Overfitting & Generalisation

Mark van der Wilk

Department of Computing
Imperial College London

@markvanderwilk
m.vdwilk@imperial.ac.uk

November 4, 2022

Announcements

- Formula sheet will be given to you during exam, and will be available beforehand. See `exercises.pdf`.
- New exercises: Keep up-to-date, and discuss with TAs. Not enough of you are taking advantage.
- New coursework next Tuesday.

Machine Learning is just

statistics done badly!

Machine Learning is just

statistics done badly!

- Statisticans work really hard to provide methods with **guarantees**

Machine Learning is just
statistics done badly!

- Statisticans work really hard to provide methods with **guarantees**
- ML applies methods in settings (e.g. high dimensions) where the guarantees don't hold

Machine Learning is just **statistics done badly!**

- Statisticans work really hard to provide methods with **guarantees**
- ML applies methods in settings (e.g. high dimensions) where the guarantees don't hold

But we have one saviour:

Machine Learning is just
statistics done badly!

- Statisticans work really hard to provide methods with **guarantees**
- ML applies methods in settings (e.g. high dimensions) where the guarantees don't hold

But we have one saviour:
The train/test split!

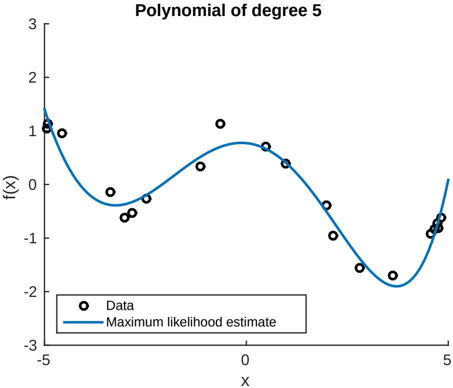
Machine Learning is just
statistics done badly!

- Statisticans work really hard to provide methods with **guarantees**
- ML applies methods in settings (e.g. high dimensions) where the guarantees don't hold

But we have one saviour:
The train/test split!

- You may already know the techniques we discuss today.
- But today, we let the mathematics **prove** why they work.

Curve fitting



- What we want: Find a curve that predicts well **even for unseen inputs**
- What we do: Minimise loss on **training points**:

$$L(\boldsymbol{\theta}) = \sum_n (f(\mathbf{x}_n; \boldsymbol{\theta}) - y_n)^2 \tag{1}$$

Curve fitting

We will investigate the consequences of

- choosing particular basis functions,
- fitting to training points (**overfitting**).

Curve fitting

We will investigate the consequences of

- choosing particular basis functions,
- fitting to training points (**overfitting**).

The two are related. Let's consider a sequence of linear regression models with polynomial basis functions:

$$\text{Model: } q \in 0, \ldots, Q : \boldsymbol{\phi}(x) = [1 \;\; \ldots \;\; x^q]^\intercal \tag{2}$$

Curve fitting

We will investigate the consequences of

- choosing particular basis functions,
- fitting to training points (**overfitting**).

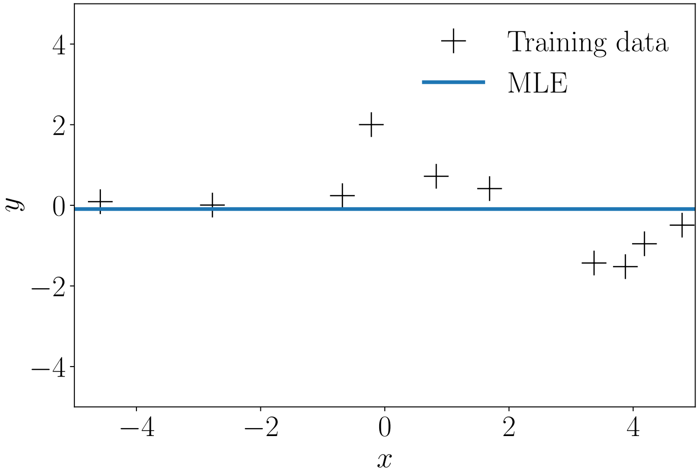
The two are related. Let's consider a sequence of linear regression models with polynomial basis functions:

$$\text{Model: } q \in 0, \ldots, Q : \boldsymbol{\phi}(x) = \begin{bmatrix} 1 & \ldots & x^q \end{bmatrix}^\mathsf{T} \tag{2}$$

Two questions: As q increases,

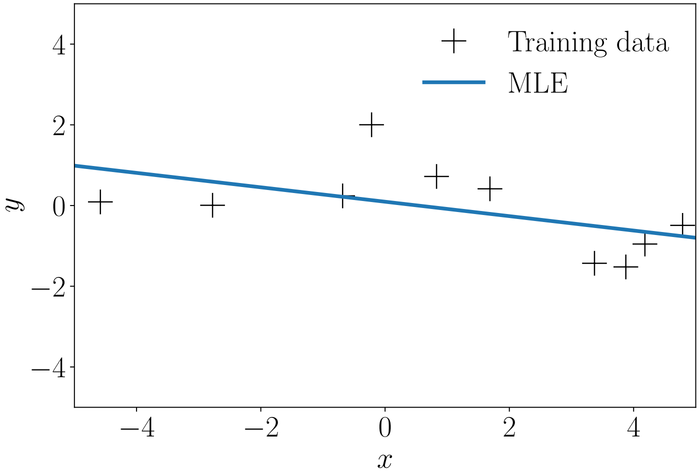
- what will happen to the **training loss**?
- how good do you think the predictions will be?

Basis functions



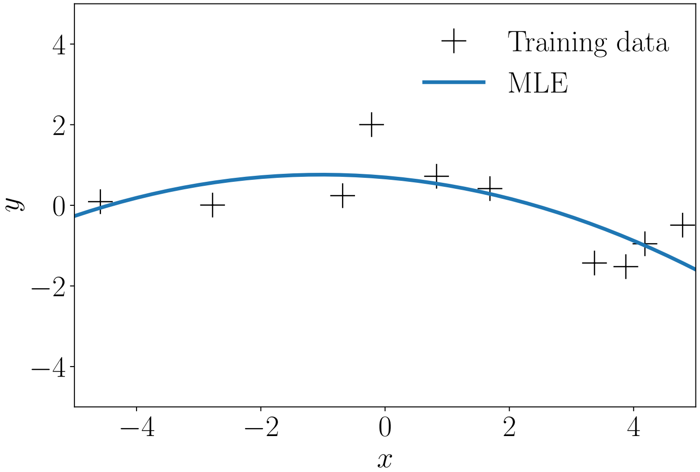
$$\boldsymbol{\phi}(x) = [1]^{\mathsf{T}} \tag{3}$$

Basis functions



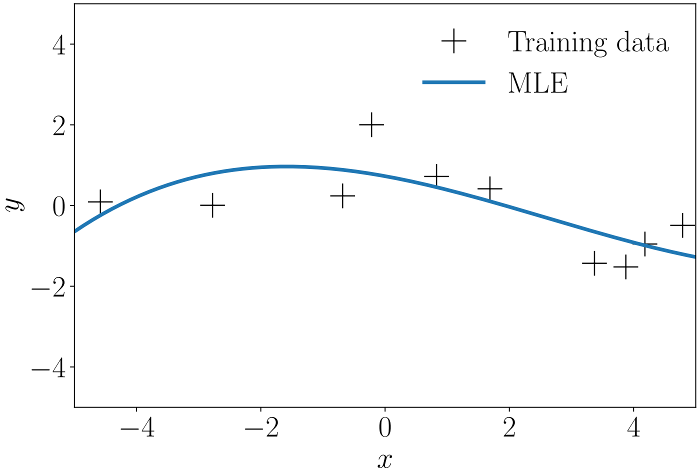
$$\boldsymbol{\phi}(x) = [1 \;\; x]^\top \tag{3}$$

Basis functions



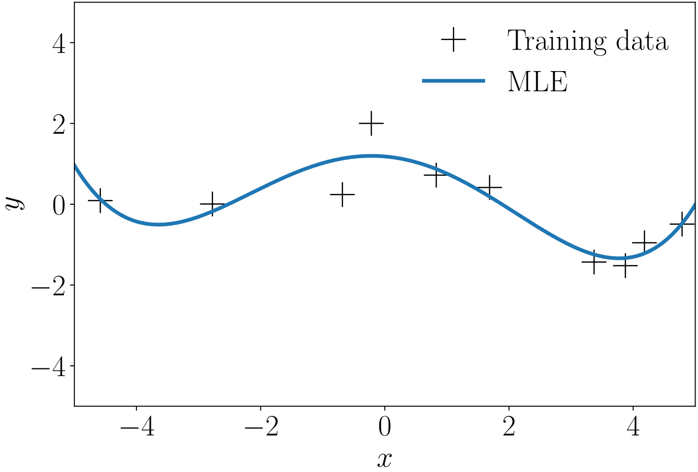
$$\boldsymbol{\phi}(x) = [1 \;\; x \;\; x^2]^\intercal \tag{3}$$

Basis functions



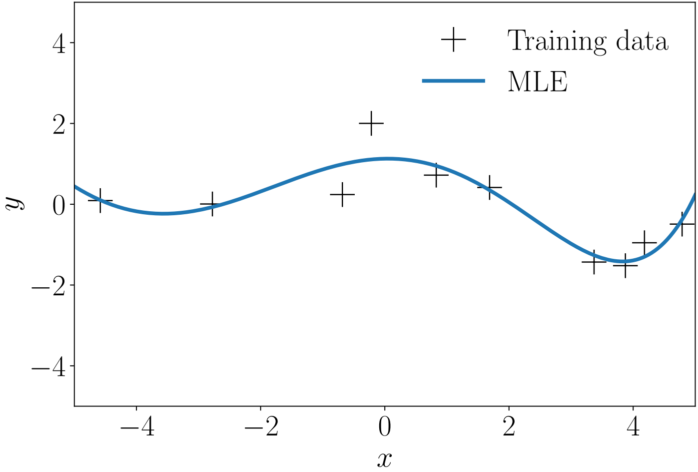
$$\boldsymbol{\phi}(x) = [1 \; x \; x^2 \; x^3]^\mathsf{T} \tag{3}$$

Basis functions



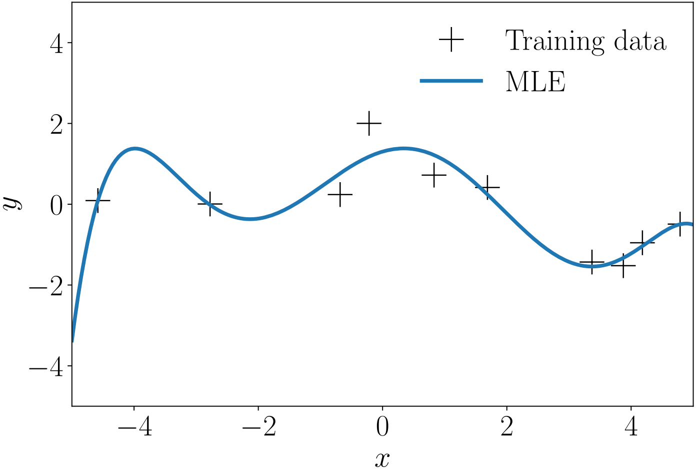
$$\boldsymbol{\phi}(x) = [1 \; x \; x^2 \; x^3, \ldots]^\intercal \tag{3}$$

Basis functions



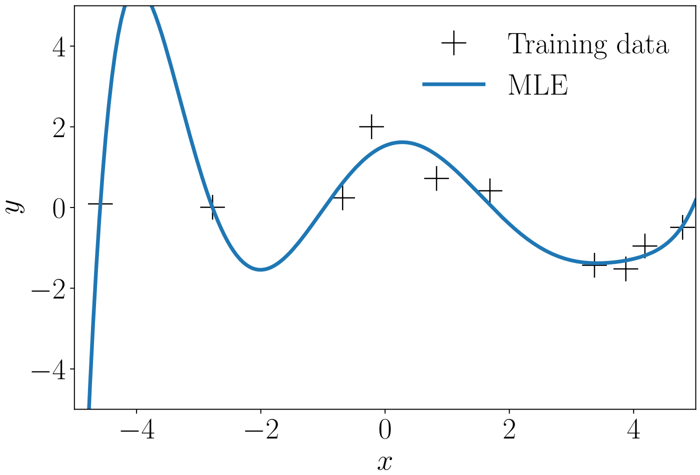
$$\boldsymbol{\phi}(x) = [1\ x\ x^2\ x^3, \dots]^\intercal \tag{3}$$

Basis functions



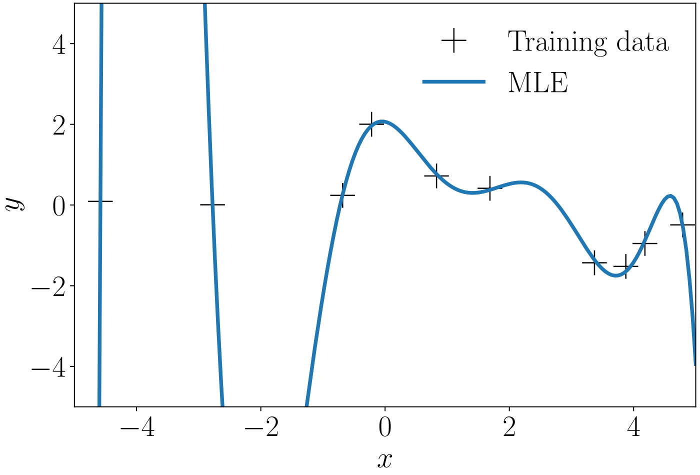
$$\boldsymbol{\phi}(x) = [1\ x\ x^2\ x^3, \dots]^\mathsf{T} \tag{3}$$

Basis functions



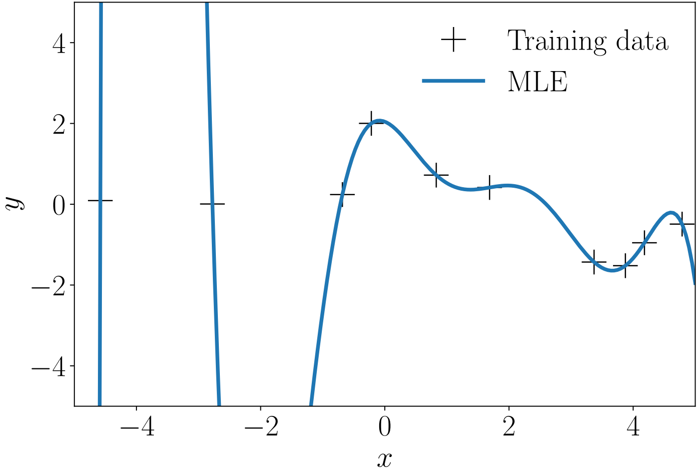
$$\boldsymbol{\phi}(x) = [1 \ x \ x^2 \ x^3, \ldots]^\mathsf{T} \tag{3}$$

Basis functions



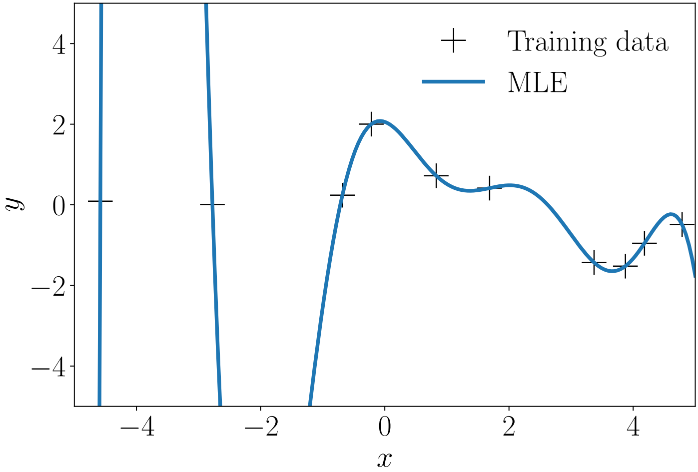
$$\boldsymbol{\phi}(x) = [1 \ x \ x^2 \ x^3, \ldots]^\intercal \tag{3}$$

Basis functions



$$\boldsymbol{\phi}(x) = [1 \ x \ x^2 \ x^3, \ldots]^\intercal \tag{3}$$

Basis functions



$$\boldsymbol{\phi}(x) = [1 \ x \ x^2 \ x^3, \ldots]^\mathsf{T} \tag{3}$$

Training loss

Q: How does the training loss change as M increases?

$$L_M(\boldsymbol{\theta}) = \sum_n (f_M(\mathbf{x}_n;\boldsymbol{\theta}) - y_n)^2\,, \qquad f_M(\mathbf{x}_n;\boldsymbol{\theta}) = \sum_{m=0}^{M} x^m \theta_m \tag{4}$$

Training loss

Q: How does the training loss change as M increases?

$$L_M(\boldsymbol{\theta}) = \sum_n (f_M(\mathbf{x}_n; \boldsymbol{\theta}) - y_n)^2 , \qquad f_M(\mathbf{x}_n; \boldsymbol{\theta}) = \sum_{m=0}^{M} x^m \theta_m \tag{4}$$

- For two models with $M_1 \leqslant M_2$, model M_2 can represent *all* functions that M_1 can, and more.

Training loss

Q: How does the training loss change as M increases?

$$L_M(\boldsymbol{\theta}) = \sum_n (f_M(\mathbf{x}_n; \boldsymbol{\theta}) - y_n)^2 , \qquad f_M(\mathbf{x}_n; \boldsymbol{\theta}) = \sum_{m=0}^{M} x^m \theta_m \tag{4}$$

- For two models with $M_1 \leqslant M_2$, model M_2 can represent *all* functions that M_1 can, and more.
- Remember: Closed-form optimisation finds the **exact** minimum.

Training loss

Q: How does the training loss change as M increases?

$$L_M(\boldsymbol{\theta}) = \sum_n (f_M(\mathbf{x}_n; \boldsymbol{\theta}) - y_n)^2 \,, \qquad f_M(\mathbf{x}_n; \boldsymbol{\theta}) = \sum_{m=0}^{M} x^m \theta_m \tag{4}$$

- For two models with $M_1 \leqslant M_2$, model M_2 can represent *all* functions that M_1 can, and more.
- Remember: Closed-form optimisation finds the **exact** minimum.

$$\implies \min_{\boldsymbol{\theta}} L_{M_2}(\boldsymbol{\theta}) \leqslant \min_{\boldsymbol{\theta}} L_{M_1}(\boldsymbol{\theta}) \tag{5}$$

Training loss

Q: How does the training loss change as M increases?

$$L_M(\boldsymbol{\theta}) = \sum_n (f_M(\mathbf{x}_n; \boldsymbol{\theta}) - y_n)^2 \,, \qquad f_M(\mathbf{x}_n; \boldsymbol{\theta}) = \sum_{m=0}^{M} x^m \theta_m \tag{4}$$

- For two models with $M_1 \leqslant M_2$, model M_2 can represent *all* functions that M_1 can, and more.
- Remember: Closed-form optimisation finds the **exact** minimum.

$$\implies \min_{\boldsymbol{\theta}} L_{M_2}(\boldsymbol{\theta}) \leqslant \min_{\boldsymbol{\theta}} L_{M_1}(\boldsymbol{\theta}) \tag{5}$$

Training loss always gets smaller as we add basis functions.

Training loss

Q: How does the training loss change as M increases?

$$L_M(\boldsymbol{\theta}) = \sum_n (f_M(\mathbf{x}_n; \boldsymbol{\theta}) - y_n)^2 \,, \qquad f_M(\mathbf{x}_n; \boldsymbol{\theta}) = \sum_{m=0}^{M} x^m \theta_m \tag{4}$$

- For two models with $M_1 \leqslant M_2$, model M_2 can represent *all* functions that M_1 can, and more.
- Remember: Closed-form optimisation finds the **exact** minimum.

$$\implies \min_{\boldsymbol{\theta}} L_{M_2}(\boldsymbol{\theta}) \leqslant \min_{\boldsymbol{\theta}} L_{M_1}(\boldsymbol{\theta}) \tag{5}$$

Training loss always gets smaller as we add basis functions.
But what about the predictions?

Train/Test Split

Some machine learning "good practice"...

How can we tell
how well a model will predict on **unseen** data?

Train/Test Split

Some machine learning "good practice"...

How can we tell
how well a model will predict on **unseen** data?

You probably already know the procedure:

Train/Test Split

Some machine learning "good practice"...

How can we tell
how well a model will predict on **unseen** data?

You probably already know the procedure:

- Split all data into a **training set** and **test set**.

Train/Test Split

Some machine learning "good practice"...

How can we tell
how well a model will predict on **unseen** data?

You probably already know the procedure:

- Split all data into a **training set** and **test set**.
- Only train on training set, and **measure** loss on the test set:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \tag{6}$$

$$L_{\text{test}} = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \ell(f(x_n; \boldsymbol{\theta}^*), y_n)\,, \qquad (x_n, y_n) \in \mathcal{S}_{\text{test}} \tag{7}$$

Train/Test Split

Some machine learning "good practice"...

How can we tell
how well a model will predict on **unseen** data?

You probably already know the procedure:

- Split all data into a **training set** and **test set**.
- Only train on training set, and **measure** loss on the test set:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \tag{6}$$

$$L_{\text{test}} = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \ell(f(x_n; \boldsymbol{\theta}^*), y_n)\,, \qquad (x_n, y_n) \in \mathcal{S}_{\text{test}} \tag{7}$$

Why does this work?

Generalisation performance

Our question is not precise enough:

Generalisation performance

Our question is not precise enough:

- How will our predictions on unseen data affect us?

Generalisation performance

Our question is not precise enough:

- How will our predictions on unseen data affect us?
- But what is unseen data?

Generalisation performance

Our question is not precise enough:

- How will our predictions on unseen data affect us?
- But what is unseen data?
- What assumptions underlie this reasoning?

Generalisation performance

Our question is not precise enough:

- How will our predictions on unseen data affect us?
- But what is unseen data?
- What assumptions underlie this reasoning?

Let's make the question precise by:

- Specifying one way for how we will use our predictions.
- Using the assumptions from lecture 1.

Predictions and Losses

- Imagine deploying a ML system in production
  (predicting ad value, engine fuel control system)

Predictions and Losses

- Imagine deploying a ML system in production
  (predicting ad value, engine fuel control system)
- If widely deployed, you will make **many** predictions

Predictions and Losses

- Imagine deploying a ML system in production
  (predicting ad value, engine fuel control system)
- If widely deployed, you will make **many** predictions
- We incur a loss for error in the predictions

Predictions and Losses

- Imagine deploying a ML system in production (predicting ad value, engine fuel control system)
- If widely deployed, you will make **many** predictions
- We incur a loss for error in the predictions
- Let's measure the loss per prediction.

$$L_{\text{deploy}} = \frac{1}{N_{\text{deploy}}} \sum_{n=1}^{N_{\text{deploy}}} \ell(f(x_n, \boldsymbol{\theta}^*), y_n) \tag{8}$$

Predictions and Losses

- Imagine deploying a ML system in production (predicting ad value, engine fuel control system)
- If widely deployed, you will make **many** predictions
- We incur a loss for error in the predictions
- Let's measure the loss per prediction.

$$L_{\text{deploy}} = \frac{1}{N_{\text{deploy}}} \sum_{n=1}^{N_{\text{deploy}}} \ell(f(x_n, \boldsymbol{\theta}^*), y_n) \tag{8}$$

- Many predictions: $N_{\text{deploy}} \to \infty$

Predictions and Losses

- Imagine deploying a ML system in production (predicting ad value, engine fuel control system)
- If widely deployed, you will make **many** predictions
- We incur a loss for error in the predictions
- Let's measure the loss per prediction.

$$L_{\text{deploy}} = \frac{1}{N_{\text{deploy}}} \sum_{n=1}^{N_{\text{deploy}}} \ell(f(x_n, \boldsymbol{\theta}^*), y_n) \tag{8}$$

- Many predictions: $N_{\text{deploy}} \to \infty$
- Does it even make sense? Is this quantity even well-defined?

Statistical View on the World

Data Generating Process

We assume that the data we observe is the outcome of some random process. Each observation is one random variable. In this course, probabilities of the data generating process are denoted with $\mathbb{P}(\cdot)$, and which has distribution $\pi(\cdot)$.

Statistical View on the World

Data Generating Process

We assume that the data we observe is the outcome of some random process. Each observation is one random variable. In this course, probabilities of the data generating process are denoted with $\mathbb{P}(\cdot)$, and which has distribution $\pi(\cdot)$.

Example:

- We observe a dataset of 3 values $\{x_n\}_{n=1}^{3}$.
- This has density $\pi_{X_1,X_2,X_3}(x_1, x_2, x_3)$.

Independent Identically Distributed

Independent Identically Distributed (iid) Assumption

Often, we assume that random variables in a dataset are independent and identically distributed, which means that each random variable has the same distribution. Groups of RVs can also be iid.

Independent Identically Distributed

Independent Identically Distributed (iid) Assumption

Often, we assume that random variables in a dataset are independent and identically distributed, which means that each random variable has the same distribution. Groups of RVs can also be iid.

Examples:

$$\pi_{X_1,X_2,X_3}(x_1,x_2,x_3) = \prod_{n=1}^{3} \pi(x_n)$$

$$\pi_{X_1,Y_1,X_2,Y_2,\ldots}(x_1,y_1,x_2,y_2,\ldots) = \pi_{X,Y}(\mathbf{x},\mathbf{y}) \qquad \mathbf{x},\mathbf{y} \in \mathbb{R}^N$$

$$= \prod_{n=1}^{N} \pi(x_n, y_n)$$

Expected Loss

- $x_n, y_n \overset{\text{iid}}{\sim} \pi$

Expected Loss

- $x_n, y_n \overset{\text{iid}}{\sim} \pi$
- $N_{\text{deploy}} \to \infty$

Expected Loss

- $x_n, y_n \overset{\text{iid}}{\sim} \pi$
- $N_{\text{deploy}} \to \infty$
- So intuitively

$$L_{\text{deploy}} = \frac{1}{N_{\text{deploy}}} \sum_{n=1}^{N_{\text{deploy}}} \ell(f(x_n, \boldsymbol{\theta}^*), y_n) \tag{9}$$

$$\approx \mathbb{E}_{\pi(x,y)}[\ell(f(x, \boldsymbol{\theta}^*), y)] \tag{10}$$

Expected Loss

- $x_n, y_n \overset{\text{iid}}{\sim} \pi$
- $N_{\text{deploy}} \to \infty$
- So intuitively

$$L_{\text{deploy}} = \frac{1}{N_{\text{deploy}}} \sum_{n=1}^{N_{\text{deploy}}} \ell(f(x_n, \boldsymbol{\theta}^*), y_n) \tag{9}$$

$$\approx \mathbb{E}_{\pi(x,y)}[\ell(f(x, \boldsymbol{\theta}^*), y)] \tag{10}$$

- Finally, a well-defined quantity

Expected Loss

- $x_n, y_n \overset{\text{iid}}{\sim} \pi$
- $N_{\text{deploy}} \to \infty$
- So intuitively

$$L_{\text{deploy}} = \frac{1}{N_{\text{deploy}}} \sum_{n=1}^{N_{\text{deploy}}} \ell(f(x_n, \boldsymbol{\theta}^*), y_n) \tag{9}$$

$$\approx \mathbb{E}_{\pi(x,y)}[\ell(f(x, \boldsymbol{\theta}^*), y)] \tag{10}$$

- Finally, a well-defined quantity
- Only meaningful because of the iid assumption!

Expected Loss

- $x_n, y_n \overset{\text{iid}}{\sim} \pi$
- $N_{\text{deploy}} \to \infty$
- So intuitively

$$L_{\text{deploy}} = \frac{1}{N_{\text{deploy}}} \sum_{n=1}^{N_{\text{deploy}}} \ell(f(x_n, \boldsymbol{\theta}^*), y_n) \tag{9}$$

$$\approx \mathbb{E}_{\pi(x,y)}[\ell(f(x, \boldsymbol{\theta}^*), y)] \tag{10}$$

- Finally, a well-defined quantity
- Only meaningful because of the iid assumption!
- Let's use a **theorem** to prove this.

Weak Law of Large Numbers

In what sense does the $\approx$ hold?

Weak Law of Large Numbers

In what sense does the $\approx$ hold?

Weak Law of Large Numbers:

- For a sequence of iid RVs $X_1, X_2, X_3, \ldots, X_N$

Weak Law of Large Numbers

In what sense does the $\approx$ hold?

Weak Law of Large Numbers:

- For a sequence of iid RVs $X_1, X_2, X_3, \ldots, X_N$
- with mean $\mu = \mathbb{E}[X]$

Weak Law of Large Numbers

In what sense does the $\approx$ hold?

Weak Law of Large Numbers:

- For a sequence of iid RVs $X_1, X_2, X_3, \ldots, X_N$
- with mean $\mu = \mathbb{E}[X]$
- we can define a new RV $\overline{X}_N = \frac{1}{N}\sum_{n=1}^{N} X_n$

Weak Law of Large Numbers

In what sense does the $\approx$ hold?

Weak Law of Large Numbers:

- For a sequence of iid RVs $X_1, X_2, X_3, \ldots, X_N$
- with mean $\mu = \mathbb{E}[X]$
- we can define a new RV $\overline{X}_N = \frac{1}{N}\sum_{n=1}^{N} X_n$
- for which will hold:

$$\lim_{N\to\infty} \mathbb{P}\left(|\overline{X}_n - \mu| < \epsilon\right) = 1 \tag{11}$$

Expected Loss

Deploy loss converging to expected loss:

- $x_n, y_n \stackrel{\text{iid}}{\sim} \pi_{X,Y} \implies \ell_n = \ell(f(x_n), y_n))$ is a RV, with density $\pi(\ell_n)$

Expected Loss

Deploy loss converging to expected loss:

- $x_n, y_n \overset{\text{iid}}{\sim} \pi_{X,Y} \implies \ell_n = \ell(f(x_n), y_n))$ is a RV, with density $\pi(\ell_n)$
- $L_{\text{deploy}}^{N_{\text{deploy}}} = \frac{1}{N_{\text{deploy}}} \sum_{n=1}^{N_{\text{deploy}}} \ell_n$

Expected Loss

Deploy loss converging to expected loss:

- $x_n, y_n \overset{\text{iid}}{\sim} \pi_{X,Y} \implies \ell_n = \ell(f(x_n), y_n))$ is a RV, with density $\pi(\ell_n)$
- $L_{\text{deploy}}^{N_{\text{deploy}}} = \frac{1}{N_{\text{deploy}}} \sum_{n=1}^{N_{\text{deploy}}} \ell_n$
- By Weak LLN

$$\lim_{N_{\text{deploy}} \to \infty} \mathbb{P}(|L_{\text{deploy}} - \mu| < \epsilon) = 1 \tag{12}$$

Expected Loss

Deploy loss converging to expected loss:

- $x_n, y_n \overset{\text{iid}}{\sim} \pi_{X,Y} \implies \ell_n = \ell(f(x_n), y_n))$ is a RV, with density $\pi(\ell_n)$
- $L_{\text{deploy}}^{N_{\text{deploy}}} = \frac{1}{N_{\text{deploy}}} \sum_{n=1}^{N_{\text{deploy}}} \ell_n$
- By Weak LLN

$$\lim_{N_{\text{deploy}} \to \infty} \mathbb{P}(|L_{\text{deploy}} - \mu| < \epsilon) = 1 \tag{12}$$

- With $\mu = \mathbb{E}_{\pi(\ell)}[\ell] = \mathbb{E}_{\pi(x,y)}[\ell(f(x, \boldsymbol{\theta}^*), y)]$ by LOTUS

Monte Carlo Estimate of the Expected Loss

- We now know that the deploy loss is well-defined

Monte Carlo Estimate of the Expected Loss

- We now know that the deploy loss is well-defined
- But, we can't compute it: don't know $\pi_{X,Y}$

Monte Carlo Estimate of the Expected Loss

- We now know that the deploy loss is well-defined
- But, we can't compute it: don't know $\pi_{X,Y}$
- Can we estimate it?

Monte Carlo Estimate of the Expected Loss

- We now know that the deploy loss is well-defined
- But, we can't compute it: don't know $\pi_{X,Y}$
- Can we estimate it?
- Previous process in reverse $\implies$ **Monte Carlo** estimation

Monte Carlo Estimation

Want to find a difficult expectation / integral:

$$I = \int p_X(x) f(x) \mathrm{d}x = \mathbb{E}_{p_X(X)}[f(X)] \tag{13}$$

Monte Carlo Estimation

Want to find a difficult expectation / integral:

$$I = \int p_X(x) f(x) \mathrm{d}x = \mathbb{E}_{p_X(X)}[f(X)] \tag{13}$$

Use estimator

$$\hat{I} = \frac{1}{N} \sum_{n=1}^{N} f(x_n) \qquad\qquad x_n \overset{\text{iid}}{\sim} p_X \tag{14}$$

Monte Carlo Estimation

Want to find a difficult expectation / integral:

$$I = \int p_X(x) f(x) \mathrm{d}x = \mathbb{E}_{p_X(X)}[f(X)] \tag{13}$$

Use estimator

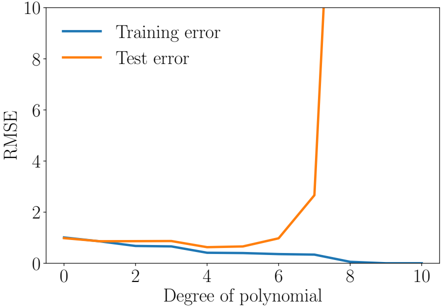
$$\hat{I} = \frac{1}{N} \sum_{n=1}^{N} f(x_n) \qquad\qquad x_n \overset{\text{iid}}{\sim} p_X \tag{14}$$

By considering it as a sum of independent RVs:

- Can show it to be **unbiased**, i.e. $\mathbb{E}_{p(x_1, x_2, \dots)}\left[\hat{I}\right] = I$ (board).
- Can show the variance reduces as c/N (board).
- Weak LLN implies $\lim_{N\to\infty} P(|\hat{I} - I| < \epsilon) = 1$ (relies on unbiasedness!)

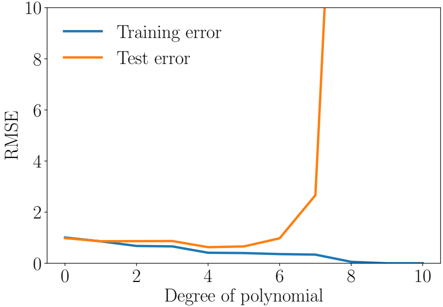
Overfitting

Can use test loss estimator to **evaluate** various models:



Overfitting

Can use test loss estimator to **evaluate** various models:



- We indeed see that training error continuously decreases.
- Test error only decreases with train error up to a point!
- When test error starts increasing $\implies$ overfitting.

Model selection

We can prevent overfitting by choosing a model which isn't too flexible.

Q: How do we decide which model to use?

Model selection

We can prevent overfitting by choosing a model which isn't too flexible.

Q: How do we decide which model to use?

$\implies$ **model selection**

Model selection

We can prevent overfitting by choosing a model which isn't too flexible.

Q: How do we decide which model to use?

$\Longrightarrow$ **model selection**

Summary

- Overfitting
- Assumptions behind test sets (many predictions + iid data generation)
- Law of Large Numbers (skill)
- Monte Carlo estimation (skill)