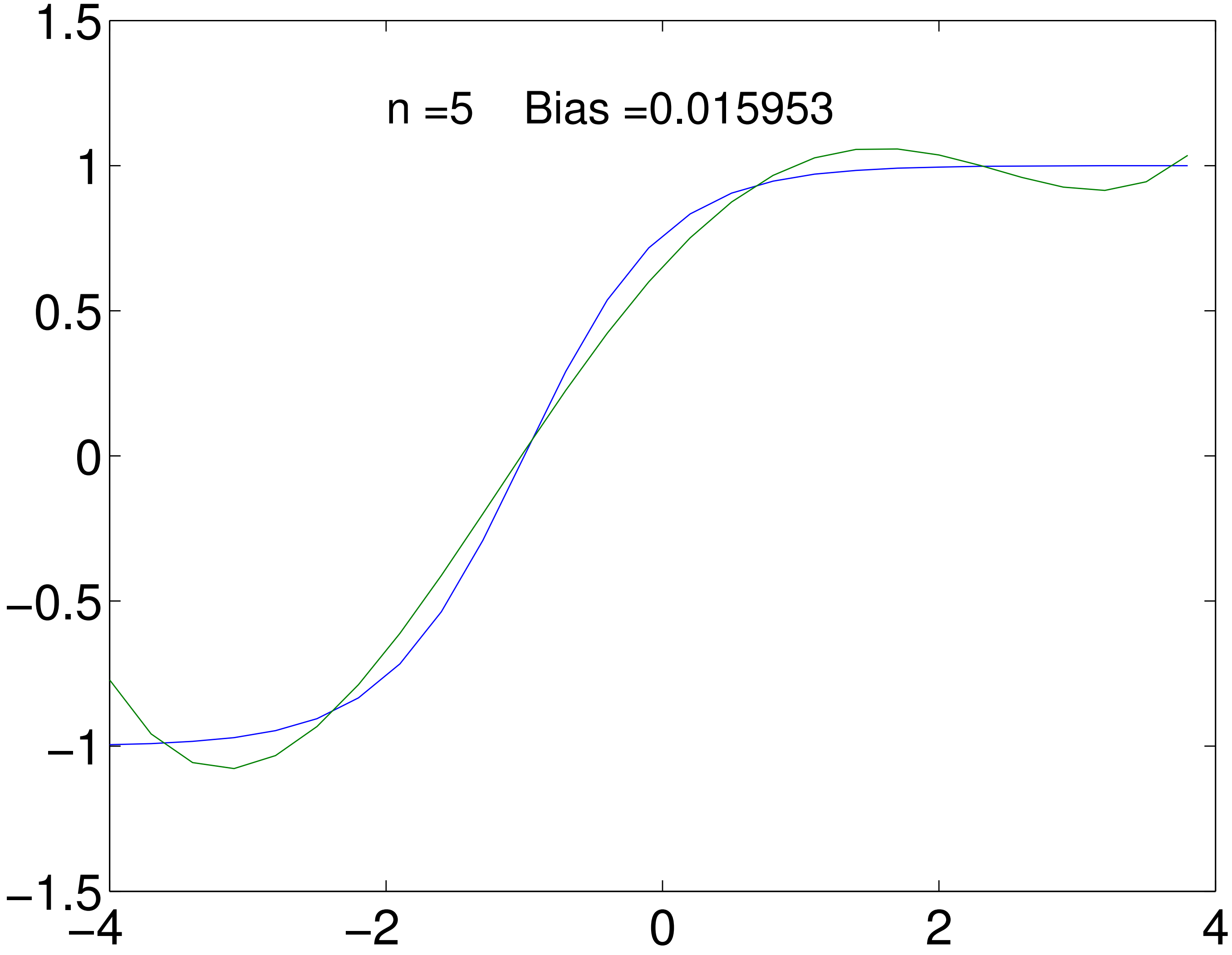
Regression Example $n = 5$



Regression Example $n = 5$



Regression Example $n = 5$



Bias and Variance

Consider the expected generalisation for data sets of size $|\mathcal{D}| = m$

$$\bar{E}_G = \mathbb{E}_{\mathcal{D}}[E_G(\mathcal{D})]$$

Bias and Variance

Consider the expected generalisation for data sets of size $|\mathcal{D}| = m$

$$\bar{E}_G = \mathbb{E}_{\mathcal{D}}[E_G(\mathcal{D})] = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - f(\boldsymbol{x})\right)^2\right]$$

Bias and Variance

Consider the expected generalisation for data sets of size $|\mathcal{D}| = m$

$$\bar{E}_G = \mathbb{E}_{\mathcal{D}}[E_G(\mathcal{D})] = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - f(\boldsymbol{x})\right)^2\right]$$

$$= \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})\, \mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - f(\boldsymbol{x})\right)^2\right]$$

Bias and Variance

Consider the expected generalisation for data sets of size $|\mathcal{D}| = m$

$$\bar{E}_G = \mathbb{E}_{\mathcal{D}}[E_G(\mathcal{D})] = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - f(\boldsymbol{x})\right)^2\right]$$

$$= \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})\, \mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - f(\boldsymbol{x})\right)^2\right]$$

$$= \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})\, \mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x}) + \hat{f}_m(\boldsymbol{x}) - f(\boldsymbol{x}) \right)^2\right]$$

Bias and Variance

Consider the expected generalisation for data sets of size $|\mathcal{D}| = m$

$$\bar{E}_G = \mathbb{E}_{\mathcal{D}}[E_G(\mathcal{D})] = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \, \left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - f(\boldsymbol{x})\right)^2\right]$$

$$= \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \, \mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - f(\boldsymbol{x})\right)^2\right]$$

$$= \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \, \mathbb{E}_{\mathcal{D}}\left[\left(\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x})\right) + \left(\hat{f}_m(\boldsymbol{x}) - f(\boldsymbol{x})\right)\right)^2\right]$$

Bias and Variance

Consider the expected generalisation for data sets of size $|\mathcal{D}| = m$

$$\bar{E}_G = \mathbb{E}_{\mathcal{D}}[E_G(\mathcal{D})] = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - f(\boldsymbol{x})\right)^2\right]$$

$$= \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})\, \mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - f(\boldsymbol{x})\right)^2\right]$$

$$= \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})\, \mathbb{E}_{\mathcal{D}}\left[\left(\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x})\right) + \left(\hat{f}_m(\boldsymbol{x}) - f(\boldsymbol{x})\right)\right)^2\right]$$

$$= \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x})\right)^2 + \left(\hat{f}_m(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right] + \mathbb{E}_{\mathcal{D}}\left[2\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x})\right)\left(\hat{f}_m(\boldsymbol{x}) - f(\boldsymbol{x})\right)\right]\right)$$

Bias and Variance

- We can write the expected generalisation as

$$\mathbb{E}_{\mathcal{D}}[E_G(\mathcal{D})] = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x})\right)^2\right] + \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}_m(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 = V + B$$

- Where B is the bias and V is the variance defined by

$$V = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x})\right)^2\right]$$

Bias and Variance

- We can write the expected generalisation as

$$\mathbb{E}_{\mathcal{D}}[E_G(\mathcal{D})] = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})\,\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x})\right)^2\right] + \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})\left(\hat{f}_m(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 = V + B$$

- Where B is the bias and V is the variance defined by

$$V = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x})\,\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x})\right)^2\right]$$

Bias and Variance

- We can write the expected generalisation as

$$\mathbb{E}_{\mathcal{D}}[E_G(\mathcal{D})] = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x})\right)^2\right] + \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}_m(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 = V + B$$

- Where B is the bias and V is the variance defined by

$$V = \mathbb{E}_{\mathcal{D}}\left[\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x})\right)^2\right]$$

Bias-Variance Dilemma

- The bias measure the generalisation performance of the *mean machine* and is large if the machine is too simple to capture the changes in the function we want to learn

- The variance measures the variation in the prediction of the machine as we change the data set we train on

$$V = \sum_{\boldsymbol{x} \in \mathcal{X}} p(\boldsymbol{x}) \, \mathbb{E}_{\mathcal{D}} \left[\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x}) \right)^2 \right]$$

- The variance is usually large if we have a complex machine

- Striking the right balance is often the key to getting good results

Bias-Variance Dilemma

- The bias measure the generalisation performance of the *mean machine* and is large if the machine is too simple to capture the changes in the function we want to learn

- The variance measures the variation in the prediction of the machine as we change the data set we train on

$$V = \sum_{\boldsymbol{x} \in \mathcal{X}} p(\boldsymbol{x}) \, \mathbb{E}_{\mathcal{D}} \left[\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x}) \right)^2 \right]$$

- The variance is usually large if we have a complex machine

- Striking the right balance is often the key to getting good results

Bias-Variance Dilemma

- The bias measure the generalisation performance of the *mean machine* and is large if the machine is too simple to capture the changes in the function we want to learn

- The variance measures the variation in the prediction of the machine as we change the data set we train on

$$V = \sum_{\boldsymbol{x} \in \mathcal{X}} p(\boldsymbol{x}) \, \mathbb{E}_{\mathcal{D}} \left[\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x}) \right)^2 \right]$$

- The variance is usually large if we have a complex machine

- Striking the right balance is often the key to getting good results

Bias-Variance Dilemma

- The bias measure the generalisation performance of the *mean machine* and is large if the machine is too simple to capture the changes in the function we want to learn

- The variance measures the variation in the prediction of the machine as we change the data set we train on

$$V = \sum_{\boldsymbol{x} \in \mathcal{X}} p(\boldsymbol{x}) \, \mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}(\boldsymbol{x}|\mathcal{D}) - \hat{f}_m(\boldsymbol{x}) \right)^2 \right]$$

- The variance is usually large if we have a complex machine

- Striking the right balance is often the key to getting good results

Balancing Bias and Variance

- We want to choose a learning machine that is complex enough to capture the underlying function we are trying to learn, but otherwise as simple as possible
- There are a number of tricks to achieve this balance
- Some require us to preprocess the data to reduce the number of inputs
- Some machines cleverly adjust their own complexity
- Today we looks at machines that achieve this balance

Balancing Bias and Variance

- We want to choose a learning machine that is complex enough to capture the underlying function we are trying to learn, but otherwise as simple as possible
- There are a number of tricks to achieve this balance
- Some require us to preprocess the data to reduce the number of inputs
- Some machines cleverly adjust their own complexity
- Today we looks at machines that achieve this balance

Balancing Bias and Variance

- We want to choose a learning machine that is complex enough to capture the underlying function we are trying to learn, but otherwise as simple as possible
- There are a number of tricks to achieve this balance
- Some require us to preprocess the data to reduce the number of inputs
- Some machines cleverly adjust their own complexity
- Today we looks at machines that achieve this balance

Balancing Bias and Variance

- We want to choose a learning machine that is complex enough to capture the underlying function we are trying to learn, but otherwise as simple as possible
- There are a number of tricks to achieve this balance
- Some require us to preprocess the data to reduce the number of inputs
- Some machines cleverly adjust their own complexity
- Today we looks at machines that achieve this balance

Balancing Bias and Variance

- We want to choose a learning machine that is complex enough to capture the underlying function we are trying to learn, but otherwise as simple as possible
- There are a number of tricks to achieve this balance
- Some require us to preprocess the data to reduce the number of inputs
- Some machines cleverly adjust their own complexity
- Today we looks at machines that achieve this balance

Over-fitting

- Complex machine can **over-fitting**

  over-fitting*: fitting the training data well at the cost of getting poorer generalisation performance*

- Three red cars. . .

- If we used an infinitely flexible machine we can fit our training data perfectly, but would have no generalisation ability

Over-fitting

- Complex machine can **over-fitting**

  ***over-fitting**: fitting the training data well at the cost of getting poorer generalisation performance*

- Three red cars. . .

- If we used an infinitely flexible machine we can fit our training data perfectly, but would have no generalisation ability

Over-fitting

- Complex machine can **over-fitting**

  ***over-fitting**: fitting the training data well at the cost of getting poorer generalisation performance*

- Three red cars. . .

- If we used an infinitely flexible machine we can fit our training data perfectly, but would have no generalisation ability

Binary Classification Task for You

Class 1

Class 2

Which Category?

- Which category does the following image belong to?

Training Examples

- As we increase the number of training examples, we make it hard to find a spurious rule
- Bigger data sets allow us to use more complicated machines
- (Labelled) data is often expensive to collect so we sometimes have no choice
- Need to control the complexity of our learning machine

Training Examples

- As we increase the number of training examples, we make it hard to find a spurious rule
- Bigger data sets allow us to use more complicated machines
- (Labelled) data is often expensive to collect so we sometimes have no choice
- Need to control the complexity of our learning machine

Training Examples

- As we increase the number of training examples, we make it hard to find a spurious rule
- Bigger data sets allow us to use more complicated machines
- (Labelled) data is often expensive to collect so we sometimes have no choice
- Need to control the complexity of our learning machine

Training Examples

- As we increase the number of training examples, we make it hard to find a spurious rule
- Bigger data sets allow us to use more complicated machines
- (Labelled) data is often expensive to collect so we sometimes have no choice
- Need to control the complexity of our learning machine

Dimensionality Reduction

- We can simplify our machines by using less features
- We can project our data onto a lower dimensional sub-space (e.g. one with the maximum variation in the data PCA)
- We can use clustering to find exemplars and recode our data in terms of differences from the exemplars (radial basis functions)
- Whether this helps depends on whether the information we discard is pertinent to the task we are trying to perform

Dimensionality Reduction

- We can simplify our machines by using less features
- We can project our data onto a lower dimensional sub-space (e.g. one with the maximum variation in the data PCA)
- We can use clustering to find exemplars and recode our data in terms of differences from the exemplars (radial basis functions)
- Whether this helps depends on whether the information we discard is pertinent to the task we are trying to perform

Dimensionality Reduction

- We can simplify our machines by using less features
- We can project our data onto a lower dimensional sub-space (e.g. one with the maximum variation in the data PCA)
- We can use clustering to find exemplars and recode our data in terms of differences from the exemplars (radial basis functions)
- Whether this helps depends on whether the information we discard is pertinent to the task we are trying to perform

Dimensionality Reduction

- We can simplify our machines by using less features
- We can project our data onto a lower dimensional sub-space (e.g. one with the maximum variation in the data PCA)
- We can use clustering to find exemplars and recode our data in terms of differences from the exemplars (radial basis functions)
- Whether this helps depends on whether the information we discard is pertinent to the task we are trying to perform

Feature Selection

- Spurious features will allow us to find spurious rules (**over-fitting**)
- We can try different combinations of features to find the best set, although it rapidly becomes intractable to do this in all ways
- We can use various heuristics to decide which features to keep, but no heuristic is fail-safe
- Feature selection however can be powerful, often we can get very good results by keeping only a few variables
- As well as possibly improving generalisation we also get a more **interpretable** rule

Feature Selection

- Spurious features will allow us to find spurious rules (**over-fitting**)
- We can try different combinations of features to find the best set, although it rapidly becomes intractable to do this in all ways
- We can use various heuristics to decide which features to keep, but no heuristic is fail-safe
- Feature selection however can be powerful, often we can get very good results by keeping only a few variables
- As well as possibly improving generalisation we also get a more **interpretable** rule

Feature Selection

- Spurious features will allow us to find spurious rules (**over-fitting**)
- We can try different combinations of features to find the best set, although it rapidly becomes intractable to do this in all ways
- We can use various heuristics to decide which features to keep, but no heuristic is fail-safe
- Feature selection however can be powerful, often we can get very good results by keeping only a few variables
- As well as possibly improving generalisation we also get a more **interpretable** rule

Feature Selection

- Spurious features will allow us to find spurious rules (**over-fitting**)
- We can try different combinations of features to find the best set, although it rapidly becomes intractable to do this in all ways
- We can use various heuristics to decide which features to keep, but no heuristic is fail-safe
- Feature selection however can be powerful, often we can get very good results by keeping only a few variables
- As well as possibly improving generalisation we also get a more **interpretable** rule

Feature Selection

- Spurious features will allow us to find spurious rules (**over-fitting**)
- We can try different combinations of features to find the best set, although it rapidly becomes intractable to do this in all ways
- We can use various heuristics to decide which features to keep, but no heuristic is fail-safe
- Feature selection however can be powerful, often we can get very good results by keeping only a few variables
- As well as possibly improving generalisation we also get a more **interpretable** rule

Explicit Regularisation

- As Niranjan showed us we can modify our error function to choose smoother functions

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n \right)^2$$

- Second term is minimised when $w_i = 0$
- If w_i is large then

$$f(\boldsymbol{x}|\boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n$$

varies rapidly as we change x_i

Explicit Regularisation

- As Niranjan showed us we can modify our error function to choose smoother functions

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^\mathsf{T} \boldsymbol{x}_n - y_n \right)^2 + \nu \left\| \boldsymbol{w} \right\|^2$$

- Second term is minimised when $w_i = 0$

- If w_i is large then

$$f(\boldsymbol{x}|\boldsymbol{w}) = \boldsymbol{w}^\mathsf{T} \boldsymbol{x}_n$$

varies rapidly as we change x_i

Explicit Regularisation

- As Niranjan showed us we can modify our error function to choose smoother functions

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n \right)^2 + \nu \left\| \boldsymbol{w} \right\|^2$$

(Good to normalise data)

- Second term is minimised when $w_i = 0$

- If w_i is large then

$$f(\boldsymbol{x}|\boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n$$

varies rapidly as we change x_i

Explicit Regularisation

- As Niranjan showed us we can modify our error function to choose smoother functions

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n \right)^2 + \nu \, \|\boldsymbol{w}\|^2$$

(Good to normalise data)

- Second term is minimised when $w_i = 0$

- If w_i is large then

$$f(\boldsymbol{x}|\boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n$$

varies rapidly as we change x_i

Explicit Regularisation

- As Niranjan showed us we can modify our error function to choose smoother functions

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n \right)^2 + \nu \, \|\boldsymbol{w}\|^2$$

(Good to normalise data)

- Second term is minimised when $w_i = 0$
- If w_i is large then

$$f(\boldsymbol{x}|\boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n$$

varies rapidly as we change x_i

Explicit Regularisation

- As Niranjan showed us we can modify our error function to choose smoother functions

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n \right)^2 + \nu \, \|\boldsymbol{w}\|^2$$

(Good to normalise data)

- Second term is minimised when $w_i = 0$

- If w_i is large then

$$f(\boldsymbol{x}|\boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n = \sum_{i=1}^{p} w_i \, x_i$$

varies rapidly as we change x_i

Explicit Regularisation

- As Niranjan showed us we can modify our error function to choose smoother functions

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n \right)^2 + \nu \, \|\boldsymbol{w}\|^2$$

(Good to normalise data)

- Second term is minimised when $w_i = 0$

- If w_i is large then

$$f(\boldsymbol{x}|\boldsymbol{w}) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n = \sum_{i=1}^{p} w_i \, x_i$$

varies rapidly as we change x_i

Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n \right)^2$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error

Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n \right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error

Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n \right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

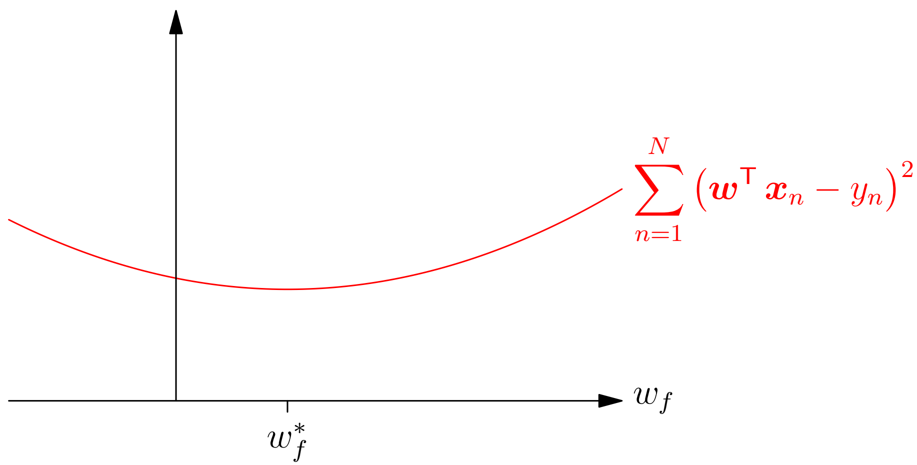
- Spurious features (e.g. shoe size) will give us a small improvement in training error

Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^\mathsf{T} \boldsymbol{x}_n - y_n \right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error

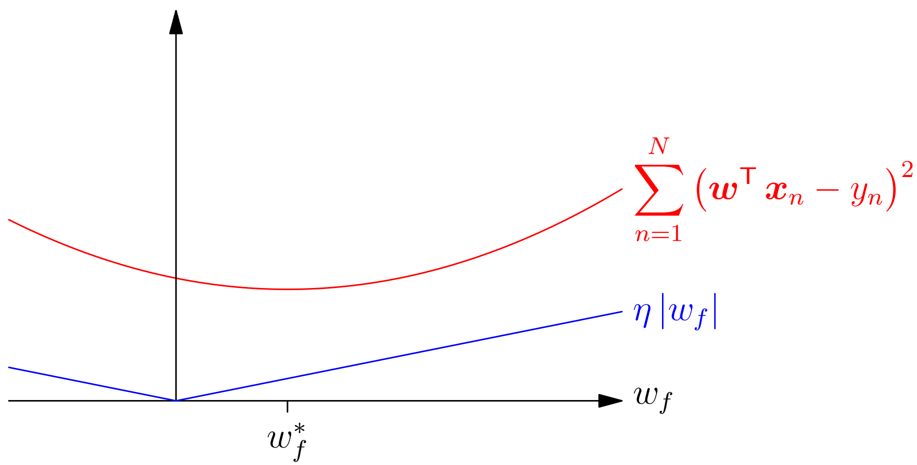


Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n\right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error

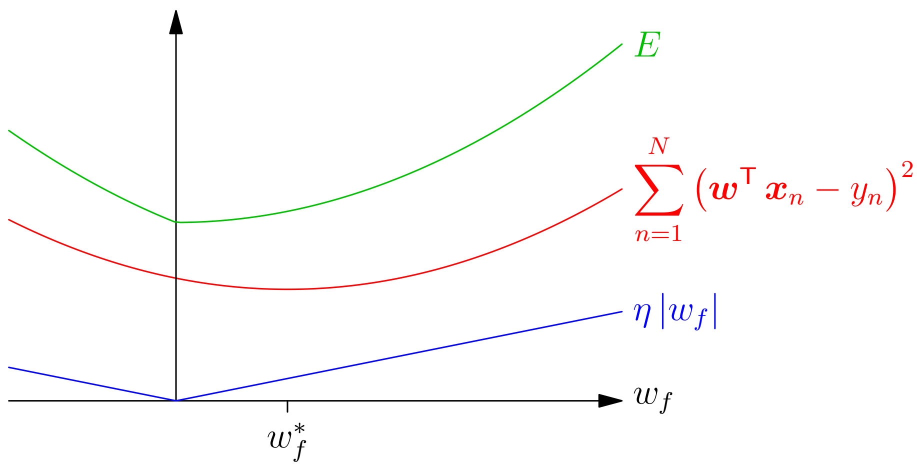


Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n\right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error

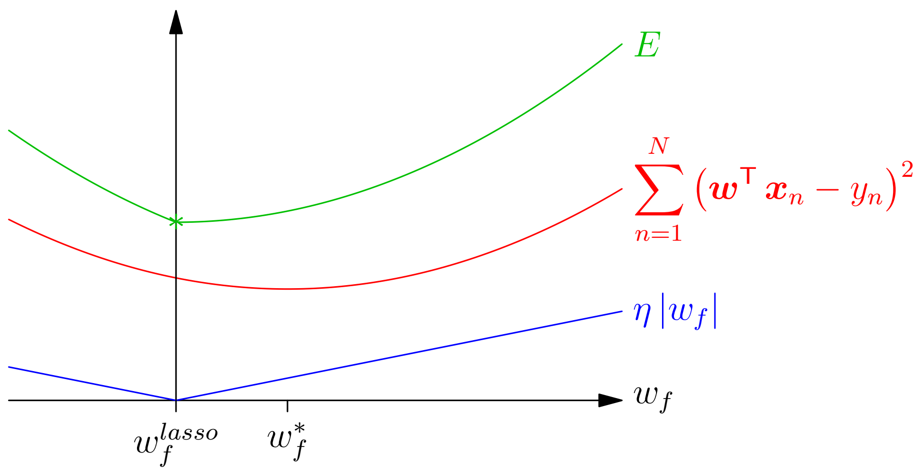


Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n\right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error

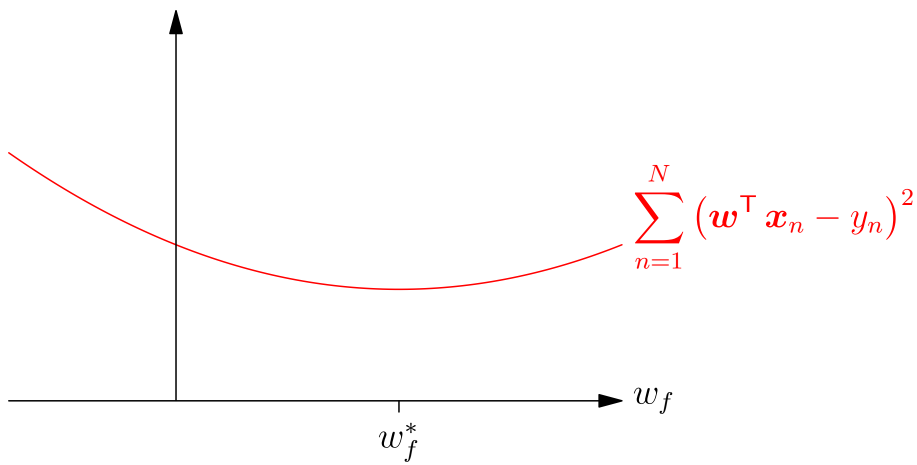


Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n \right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error

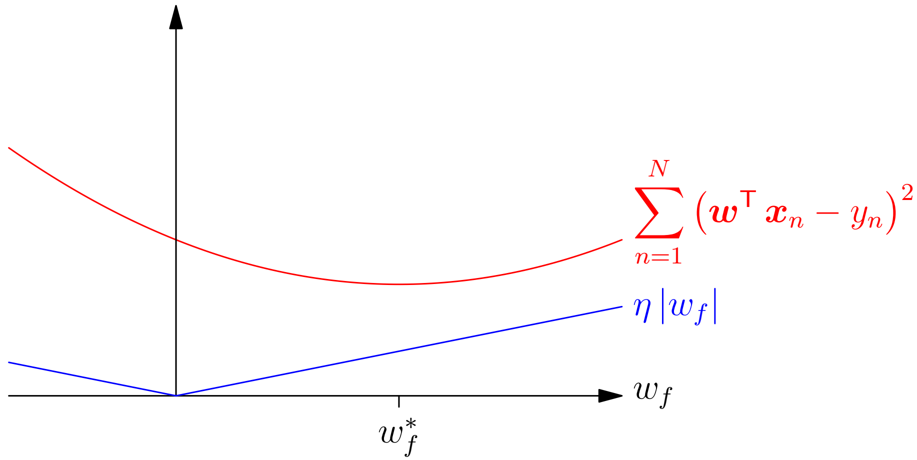


Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n\right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error

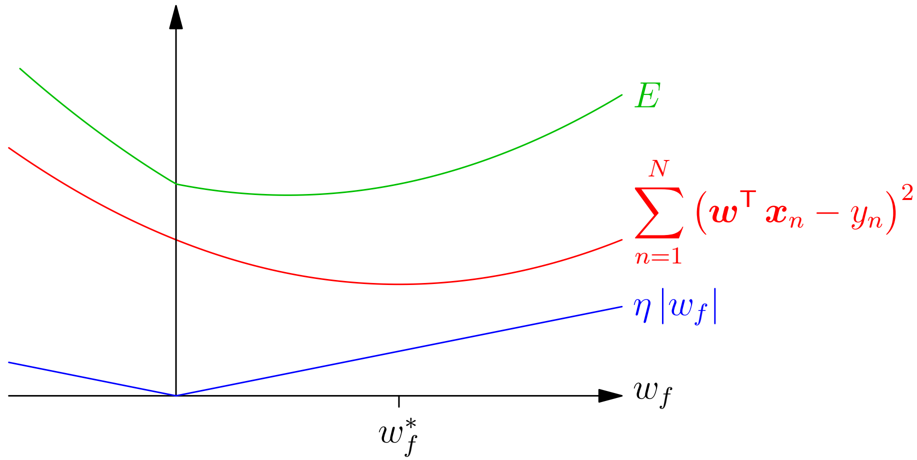


Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n\right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error

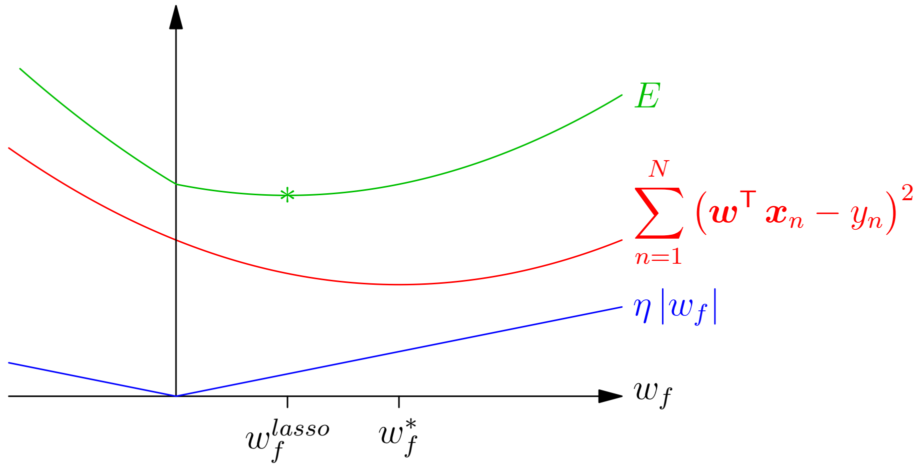


Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n\right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error

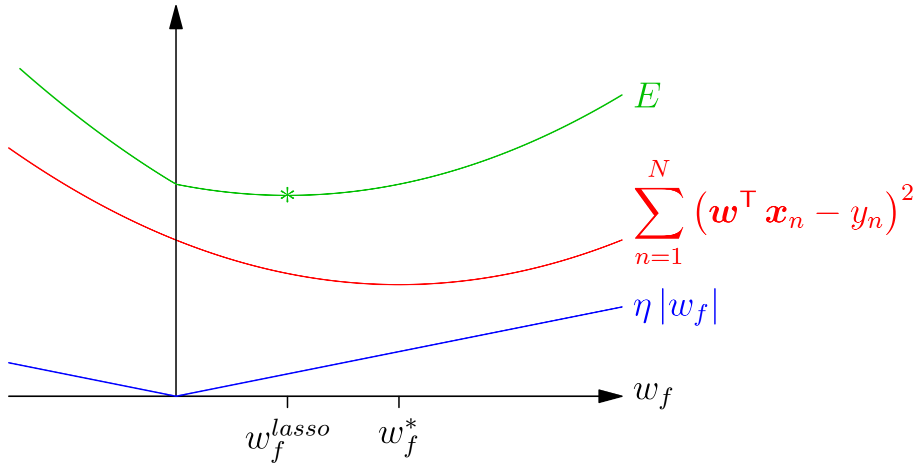


Lasso

- We can us other regularisers

$$E = \sum_{n=1}^{N} \left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_n - y_n\right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- Spurious features (e.g. shoe size) will give us a small improvement in training error



- Does automatic feature selection

Implicit Regularisation

- In the last two examples we added an explicit regularisation term that made the function we learnt simpler
- Some learning machines do this less explicitly
- Some deep learning architectures do subtle averaging
- Sometimes the architecture biases the machine to find a simple solution
- We will see this in support vector machines shortly

Implicit Regularisation

- In the last two examples we added an explicit regularisation term that made the function we learnt simpler
- Some learning machines do this less explicitly
- Some deep learning architectures do subtle averaging
- Sometimes the architecture biases the machine to find a simple solution
- We will see this in support vector machines shortly

Implicit Regularisation

- In the last two examples we added an explicit regularisation term that made the function we learnt simpler
- Some learning machines do this less explicitly
- Some deep learning architectures do subtle averaging
- Sometimes the architecture biases the machine to find a simple solution
- We will see this in support vector machines shortly

Implicit Regularisation

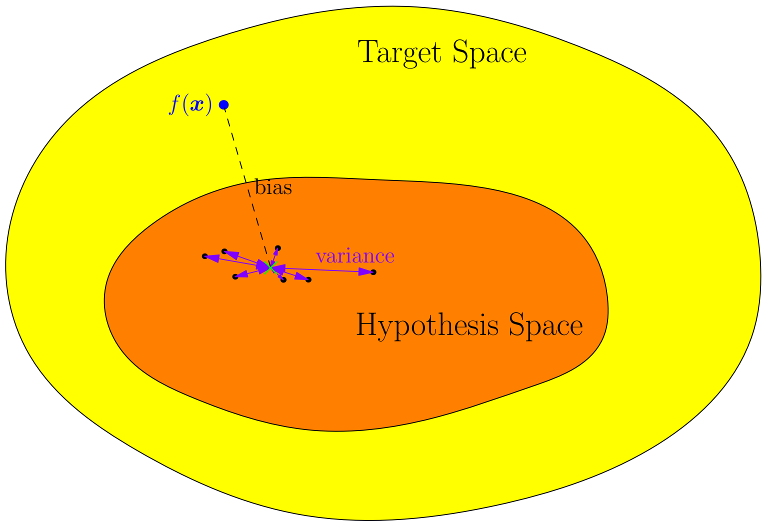
- In the last two examples we added an explicit regularisation term that made the function we learnt simpler
- Some learning machines do this less explicitly
- Some deep learning architectures do subtle averaging
- Sometimes the architecture biases the machine to find a simple solution
- We will see this in support vector machines shortly

Implicit Regularisation

- In the last two examples we added an explicit regularisation term that made the function we learnt simpler
- Some learning machines do this less explicitly
- Some deep learning architectures do subtle averaging
- Sometimes the architecture biases the machine to find a simple solution
- We will see this in support vector machines shortly

Outline

1. What Makes a Good Learning Machine?
2. **SVMs**
3. Ensemble Methods
4. Bayesian Inference



Support Vector Machines

- Support vector machines, when used right, often have the best generalisation results
- They are typically used on numerical data, but can and have been adapted to text, sequences, etc.
- Although not as trendy as deep learning, they will often be the method of choice on small data sets
- They subtly regularise themselves, choosing a solution that generalises well from a host of different solutions

Support Vector Machines

- Support vector machines, when used right, often have the best generalisation results
- They are typically used on numerical data, but can and have been adapted to text, sequences, etc.
- Although not as trendy as deep learning, they will often be the method of choice on small data sets
- They subtly regularise themselves, choosing a solution that generalises well from a host of different solutions

Support Vector Machines

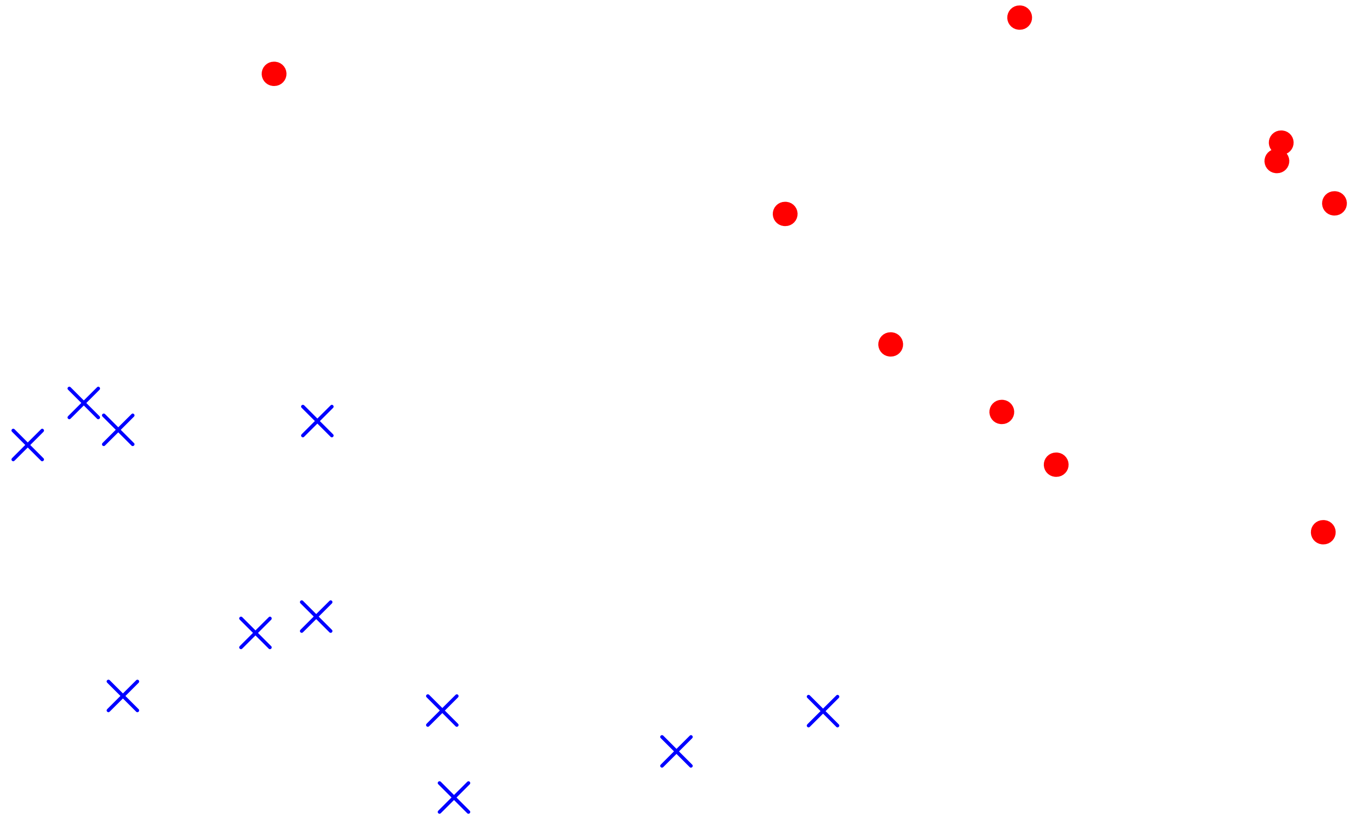
- Support vector machines, when used right, often have the best generalisation results
- They are typically used on numerical data, but can and have been adapted to text, sequences, etc.
- Although not as trendy as deep learning, they will often be the method of choice on small data sets
- They subtly regularise themselves, choosing a solution that generalises well from a host of different solutions

Support Vector Machines

- Support vector machines, when used right, often have the best generalisation results
- They are typically used on numerical data, but can and have been adapted to text, sequences, etc.
- Although not as trendy as deep learning, they will often be the method of choice on small data sets
- They subtly regularise themselves, choosing a solution that generalises well from a host of different solutions

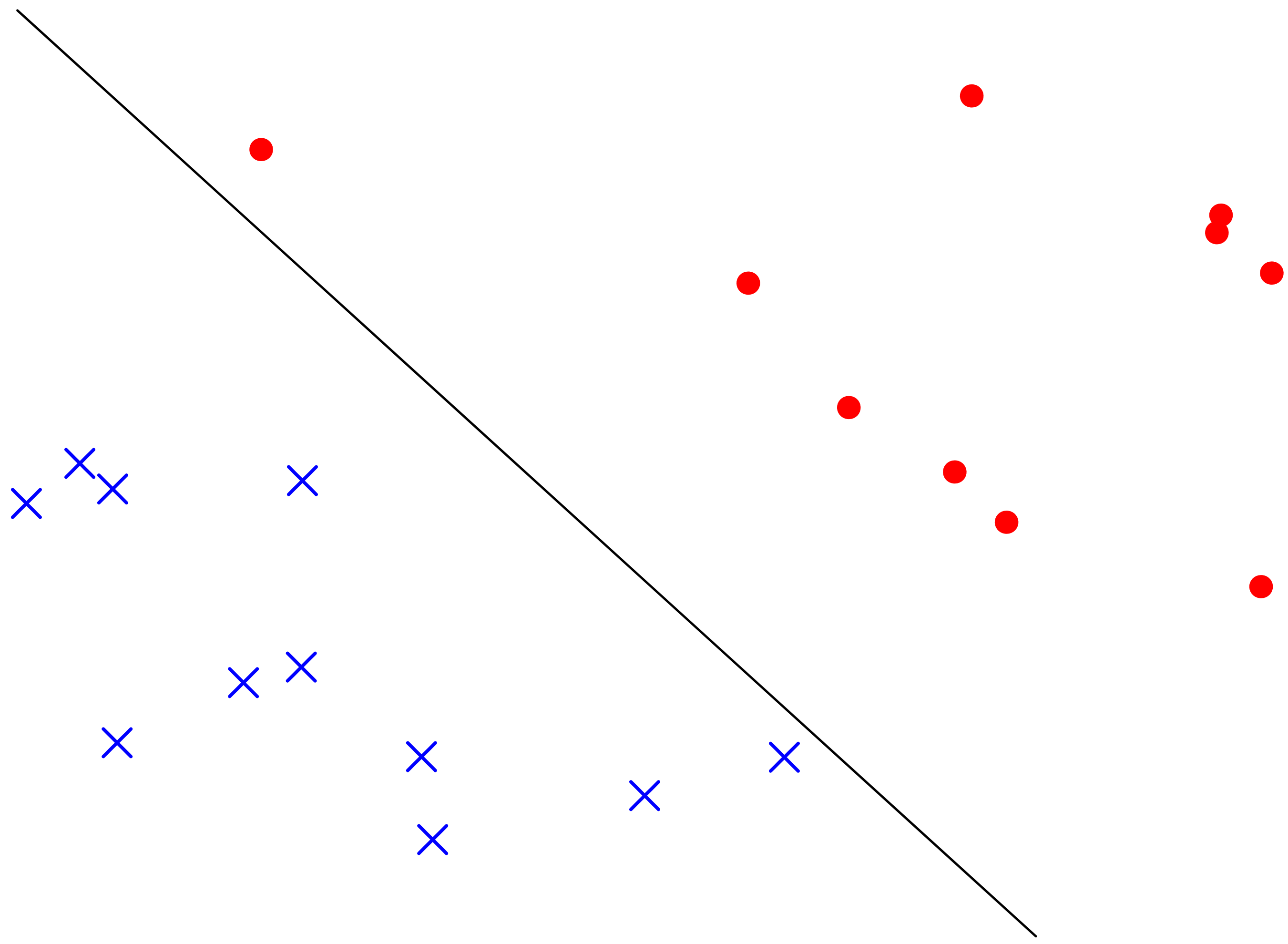
Linear Separation of Data

- SVMs classify linearly separable data



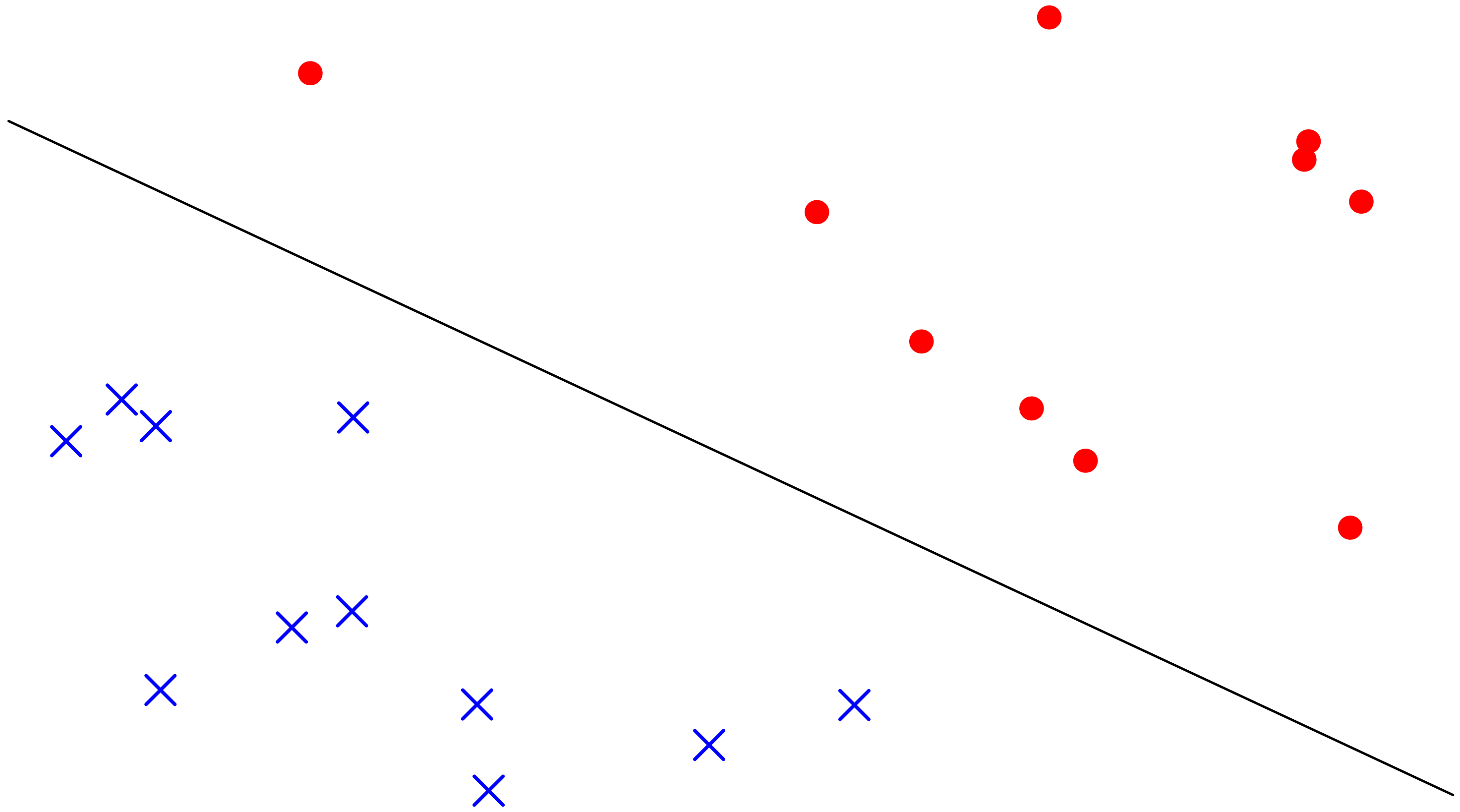
Linear Separation of Data

- SVMs classify linearly separable data



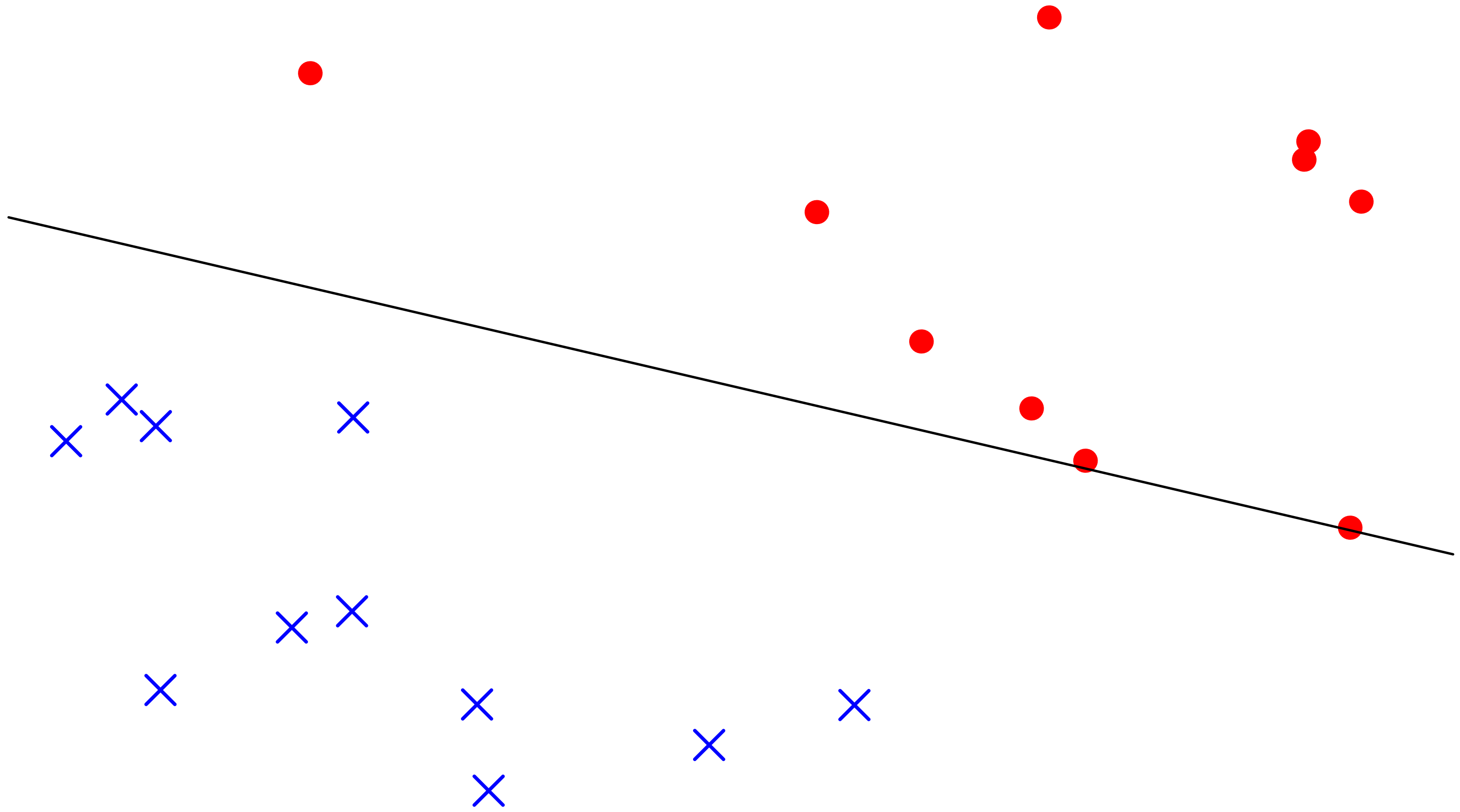
Linear Separation of Data

- SVMs classify linearly separable data



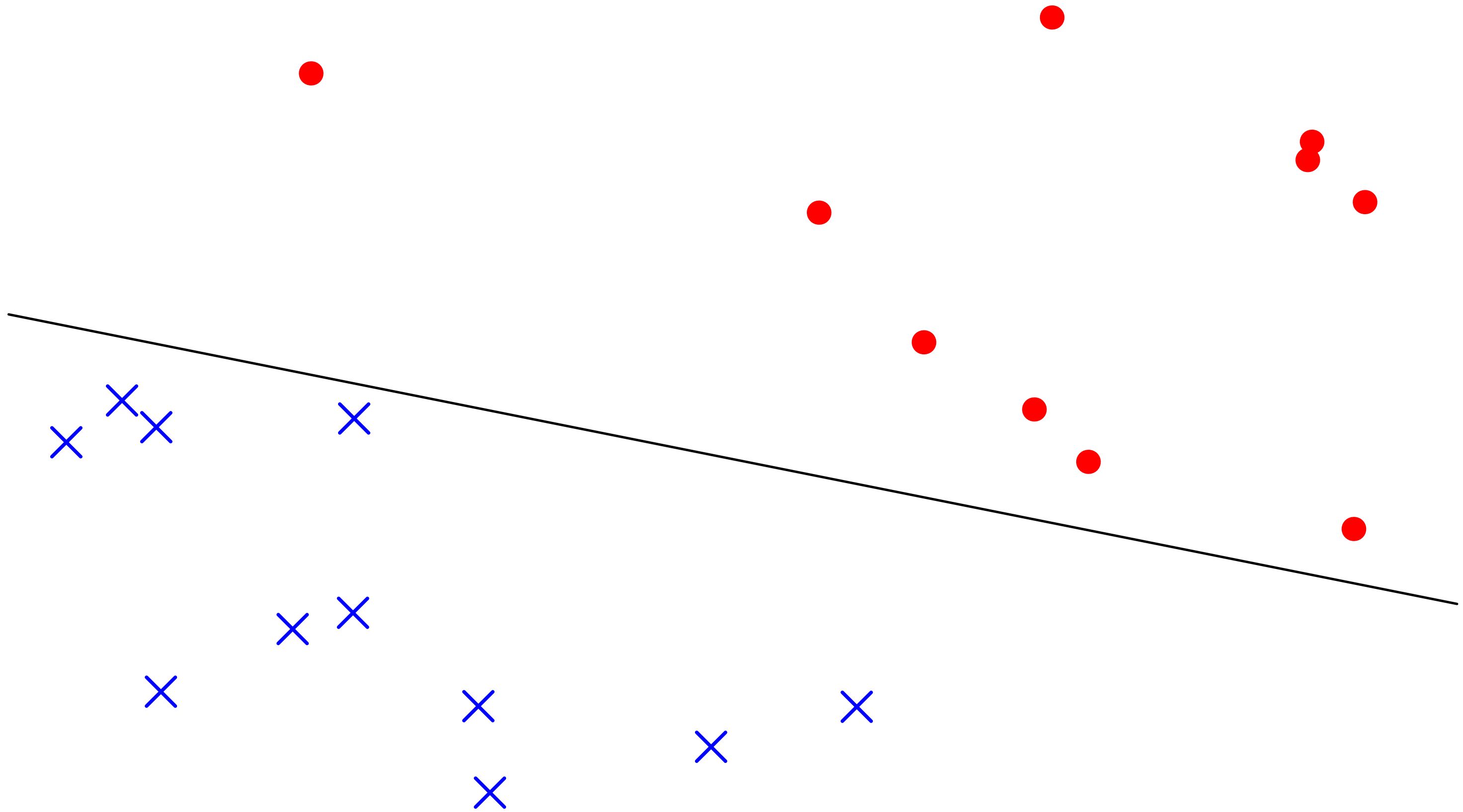
Linear Separation of Data

- SVMs classify linearly separable data



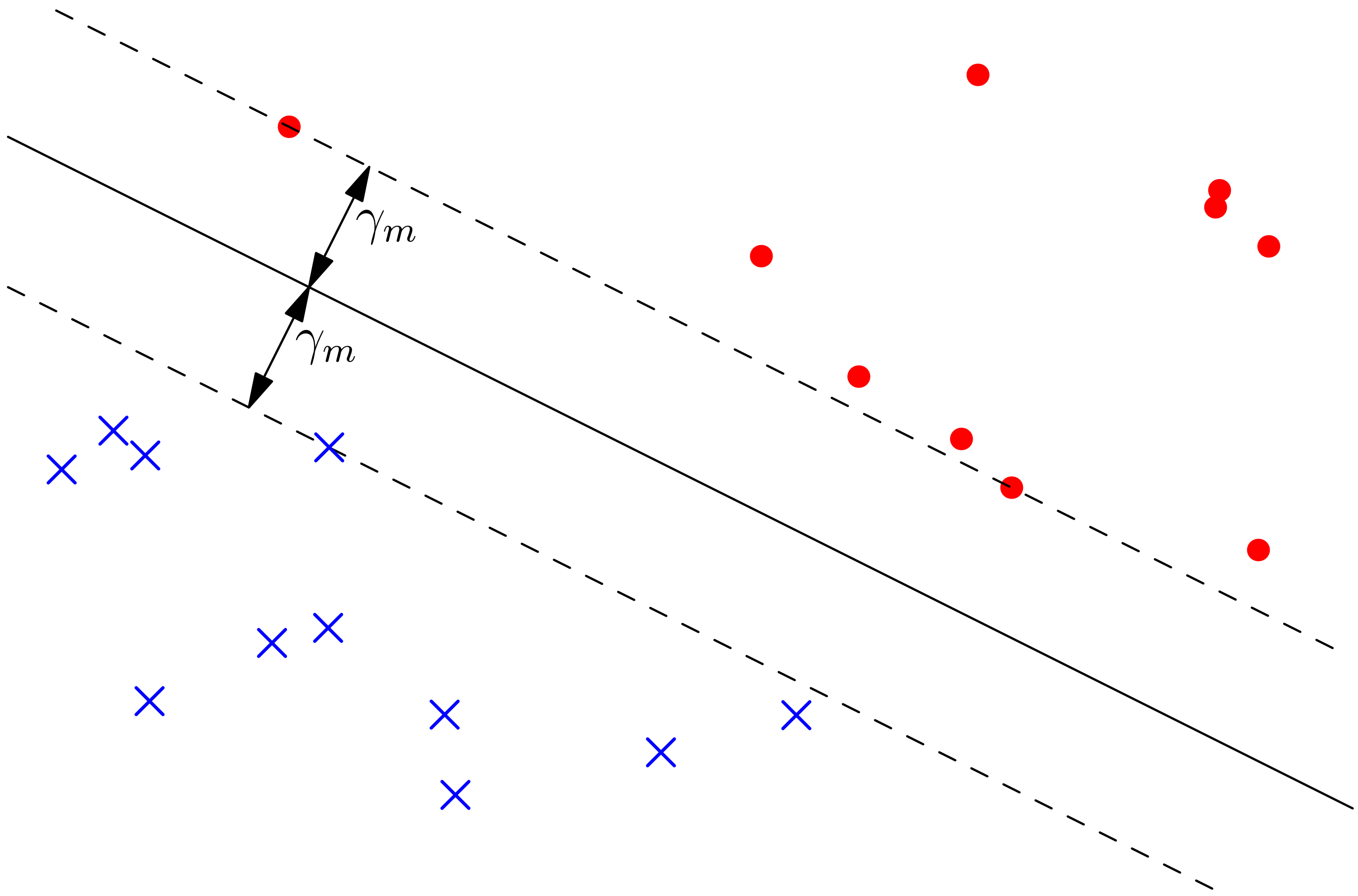
Linear Separation of Data

- SVMs classify linearly separable data



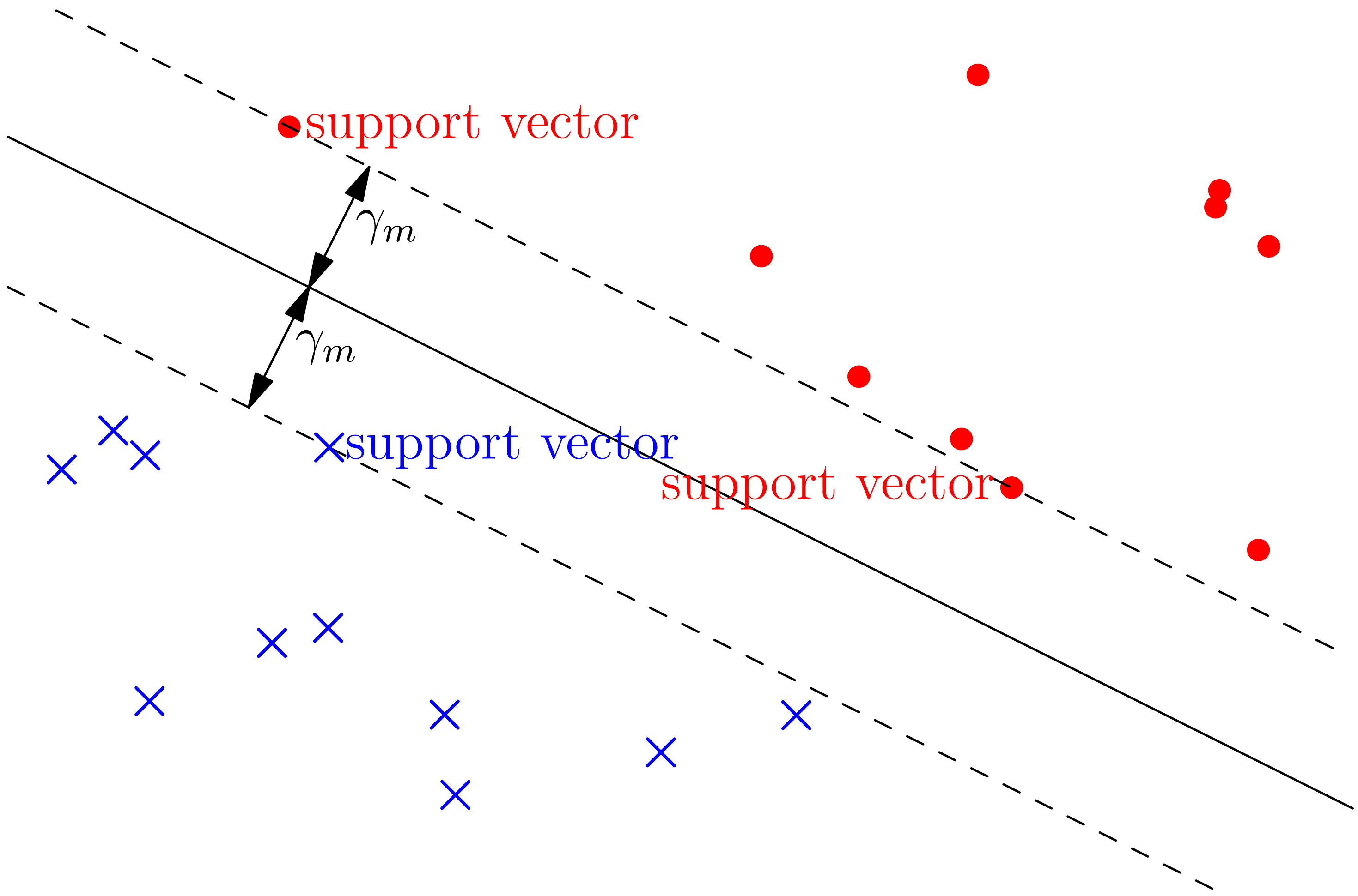
Linear Separation of Data

- SVMs classify linearly separable data



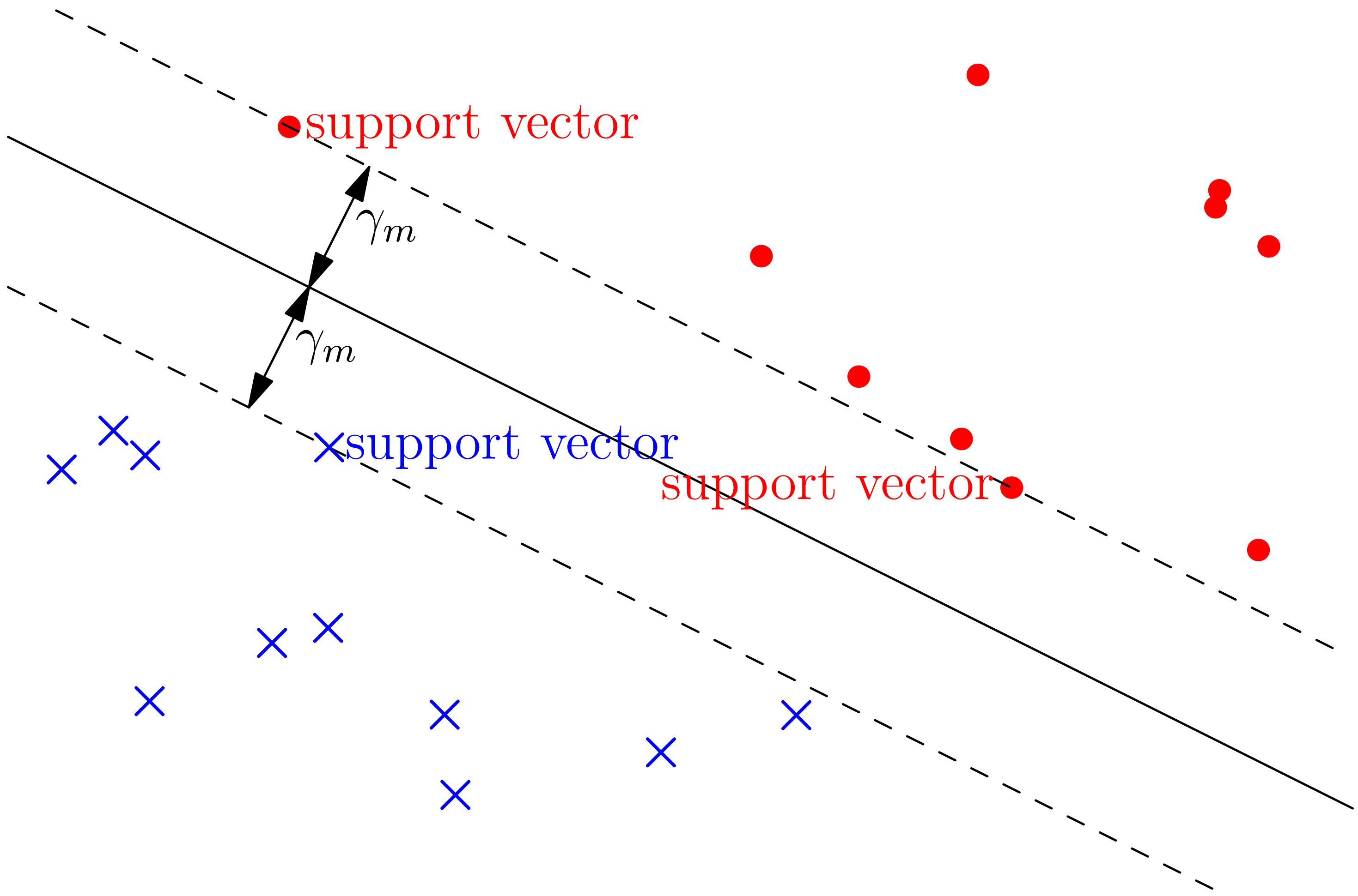
Linear Separation of Data

- SVMs classify linearly separable data



Linear Separation of Data

- SVMs classify linearly separable data



- Finds maximum-margin separating plane

Extended Feature Space

- To increase the likelihood of linear-separability we often use a high-dimensional mapping

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_p) \to \boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_m(\boldsymbol{x}))$$

$m \gg p$

- Finding the maximum margin hyper-plane is time consuming in "primal" form if m is large

- We can work in the "dual" space of patterns, then we only need to compute dot products

$$\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j)$$

Extended Feature Space

- To increase the likelihood of linear-separability we often use a high-dimensional mapping

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_p) \to \boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_m(\boldsymbol{x}))$$

$m \gg p$

- Finding the maximum margin hyper-plane is time consuming in "primal" form if m is large

- We can work in the "dual" space of patterns, then we only need to compute dot products

$$\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j)$$

Extended Feature Space

- To increase the likelihood of linear-separability we often use a high-dimensional mapping

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_p) \to \boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_m(\boldsymbol{x}))$$

$m \gg p$

- Finding the maximum margin hyper-plane is time consuming in "primal" form if m is large

- We can work in the "dual" space of patterns, then we only need to compute dot products

$$\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j)$$

Extended Feature Space

- To increase the likelihood of linear-separability we often use a high-dimensional mapping

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_p) \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_m(\boldsymbol{x}))$$

$m \gg p$

- Finding the maximum margin hyper-plane is time consuming in "primal" form if m is large

- We can work in the "dual" space of patterns, then we only need to compute dot products

$$\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j) = \sum_{k=1}^{m} \phi_k(\boldsymbol{x}_i)\, \phi_k(\boldsymbol{x}_j)$$

Kernel Trick

- If we choose a **positive semi-definite** kernel function $K(\boldsymbol{x}, \boldsymbol{y})$ then there exists functions $\phi_k(\boldsymbol{x})$, such that

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j)$$

  (like an eigenvector decomposition of a matrix)

- Never need to compute $\phi_k(\boldsymbol{x}_i)$ explicitly as we only need the dot-product $\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to compute maximum margin separating hyper-plane

- Sometimes $\boldsymbol{\phi}(\boldsymbol{x}_i)$ is an infinite dimensional vector so its good we don't have to compute it!

Kernel Trick

- If we choose a **positive semi-definite** kernel function $K(\boldsymbol{x}, \boldsymbol{y})$ then there exists functions $\phi_k(\boldsymbol{x})$, such that

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j)$$

(like an eigenvector decomposition of a matrix)

- Never need to compute $\phi_k(\boldsymbol{x}_i)$ explicitly as we only need the dot-product $\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to compute maximum margin separating hyper-plane

- Sometimes $\boldsymbol{\phi}(\boldsymbol{x}_i)$ is an infinite dimensional vector so its good we don't have to compute it!

Kernel Trick

- If we choose a **positive semi-definite** kernel function $K(\boldsymbol{x}, \boldsymbol{y})$ then there exists functions $\phi_k(\boldsymbol{x})$, such that

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j)$$

  (like an eigenvector decomposition of a matrix)

- Never need to compute $\phi_k(\boldsymbol{x}_i)$ explicitly as we only need the dot-product $\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to compute maximum margin separating hyper-plane

- Sometimes $\boldsymbol{\phi}(\boldsymbol{x}_i)$ is an infinite dimensional vector so its good we don't have to compute it!

Kernel Trick

- If we choose a **positive semi-definite** kernel function $K(\boldsymbol{x}, \boldsymbol{y})$ then there exists functions $\phi_k(\boldsymbol{x})$, such that

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j)$$

  (like an eigenvector decomposition of a matrix)

- Never need to compute $\phi_k(\boldsymbol{x}_i)$ explicitly as we only need the dot-product $\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to compute maximum margin separating hyper-plane

- Sometimes $\boldsymbol{\phi}(\boldsymbol{x}_i)$ is an infinite dimensional vector so its good we don't have to compute it!

Kernel Functions

- Kernel functions are symmetric functions of two variable
- Strong restriction: *positive semi-definite*
- Examples

$$\text{Quadratic kernel:} \qquad K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left(\boldsymbol{x}_1^{\mathsf{T}} \boldsymbol{x}_2\right)^2$$

$$\text{Gaussian (RBF) kernel:} \qquad K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathrm{e}^{-\gamma \, \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

- Consider the mapping

$$\boldsymbol{x}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}_i) = \begin{pmatrix} x_i^2 \\ y_i^2 \\ \sqrt{2}\, x_i \, y_i \end{pmatrix}$$

Kernel Functions

- Kernel functions are symmetric functions of two variable
- Strong restriction: *positive semi-definite*
- Examples

$$\text{Quadratic kernel:} \qquad K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left(\boldsymbol{x}_1^{\mathsf{T}} \boldsymbol{x}_2\right)^2$$

$$\text{Gaussian (RBF) kernel:} \qquad K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathrm{e}^{-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

- Consider the mapping

$$\boldsymbol{x}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \to \boldsymbol{\phi}(\boldsymbol{x}_i) = \begin{pmatrix} x_i^2 \\ y_i^2 \\ \sqrt{2}\, x_i\, y_i \end{pmatrix}$$

Kernel Functions

- Kernel functions are symmetric functions of two variable
- Strong restriction: *positive semi-definite*
- Examples

Quadratic kernel: $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left(\boldsymbol{x}_1^{\mathsf{T}} \boldsymbol{x}_2\right)^2$

Gaussian (RBF) kernel: $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathrm{e}^{-\gamma \, \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$

- Consider the mapping

$$\boldsymbol{x}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \to \boldsymbol{\phi}(\boldsymbol{x}_i) = \begin{pmatrix} x_i^2 \\ y_i^2 \\ \sqrt{2}\, x_i \, y_i \end{pmatrix}$$

Kernel Functions

- Kernel functions are symmetric functions of two variable
- Strong restriction: *positive semi-definite*
- Examples

$$\text{Quadratic kernel:} \qquad K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left(\boldsymbol{x}_1^{\mathsf{T}} \boldsymbol{x}_2\right)^2$$

$$\text{Gaussian (RBF) kernel:} \qquad K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathrm{e}^{-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

- Consider the mapping

$$\boldsymbol{x}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \to \boldsymbol{\phi}(\boldsymbol{x}_i) = \begin{pmatrix} x_i^2 \\ y_i^2 \\ \sqrt{2}\, x_i\, y_i \end{pmatrix}$$

Non-linearly Separation of Data

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_1)\boldsymbol{\phi}(\boldsymbol{x}_2)$$

Non-linearly Separation of Data

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_1)\boldsymbol{\phi}(\boldsymbol{x}_2) = \begin{pmatrix} x_1^2 & y_1^2 & \sqrt{2}\,x_1\,y_1 \end{pmatrix} \begin{pmatrix} x_2^2 \\ y_2^2 \\ \sqrt{2}\,x_2\,y_2 \end{pmatrix}$$

Non-linearly Separation of Data

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_1)\boldsymbol{\phi}(\boldsymbol{x}_2) = \begin{pmatrix} x_1^2 & y_1^2 & \sqrt{2}\, x_1\, y_1 \end{pmatrix} \begin{pmatrix} x_2^2 \\ y_2^2 \\ \sqrt{2}\, x_2\, y_2 \end{pmatrix}$$

$$= x_1^2\, x_2^2 + y_1^2\, y_2^2 + 2\, x_1\, y_1\, x_2\, y_2$$

Non-linearly Separation of Data

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_1)\boldsymbol{\phi}(\boldsymbol{x}_2) = \begin{pmatrix} x_1^2 & y_1^2 & \sqrt{2}\, x_1\, y_1 \end{pmatrix} \begin{pmatrix} x_2^2 \\ y_2^2 \\ \sqrt{2}\, x_2\, y_2 \end{pmatrix}$$

$$= x_1^2\, x_2^2 + y_1^2\, y_2^2 + 2\, x_1\, y_1\, x_2\, y_2 = (x_1\, x_2 + y_1\, y_2)^2$$

Non-linearly Separation of Data

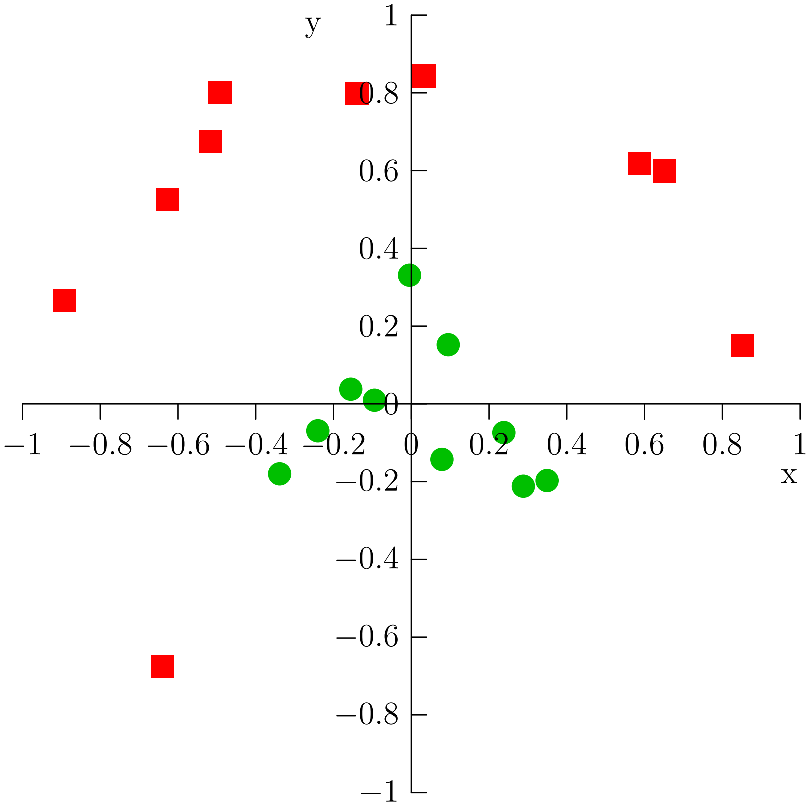
$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_1)\boldsymbol{\phi}(\boldsymbol{x}_2) = \begin{pmatrix} x_1^2 & y_1^2 & \sqrt{2}\, x_1\, y_1 \end{pmatrix} \begin{pmatrix} x_2^2 \\ y_2^2 \\ \sqrt{2}\, x_2\, y_2 \end{pmatrix}$$

$$= x_1^2\, x_2^2 + y_1^2\, y_2^2 + 2\, x_1\, y_1\, x_2\, y_2 = (x_1\, x_2 + y_1\, y_2)^2 = \left(\boldsymbol{x}_1^T \boldsymbol{x}_2\right)^2$$

Non-linearly Separation of Data

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_1)\boldsymbol{\phi}(\boldsymbol{x}_2) = \begin{pmatrix} x_1^2 & y_1^2 & \sqrt{2}\,x_1\,y_1 \end{pmatrix} \begin{pmatrix} x_2^2 \\ y_2^2 \\ \sqrt{2}\,x_2\,y_2 \end{pmatrix}$$

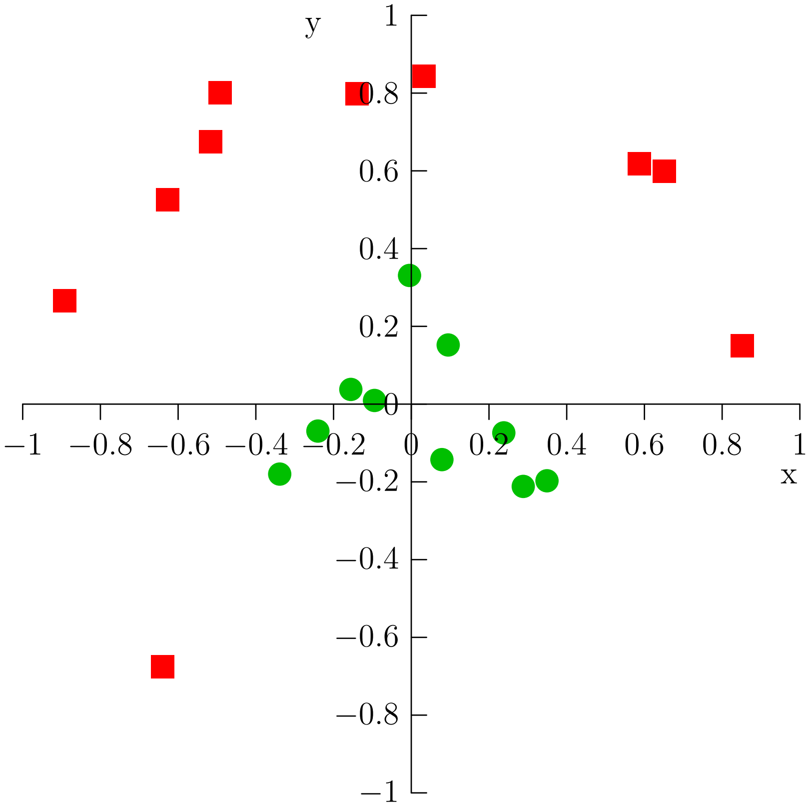
$$= x_1^2\,x_2^2 + y_1^2\,y_2^2 + 2\,x_1\,y_1\,x_2\,y_2 = (x_1\,x_2 + y_1\,y_2)^2 = \left(\boldsymbol{x}_1^T\boldsymbol{x}_2\right)^2$$



Non-linearly Separation of Data

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_1)\boldsymbol{\phi}(\boldsymbol{x}_2) = \begin{pmatrix} x_1^2 & y_1^2 & \sqrt{2}\, x_1\, y_1 \end{pmatrix} \begin{pmatrix} x_2^2 \\ y_2^2 \\ \sqrt{2}\, x_2\, y_2 \end{pmatrix}$$

$$= x_1^2\, x_2^2 + y_1^2\, y_2^2 + 2\, x_1\, y_1\, x_2\, y_2 = (x_1\, x_2 + y_1\, y_2)^2 = \left(\boldsymbol{x}_1^T \boldsymbol{x}_2\right)^2$$

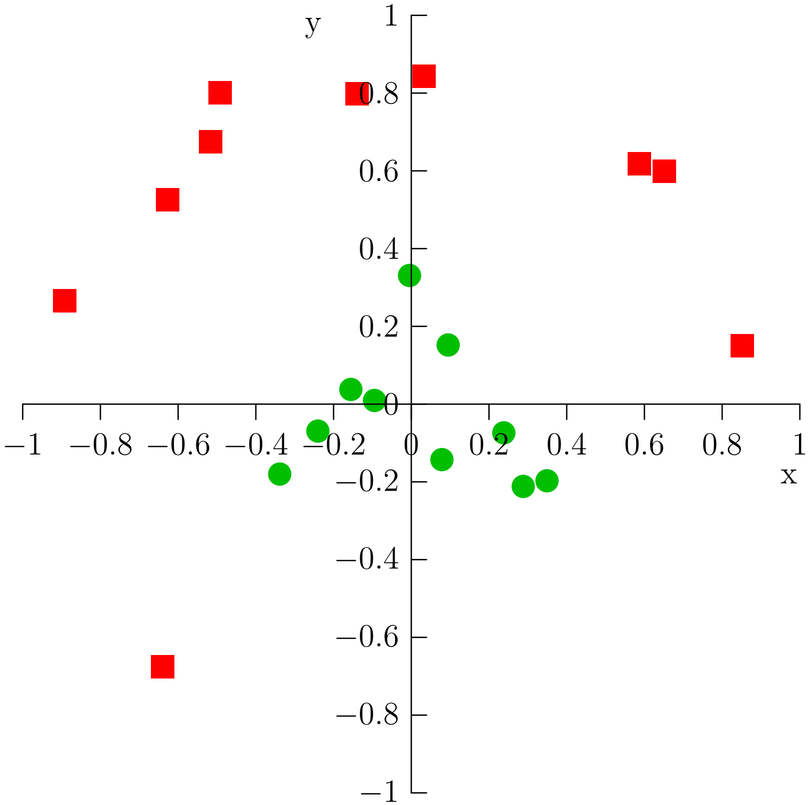


$\boldsymbol{x} \rightarrow \boldsymbol{\phi}(\boldsymbol{x})$
$\longrightarrow$

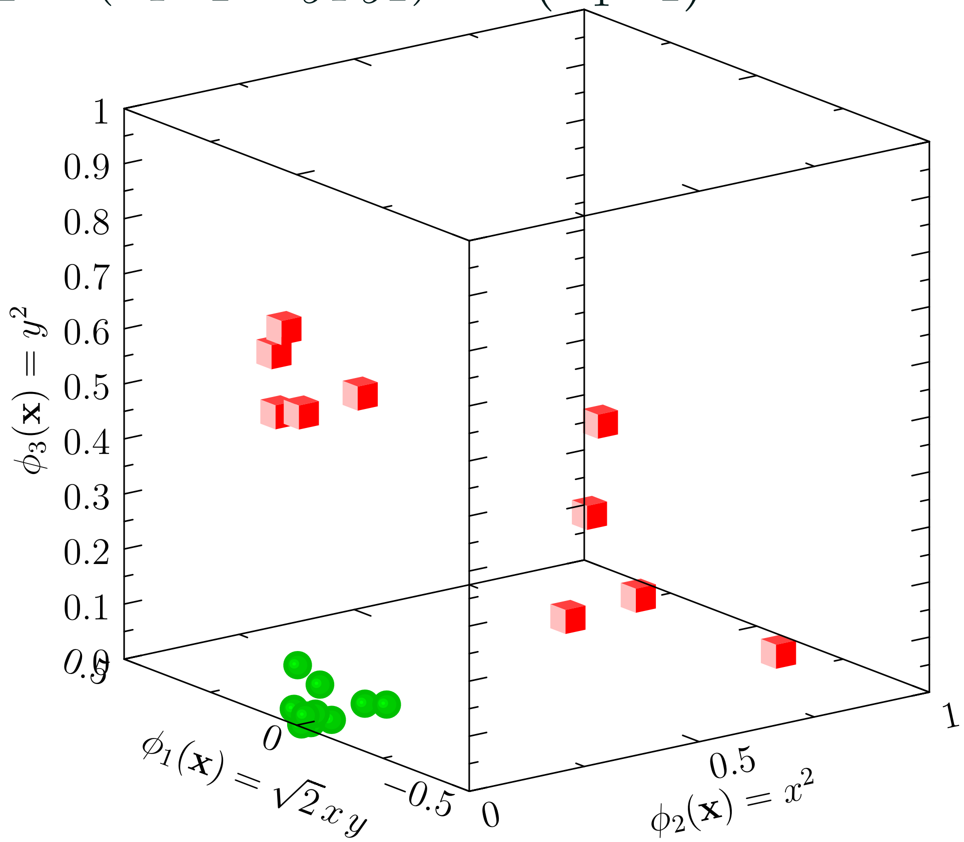
Non-linearly Separation of Data

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_1)\boldsymbol{\phi}(\boldsymbol{x}_2) = \begin{pmatrix} x_1^2 & y_1^2 & \sqrt{2}\,x_1\,y_1 \end{pmatrix} \begin{pmatrix} x_2^2 \\ y_2^2 \\ \sqrt{2}\,x_2\,y_2 \end{pmatrix}$$

$$= x_1^2\,x_2^2 + y_1^2\,y_2^2 + 2\,x_1\,y_1\,x_2\,y_2 = (x_1\,x_2 + y_1\,y_2)^2 = \left(\boldsymbol{x}_1^T\boldsymbol{x}_2\right)^2$$



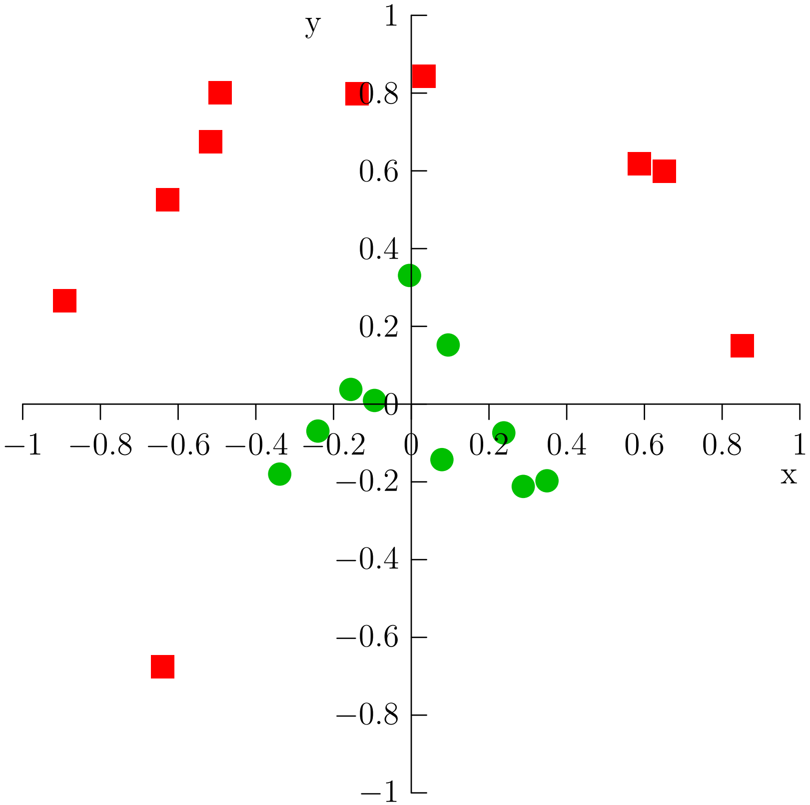
$$\xrightarrow{\boldsymbol{x}\to\boldsymbol{\phi}(\boldsymbol{x})}$$



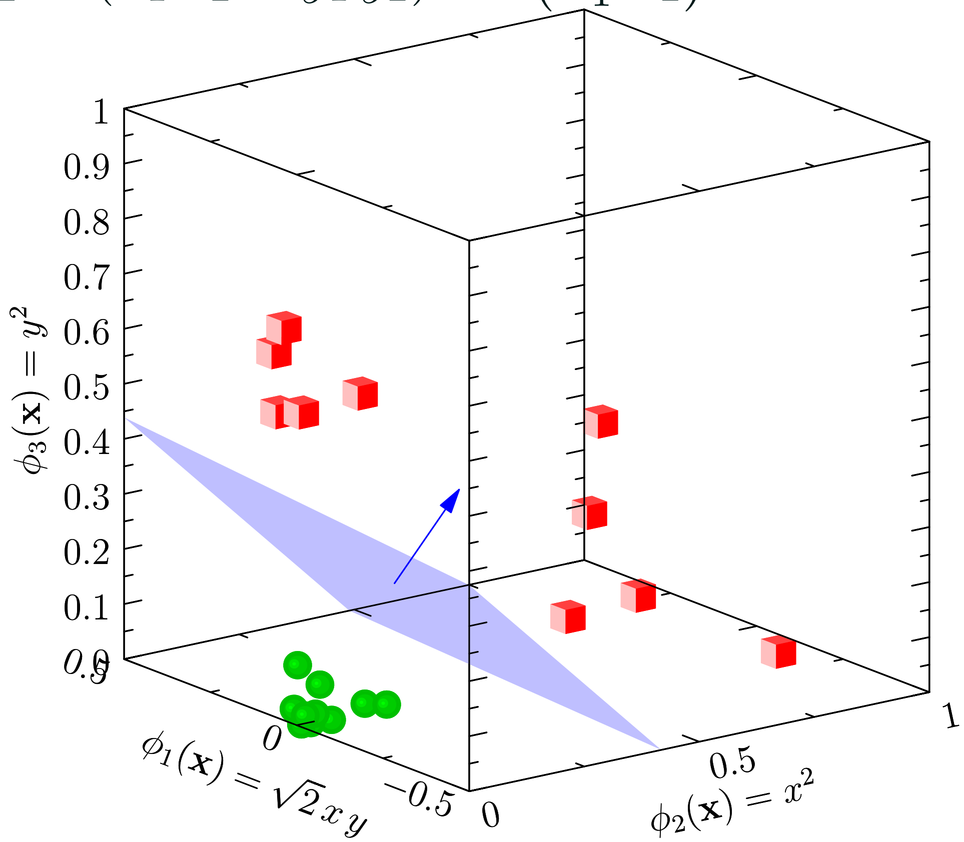
Non-linearly Separation of Data

$$K(\boldsymbol{x}_1,\,\boldsymbol{x}_2) = \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_1)\boldsymbol{\phi}(\boldsymbol{x}_2) = \begin{pmatrix} x_1^2 & y_1^2 & \sqrt{2}\,x_1\,y_1 \end{pmatrix} \begin{pmatrix} x_2^2 \\ y_2^2 \\ \sqrt{2}\,x_2\,y_2 \end{pmatrix}$$

$$= x_1^2\,x_2^2 + y_1^2\,y_2^2 + 2\,x_1\,y_1\,x_2\,y_2 = (x_1\,x_2 + y_1\,y_2)^2 = \left(\boldsymbol{x}_1^T\boldsymbol{x}_2\right)^2$$



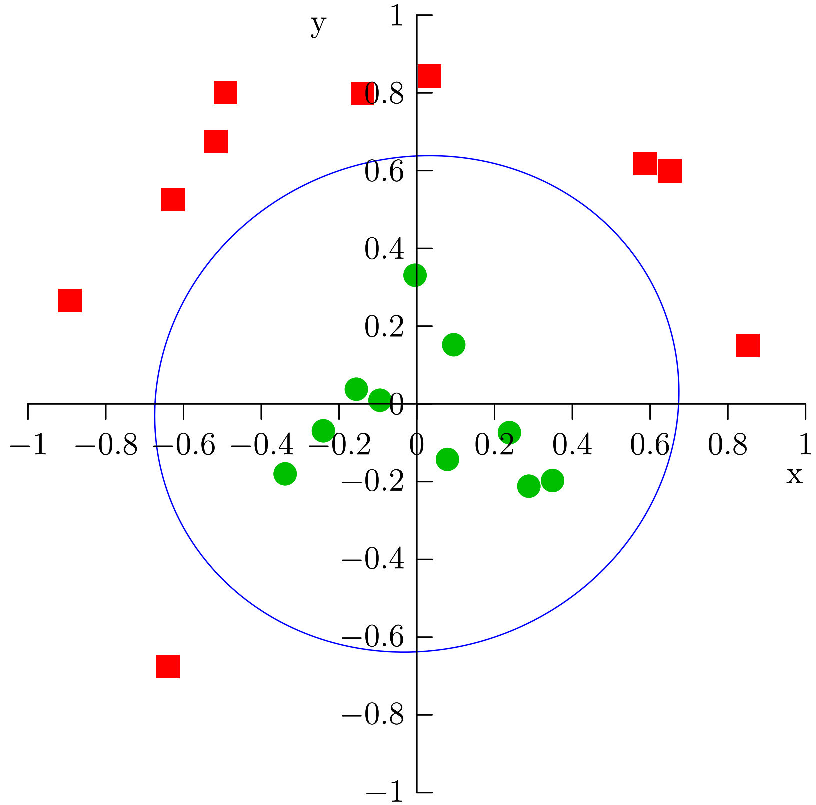
$\xrightarrow{\boldsymbol{x}\to\boldsymbol{\phi}(\boldsymbol{x})}$



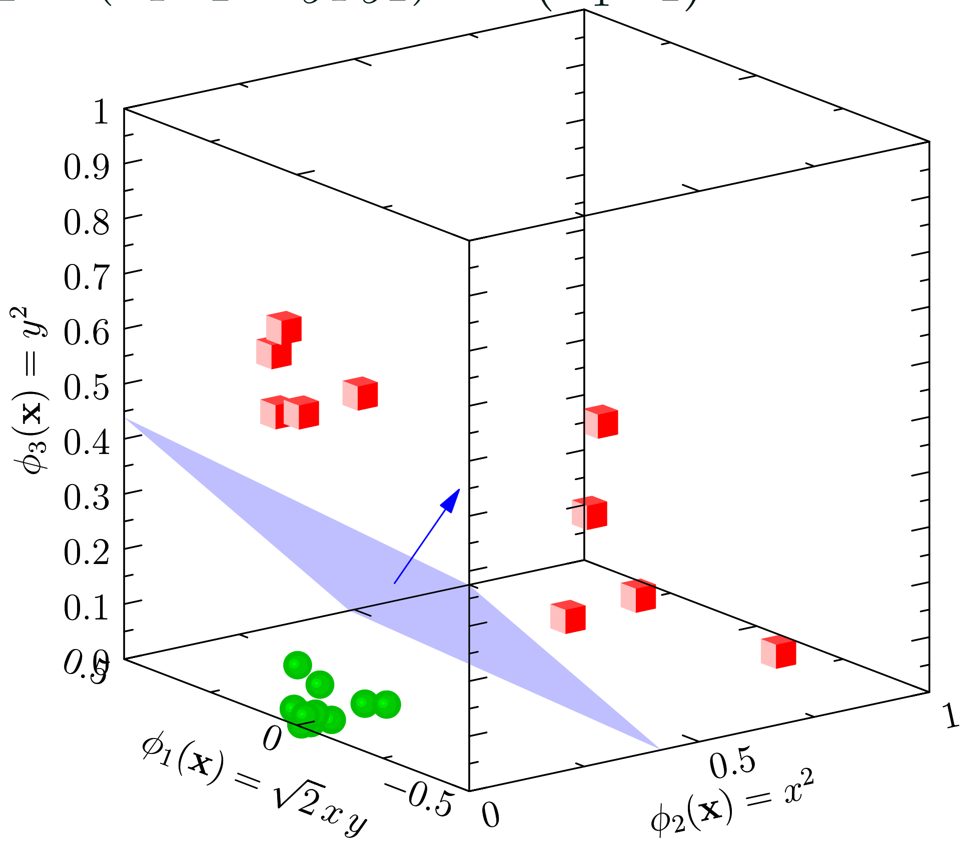
Non-linearly Separation of Data

$$K(\boldsymbol{x}_1,\, \boldsymbol{x}_2) = \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_1)\boldsymbol{\phi}(\boldsymbol{x}_2) = \begin{pmatrix} x_1^2 & y_1^2 & \sqrt{2}\, x_1\, y_1 \end{pmatrix} \begin{pmatrix} x_2^2 \\ y_2^2 \\ \sqrt{2}\, x_2\, y_2 \end{pmatrix}$$

$$= x_1^2\, x_2^2 + y_1^2\, y_2^2 + 2\, x_1\, y_1\, x_2\, y_2 = (x_1\, x_2 + y_1\, y_2)^2 = \left(\boldsymbol{x}_1^T \boldsymbol{x}_2\right)^2$$



$$\xrightarrow{\boldsymbol{x} \to \boldsymbol{\phi}(\boldsymbol{x})}$$



Computing the Maximum-Margin Hyper-plane

- Although it is not hugely difficult to derive the equations for finding the maximum-margin hyper-plane it is not that illuminating (although very elegant)

- Never need to compute $\phi_i(\boldsymbol{x})$ only need to compute $\boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_i)\boldsymbol{\phi}(\boldsymbol{x}_j)$

- We end up with a quadratic programming problem, which requires some sophisticated algorithms to solve

- When we use the kernel trick the time to compute the solution to the quadratic programming problem is $p\,N^3$ where N is the number of training examples and p is the number of features

Computing the Maximum-Margin Hyper-plane

- Although it is not hugely difficult to derive the equations for finding the maximum-margin hyper-plane it is not that illuminating (although very elegant)

- Never need to compute $\phi_i(\boldsymbol{x})$ only need to compute $\boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_i)\boldsymbol{\phi}(\boldsymbol{x}_j)$

- We end up with a quadratic programming problem, which requires some sophisticated algorithms to solve

- When we use the kernel trick the time to compute the solution to the quadratic programming problem is $p\,N^3$ where N is the number of training examples and p is the number of features

Computing the Maximum-Margin Hyper-plane

- Although it is not hugely difficult to derive the equations for finding the maximum-margin hyper-plane it is not that illuminating (although very elegant)
- Never need to compute $\phi_i(\boldsymbol{x})$ only need to compute $\boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_i)\boldsymbol{\phi}(\boldsymbol{x}_j)$
- We end up with a quadratic programming problem, which requires some sophisticated algorithms to solve
- When we use the kernel trick the time to compute the solution to the quadratic programming problem is $p\,N^3$ where N is the number of training examples and p is the number of features

Computing the Maximum-Margin Hyper-plane

- Although it is not hugely difficult to derive the equations for finding the maximum-margin hyper-plane it is not that illuminating (although very elegant)

- Never need to compute $\phi_i(\boldsymbol{x})$ only need to compute $\boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_i)\boldsymbol{\phi}(\boldsymbol{x}_j)$

- We end up with a quadratic programming problem, which requires some sophisticated algorithms to solve

- When we use the kernel trick the time to compute the solution to the quadratic programming problem is $p\,N^3$ where N is the number of training examples and p is the number of features

Getting SVMs to Work Well

- SVMs rely on distances between data points
- These will change relative to each other if we rescale some features but not other—giving different maximum-margin hyper-planes
- If we don't know what features are important (most often the case), then it is worth scaling each feature (for example, so their range is between 0 and 1 or their variance is 1)

Getting SVMs to Work Well

- SVMs rely on distances between data points
- These will change relative to each other if we rescale some features but not other—giving different maximum-margin hyper-planes
- If we don't know what features are important (most often the case), then it is worth scaling each feature (for example, so their range is between 0 and 1 or their variance is 1)

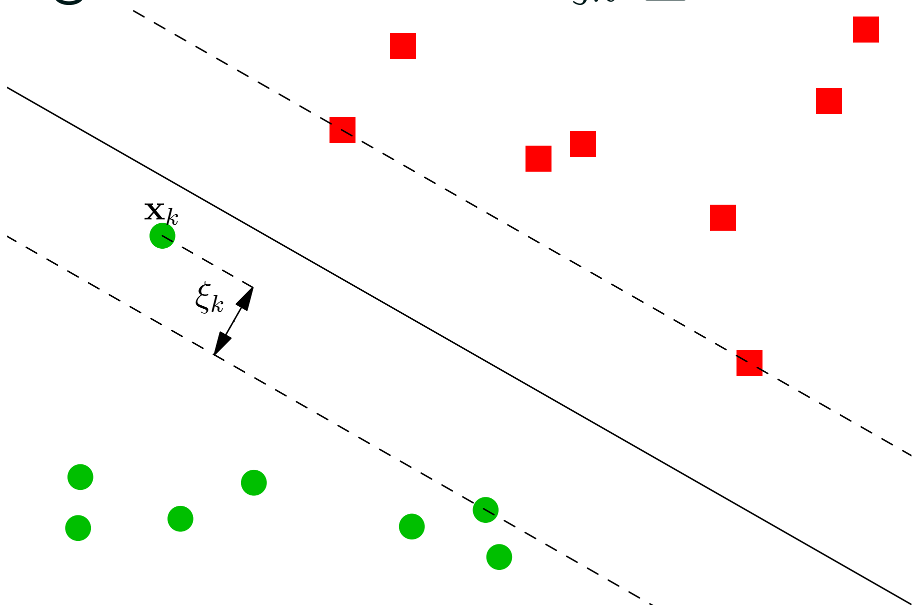
Getting SVMs to Work Well

- SVMs rely on distances between data points
- These will change relative to each other if we rescale some features but not other—giving different maximum-margin hyper-planes
- If we don't know what features are important (most often the case), then it is worth scaling each feature (for example, so their range is between 0 and 1 or their variance is 1)

Soft Margins

- Sometimes the margin constraint is too severe
- Relax constraints by introducing *slack variables*, $\xi_k \geq 0$

$$y_k(\boldsymbol{x}_k^{\mathsf{T}}\boldsymbol{w} - b) \geq 1 - \xi_k$$

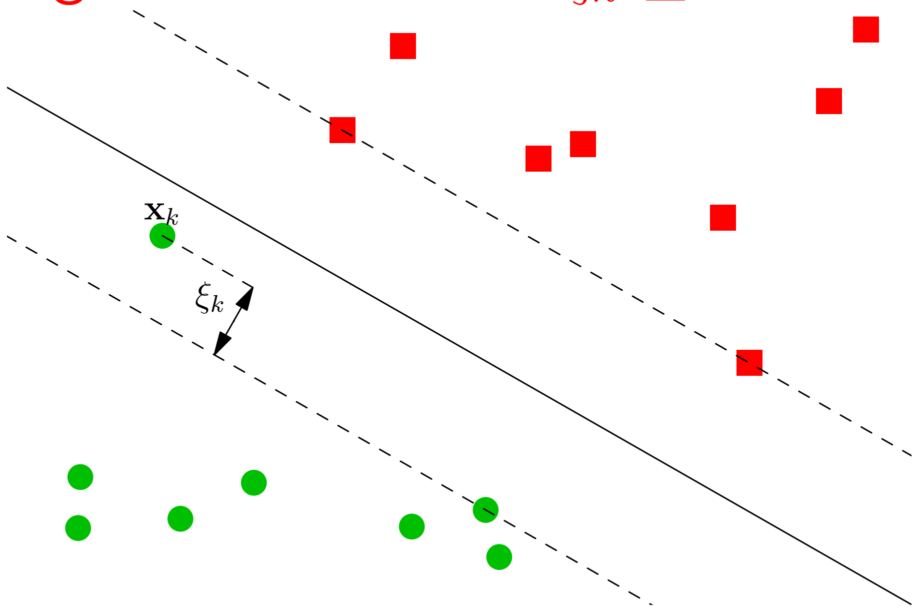


- Minimise $\frac{\|\boldsymbol{w}\|^2}{2} + C\sum_{k=1}^{n} \xi_k$ subject to constraints
- Large C punishes slack variables

Soft Margins

- Sometimes the margin constraint is too severe
- Relax constraints by introducing *slack variables*, $\xi_k \geq 0$

$$y_k(\boldsymbol{x}_k^\mathsf{T}\boldsymbol{w} - b) \geq 1 - \xi_k$$

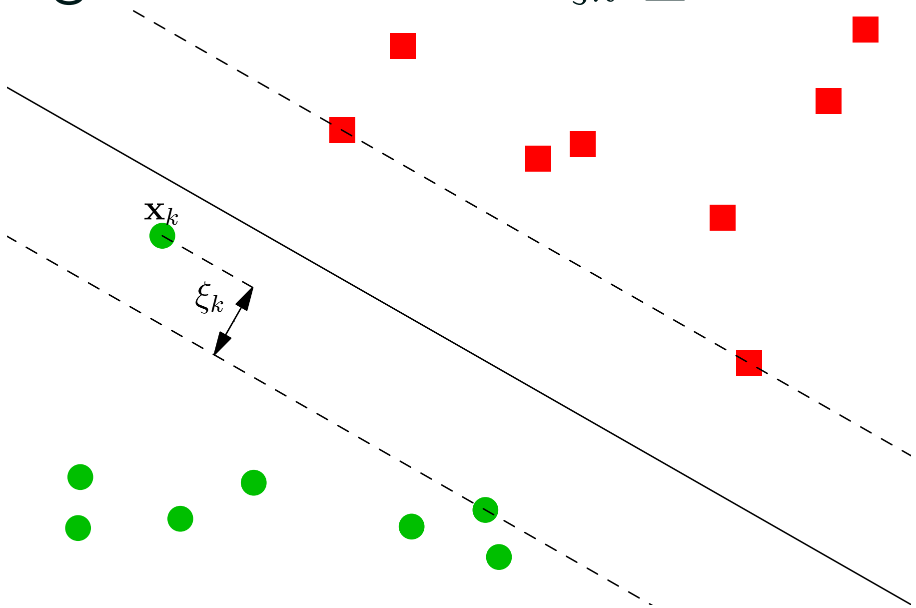


- Minimise $\frac{\|\boldsymbol{w}\|^2}{2} + C\sum_{k=1}^{n}\xi_k$ subject to constraints
- Large C punishes slack variables

Soft Margins

- Sometimes the margin constraint is too severe
- Relax constraints by introducing *slack variables*, $\xi_k \geq 0$

$$y_k(\boldsymbol{x}_k^{\mathsf{T}}\boldsymbol{w} - b) \geq 1 - \xi_k$$



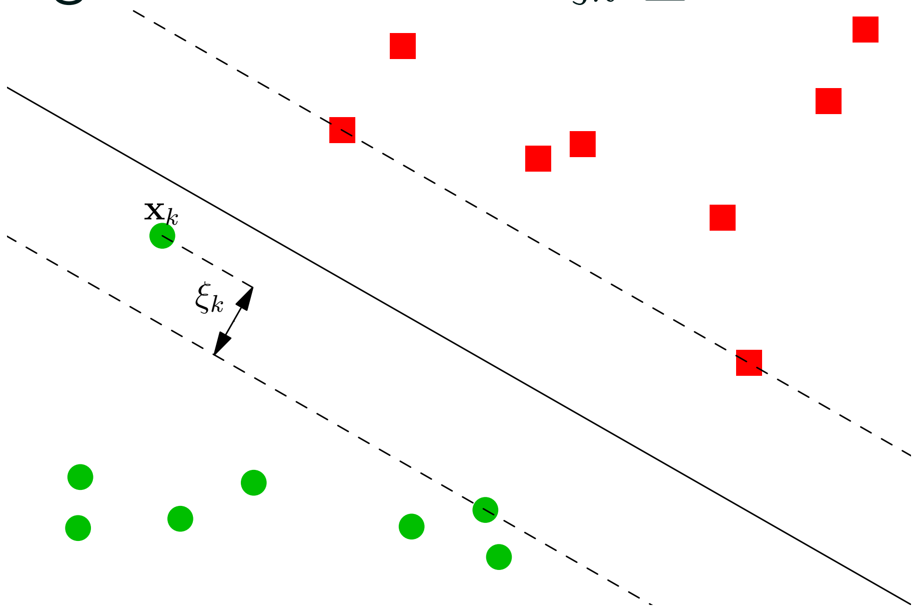
- Minimise $\frac{\|\boldsymbol{w}\|^2}{2} + C\sum_{k=1}^{n} \xi_k$ subject to constraints
- Large C punishes slack variables

Soft Margins

- Sometimes the margin constraint is too severe

- Relax constraints by introducing *slack variables*, $\xi_k \geq 0$

$$y_k(\boldsymbol{x}_k^{\mathsf{T}}\boldsymbol{w} - b) \geq 1 - \xi_k$$



- Minimise $\frac{\|\boldsymbol{w}\|^2}{2} + C\sum_{k=1}^{n} \xi_k$ subject to constraints

- Large C punishes slack variables

Optimising C

- In practice it can make a huge difference to the performance if we change C
- Optimal C values changes by many orders of magnitude e.g. 2^{-5}–2^{15}
- Typically optimised by a grid search (start from 2^{-5} say and double until you reach 2^{15})

Optimising C

- In practice it can make a huge difference to the performance if we change C
- Optimal C values changes by many orders of magnitude e.g. 2^{-5}–2^{15}
- Typically optimised by a grid search (start from 2^{-5} say and double until you reach 2^{15})

Optimising C

- In practice it can make a huge difference to the performance if we change C

- Optimal C values changes by many orders of magnitude e.g. 2^{-5}–2^{15}

- Typically optimised by a grid search (start from 2^{-5} say and double until you reach 2^{15})

Choosing the Right Kernel Function

- There are kernels design for particular data types (e.g. string kernels for text or biological sequences)
- For numerical data people tend to look at using no kernel (linear SVM), a radial basis function (Gaussian) kernel or polynomial kernels
- Kernel's often come with parameters, e.g. the popular radial basis function kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{e}^{-\gamma \|\boldsymbol{x}-\boldsymbol{y}\|^2}$$

- Optimal γ values range over 2^{-15}–2^{3}

Choosing the Right Kernel Function

- There are kernels design for particular data types (e.g. string kernels for text or biological sequences)

- For numerical data people tend to look at using no kernel (linear SVM), a radial basis function (Gaussian) kernel or polynomial kernels

- Kernel's often come with parameters, e.g. the popular radial basis function kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{e}^{-\gamma \, \|\boldsymbol{x}-\boldsymbol{y}\|^2}$$

- Optimal γ values range over 2^{-15}–2^3

Choosing the Right Kernel Function

- There are kernels design for particular data types (e.g. string kernels for text or biological sequences)

- For numerical data people tend to look at using no kernel (linear SVM), a radial basis function (Gaussian) kernel or polynomial kernels

- Kernel's often come with parameters, e.g. the popular radial basis function kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{e}^{-\gamma \, \|\boldsymbol{x}-\boldsymbol{y}\|^2}$$

- Optimal γ values range over 2^{-15}–2^3

Choosing the Right Kernel Function

- There are kernels design for particular data types (e.g. string kernels for text or biological sequences)
- For numerical data people tend to look at using no kernel (linear SVM), a radial basis function (Gaussian) kernel or polynomial kernels
- Kernel's often come with parameters, e.g. the popular radial basis function kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{e}^{-\gamma \, \|\boldsymbol{x}-\boldsymbol{y}\|^2}$$

- Optimal γ values range over 2^{-15}–2^3

SVM Libraries

- Although SVMs have unique solutions, they require very well written optimisers
- If you have a large data set they can be very slow
- There are good libraries out there, svmlib, svm-lite, etc.
- These will often automate normalisation of data and grid search for parameters

SVM Libraries

- Although SVMs have unique solutions, they require very well written optimisers
- If you have a large data set they can be very slow
- There are good libraries out there, svmlib, svm-lite, etc.
- These will often automate normalisation of data and grid search for parameters

SVM Libraries

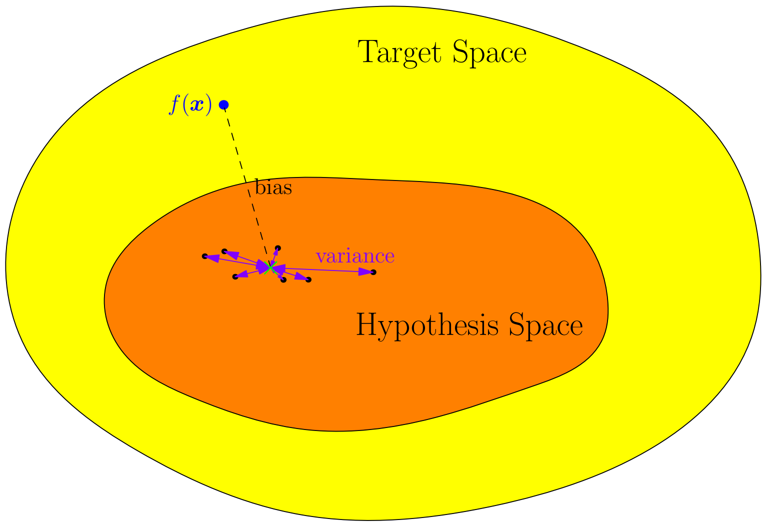
- Although SVMs have unique solutions, they require very well written optimisers
- If you have a large data set they can be very slow
- There are good libraries out there, svmlib, svm-lite, etc.
- These will often automate normalisation of data and grid search for parameters

SVM Libraries

- Although SVMs have unique solutions, they require very well written optimisers
- If you have a large data set they can be very slow
- There are good libraries out there, svmlib, svm-lite, etc.
- These will often automate normalisation of data and grid search for parameters

Outline

1. What Makes a Good Learning Machine?
2. SVMs
3. **Ensemble Methods**
4. Bayesian Inference



Removing Variance By Averaging

- We can reduce the variance and hence improve our generalisation error by averaging over different learning machines
- There are a number of different techniques for doing this that go by the name of **ensemble methods** or **ensemble learning**
- This trick can be used with many different learning machines, but is clearly most practical for machine that can be trained quickly (nevertheless, even for deep learning taking the average response of many machines is usually done to win competitions)

Removing Variance By Averaging

- We can reduce the variance and hence improve our generalisation error by averaging over different learning machines

- There are a number of different techniques for doing this that go by the name of **ensemble methods** or **ensemble learning**

- This trick can be used with many different learning machines, but is clearly most practical for machine that can be trained quickly (nevertheless, even for deep learning taking the average response of many machines is usually done to win competitions)

Removing Variance By Averaging

- We can reduce the variance and hence improve our generalisation error by averaging over different learning machines

- There are a number of different techniques for doing this that go by the name of **ensemble methods** or **ensemble learning**

- This trick can be used with many different learning machines, but is clearly most practical for machine that can be trained quickly (nevertheless, even for deep learning taking the average response of many machines is usually done to win competitions)

Ensembling of Decision Trees

- One set of algorithms where ensembling are common place are decision trees

- These are particularly good for handling messy data
  - ⋆ categorical data
  - ⋆ mixture of data types
  - ⋆ missing data
  - ⋆ large data sets
  - ⋆ multiclass

- In many competitions ensembled trees, particularly *random forests* and *gradient boosing* beats all other techniques

Ensembling of Decision Trees

- One set of algorithms where ensembling are common place are decision trees

- These are particularly good for handling messy data
  - ⋆ categorical data
  - ⋆ mixture of data types
  - ⋆ missing data
  - ⋆ large data sets
  - ⋆ multiclass

- In many competitions ensembled trees, particularly *random forests* and *gradient boosing* beats all other techniques

Ensembling of Decision Trees

- One set of algorithms where ensembling are common place are decision trees
- These are particularly good for handling messy data
  - ⋆ categorical data
  - ⋆ mixture of data types
  - ⋆ missing data
  - ⋆ large data sets
  - ⋆ multiclass
- In many competitions ensembled trees, particularly *random forests* and *gradient boosing* beats all other techniques

Ensembling of Decision Trees

- One set of algorithms where ensembling are common place are decision trees
- These are particularly good for handling messy data
  - ⋆ categorical data
  - ⋆ mixture of data types
  - ⋆ missing data
  - ⋆ large data sets
  - ⋆ multiclass
- In many competitions ensembled trees, particularly *random forests* and *gradient boosing* beats all other techniques

Ensembling of Decision Trees

- One set of algorithms where ensembling are common place are decision trees
- These are particularly good for handling messy data
  - ⋆ categorical data
  - ⋆ mixture of data types
  - ⋆ missing data
  - ⋆ large data sets
  - ⋆ multiclass
- In many competitions ensembled trees, particularly $random$ $forests$ and $gradient$ $boosing$ beats all other techniques

Ensembling of Decision Trees

- One set of algorithms where ensembling are common place are decision trees

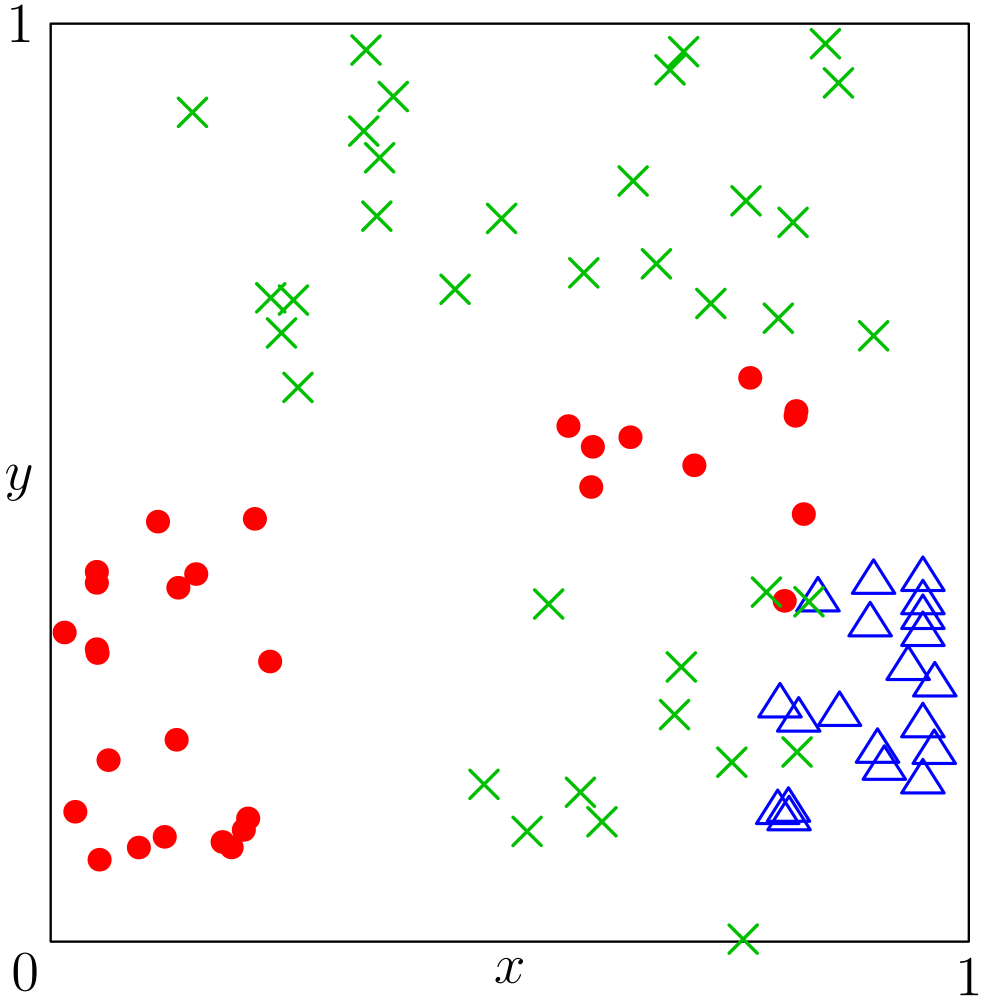
- These are particularly good for handling messy data
  - ⋆ categorical data
  - ⋆ mixture of data types
  - ⋆ missing data
  - ⋆ large data sets
  - ⋆ multiclass

- In many competitions ensembled trees, particularly *random forests* and *gradient boosing* beats all other techniques

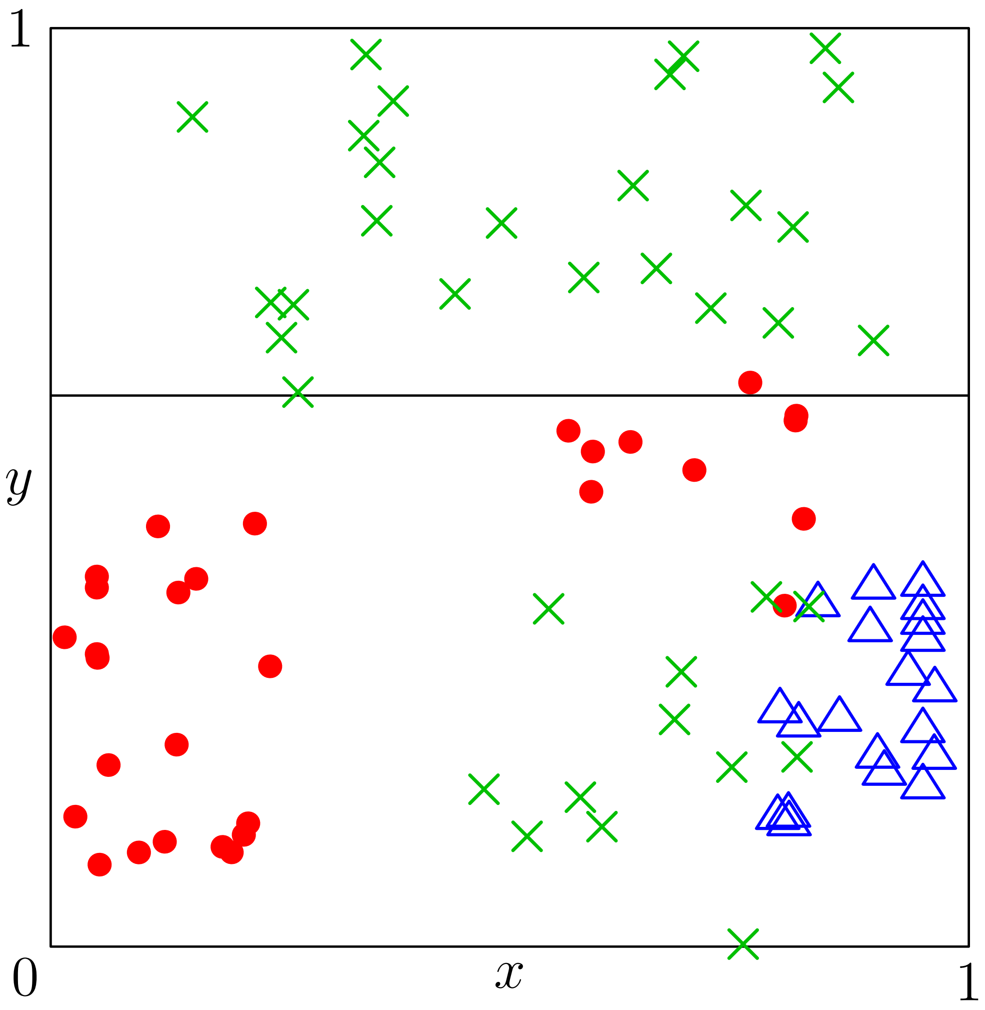
Ensembling of Decision Trees

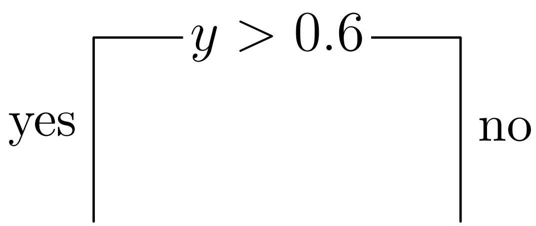
- One set of algorithms where ensembling are common place are decision trees
- These are particularly good for handling messy data
  - ⋆ categorical data
  - ⋆ mixture of data types
  - ⋆ missing data
  - ⋆ large data sets
  - ⋆ multiclass
- In many competitions ensembled trees, particularly *random forests* and *gradient boosing* beats all other techniques

Building Decision Trees

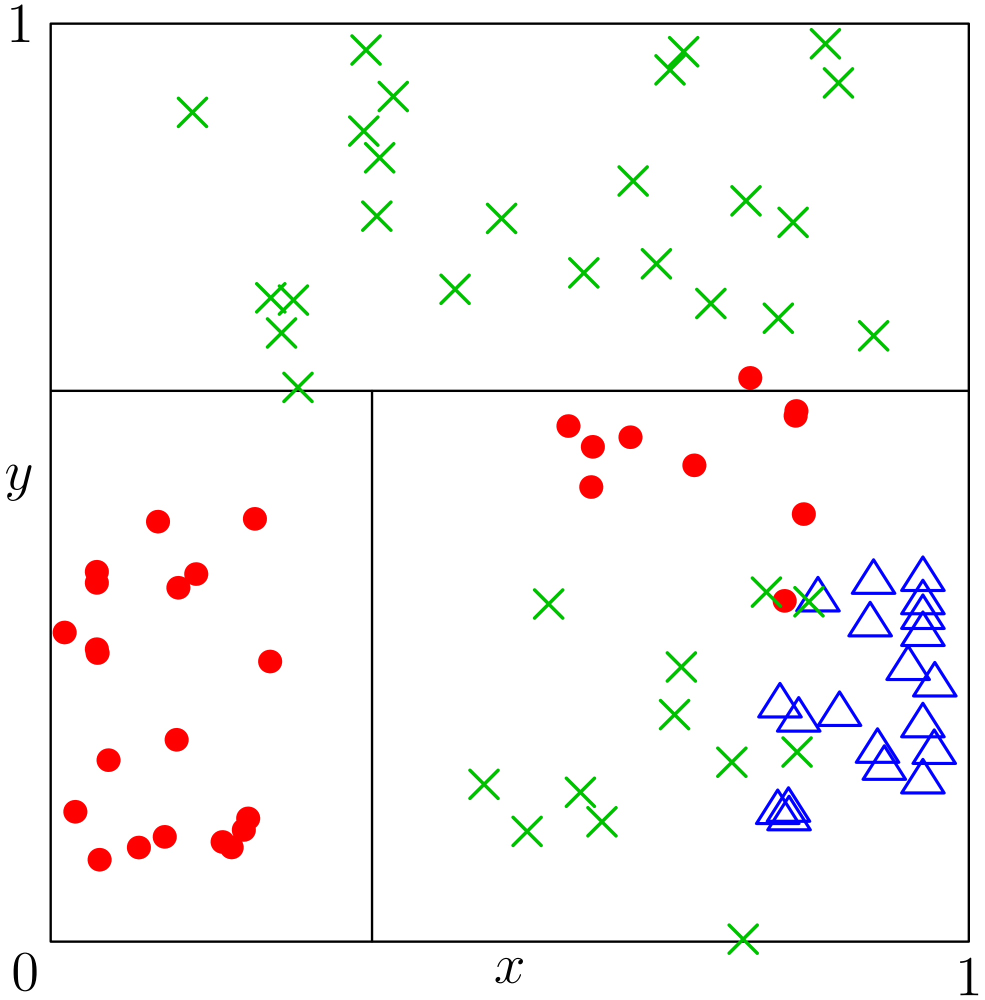


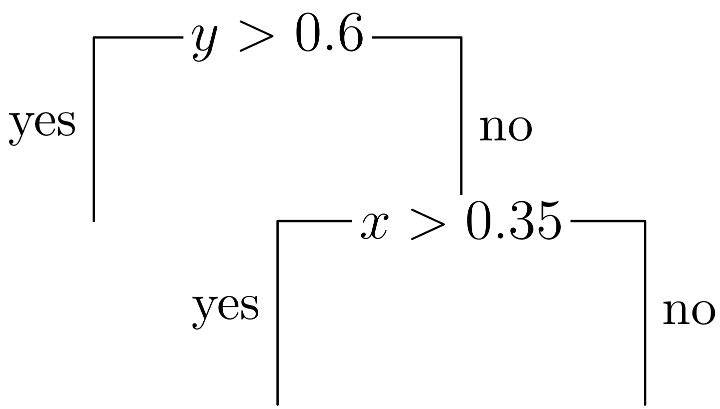
Building Decision Trees



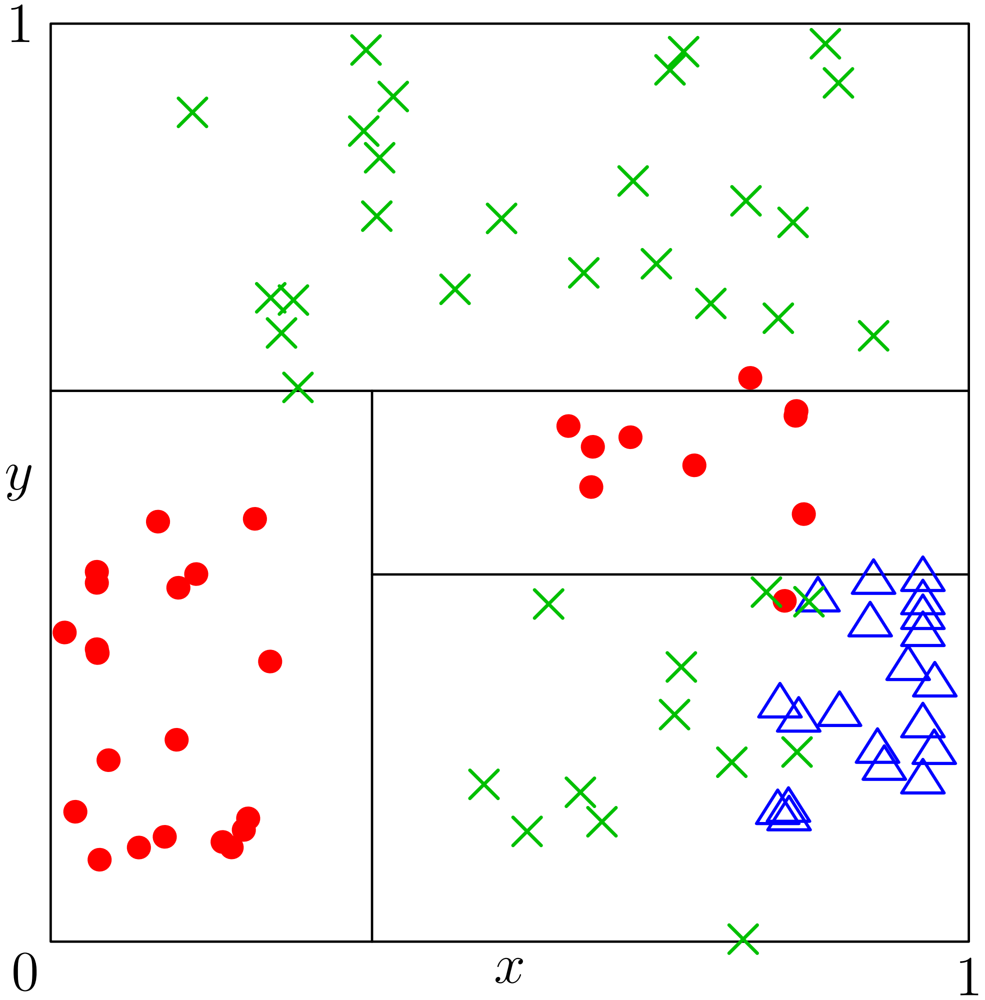


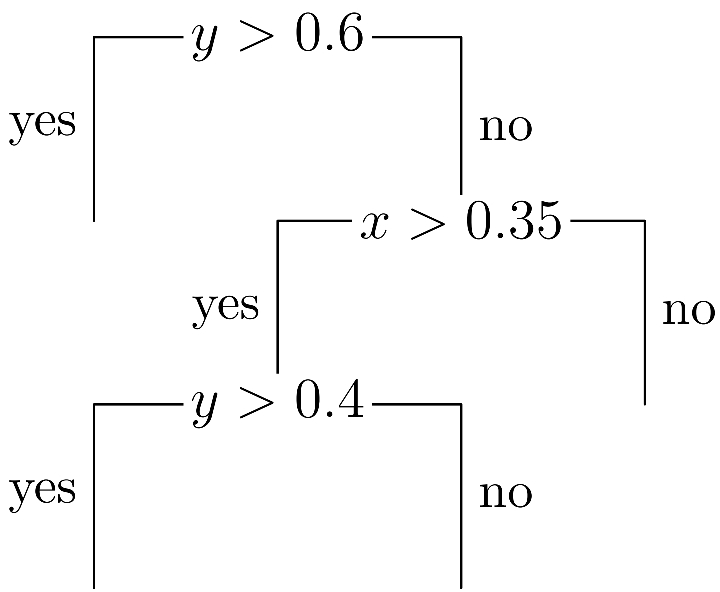
Building Decision Trees



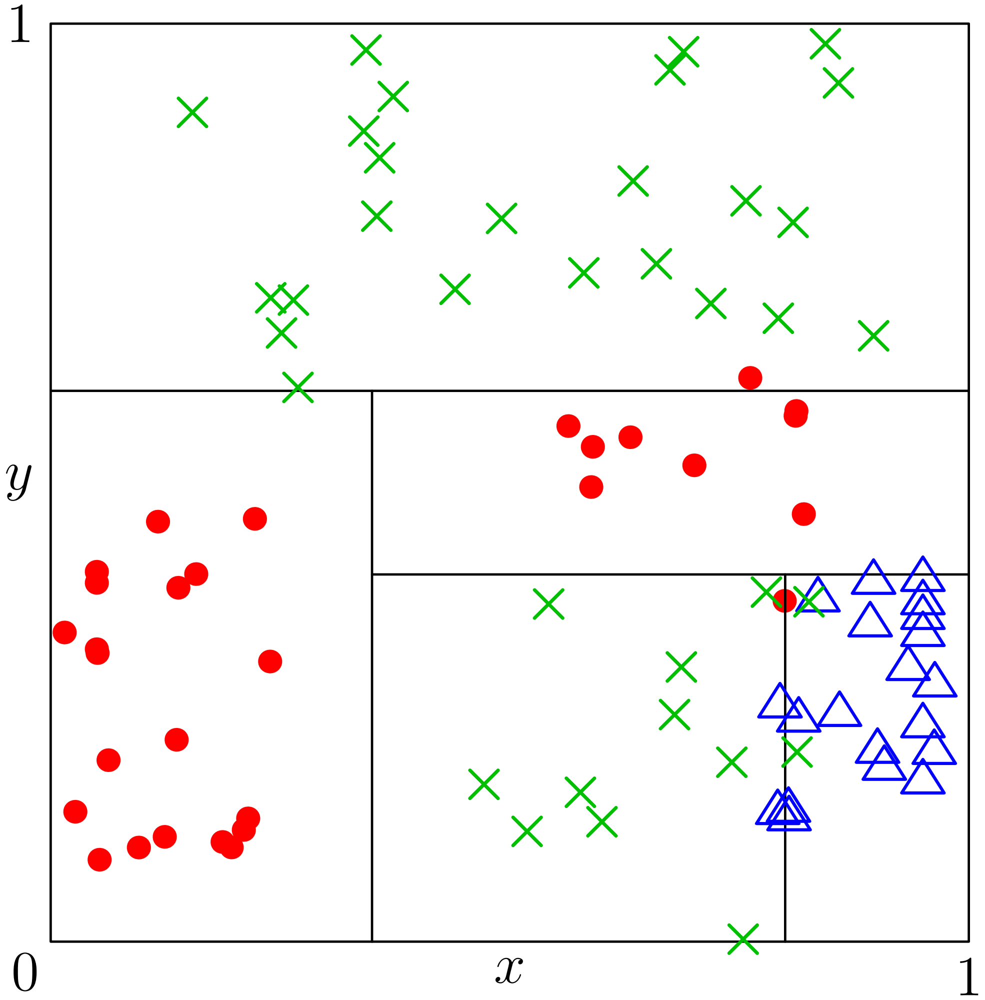


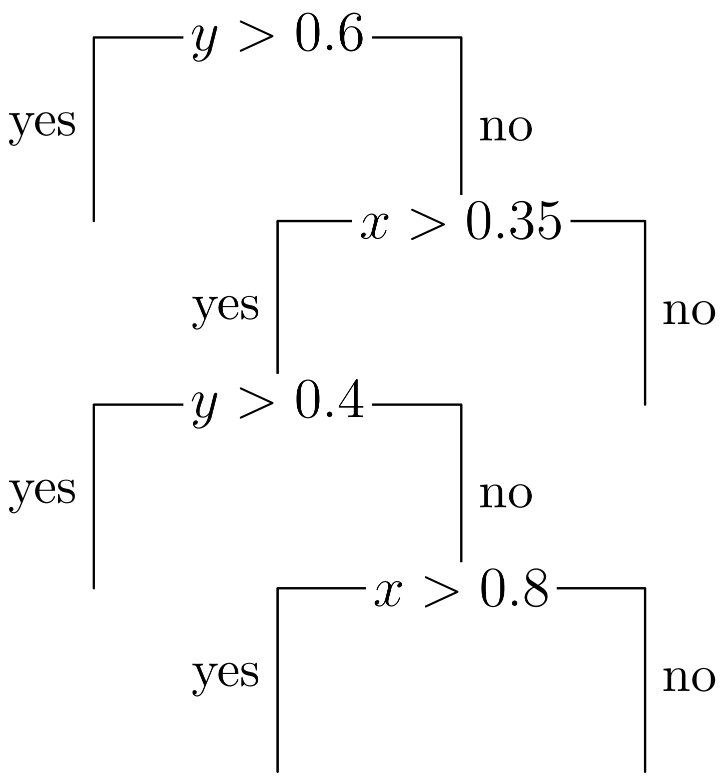
Building Decision Trees



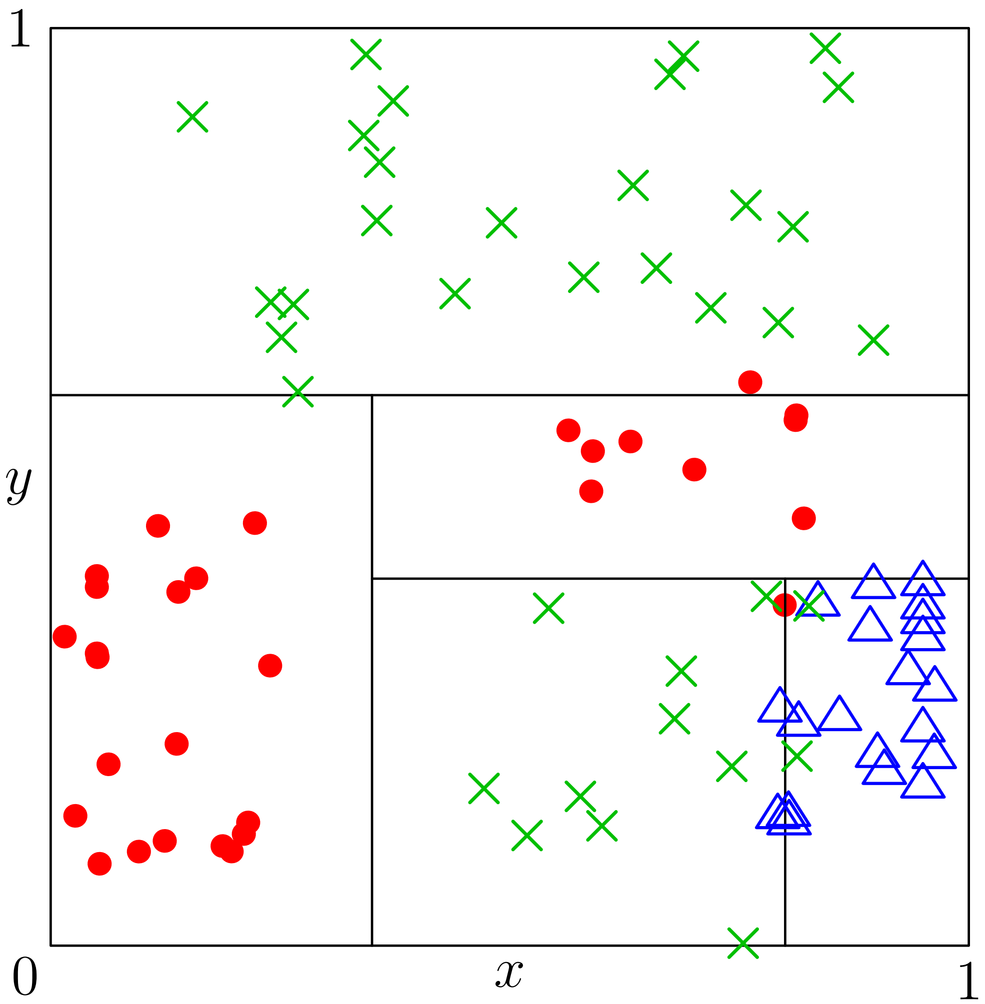


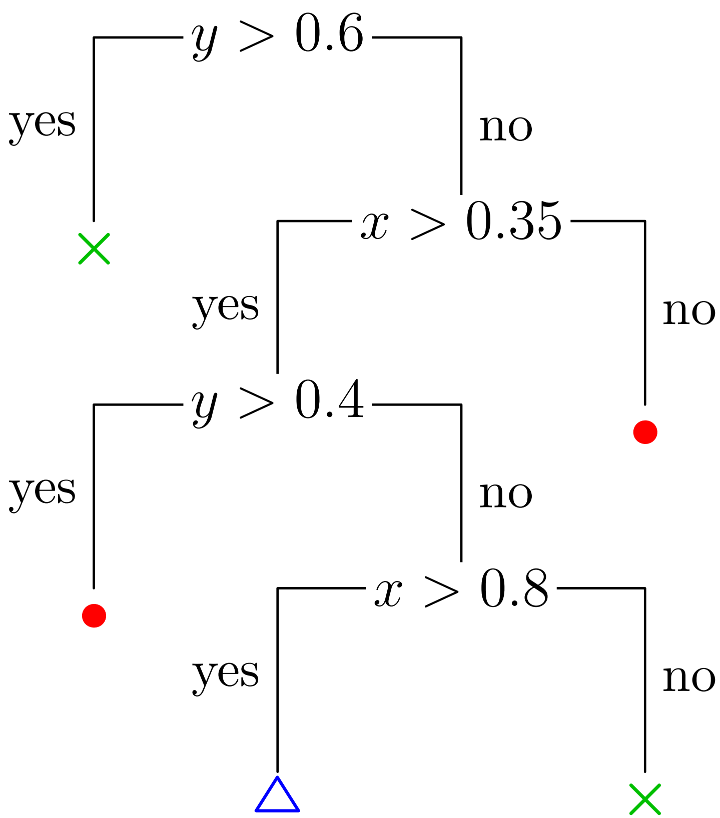
Building Decision Trees





Building Decision Trees





Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different

- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$(\boldsymbol{x}_1, y_1)$ $(\boldsymbol{x}_2, y_2)$ $(\boldsymbol{x}_3, y_3)$ $(\boldsymbol{x}_4, y_4)$ $(\boldsymbol{x}_5, y_5)$ $(\boldsymbol{x}_6, y_6)$

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$(\boldsymbol{x}_5,\ y_5)$

$(\boldsymbol{x}_1,\ y_1)$ $(\boldsymbol{x}_2,\ y_2)$ $(\boldsymbol{x}_3,\ y_3)$ $(\boldsymbol{x}_4,\ y_4)$ $(\boldsymbol{x}_5,\ y_5)$ $(\boldsymbol{x}_6,\ y_6)$

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$(\boldsymbol{x}_5,\, y_5)\ \ (\boldsymbol{x}_4,\, y_4)$

$(\boldsymbol{x}_1,\, y_1)\ \ (\boldsymbol{x}_2,\, y_2)\ \ (\boldsymbol{x}_3,\, y_3)\ \ (\boldsymbol{x}_4,\, y_4)\ \ (\boldsymbol{x}_5,\, y_5)\ \ (\boldsymbol{x}_6,\, y_6)$

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$(\boldsymbol{x}_5, y_5)$ $(\boldsymbol{x}_4, y_4)$ $(\boldsymbol{x}_5, y_5)$

$(\boldsymbol{x}_1, y_1)$ $(\boldsymbol{x}_2, y_2)$ $(\boldsymbol{x}_3, y_3)$ $(\boldsymbol{x}_4, y_4)$ $(\boldsymbol{x}_5, y_5)$ $(\boldsymbol{x}_6, y_6)$

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$(\boldsymbol{x}_5,\, y_5)\;\; (\boldsymbol{x}_4,\, y_4)\;\; (\boldsymbol{x}_5,\, y_5)\;\; (\boldsymbol{x}_6,\, y_6)$

$(\boldsymbol{x}_1,\, y_1)\;\; (\boldsymbol{x}_2,\, y_2)\;\; (\boldsymbol{x}_3,\, y_3)\;\; (\boldsymbol{x}_4,\, y_4)\;\; (\boldsymbol{x}_5,\, y_5)\;\; (\boldsymbol{x}_6,\, y_6)$

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$(\boldsymbol{x}_5, y_5)$ $(\boldsymbol{x}_4, y_4)$ $(\boldsymbol{x}_5, y_5)$ $(\boldsymbol{x}_6, y_6)$ $(\boldsymbol{x}_2, y_2)$

$(\boldsymbol{x}_1, y_1)$ $(\boldsymbol{x}_2, y_2)$ $(\boldsymbol{x}_3, y_3)$ $(\boldsymbol{x}_4, y_4)$ $(\boldsymbol{x}_5, y_5)$ $(\boldsymbol{x}_6, y_6)$

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$(\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_4,\, y_4)\ (\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_6,\, y_6)\ (\boldsymbol{x}_2,\, y_2)\ (\boldsymbol{x}_1,\, y_1)$

$(\boldsymbol{x}_1,\, y_1)\ (\boldsymbol{x}_2,\, y_2)\ (\boldsymbol{x}_3,\, y_3)\ (\boldsymbol{x}_4,\, y_4)\ (\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_6,\, y_6)$

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$(\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_4,\, y_4)\ (\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_6,\, y_6)\ (\boldsymbol{x}_2,\, y_2)\ (\boldsymbol{x}_1,\, y_1)$

$(\boldsymbol{x}_1,\, y_1)\ (\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_3,\, y_3)\ (\boldsymbol{x}_2,\, y_2)\ (\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_2,\, y_2)$

$(\boldsymbol{x}_1,\, y_1)\ (\boldsymbol{x}_2,\, y_2)\ (\boldsymbol{x}_3,\, y_3)\ (\boldsymbol{x}_4,\, y_4)\ (\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_6,\, y_6)$

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$(\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_4,\, y_4)\ (\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_6,\, y_6)\ (\boldsymbol{x}_2,\, y_2)\ (\boldsymbol{x}_1,\, y_1)$

$(\boldsymbol{x}_1,\, y_1)\ (\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_3,\, y_3)\ (\boldsymbol{x}_2,\, y_2)\ (\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_2,\, y_2)$

$(\boldsymbol{x}_1,\, y_1)\ (\boldsymbol{x}_2,\, y_2)\ (\boldsymbol{x}_3,\, y_3)\ (\boldsymbol{x}_4,\, y_4)\ (\boldsymbol{x}_5,\, y_5)\ (\boldsymbol{x}_6,\, y_6)$ $\qquad$ $(\boldsymbol{x}_6,\, y_6)\ (\boldsymbol{x}_1,\, y_1)\ (\boldsymbol{x}_3,\, y_3)\ (\boldsymbol{x}_6,\, y_6)\ (\boldsymbol{x}_3,\, y_3)\ (\boldsymbol{x}_4,\, y_4)$

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$$(\boldsymbol{x}_1, y_1)\ (\boldsymbol{x}_2, y_2)\ (\boldsymbol{x}_3, y_3)\ (\boldsymbol{x}_4, y_4)\ (\boldsymbol{x}_5, y_5)\ (\boldsymbol{x}_6, y_6)$$

$$\begin{array}{cccccc}
(\boldsymbol{x}_5, y_5) & (\boldsymbol{x}_4, y_4) & (\boldsymbol{x}_5, y_5) & (\boldsymbol{x}_6, y_6) & (\boldsymbol{x}_2, y_2) & (\boldsymbol{x}_1, y_1) \\
(\boldsymbol{x}_1, y_1) & (\boldsymbol{x}_5, y_5) & (\boldsymbol{x}_3, y_3) & (\boldsymbol{x}_2, y_2) & (\boldsymbol{x}_5, y_5) & (\boldsymbol{x}_2, y_2) \\
(\boldsymbol{x}_6, y_6) & (\boldsymbol{x}_1, y_1) & (\boldsymbol{x}_3, y_3) & (\boldsymbol{x}_6, y_6) & (\boldsymbol{x}_3, y_3) & (\boldsymbol{x}_4, y_4) \\
(\boldsymbol{x}_4, y_4) & (\boldsymbol{x}_3, y_3) & (\boldsymbol{x}_1, y_1) & (\boldsymbol{x}_6, y_6) & (\boldsymbol{x}_4, y_4) & (\boldsymbol{x}_6, y_6)
\end{array}$$

Bootstrap Aggregation (Bagging)

- In order to average machine they must learn something different
- We only have one data set, but we can resample from the data set to make them look a bit different—this is known a **bootstrapping**

$(\boldsymbol{x}_1, y_1)\ (\boldsymbol{x}_2, y_2)\ (\boldsymbol{x}_3, y_3)\ (\boldsymbol{x}_4, y_4)\ (\boldsymbol{x}_5, y_5)\ (\boldsymbol{x}_6, y_6)$

$(\boldsymbol{x}_5, y_5)\ (\boldsymbol{x}_4, y_4)\ (\boldsymbol{x}_5, y_5)\ (\boldsymbol{x}_6, y_6)\ (\boldsymbol{x}_2, y_2)\ (\boldsymbol{x}_1, y_1)$

$(\boldsymbol{x}_1, y_1)\ (\boldsymbol{x}_5, y_5)\ (\boldsymbol{x}_3, y_3)\ (\boldsymbol{x}_2, y_2)\ (\boldsymbol{x}_5, y_5)\ (\boldsymbol{x}_2, y_2)$

$(\boldsymbol{x}_6, y_6)\ (\boldsymbol{x}_1, y_1)\ (\boldsymbol{x}_3, y_3)\ (\boldsymbol{x}_6, y_6)\ (\boldsymbol{x}_3, y_3)\ (\boldsymbol{x}_4, y_4)$

$(\boldsymbol{x}_4, y_4)\ (\boldsymbol{x}_3, y_3)\ (\boldsymbol{x}_1, y_1)\ (\boldsymbol{x}_6, y_6)\ (\boldsymbol{x}_4, y_4)\ (\boldsymbol{x}_6, y_6)$

$(\boldsymbol{x}_3, y_3)\ (\boldsymbol{x}_2, y_2)\ (\boldsymbol{x}_3, y_3)\ (\boldsymbol{x}_4, y_4)\ (\boldsymbol{x}_6, y_6)\ (\boldsymbol{x}_2, y_2)$

Random Forest

- To make decision trees more distinct we can choose a subset of variables on which to split the tree
- Bagging with trees is known as **random forest** and is a very powerful technique
- In many instances it has been superseded by boosting algorithms

Random Forest

- To make decision trees more distinct we can choose a subset of variables on which to split the tree
- Bagging with trees is known as **random forest** and is a very powerful technique
- In many instances it has been superseded by boosting algorithms

Random Forest

- To make decision trees more distinct we can choose a subset of variables on which to split the tree
- Bagging with trees is known as **random forest** and is a very powerful technique
- In many instances it has been superseded by boosting algorithms

Boosting

- In boosting we make a **strong learner** by using a weighted sum of **weak learners**

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \, \hat{h}_i(\boldsymbol{x})$$

- Weak learners ($\hat{h}_i(\boldsymbol{x})$) are learning machine that do a little better than chance

- The trick is to choose the weights w_i

Boosting

- In boosting we make a **strong learner** by using a weighted sum of **weak learners**

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \, \hat{h}_i(\boldsymbol{x})$$

- Weak learners $(\hat{h}_i(\boldsymbol{x}))$ are learning machine that do a little better than chance

- The trick is to choose the weights w_i

Boosting

- In boosting we make a **strong learner** by using a weighted sum of **weak learners**

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \, \hat{h}_i(\boldsymbol{x})$$

- Weak learners ($\hat{h}_i(\boldsymbol{x})$) are learning machine that do a little better than chance

- The trick is to choose the weights w_i

Shallow Trees

- One of the most effective type of weak learner are very shallow trees, (e.g. $\sqrt{p}$)
- Use at most only small number of random variables, (e.g. $\sqrt{p}$)
- There are different algorithms for choosing the weights
  - ⋆ adaboost
  - ⋆ gradient boosting
- XGBoost is a very efficient library for gradient boosting on large data sets and often wins competition

Shallow Trees

- One of the most effective type of weak learner are very shallow trees, (e.g. $\sqrt{p}$)
- Use at most only small number of random variables, (e.g. $\sqrt{p}$)
- There are different algorithms for choosing the weights
  - ⋆ adaboost
  - ⋆ gradient boosting
- XGBoost is a very efficient library for gradient boosting on large data sets and often wins competition

Shallow Trees

- One of the most effective type of weak learner are very shallow trees, (e.g. $\sqrt{p}$)
- Use at most only small number of random variables, (e.g. $\sqrt{p}$)
- There are different algorithms for choosing the weights
  - ⋆ adaboost
  - ⋆ gradient boosting
- XGBoost is a very efficient library for gradient boosting on large data sets and often wins competition

Shallow Trees

- One of the most effective type of weak learner are very shallow trees, (e.g. $\sqrt{p}$)
- Use at most only small number of random variables, (e.g. $\sqrt{p}$)
- There are different algorithms for choosing the weights
  - adaboost—a classic algorithm for binary problems
  - gradient boosting
- XGBoost is a very efficient library for gradient boosting on large data sets and often wins competition

Shallow Trees

- One of the most effective type of weak learner are very shallow trees, (e.g. $\sqrt{p}$)

- Use at most only small number of random variables, (e.g. $\sqrt{p}$)

- There are different algorithms for choosing the weights
  - ⋆ adaboost—a classic algorithm for binary problems
  - ⋆ gradient boosting

- XGBoost is a very efficient library for gradient boosting on large data sets and often wins competition

Shallow Trees

- One of the most effective type of weak learner are very shallow trees, (e.g. $\sqrt{p}$)
- Use at most only small number of random variables, (e.g. $\sqrt{p}$)
- There are different algorithms for choosing the weights
  - ⋆ adaboost—a classic algorithm for binary problems
  - ⋆ gradient boosting—now the dominant method, train a classifier on the residual errors
- XGBoost is a very efficient library for gradient boosting on large data sets and often wins competition

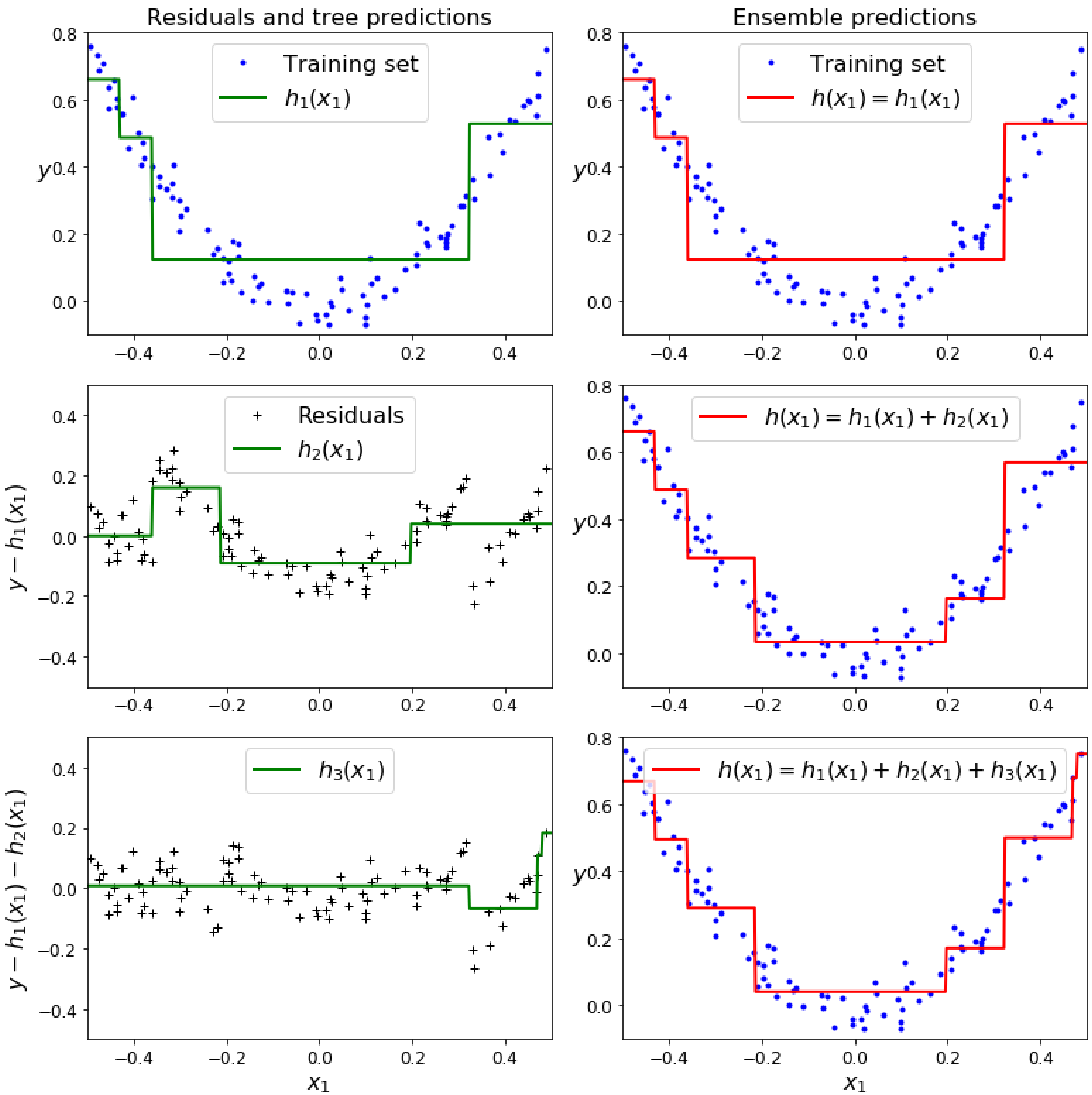
Shallow Trees

- One of the most effective type of weak learner are very shallow trees, (e.g. $\sqrt{p}$)

- Use at most only small number of random variables, (e.g. $\sqrt{p}$)

- There are different algorithms for choosing the weights
  - ⋆ adaboost—a classic algorithm for binary problems
  - ⋆ gradient boosting—now the dominant method, train a classifier on the residual errors

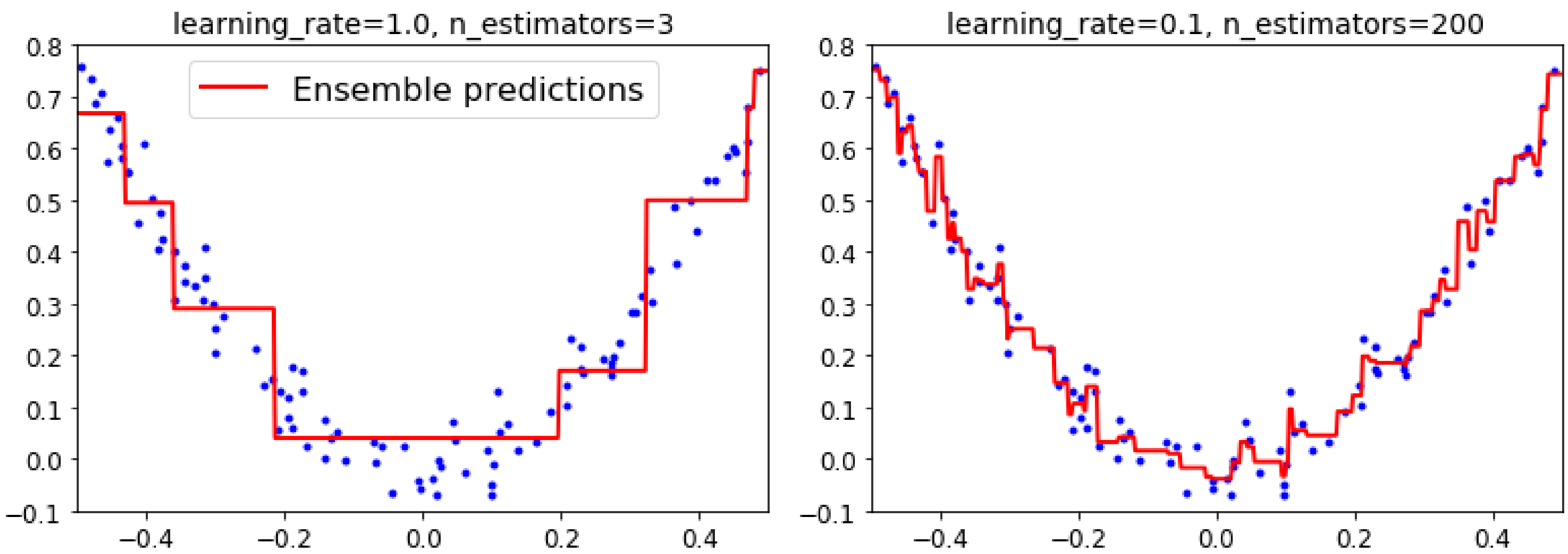
- XGBoost is a very efficient library for gradient boosting on large data sets and often wins competition



Residuals and tree predictions
Ensemble predictions
Training set
$h_1(x_1)$
$h(x_1) = h_1(x_1)$
Residuals
$h_2(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1)$
$h_3(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1) + h_3(x_1)$
y
$y - h_1(x_1)$
$y - h_1(x_1) - h_2(x_1)$
x_1

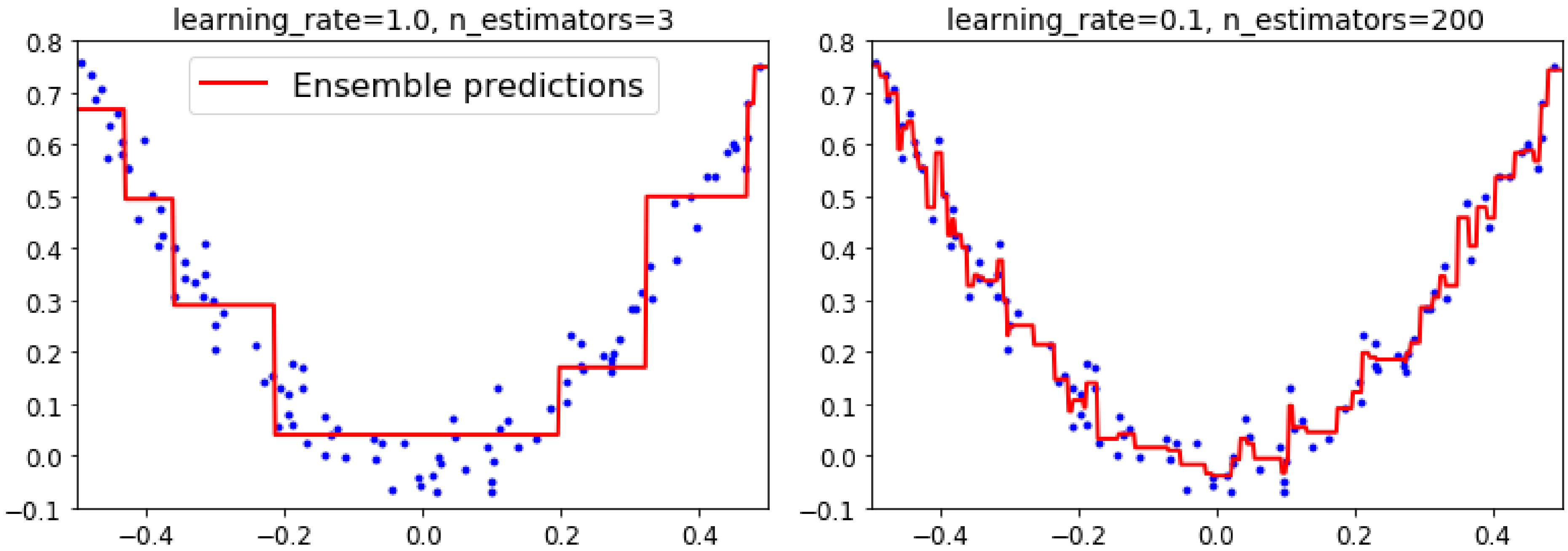
Keep On Going

- We can keep on going



Keep On Going

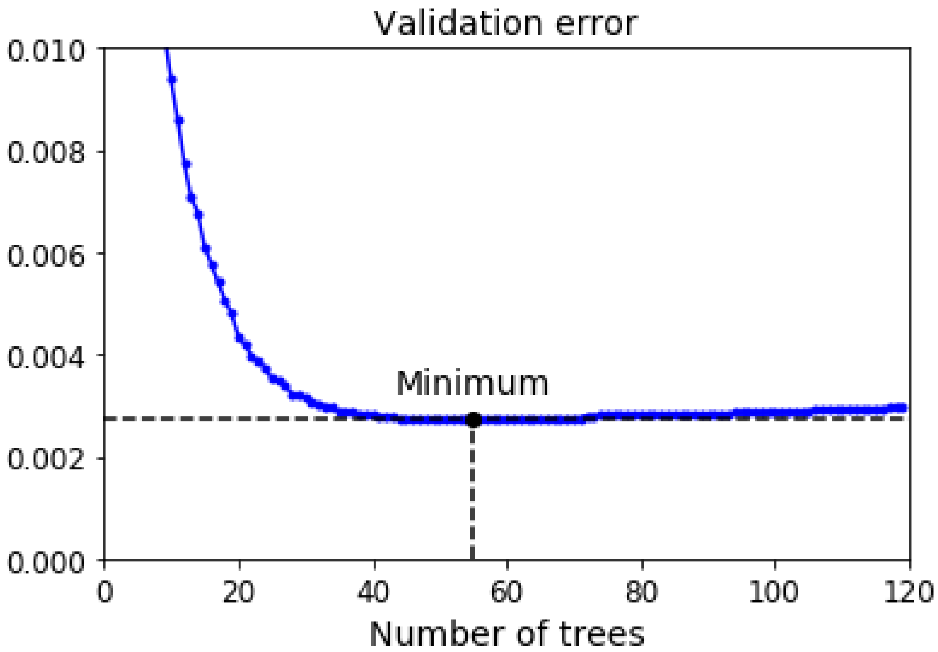
- We can keep on going

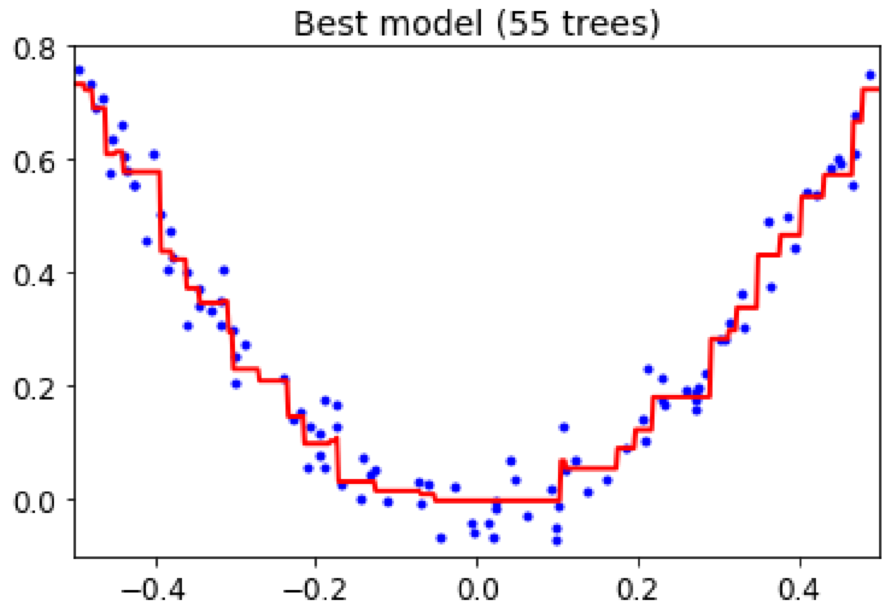


- But we will over-fit eventually

Early Stopping

- Like many algorithms we often get better results by early stopping

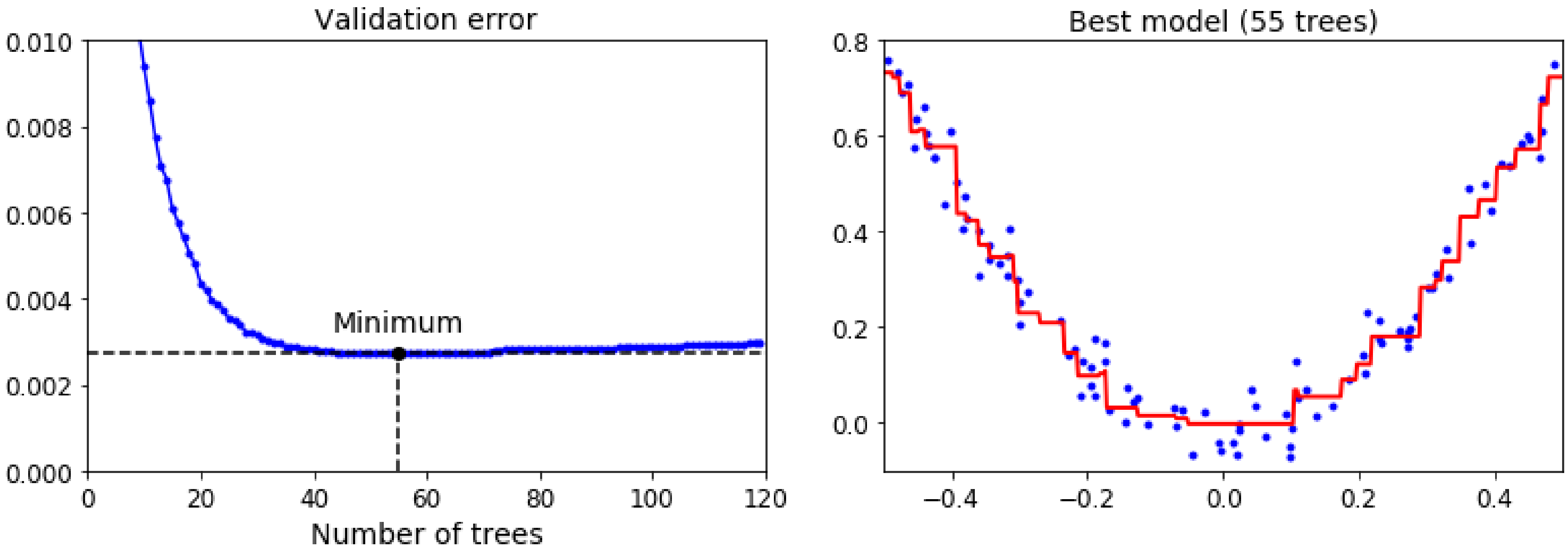




- Use cross-validation against a validation set to decide when to stop

Early Stopping

- Like many algorithms we often get better results by early stopping



- Use cross-validation against a validation set to decide when to stop

XGBoost

- XGBoost is an implementation of gradient boosting that won the Higg's Boson challenge and regularly wins Kaggle competitions

- XGBoost stands for eXtreme Gradient Boosting

- It is much faster than most gradient boosting algorithms and scales to billions of training data points

- It uses a cleverly chosen regularisation term to favour simple trees

- Finds a clever way to approximately minimise error plus regulariser very fast

XGBoost

- XGBoost is an implementation of gradient boosting that won the Higg's Boson challenge and regularly wins Kaggle competitions
- XGBoost stands for eXtreme Gradient Boosting
- It is much faster than most gradient boosting algorithms and scales to billions of training data points
- It uses a cleverly chosen regularisation term to favour simple trees
- Finds a clever way to approximately minimise error plus regulariser very fast

XGBoost

- XGBoost is an implementation of gradient boosting that won the Higg's Boson challenge and regularly wins Kaggle competitions
- XGBoost stands for eXtreme Gradient Boosting
- It is much faster than most gradient boosting algorithms and scales to billions of training data points
- It uses a cleverly chosen regularisation term to favour simple trees
- Finds a clever way to approximately minimise error plus regulariser very fast

XGBoost

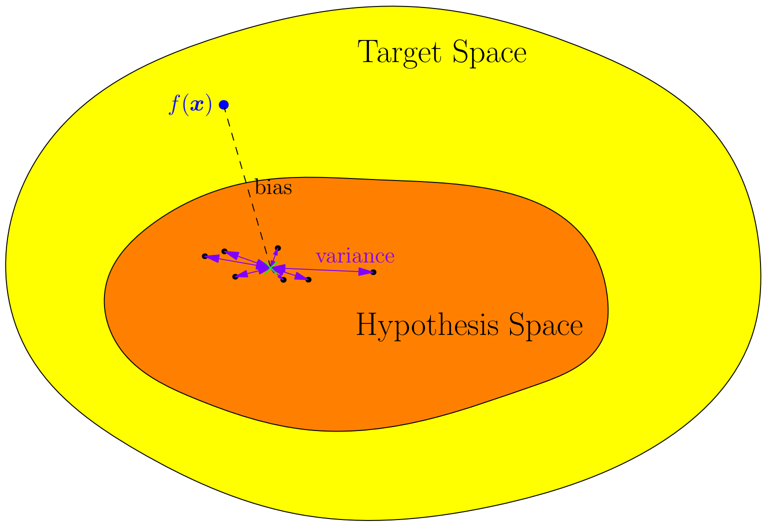
- XGBoost is an implementation of gradient boosting that won the Higg's Boson challenge and regularly wins Kaggle competitions
- XGBoost stands for eXtreme Gradient Boosting
- It is much faster than most gradient boosting algorithms and scales to billions of training data points
- It uses a cleverly chosen regularisation term to favour simple trees
- Finds a clever way to approximately minimise error plus regulariser very fast

XGBoost

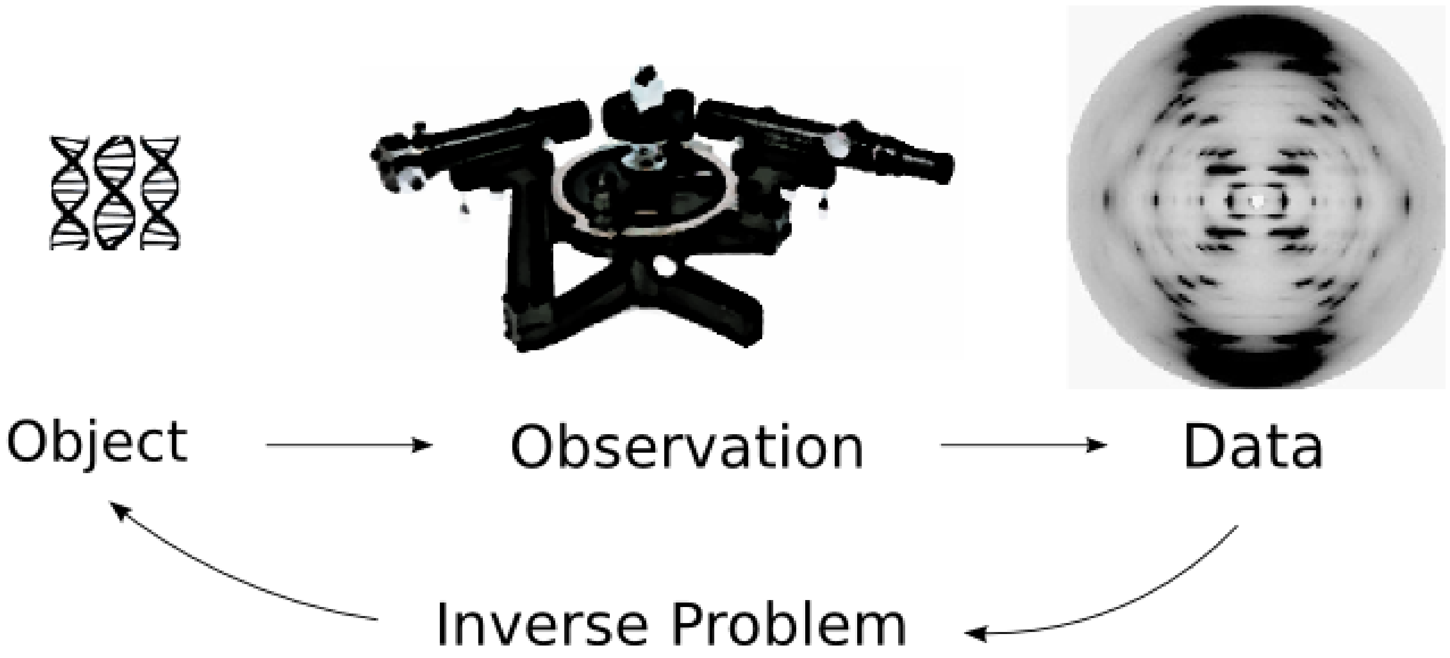
- XGBoost is an implementation of gradient boosting that won the Higg's Boson challenge and regularly wins Kaggle competitions
- XGBoost stands for eXtreme Gradient Boosting
- It is much faster than most gradient boosting algorithms and scales to billions of training data points
- It uses a cleverly chosen regularisation term to favour simple trees
- Finds a clever way to approximately minimise error plus regulariser very fast

Outline

1. What Makes a Good Learning Machine?
2. SVMs
3. Ensemble Methods
4. **Bayesian Inference**

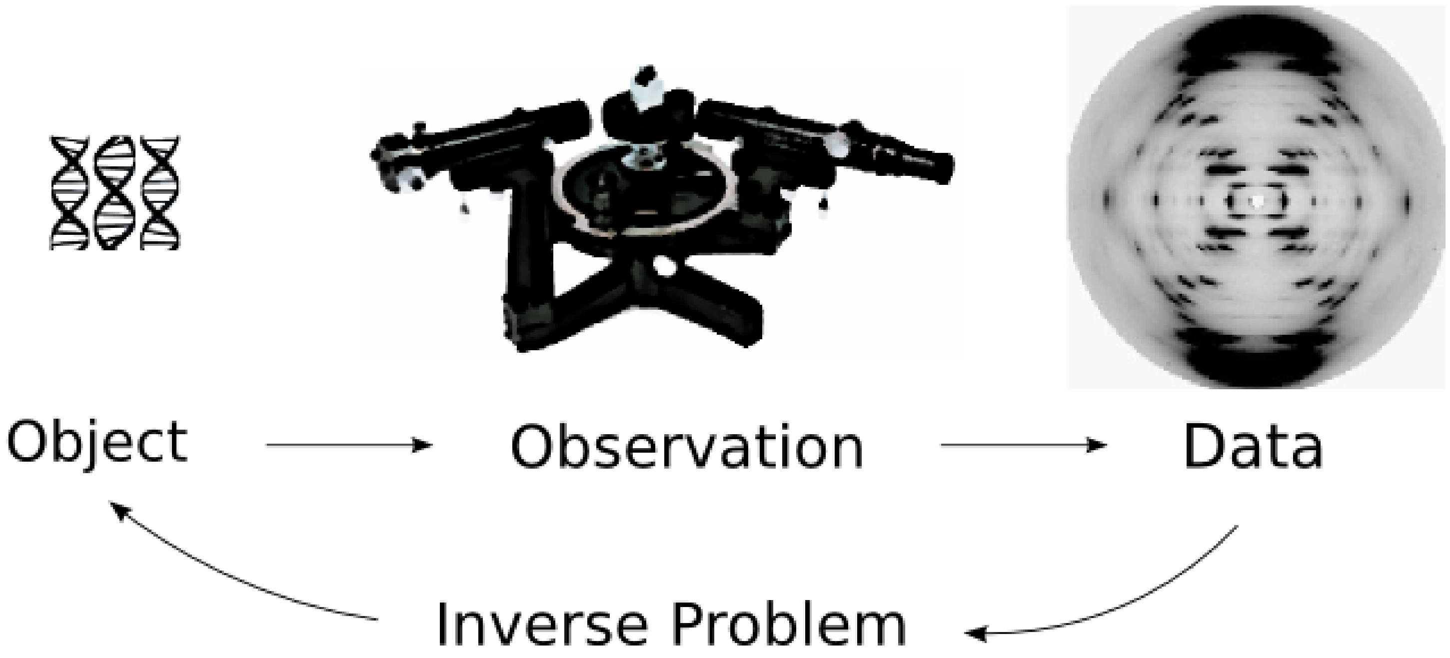


Inverse Problems



- Machine learning can often be seen as solving inverse problems
- We are given some data generated by the world and we want to infer something about the world

Inverse Problems



- Machine learning can often be seen as solving inverse problems
- We are given some data generated by the world and we want to infer something about the world

Bayes' Rule

- A trivial identity in probability known as Bayes' rules tells you how to solve inverse problems

$$\mathbb{P}\left(W|\mathcal{D}\right) = \frac{\mathbb{P}\left(\mathcal{D}|W\right)\,\mathbb{P}\left(W\right)}{\mathbb{P}\left(D\right)}$$

- What we want is to know the probability of the world, W, given the data, $\mathcal{D}$ we have observered—this is known as the **posteriori** probability

- This depends on the **likelihood** of the data given the world $\mathbb{P}\left(\mathcal{D}|W\right)$

- Multiplied by the **prior probability** of the world, $\mathbb{P}\left(W\right)$

Bayes' Rule

- A trivial identity in probability known as Bayes' rules tells you how to solve inverse problems

$$\mathbb{P}\left(W|\mathcal{D}\right) = \frac{\mathbb{P}\left(\mathcal{D}|W\right)\,\mathbb{P}\left(W\right)}{\mathbb{P}\left(D\right)}$$

- What we want is to know the probability of the world, W, given the data, $\mathcal{D}$ we have observered—this is known as the **posteriori** probability

- This depends on the **likelihood** of the data given the world $\mathbb{P}\left(\mathcal{D}|W\right)$

- Multiplied by the **prior probability** of the world, $\mathbb{P}\left(W\right)$

Bayes' Rule

- A trivial identity in probability known as Bayes' rules tells you how to solve inverse problems

$$\mathbb{P}(W|\mathcal{D}) = \frac{\mathbb{P}(\mathcal{D}|W)\ \mathbb{P}(W)}{\mathbb{P}(D)}$$

- What we want is to know the probability of the world, W, given the data, $\mathcal{D}$ we have observered—this is known as the **posteriori** probability

- This depends on the **likelihood** of the data given the world $\mathbb{P}(\mathcal{D}|W)$

- Multiplied by the **prior probability** of the world, $\mathbb{P}(W)$

Bayes' Rule

- A trivial identity in probability known as Bayes' rules tells you how to solve inverse problems

$$\mathbb{P}\left(W|\mathcal{D}\right) = \frac{\mathbb{P}\left(\mathcal{D}|W\right)\,\mathbb{P}\left(W\right)}{\mathbb{P}\left(D\right)}$$

- What we want is to know the probability of the world, W, given the data, $\mathcal{D}$ we have observered—this is known as the **posteriori** probability

- This depends on the **likelihood** of the data given the world $\mathbb{P}\left(\mathcal{D}|W\right)$

- Multiplied by the **prior probability** of the world, $\mathbb{P}\left(W\right)$

Bayes' Rule

- A trivial identity in probability known as Bayes' rules tells you how to solve inverse problems

$$\mathbb{P}\left(W|\mathcal{D}\right) = \frac{\mathbb{P}\left(\mathcal{D}|W\right)\,\mathbb{P}\left(W\right)}{\mathbb{P}\left(D\right)}$$

- What we want is to know the probability of the world, W, given the data, $\mathcal{D}$ we have observered—this is known as the **posteriori** probability

- This depends on the **likelihood** of the data given the world $\mathbb{P}\left(\mathcal{D}|W\right)$

- Multiplied by the **prior probability** of the world, $\mathbb{P}\left(W\right)$

Normalisation

- The denumerator is a normalisation constant

$$\mathbb{P}(D) = \sum_{W} \mathbb{P}(\mathcal{D}|W)\, \mathbb{P}(W)$$

- It is useful for comparing between different models of the world, W, and is sometimes called the **evidence**

- The model with the largest evidence is the most likely model to be correct

Normalisation

- The denumerator is a normalisation constant

$$\mathbb{P}(D) = \sum_{W} \mathbb{P}(\mathcal{D}|W)\, \mathbb{P}(W)$$

- It is useful for comparing between different models of the world, W, and is sometimes called the **evidence**

- The model with the largest evidence is the most likely model to be correct

Normalisation

- The denumerator is a normalisation constant

$$\mathbb{P}(D) = \sum_{W} \mathbb{P}(\mathcal{D}|W)\, \mathbb{P}(W)$$

- It is useful for comparing between different models of the world, W, and is sometimes called the **evidence**

- The model with the largest evidence is the most likely model to be correct

Normalisation

- The denumerator is a normalisation constant

$$\mathbb{P}(D) = \sum_{W} \mathbb{P}(\mathcal{D}|W)\,\mathbb{P}(W)$$

- It is useful for comparing between different models of the world, W, and is sometimes called the **evidence**

- The model with the largest evidence is the most likely model to be correct—used for model selection

Worlds

- W is some model of the world—technically they are parameters of the likelihood function
- If we trying to infer the length of rod then it would be a length
- If we are trying to infer the DNA sequence given a set of short reads then W would be a DNA sequence
- If we are trying to infer the crystal structure of molecules then W would be a structure

Worlds

- W is some model of the world—technically they are parameters of the likelihood function

- If we trying to infer the length of rod then it would be a length

- If we are trying to infer the DNA sequence given a set of short reads then W would be a DNA sequence

- If we are trying to infer the crystal structure of molecules then W would be a structure

Worlds

- W is some model of the world—technically they are parameters of the likelihood function
- If we trying to infer the length of rod then it would be a length
- If we are trying to infer the DNA sequence given a set of short reads then W would be a DNA sequence
- If we are trying to infer the crystal structure of molecules then W would be a structure

Worlds

- W is some model of the world—technically they are parameters of the likelihood function
- If we trying to infer the length of rod then it would be a length
- If we are trying to infer the DNA sequence given a set of short reads then W would be a DNA sequence
- If we are trying to infer the crystal structure of molecules then W would be a structure

Worlds

- W is some model of the world—technically they are parameters of the likelihood function

- If we trying to infer the length of rod then it would be a length

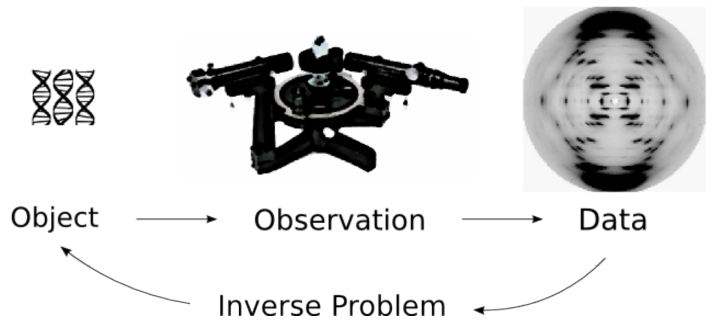
- If we are trying to infer the DNA sequence given a set of short reads then W would be a DNA sequence

- If we are trying to infer the crystal structure of molecules then W would be a structure

Advantages of Bayesian Inference

- Its optimal (follows from a simple probabilistic identity)

- It is relatively easy to model the likelihood (this is a forward process not an inverse process)



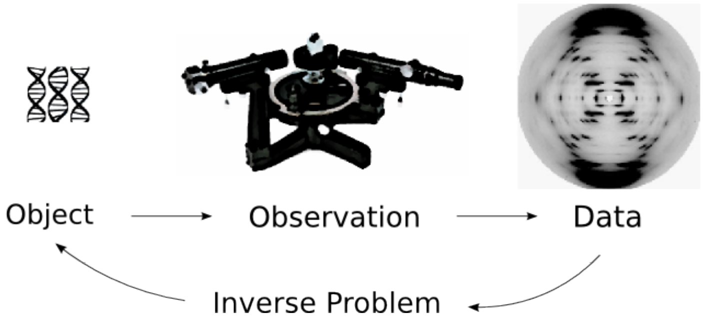
- No problem with missing data—we just calculate the likelihood of the data we see

- Won't over-fit

- Provides a full probabilistic description of the inference

Advantages of Bayesian Inference

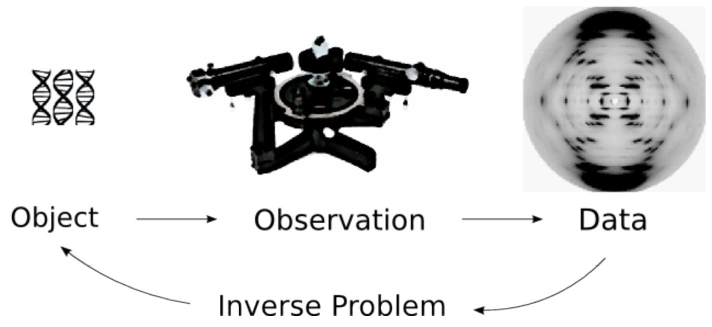
- Its optimal (follows from a simple probabilistic identity)
- It is relatively easy to model the likelihood (this is a forward process not an inverse process)



- No problem with missing data—we just calculate the likelihood of the data we see
- Won't over-fit
- Provides a full probabilistic description of the inference

Advantages of Bayesian Inference

- Its optimal (follows from a simple probabilistic identity)
- It is relatively easy to model the likelihood (this is a forward process not an inverse process)

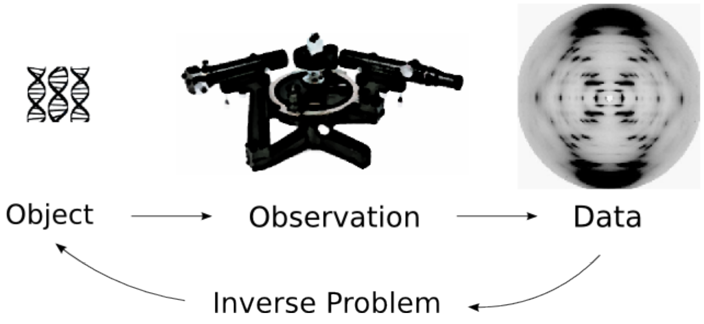


- No problem with missing data—we just calculate the likelihood of the data we see
- Won't over-fit
- Provides a full probabilistic description of the inference

Advantages of Bayesian Inference

- Its optimal (follows from a simple probabilistic identity)

- It is relatively easy to model the likelihood (this is a forward process not an inverse process)



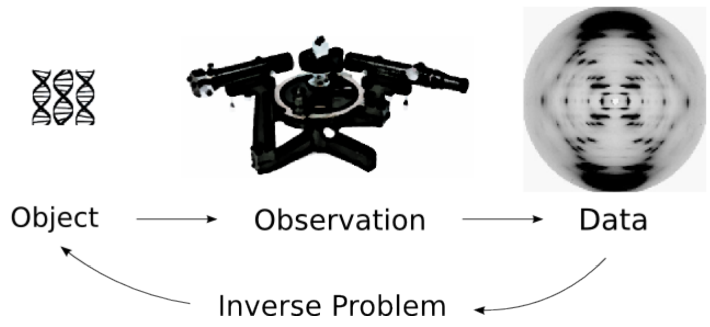
- No problem with missing data—we just calculate the likelihood of the data we see

- Won't over-fit

- Provides a full probabilistic description of the inference

Advantages of Bayesian Inference

- Its optimal (follows from a simple probabilistic identity)
- It is relatively easy to model the likelihood (this is a forward process not an inverse process)



- No problem with missing data—we just calculate the likelihood of the data we see
- Won't over-fit
- Provides a full probabilistic description of the inference

Disadvantages of Bayesian Inference

- It is a bespoke method, you need to model the likelihood and put a prior probability on the world
- Need to be able to sum over all possible worlds
- This is technically challenging and/or computationally slow
- Markov Chain Monte Carlo (MCMC) is a very established framework for suming over worlds
  - ⋆ There exist mature libraries
  - ⋆ Large number of tricks to speed things up
- Various approximations also exist for averaging over worlds

Disadvantages of Bayesian Inference

- It is a bespoke method, you need to model the likelihood and put a prior probability on the world—not out of the box

- Need to be able to sum over all possible worlds

- This is technically challenging and/or computationally slow

- Markov Chain Monte Carlo (MCMC) is a very established framework for suming over worlds
  - ⋆ There exist mature libraries
  - ⋆ Large number of tricks to speed things up

- Various approximations also exist for averaging over worlds

Disadvantages of Bayesian Inference

- It is a bespoke method, you need to model the likelihood and put a prior probability on the world—not out of the box
- Need to be able to sum over all possible worlds
- This is technically challenging and/or computationally slow
- Markov Chain Monte Carlo (MCMC) is a very established framework for suming over worlds
  - ⋆ There exist mature libraries
  - ⋆ Large number of tricks to speed things up
- Various approximations also exist for averaging over worlds

Disadvantages of Bayesian Inference

- It is a bespoke method, you need to model the likelihood and put a prior probability on the world—not out of the box
- Need to be able to sum over all possible worlds
- This is technically challenging and/or computationally slow
- Markov Chain Monte Carlo (MCMC) is a very established framework for suming over worlds
  - ⋆ There exist mature libraries
  - ⋆ Large number of tricks to speed things up
- Various approximations also exist for averaging over worlds

Disadvantages of Bayesian Inference

- It is a bespoke method, you need to model the likelihood and put a prior probability on the world—not out of the box
- Need to be able to sum over all possible worlds
- This is technically challenging and/or computationally slow
- Markov Chain Monte Carlo (MCMC) is a very established framework for suming over worlds
  - ⋆ There exist mature libraries
  - ⋆ Large number of tricks to speed things up
- Various approximations also exist for averaging over worlds

Disadvantages of Bayesian Inference

- It is a bespoke method, you need to model the likelihood and put a prior probability on the world—not out of the box
- Need to be able to sum over all possible worlds
- This is technically challenging and/or computationally slow
- Markov Chain Monte Carlo (MCMC) is a very established framework for suming over worlds
  - ⋆ There exist mature libraries
  - ⋆ Large number of tricks to speed things up
- Various approximations also exist for averaging over worlds

Simple Bayes

- There are a few cases where Bayesian Inference is relatively simple
- When we are inferring quantities with a very simple likelihood, sometimes Bayesian approach becomes trivial
- There are a few cases where we can use the same model for different situations and the model is easy to compute
  - ⋆ Gaussian processes for regression
  - ⋆ Naive Bayes (e.g. for text)

Simple Bayes

- There are a few cases where Bayesian Inference is relatively simple
- When we are inferring quantities with a very simple likelihood, sometimes Bayesian approach becomes trivial
- There are a few cases where we can use the same model for different situations and the model is easy to compute
  - ⋆ Gaussian processes for regression
  - ⋆ Naive Bayes (e.g. for text)

Simple Bayes

- There are a few cases where Bayesian Inference is relatively simple
- When we are inferring quantities with a very simple likelihood, sometimes Bayesian approach becomes trivial
- There are a few cases where we can use the same model for different situations and the model is easy to compute
  - ⋆ Gaussian processes for regression
  - ⋆ Naive Bayes (e.g. for text)

Simple Bayes

- There are a few cases where Bayesian Inference is relatively simple
- When we are inferring quantities with a very simple likelihood, sometimes Bayesian approach becomes trivial
- There are a few cases where we can use the same model for different situations and the model is easy to compute
  - ⋆ Gaussian processes for regression
  - ⋆ Naive Bayes (e.g. for text)

Simple Bayes

- There are a few cases where Bayesian Inference is relatively simple
- When we are inferring quantities with a very simple likelihood, sometimes Bayesian approach becomes trivial
- There are a few cases where we can use the same model for different situations and the model is easy to compute
  - ⋆ Gaussian processes for regression
  - ⋆ Naive Bayes (e.g. for text)

Gaussian Processes

- Gaussian Processes is an off the shelf method for regression (multi-dimensional curve fitting)

- It makes a Gaussian (normal distribution) assumption
  - ⋆ The probability of a point taking a value is normally distributed with a mean and variance that depends on all the observed points
  - ⋆ The dependence is controlled by a covariance function which you are allowed to choose

- Because everything is Gaussian, all sums over worlds can be done

- Mathematics is all doable

Gaussian Processes

- Gaussian Processes is an off the shelf method for regression (multi-dimensional curve fitting)
- It makes a Gaussian (normal distribution) assumption
  - ⋆ The probability of a point taking a value is normally distributed with a mean and variance that depends on all the observed points
  - ⋆ The dependence is controlled by a covariance function which you are allowed to choose
- Because everything is Gaussian, all sums over worlds can be done
- Mathematics is all doable

Gaussian Processes

- Gaussian Processes is an off the shelf method for regression (multi-dimensional curve fitting)

- It makes a Gaussian (normal distribution) assumption
  - ⋆ The probability of a point taking a value is normally distributed with a mean and variance that depends on all the observed points
  - ⋆ The dependence is controlled by a covariance function which you are allowed to choose

- Because everything is Gaussian, all sums over worlds can be done

- Mathematics is all doable

Gaussian Processes

- Gaussian Processes is an off the shelf method for regression (multi-dimensional curve fitting)

- It makes a Gaussian (normal distribution) assumption
  - ⋆ The probability of a point taking a value is normally distributed with a mean and variance that depends on all the observed points
  - ⋆ The dependence is controlled by a covariance function which you are allowed to choose

- Because everything is Gaussian, all sums over worlds can be done

- Mathematics is all doable

Gaussian Processes

- Gaussian Processes is an off the shelf method for regression (multi-dimensional curve fitting)

- It makes a Gaussian (normal distribution) assumption
  - ⋆ The probability of a point taking a value is normally distributed with a mean and variance that depends on all the observed points
  - ⋆ The dependence is controlled by a covariance function which you are allowed to choose

- Because everything is Gaussian, all sums over worlds can be done

- Mathematics is all doable

Gaussian Processes

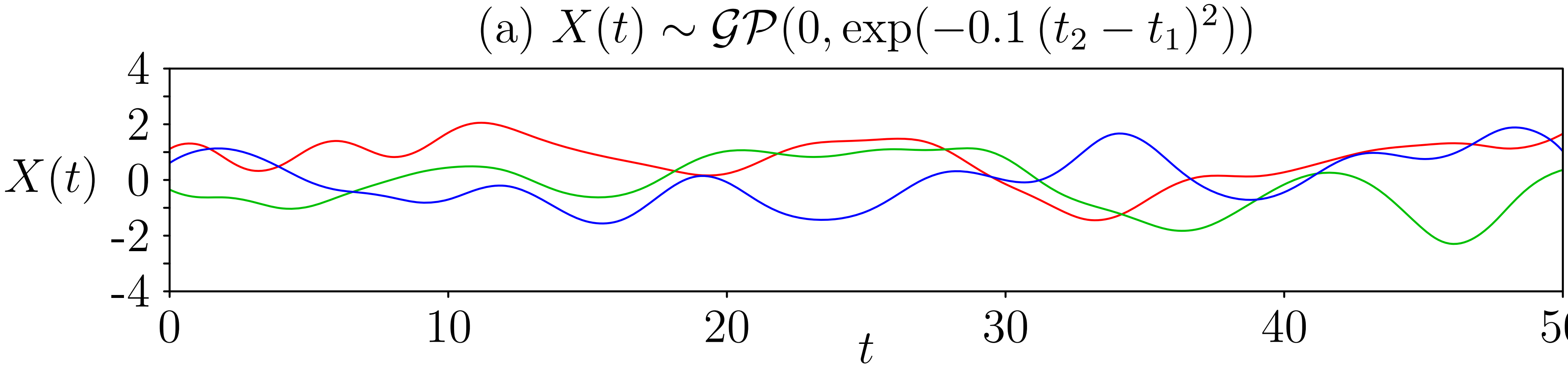
- Gaussian Processes is an off the shelf method for regression (multi-dimensional curve fitting)

- It makes a Gaussian (normal distribution) assumption
  - ⋆ The probability of a point taking a value is normally distributed with a mean and variance that depends on all the observed points
  - ⋆ The dependence is controlled by a covariance function which you are allowed to choose

- Because everything is Gaussian, all sums over worlds can be done

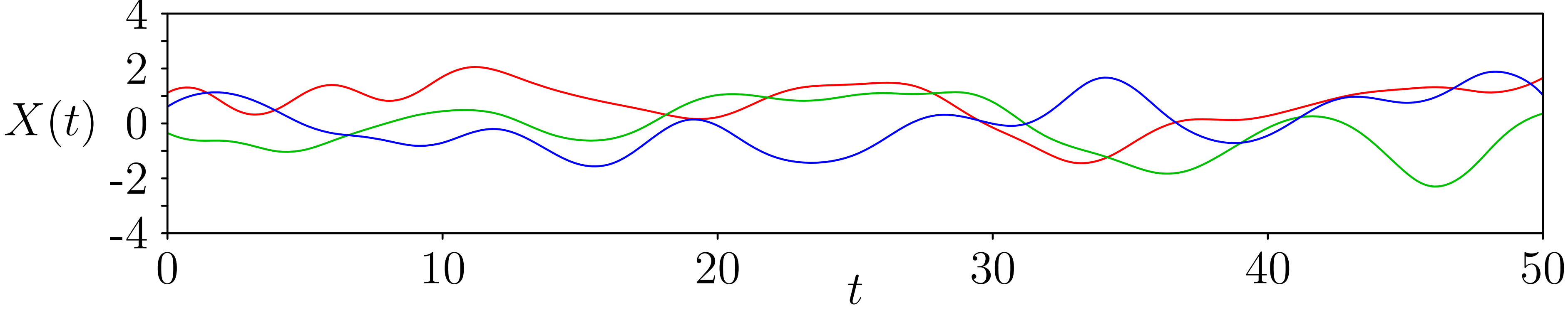
- Mathematics is all doable, but a real pain!

Gaussian Process Worlds

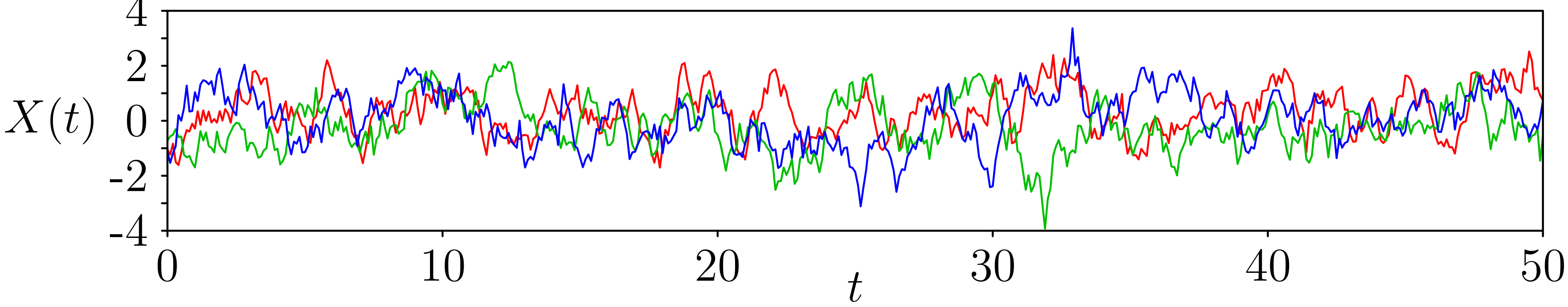


Gaussian Process Worlds

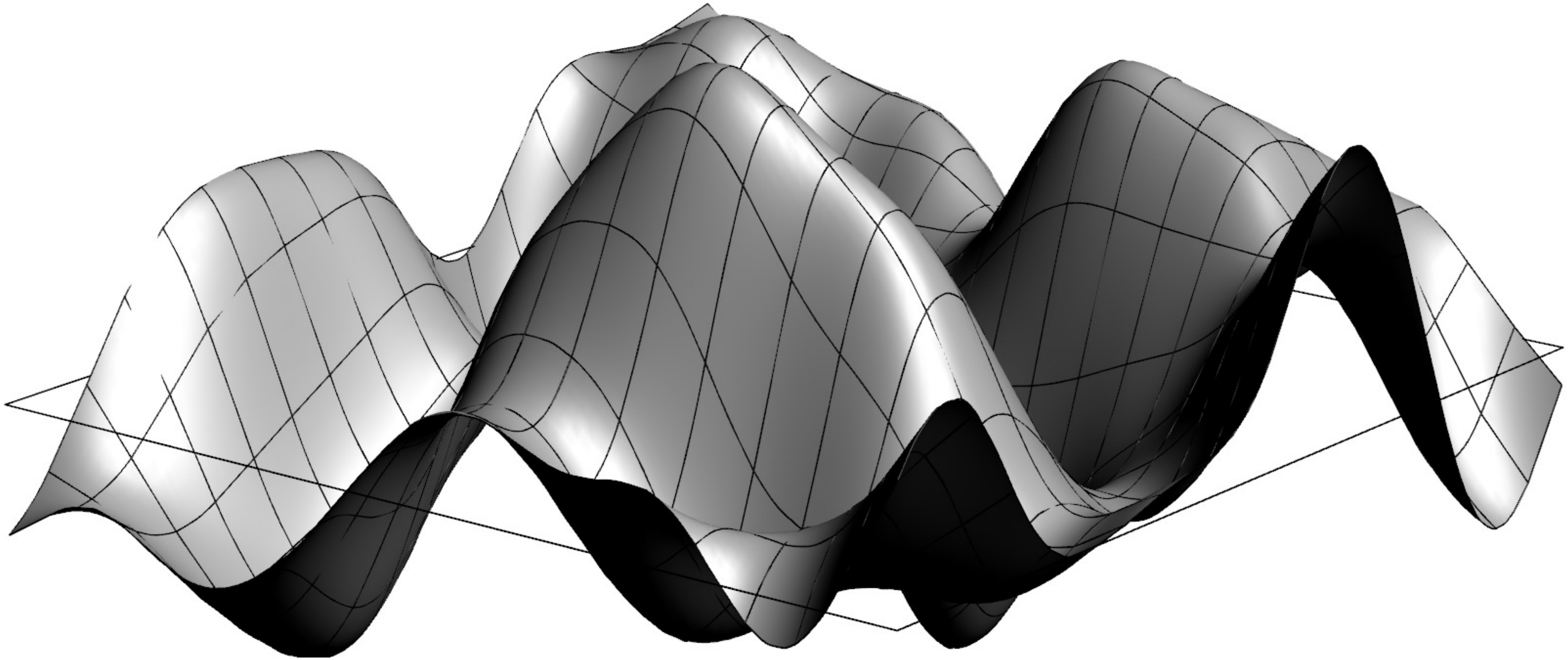
(a) $X(t) \sim \mathcal{GP}(0, \exp(-0.1\,(t_2 - t_1)^2))$



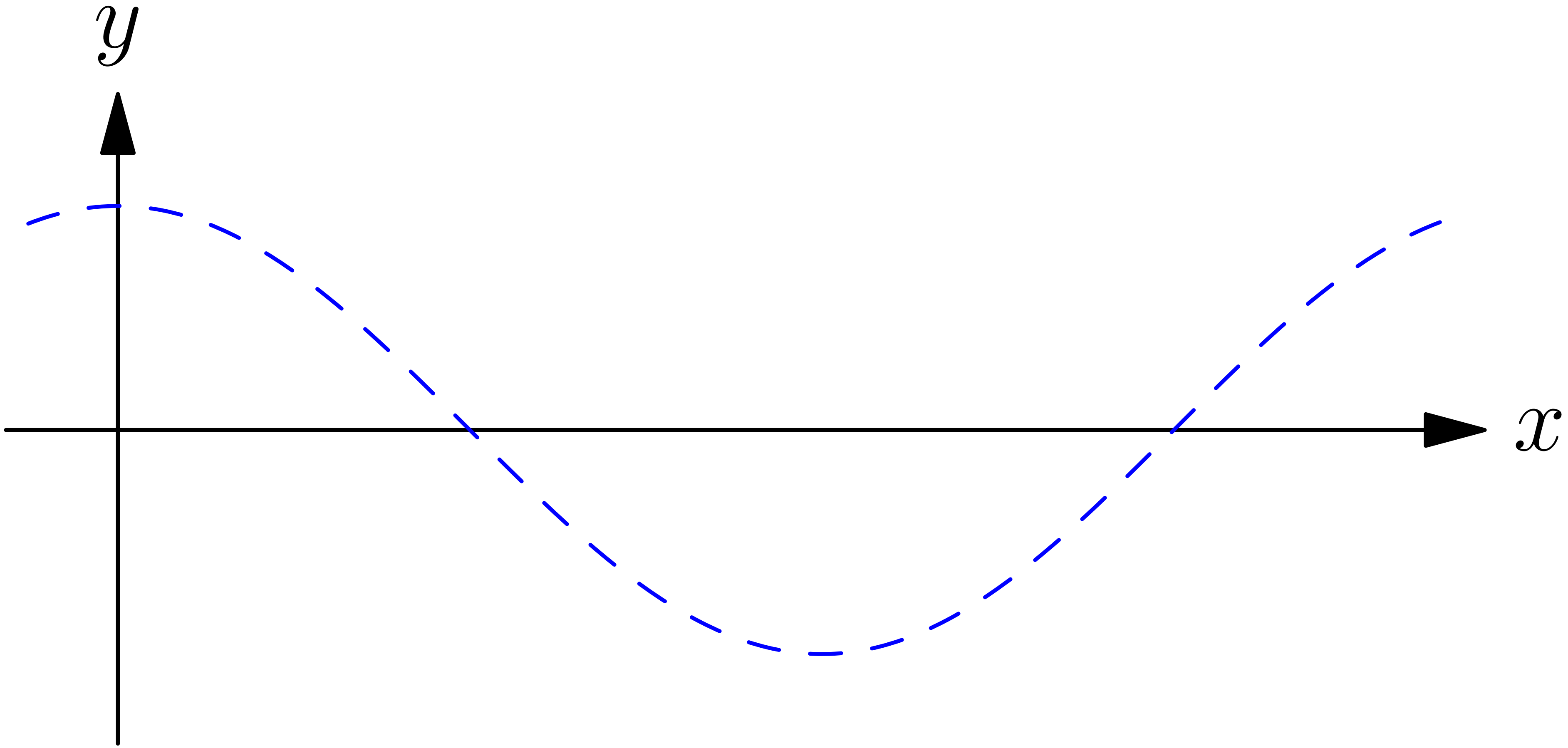
(b) $X(t) \sim \mathcal{GP}(0, \exp(-|t_2 - t_1||))$



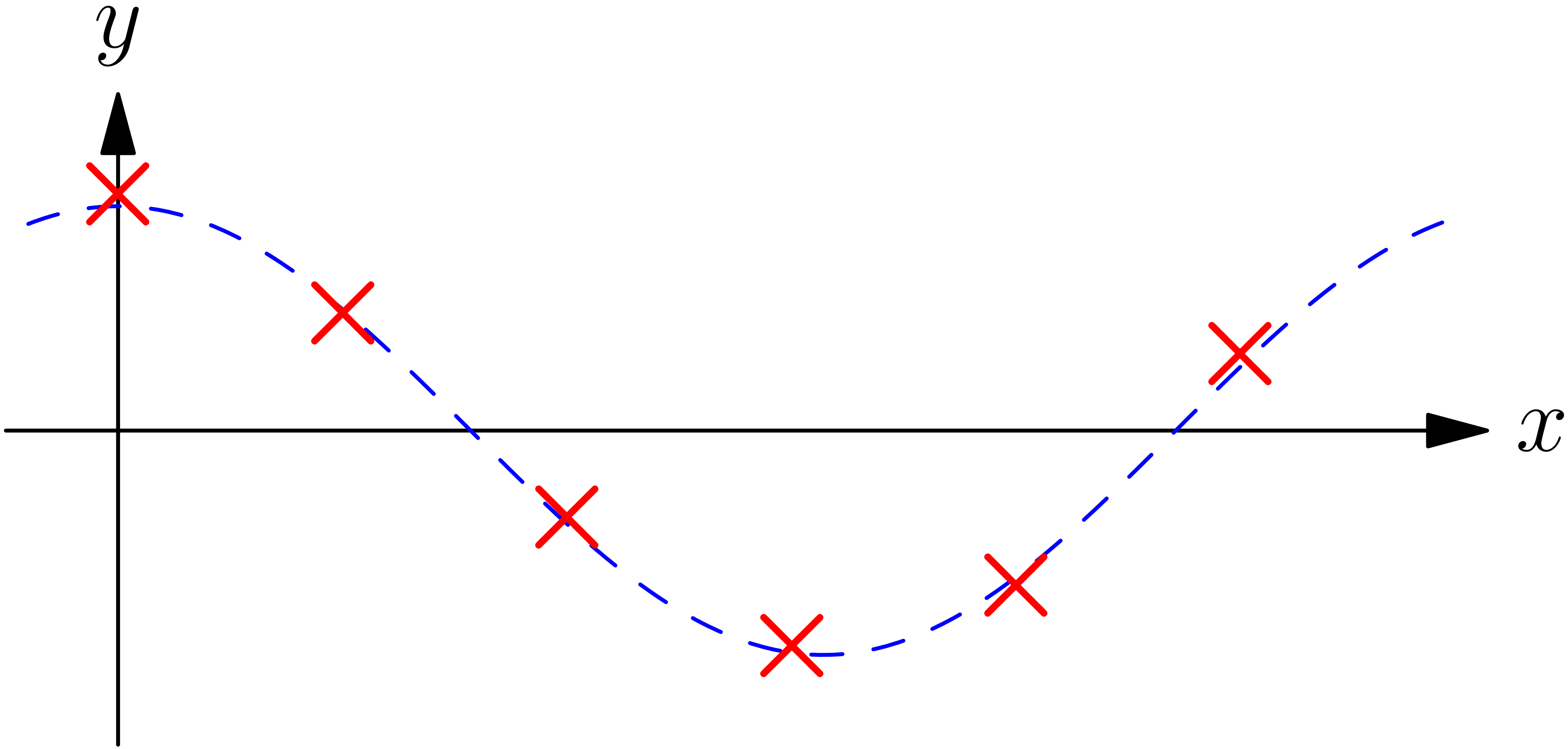
2-D Gaussian Processes



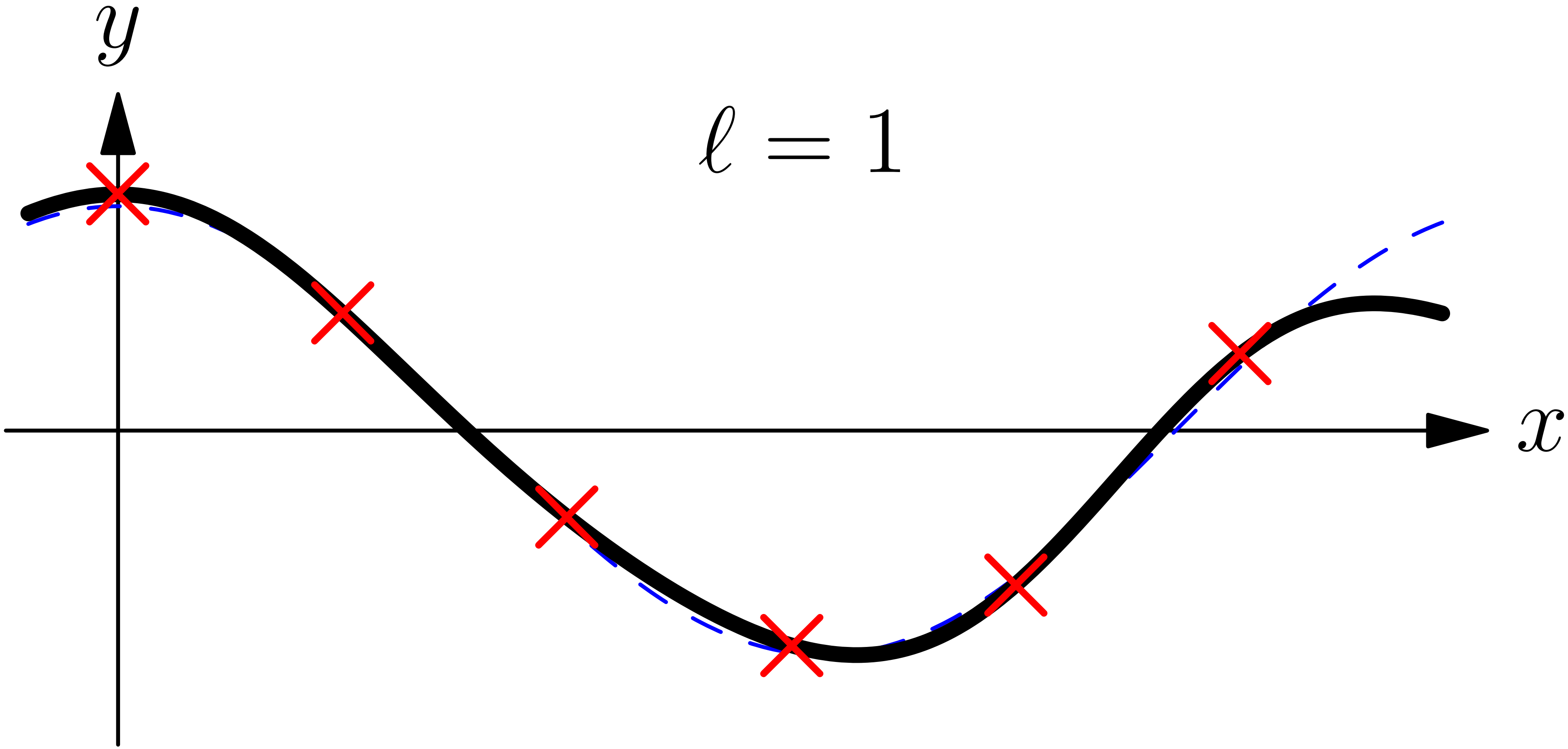
$$K(x, x') = \exp(-(x - x')^2/(2\,\ell^2))$$



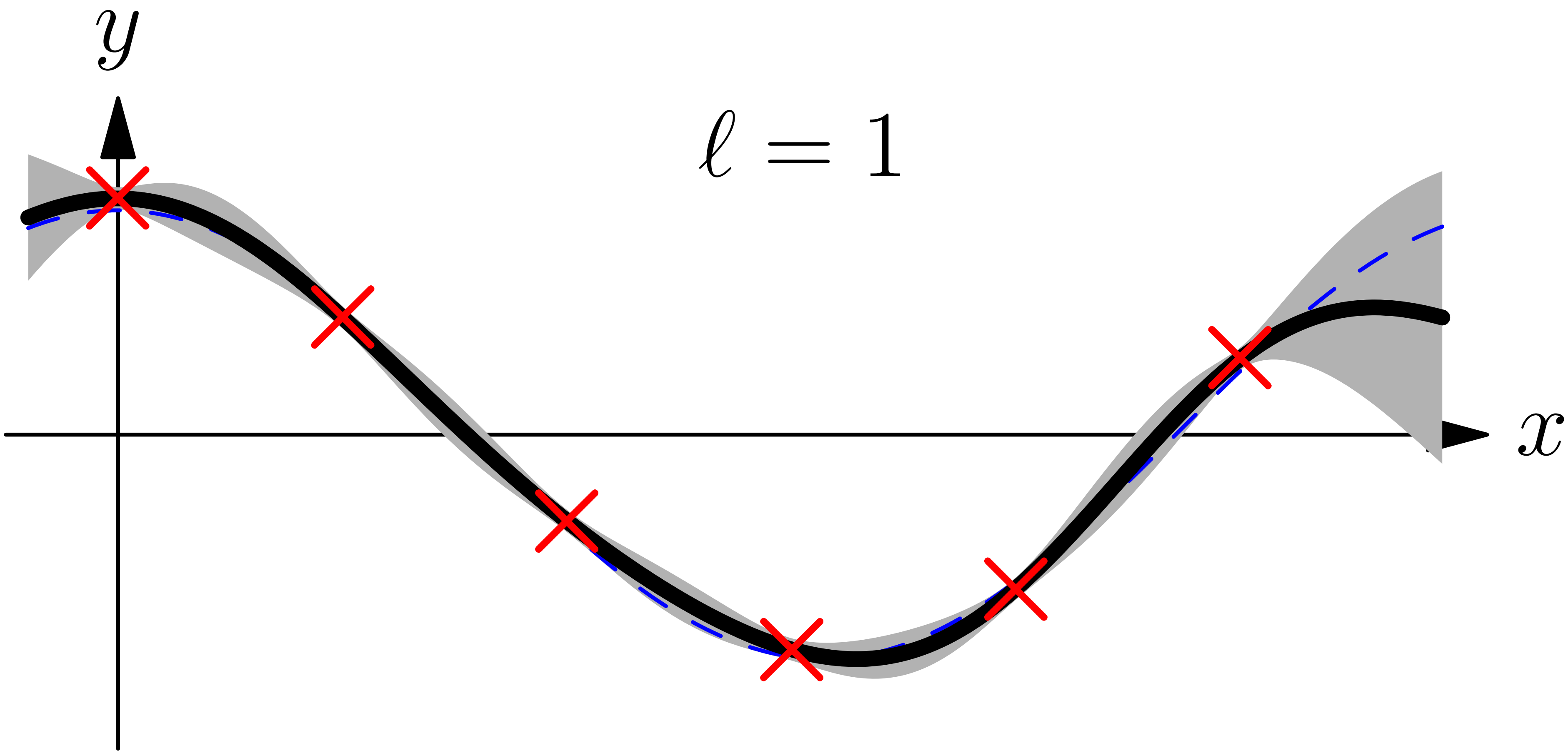
$$K(x, x') = \exp(-(x - x')^2/(2\,\ell^2))$$



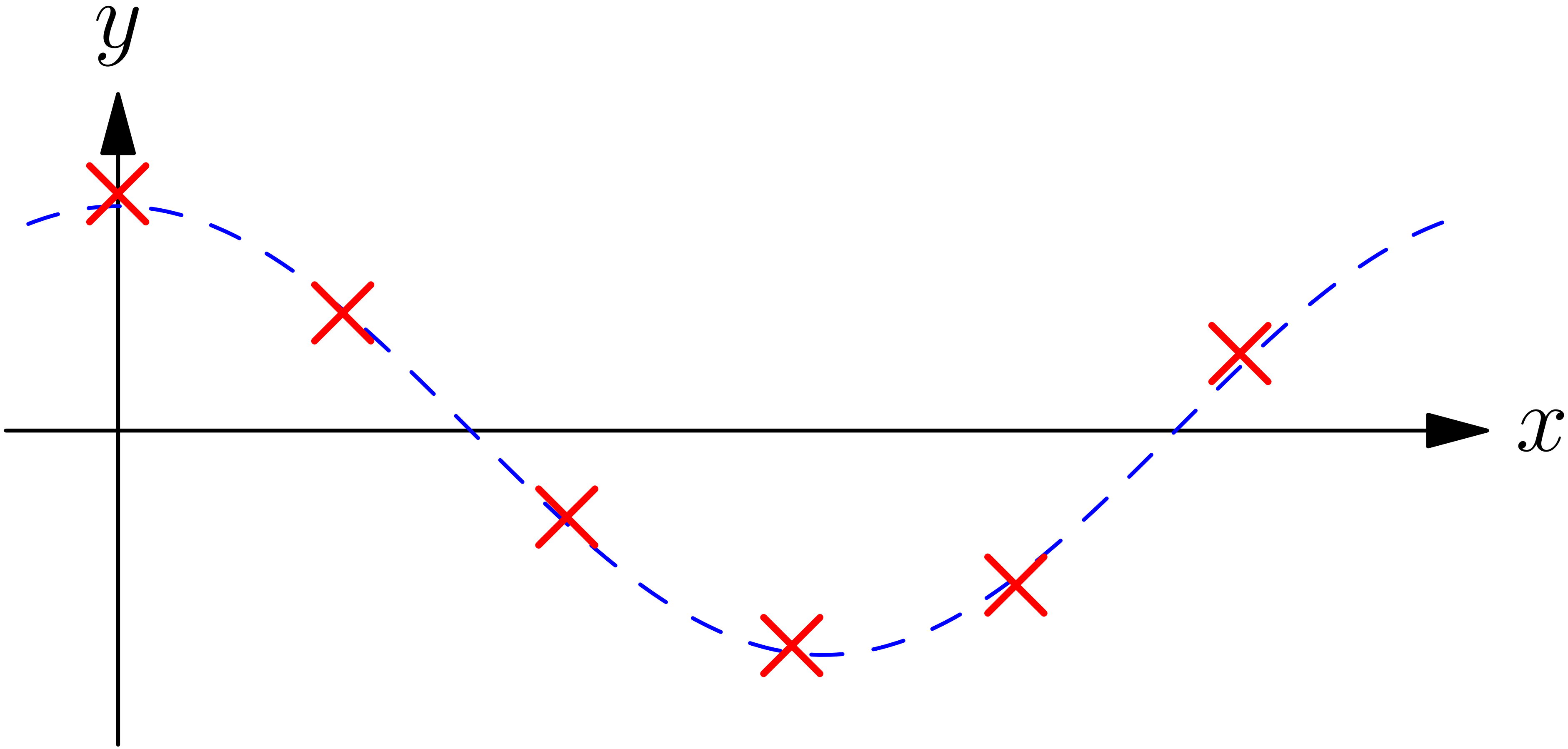
$$K(x, x') = \exp(-(x - x')^2/(2\,\ell^2))$$



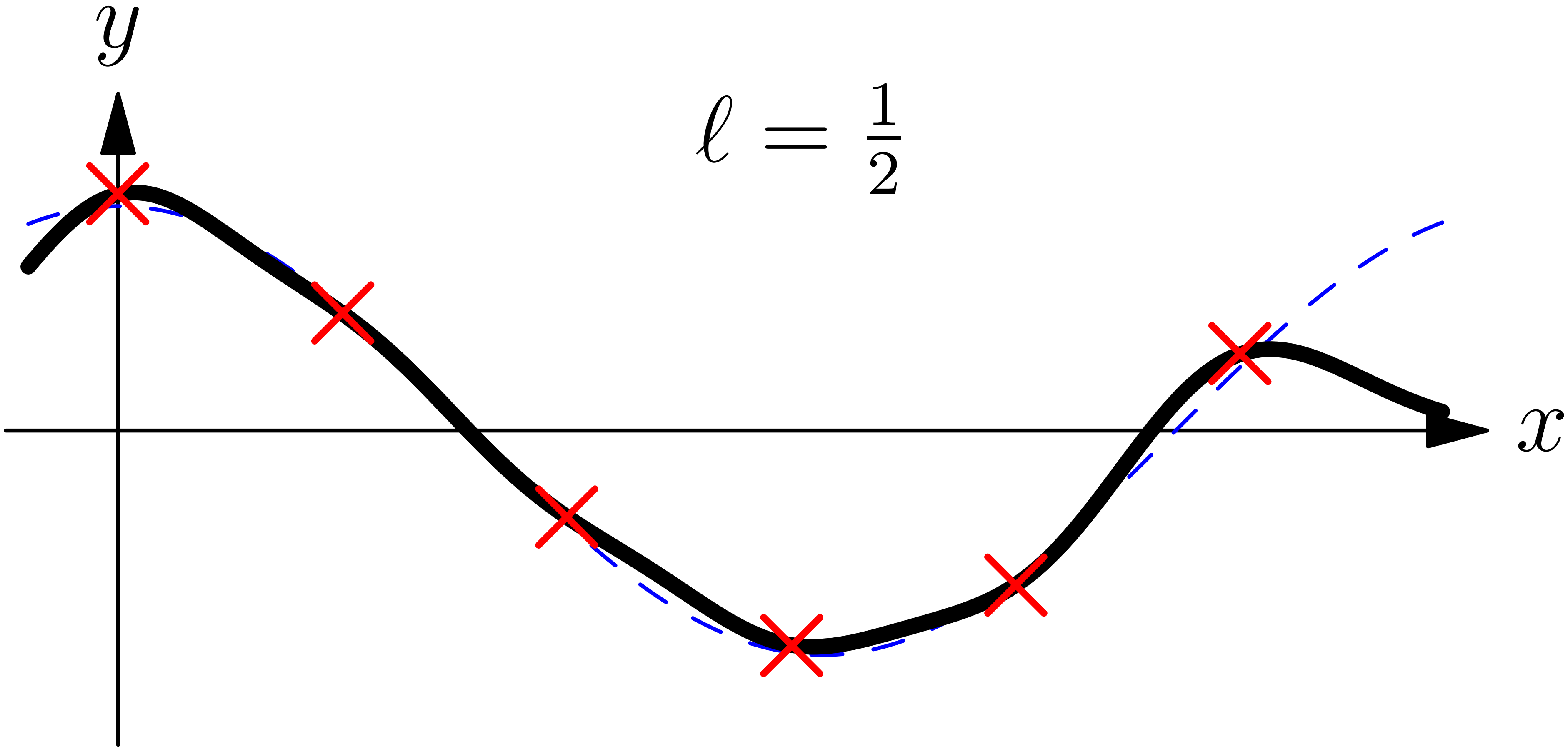
$$K(x, x') = \exp(-(x - x')^2/(2\,\ell^2))$$



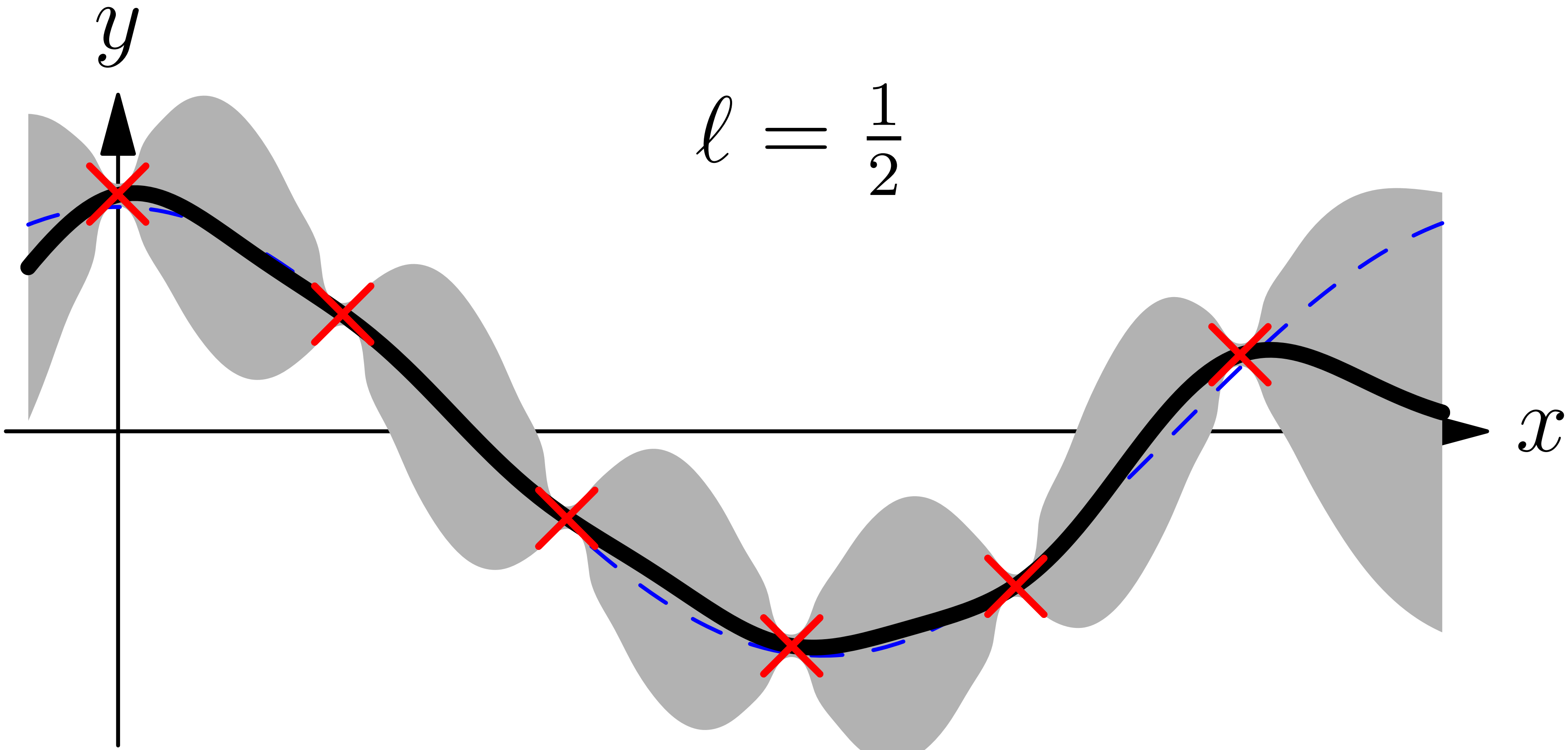
$$K(x, x') = \exp(-(x - x')^2/(2\,\ell^2))$$



$$K(x, x') = \exp(-(x - x')^2/(2\,\ell^2))$$



$$K(x, x') = \exp(-(x - x')^2/(2\,\ell^2))$$



Using Gaussian Processes

- We are usually not doing 1-d curve fitting but are doing high-dimensional regression (trying to infer a number given a set of (numerical) features
- You need to give it the right covariance functional, although (hyper-)parameters can be chosen using automatic model selection
- Gaussian processes are often the best method for regression
- Optimal if and only if Gaussian assumption is correct

Using Gaussian Processes

- We are usually not doing 1-d curve fitting but are doing high-dimensional regression (trying to infer a number given a set of (numerical) features
- You need to give it the right covariance functional, although (hyper-)parameters can be chosen using automatic model selection
- Gaussian processes are often the best method for regression
- Optimal if and only if Gaussian assumption is correct

Using Gaussian Processes

- We are usually not doing 1-d curve fitting but are doing high-dimensional regression (trying to infer a number given a set of (numerical) features

- You need to give it the right covariance functional, although (hyper-)parameters can be chosen using automatic model selection

- Gaussian processes are often the best method for regression

- Optimal if and only if Gaussian assumption is correct

Using Gaussian Processes

- We are usually not doing 1-d curve fitting but are doing high-dimensional regression (trying to infer a number given a set of (numerical) features
- You need to give it the right covariance functional, although (hyper-)parameters can be chosen using automatic model selection
- Gaussian processes are often the best method for regression
- Optimal if and only if Gaussian assumption is correct

Naive Bayes

- We finish with *Naive Bayes* which is unreasonably successful at many text problems
- It is the basis of one of the first big successes of machine learning, namely spam filtering
- It makes the crude approximation that the probability of a document depends only on the words, with all words being independent

$$\mathbb{P}\left(\text{Doc}|\text{Something}\right) = \prod_{\text{word}\in\text{Doc}} \mathbb{P}\left(\text{word}|\text{Something}\right)$$

Naive Bayes

- We finish with *Naive Bayes* which is unreasonably successful at many text problems
- It is the basis of one of the first big successes of machine learning, namely spam filtering
- It makes the crude approximation that the probability of a document depends only on the words, with all words being independent

$$\mathbb{P}(\text{Doc}|\text{Something}) = \prod_{\text{word} \in \text{Doc}} \mathbb{P}(\text{word}|\text{Something})$$

Naive Bayes

- We finish with *Naive Bayes* which is unreasonably successful at many text problems
- It is the basis of one of the first big successes of machine learning, namely spam filtering
- It makes the crude approximation that the probability of a document depends only on the words, with all words being independent

$$\mathbb{P}\left(\text{Doc}|\text{Something}\right) = \prod_{\text{word}\in\text{Doc}} \mathbb{P}\left(\text{word}|\text{Something}\right)$$

Naive Bayes

- We finish with *Naive Bayes* which is unreasonably successful at many text problems
- It is the basis of one of the first big successes of machine learning, namely spam filtering
- It makes the crude approximation that the probability of a document depends only on the words, with all words being independent

$$\mathbb{P}\left(\text{Doc}|\text{Something}\right) = \prod_{\text{word}\in\text{Doc}} \mathbb{P}\left(\text{word}|\text{Something}\right)$$

Estimating $\mathbb{P}(\textbf{word}|\textbf{Something})$

- We can estimate $\mathbb{P}(\text{word}|\text{Something})$ from the empirical frequency of occurrence of the word
- Also need to estimate $\mathbb{P}(\text{word}|\neg\text{Something})$
- Often add a pseudo-count (pretend we have seen all words one more time than we have)
- This acts as a regulariser (makes the learning machine less susceptible to the training data)

Estimating $\mathbb{P}(\text{word}|\text{Something})$

- We can estimate $\mathbb{P}(\text{word}|\text{Something})$ from the empirical frequency of occurrence of the word
- Also need to estimate $\mathbb{P}(\text{word}|\neg\text{Something})$
- Often add a pseudo-count (pretend we have seen all words one more time than we have)
- This acts as a regulariser (makes the learning machine less susceptible to the training data)

Estimating $\mathbb{P}(\textbf{word}|\textbf{Something})$

- We can estimate $\mathbb{P}(\text{word}|\text{Something})$ from the empirical frequency of occurrence of the word
- Also need to estimate $\mathbb{P}(\text{word}|\neg\text{Something})$
- Often add a pseudo-count (pretend we have seen all words one more time than we have)
- This acts as a regulariser (makes the learning machine less susceptible to the training data)

Estimating $\mathbb{P}(\text{word}|\text{Something})$

- We can estimate $\mathbb{P}(\text{word}|\text{Something})$ from the empirical frequency of occurrence of the word
- Also need to estimate $\mathbb{P}(\text{word}|\neg\text{Something})$
- Often add a pseudo-count (pretend we have seen all words one more time than we have)
- This acts as a regulariser (makes the learning machine less susceptible to the training data)

Unreasonable Success

- Naive Bayes is relatively simple to implement
- It uses the crude approximation that the order of words is irrelevant!
- Also assume that words of independent $\{$not, realistic$\}$
- However, often gives remarkably good performance

Unreasonable Success

- Naive Bayes is relatively simple to implement
- It uses the crude approximation that the order of words is irrelevant!
- Also assume that words of independent $\{$not, realistic$\}$
- However, often gives remarkably good performance

Unreasonable Success

- Naive Bayes is relatively simple to implement
- It uses the crude approximation that the order of words is irrelevant!—same assumption as bag-of-words
- Also assume that words of independent $\{$not, realistic$\}$
- However, often gives remarkably good performance

Unreasonable Success

- Naive Bayes is relatively simple to implement
- It uses the crude approximation that the order of words is irrelevant!—same assumption as bag-of-words
- Also assume that words of independent {not, realistic}
- However, often gives remarkably good performance

Unreasonable Success

- Naive Bayes is relatively simple to implement
- It uses the crude approximation that the order of words is irrelevant!—same assumption as bag-of-words
- Also assume that words of independent {not, realistic}
- However, often gives remarkably good performance

Recap

- SVMs are often best when you have smallish data sets and can encode all the features numerically
- Random Forest and Gradient Boosted Trees are used for large, messy (tabular) data sets
- Bayesian inference is used when you can build a strong model of the likelihood of the data, but it takes work
- Gaussian processes often gives excellent results for regression of numeric inputs
- Deep learning (which Jon covers later) is often the model of choice for images and signals (including text)

Recap

- SVMs are often best when you have smallish data sets and can encode all the features numerically
- Random Forest and Gradient Boosted Trees are used for large, messy (tabular) data sets
- Bayesian inference is used when you can build a strong model of the likelihood of the data, but it takes work
- Gaussian processes often gives excellent results for regression of numeric inputs
- Deep learning (which Jon covers later) is often the model of choice for images and signals (including text)

Recap

- SVMs are often best when you have smallish data sets and can encode all the features numerically
- Random Forest and Gradient Boosted Trees are used for large, messy (tabular) data sets
- Bayesian inference is used when you can build a strong model of the likelihood of the data, but it takes work
- Gaussian processes often gives excellent results for regression of numeric inputs
- Deep learning (which Jon covers later) is often the model of choice for images and signals (including text)

Recap

- SVMs are often best when you have smallish data sets and can encode all the features numerically
- Random Forest and Gradient Boosted Trees are used for large, messy (tabular) data sets
- Bayesian inference is used when you can build a strong model of the likelihood of the data, but it takes work
- Gaussian processes often gives excellent results for regression of numeric inputs
- Deep learning (which Jon covers later) is often the model of choice for images and signals (including text)

Recap

- SVMs are often best when you have smallish data sets and can encode all the features numerically
- Random Forest and Gradient Boosted Trees are used for large, messy (tabular) data sets
- Bayesian inference is used when you can build a strong model of the likelihood of the data, but it takes work
- Gaussian processes often gives excellent results for regression of numeric inputs
- Deep learning (which Jon covers later) is often the model of choice for images and signals (including text)

Conclusion

- We started by discussing generalisation performance and the bias-variance dilemma
- We've looked at many of the machine learning methods that are currently state-of-the-art
- They nearly all try to address the bias-variance dilemma one way or another
- When to use them depends on what data you have

Conclusion

- We started by discussing generalisation performance and the bias-variance dilemma
- We've looked at many of the machine learning methods that are currently state-of-the-art
- They nearly all try to address the bias-variance dilemma one way or another
- When to use them depends on what data you have

Conclusion

- We started by discussing generalisation performance and the bias-variance dilemma
- We've looked at many of the machine learning methods that are currently state-of-the-art
- They nearly all try to address the bias-variance dilemma one way or another
- When to use them depends on what data you have

Conclusion

- We started by discussing generalisation performance and the bias-variance dilemma
- We've looked at many of the machine learning methods that are currently state-of-the-art
- They nearly all try to address the bias-variance dilemma one way or another
- When to use them depends on what data you have