THE UNIVERSITY of EDINBURGH
informatics

Applied Machine Learning (AML)

Generalisation

Oisin Mac Aodha • Siddharth N.

Generalisation

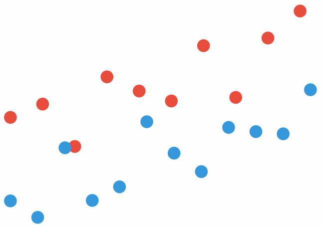
Outline

- What is Generalisation?
- How do we characterise/measure it?
- What can we do to improve it?

Generalisation

What is Generalisation?

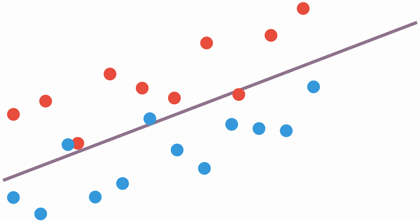
Generalisation



Machine Learning

- observe data

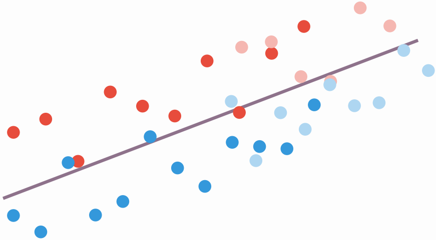
Generalisation



Machine Learning

- observe data
- learn to model observed data (*training* data)

Generalisation



Machine Learning

- observe data
- learn to model observed data (*training* data)
- generalise to unseen, novel data (*test* data)

Reasoning about Generalisation

Overfitting

Underfitting

Reasoning about Generalisation

Overfitting

- Fit training data well; unseen data poorly

Underfitting

- Fits both training and unseen data poorly

Reasoning about Generalisation

Overfitting

- Fit training data well; unseen data poorly
- Reason: accidental regularities

Underfitting

- Fits both training and unseen data poorly
- Reason: insufficient regularities

Reasoning about Generalisation

Overfitting

- Fit training data well; unseen data poorly
- Reason: accidental regularities
- Reason: memorisation

Underfitting

- Fits both training and unseen data poorly
- Reason: insufficient regularities

Reasoning about Generalisation

Overfitting

- Fit training data well; unseen data poorly
- Reason: accidental regularities
- Reason: memorisation
- Model has very large capacity

Underfitting

- Fits both training and unseen data poorly
- Reason: insufficient regularities
- Model has insufficient capacity

Reasoning about Generalisation

Overfitting

- Fit training data well; unseen data poorly
- Reason: accidental regularities
- Reason: memorisation
- Model has very large capacity

Underfitting

- Fits both training and unseen data poorly
- Reason: insufficient regularities
- Model has insufficient capacity

Capacity $\approx$ # model parameters

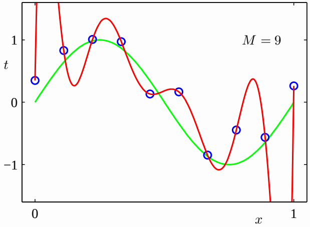
Overfitting vs. Underfitting: Example

Regression

Figures: C. Bishop - PRML

Overfitting vs. Underfitting: Example

Regression

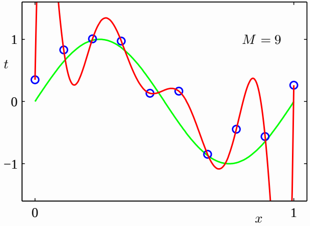


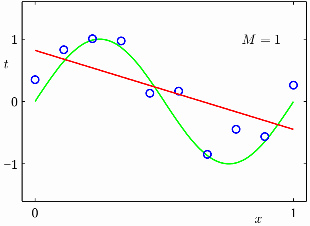
model too flexible:
fits noise

Figures: C. Bishop - PRML

Overfitting vs. Underfitting: Example

Regression





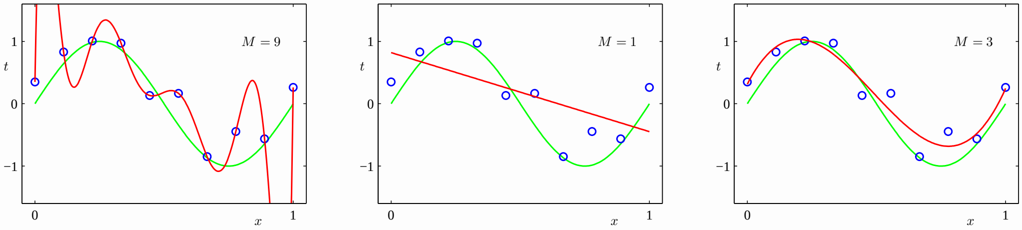
model too flexible:
fits noise

model too inflexible:
cannot capture pattern

Figures: C. Bishop - PRML

Overfitting vs. Underfitting: Example

Regression



model too flexible:
fits noise

model too inflexible:
cannot capture pattern

model just right

Figures: C. Bishop - PRML

Reasoning about Generalisation: Qualitative

Training Data

Reasoning about Generalisation: Qualitative

Training Data

- More $\Longrightarrow$ better generalisation
  - close training example likely

Reasoning about Generalisation: Qualitative

Training Data

- More $\Longrightarrow$ better generalisation
  - close training example likely
  - fewer accidental regularities

Reasoning about Generalisation: Qualitative

Training Data

- More $\Longrightarrow$ better generalisation
  - close training example likely
  - fewer accidental regularities
- Less $\Longrightarrow$ lower training error

Reasoning about Generalisation: Qualitative

Training Data

- More $\implies$ better generalisation
  - close training example likely
  - fewer accidental regularities
- Less $\implies$ lower training error
  - easier to memorise

Reasoning about Generalisation: Qualitative

Training Data

- More $\implies$ better generalisation
  - close training example likely
  - fewer accidental regularities
- Less $\implies$ lower training error
  - easier to memorise
  - fewer regularities to capture

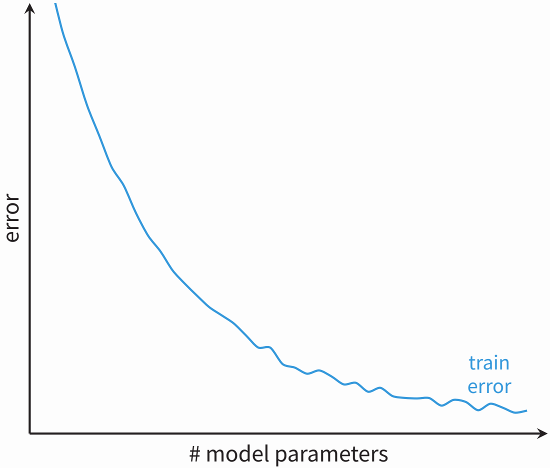
Reasoning about Generalisation: Qualitative

Model Parameters

Reasoning about Generalisation: Qualitative

Model Parameters

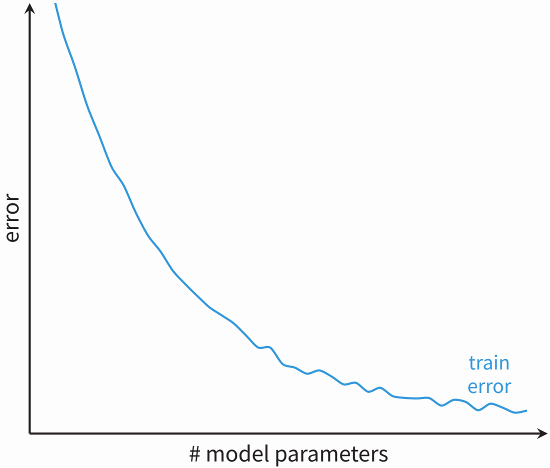
- More $\implies$ better training error
  - better flexibility



Reasoning about Generalisation: Qualitative

Model Parameters

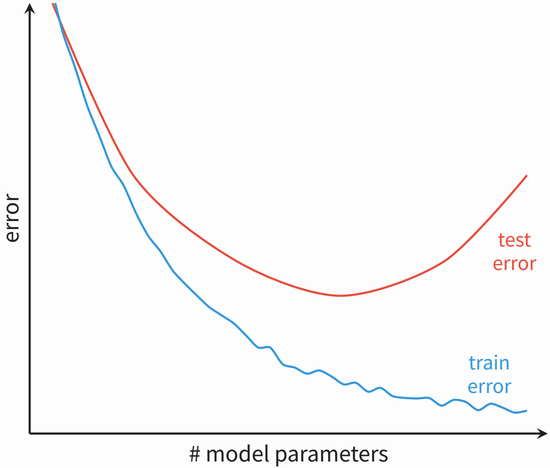
- More $\Longrightarrow$ better training error
  - better flexibility
  - easier to fit true and accidental regularities



Reasoning about Generalisation: Qualitative

Model Parameters

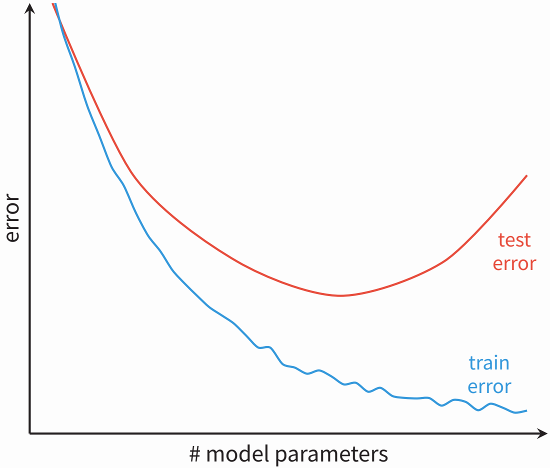
- More $\implies$ better training error
  - better flexibility
  - easier to fit true and accidental regularities
- Much more $\implies$ poor generalisation



Reasoning about Generalisation: Qualitative

Model Parameters

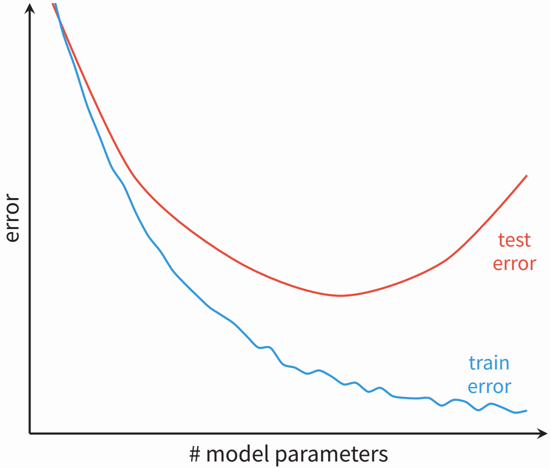
- More $\implies$ better training error
  - better flexibility
  - easier to fit true and accidental regularities
- Much more $\implies$ poor generalisation
  - easier to memorise



Reasoning about Generalisation: Qualitative

Model Parameters

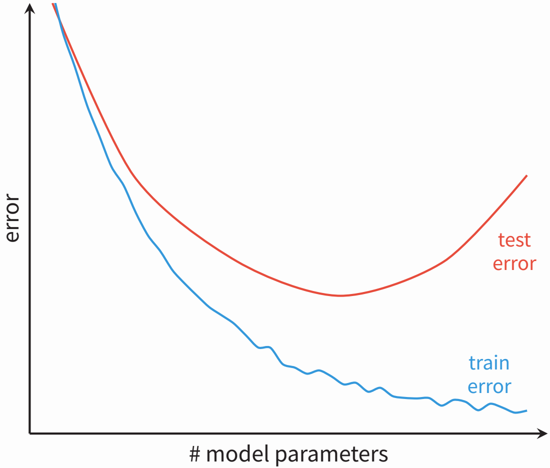
- More $\implies$ better training error
  - better flexibility
  - easier to fit true and accidental regularities
- Much more $\implies$ poor generalisation
  - easier to memorise
- Much less $\implies$ poor generalisation



Reasoning about Generalisation: Qualitative

Model Parameters

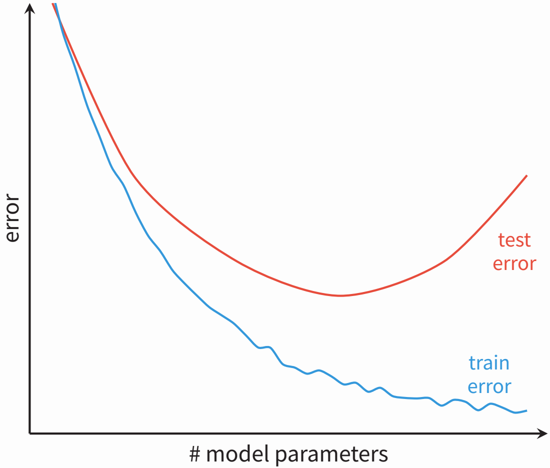
- More $\implies$ better training error
  - better flexibility
  - easier to fit true and accidental regularities
- Much more $\implies$ poor generalisation
  - easier to memorise
- Much less $\implies$ poor generalisation
  - struggle to capture regularities



Reasoning about Generalisation: Qualitative

Model Parameters

- More $\implies$ better training error
  - better flexibility
  - easier to fit true and accidental regularities
- Much more $\implies$ poor generalisation
  - easier to memorise
- Much less $\implies$ poor generalisation
  - struggle to capture regularities



Goldilocks Zone: Sufficient capacity to learn true regularities, but not enough to memorise or exploit accidental regularities.

Tuning Model Capacity

Data requirements

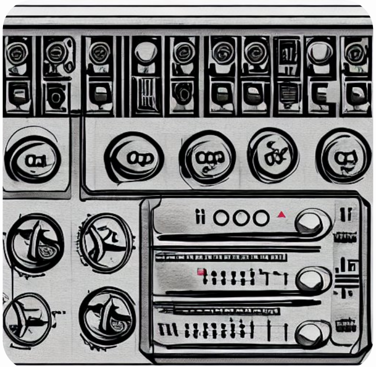
- Different data requires different capacity

Figures: Stable Diffusion (Huggingface)

Tuning Model Capacity

Data requirements

- Different data requires different capacity
- Need "controls" to control capacity

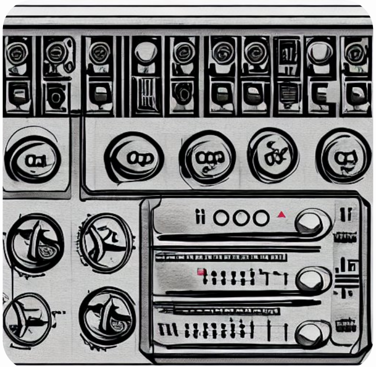


Figures: Stable Diffusion (Huggingface)

Tuning Model Capacity

Data requirements

- Different data requires different capacity
- Need “controls” to control capacity
- “controls” ≡ model hyper-parameters

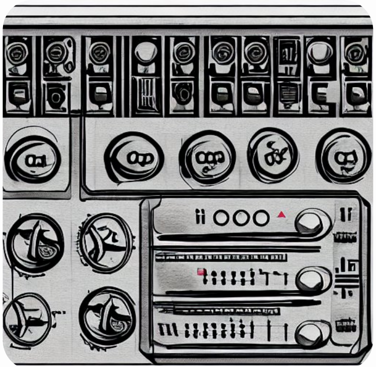


Figures: Stable Diffusion (Huggingface)

Tuning Model Capacity

Data requirements

- Different data requires different capacity
- Need “controls” to control capacity
- “controls” ≡ model hyper-parameters
  - Regression: polynomial order

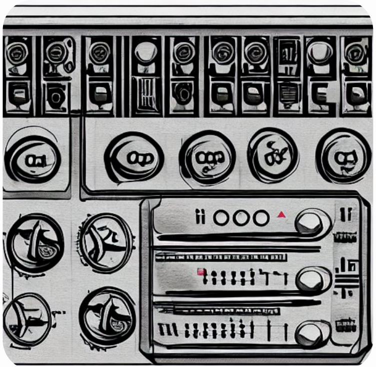


Figures: Stable Diffusion (Huggingface)

Tuning Model Capacity

Data requirements

- Different data requires different capacity
- Need "controls" to control capacity
- "controls" ≡ model hyper-parameters
  - Regression: polynomial order
  - Naive Bayes: # attributes, bounds on σ^2

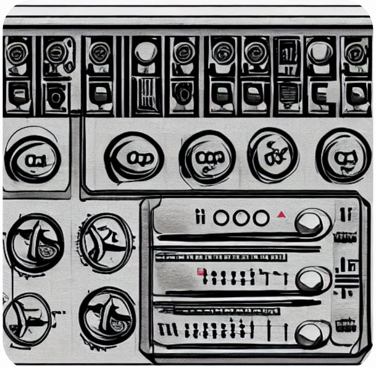


Figures: Stable Diffusion (Huggingface)

Tuning Model Capacity

Data requirements

- Different data requires different capacity
- Need "controls" to control capacity
- "controls" ≡ model hyper-parameters
  - Regression: polynomial order
  - Naive Bayes: # attributes, bounds on σ^2
  - Decision Trees: # nodes

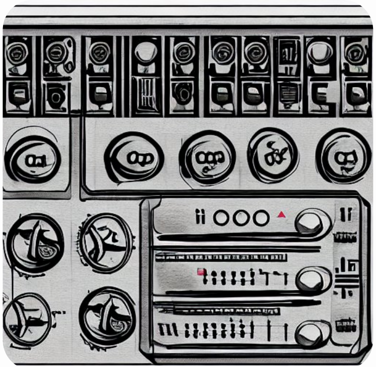


Figures: Stable Diffusion (Huggingface)

Tuning Model Capacity

Data requirements

- Different data requires different capacity
- Need "controls" to control capacity
- "controls" ≡ model hyper-parameters
  - Regression: polynomial order
  - Naive Bayes: # attributes, bounds on σ^2
  - Decision Trees: # nodes



Tune to minimise generalisation error

Figures: Stable Diffusion (Huggingface)

Generalisation

Measuring Generalisation

Beyond Fitting Training Data

Optimising an error function defined as the average loss over *training* set:

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\big(\hat{y}_i, y_i\big), \text{where } \hat{y}_i = f(\boldsymbol{x}_i; \boldsymbol{w})$$

Beyond Fitting Training Data

Optimising an error function defined as the average loss over *training* set:

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(\hat{y}_i, y_i\right), \text{ where } \hat{y}_i = f(\boldsymbol{x}_i; \boldsymbol{w})$$

Want

- not just fit training data well
- generalise to *novel* and *unseen* instances

Setup to Estimate Generalisation

Need to estimate error on test data *without* training on test data!

Setup to Estimate Generalisation

Need to estimate error on test data *without* training on test data!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{test}}\}$$

Setup to Estimate Generalisation

Need to estimate error on test data *without* training on test data!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{test}}\}$$

Cross Validation

- $\{\mathcal{D}_{\text{train}_1}; \mathcal{D}_{\text{test}_1}\}, \ldots, \{\mathcal{D}_{\text{train}_K}; \mathcal{D}_{\text{test}_K}\}$

Setup to Estimate Generalisation

Need to estimate error on test data *without* training on test data!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{test}}\}$$

Cross Validation

- $\{\mathcal{D}_{\text{train}_1}; \mathcal{D}_{\text{test}_1}\}, \ldots, \{\mathcal{D}_{\text{train}_K}; \mathcal{D}_{\text{test}_K}\}$
- partition data into train/test in *different* ways

Setup to Estimate Generalisation

Need to estimate error on test data *without* training on test data!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{test}}\}$$

Cross Validation

- $\{\mathcal{D}_{\text{train}_1}; \mathcal{D}_{\text{test}_1}\}, \ldots, \{\mathcal{D}_{\text{train}_K}; \mathcal{D}_{\text{test}_K}\}$
- partition data into train/test in *different* ways
  - Leave-1-out cross validation

Setup to Estimate Generalisation

Need to estimate error on test data *without* training on test data!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{test}}\}$$

Cross Validation

- $\{\mathcal{D}_{\text{train}_1}; \mathcal{D}_{\text{test}_1}\}, \ldots, \{\mathcal{D}_{\text{train}_K}; \mathcal{D}_{\text{test}_K}\}$
- partition data into train/test in *different* ways
  - Leave-1-out cross validation
  - Leave-K-out cross validation

Setup to Estimate Generalisation

Need to estimate error on test data *without* training on test data!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{test}}\}$$

Cross Validation

- $\{\mathcal{D}_{\text{train}_1}; \mathcal{D}_{\text{test}_1}\}, \ldots, \{\mathcal{D}_{\text{train}_K}; \mathcal{D}_{\text{test}_K}\}$
- partition data into train/test in *different* ways
  - Leave-1-out cross validation
  - Leave-K-out cross validation
- for each partition: train model on training data $\longrightarrow$ test error on test data

Setup to Estimate Generalisation

Need to estimate error on test data *without* training on test data!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{test}}\}$$

Cross Validation

- $\{\mathcal{D}_{\text{train}_1}; \mathcal{D}_{\text{test}_1}\}, \ldots, \{\mathcal{D}_{\text{train}_K}; \mathcal{D}_{\text{test}_K}\}$
- partition data into train/test in *different* ways
    - Leave-1-out cross validation
    - Leave-K-out cross validation
- for each partition: train model on training data $\longrightarrow$ test error on test data
- 'best' model $\equiv$ model from partition with lowest test error

Setup to Estimate Generalisation

Need to estimate error on test data *without* training on test data!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{test}}\}$$

Cross Validation

- $\{\mathcal{D}_{\text{train}_1}; \mathcal{D}_{\text{test}_1}\}, \ldots, \{\mathcal{D}_{\text{train}_K}; \mathcal{D}_{\text{test}_K}\}$
- partition data into train/test in *different* ways
  - Leave-1-out cross validation
  - Leave-K-out cross validation
- for each partition: train model on training data $\longrightarrow$ test error on test data
- ‘best’ model $\equiv$ model from partition with lowest test error
- typically used for ‘small’ data

Setup to Estimate Generalisation

But models have hyper-parameters!

Setup to Estimate Generalisation

But models have hyper-parameters!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{val}}; \mathcal{D}_{\text{test}}\}$$

Setup to Estimate Generalisation

But models have hyper-parameters!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{val}}; \mathcal{D}_{\text{test}}\}$$

Train–Val–Test

- cannot tune on training set—need values that generalise!

Setup to Estimate Generalisation

But models have hyper-parameters!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{val}}; \mathcal{D}_{\text{test}}\}$$

Train–Val–Test

- cannot tune on training set—need values that generalise!
- cannot tune on test set—peeking at 'unseen' data!

Setup to Estimate Generalisation

But models have hyper-parameters!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{val}}; \mathcal{D}_{\text{test}}\}$$

Train–Val–Test

- cannot tune on training set—need values that generalise!
- cannot tune on test set—peeking at 'unseen' data!
- tune hyper-parameters on $\mathcal{D}_{\text{val}}$

Setup to Estimate Generalisation

But models have hyper-parameters!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{val}}; \mathcal{D}_{\text{test}}\}$$

Train–Val–Test

- cannot tune on training set—need values that generalise!
- cannot tune on test set—peeking at ‘unseen’ data!
- tune hyper-parameters on $\mathcal{D}_{\text{val}}$
  - for every candidate set of hyper-parameters, train on $\mathcal{D}_{\text{train}}$

Setup to Estimate Generalisation

But models have hyper-parameters!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{val}}; \mathcal{D}_{\text{test}}\}$$

Train–Val–Test

- cannot tune on training set—need values that generalise!
- cannot tune on test set—peeking at ‘unseen’ data!
- tune hyper-parameters on $\mathcal{D}_{\text{val}}$
  - for every candidate set of hyper-parameters, train on $\mathcal{D}_{\text{train}}$
  - evaluate error on $\mathcal{D}_{\text{val}}$

Setup to Estimate Generalisation

But models have hyper-parameters!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{val}}; \mathcal{D}_{\text{test}}\}$$

Train–Val–Test

- cannot tune on training set—need values that generalise!
- cannot tune on test set—peeking at ‘unseen’ data!
- tune hyper-parameters on $\mathcal{D}_{\text{val}}$
  - for every candidate set of hyper-parameters, train on $\mathcal{D}_{\text{train}}$
  - evaluate error on $\mathcal{D}_{\text{val}}$
  - ‘best’ hyper-parameters ≡ lowest error on $\mathcal{D}_{\text{val}}$

Setup to Estimate Generalisation

But models have hyper-parameters!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{val}}; \mathcal{D}_{\text{test}}\}$$

Train–Val–Test

- cannot tune on training set—need values that generalise!
- cannot tune on test set—peeking at 'unseen' data!
- tune hyper-parameters on $\mathcal{D}_{\text{val}}$
  - for every candidate set of hyper-parameters, train on $\mathcal{D}_{\text{train}}$
  - evaluate error on $\mathcal{D}_{\text{val}}$
  - 'best' hyper-parameters $\equiv$ lowest error on $\mathcal{D}_{\text{val}}$
- use model trained with 'best' hyper-parameters $\longrightarrow$ test error on test data

Setup to Estimate Generalisation

But models have hyper-parameters!

$$\mathcal{D} = \{\mathcal{D}_{\text{train}}; \mathcal{D}_{\text{val}}; \mathcal{D}_{\text{test}}\}$$

Train–Val–Test

- cannot tune on training set—need values that generalise!
- cannot tune on test set—peeking at ‘unseen’ data!
- tune hyper-parameters on $\mathcal{D}_{\text{val}}$
  - for every candidate set of hyper-parameters, train on $\mathcal{D}_{\text{train}}$
  - evaluate error on $\mathcal{D}_{\text{val}}$
  - ‘best’ hyper-parameters $\equiv$ lowest error on $\mathcal{D}_{\text{val}}$
- use model trained with ‘best’ hyper-parameters $\longrightarrow$ test error on test data
- typically used for ‘big’ data; hard to cross validate with partitions

Modelling Generalisation Error

Setup

$$\mathcal{D} \coloneqq \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \sim p_{\mathcal{D}}(\boldsymbol{x}, y)$$

Modelling Generalisation Error

Setup

$$\mathcal{D} := \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \sim p_{\mathcal{D}}(\boldsymbol{x}, y)$$

Targets need not be unique

$$y \sim p_{\mathcal{D}}(y|\boldsymbol{x})$$

Modelling Generalisation Error

Setup

$$\mathcal{D} \coloneqq \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \sim p_{\mathcal{D}}(\boldsymbol{x}, y)$$

Targets need not be unique

$$y \sim p_{\mathcal{D}}(y|\boldsymbol{x})$$

House 1 = $\boldsymbol{x}_1$ = {3BHK, `garden`=T, `sqft`=1600} y_1 = sale price = 425K
House 2 = $\boldsymbol{x}_2$ = {3BHK, `garden`=T, `sqft`=1600} y_2 = sale price = 415K

Modelling Generalisation Error

Setup

$$\mathcal{D} \coloneqq \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \sim p_{\mathcal{D}}(\boldsymbol{x}, y)$$

Targets need not be unique

$$y \sim p_{\mathcal{D}}(y|\boldsymbol{x})$$

House 1 = $\boldsymbol{x}_1$ = {3BHK, `garden=T`, `sqft=1600`} y_1 = sale price = 425K

House 2 = $\boldsymbol{x}_2$ = {3BHK, `garden=T`, `sqft=1600`} y_2 = sale price = 415K

Model prediction

$$\hat{y} \sim p_{\boldsymbol{w}}(\hat{y}|\boldsymbol{x})$$

Bias and Variance

Expected Target Error

Targets sampled as $y \sim p_{\mathcal{D}}(y|\boldsymbol{x})$.

Bias and Variance

Expected Target Error

Targets sampled as $y \sim p_{\mathcal{D}}(y|\boldsymbol{x})$.

$$\mathbb{E}\left[(\hat{y} - y)^2 | \boldsymbol{x}\right] = \mathbb{E}\left[\hat{y}^2 - 2\hat{y}y + y^2 | \boldsymbol{x}\right]$$

Bias and Variance

Expected Target Error

Targets sampled as $y \sim p_{\mathcal{D}}(y|\boldsymbol{x})$.

$$\begin{aligned}\mathbb{E}\big[(\hat{y}-y)^2|\boldsymbol{x}\big] &= \mathbb{E}\big[\hat{y}^2 - 2\hat{y}y + y^2|\boldsymbol{x}\big] \\ &= \hat{y}^2 - 2\hat{y}\,\mathbb{E}[y|\boldsymbol{x}] + \mathbb{E}\big[y^2|\boldsymbol{x}\big] \qquad \text{(linearity of expectation)}\end{aligned}$$

Bias and Variance

Expected Target Error

Targets sampled as $y \sim p_{\mathcal{D}}(y|\boldsymbol{x})$.

$$\begin{aligned}
\mathbb{E}\left[(\hat{y} - y)^2|\boldsymbol{x}\right] &= \mathbb{E}\left[\hat{y}^2 - 2\hat{y}y + y^2|\boldsymbol{x}\right] \\
&= \hat{y}^2 - 2\hat{y}\,\mathbb{E}[y|\boldsymbol{x}] + \mathbb{E}\left[y^2|\boldsymbol{x}\right] && \text{(linearity of expectation)} \\
&= \hat{y}^2 - 2\hat{y}\,\mathbb{E}[y|\boldsymbol{x}] + \mathbb{E}[y|\boldsymbol{x}]^2 + \mathrm{Var}[y|\boldsymbol{x}] && \text{(expression for variance)}
\end{aligned}$$

Bias and Variance

Expected Target Error

Targets sampled as $y \sim p_{\mathcal{D}}(y|\boldsymbol{x})$.

$$\begin{aligned}
\mathbb{E}\left[(\hat{y} - y)^2 | \boldsymbol{x}\right] &= \mathbb{E}\left[\hat{y}^2 - 2\hat{y}y + y^2 | \boldsymbol{x}\right] \\
&= \hat{y}^2 - 2\hat{y}\,\mathbb{E}[y|\boldsymbol{x}] + \mathbb{E}\left[y^2|\boldsymbol{x}\right] && \text{(linearity of expectation)} \\
&= \hat{y}^2 - 2\hat{y}\,\mathbb{E}[y|\boldsymbol{x}] + \mathbb{E}[y|\boldsymbol{x}]^2 + \mathrm{Var}[y|\boldsymbol{x}] && \text{(expression for variance)} \\
&= (\hat{y} - \mathbb{E}[y|\boldsymbol{x}])^2 + \mathrm{Var}[y|\boldsymbol{x}]
\end{aligned}$$

Bias and Variance

Expected Target Error

Targets sampled as $y \sim p_{\mathcal{D}}(y|\boldsymbol{x})$.

$$\begin{aligned}
\mathbb{E}\left[(\hat{y} - y)^2 | \boldsymbol{x}\right] &= \mathbb{E}\left[\hat{y}^2 - 2\hat{y}y + y^2 | \boldsymbol{x}\right] \\
&= \hat{y}^2 - 2\hat{y}\,\mathbb{E}[y|\boldsymbol{x}] + \mathbb{E}\left[y^2|\boldsymbol{x}\right] && \text{(linearity of expectation)} \\
&= \hat{y}^2 - 2\hat{y}\,\mathbb{E}[y|\boldsymbol{x}] + \mathbb{E}[y|\boldsymbol{x}]^2 + \mathrm{Var}[y|\boldsymbol{x}] && \text{(expression for variance)} \\
&= (\hat{y} - \mathbb{E}[y|\boldsymbol{x}])^2 + \mathrm{Var}[y|\boldsymbol{x}] \\
&\triangleq \underbrace{(\hat{y} - y_\star)^2}_{\text{residual}} + \underbrace{\mathrm{Var}[y|\boldsymbol{x}]}_{\text{Bayes error}}
\end{aligned}$$

Bias and Variance

Expected Test Error

Assume model $(p_{\boldsymbol{w}})$ trained on $\mathcal{D} \sim p_{\mathcal{D}}(\boldsymbol{x}, y)$; compute predictions on $\boldsymbol{x}$.

Predictions generated as $\hat{y} \sim p_{\boldsymbol{w}}(\hat{y}|\boldsymbol{x})$.

Bias and Variance

Expected Test Error

Assume model $(p_{\boldsymbol{w}})$ trained on $\mathcal{D} \sim p_{\mathcal{D}}(\boldsymbol{x}, y)$; compute predictions on $\boldsymbol{x}$.

Predictions generated as $\hat{y} \sim p_{\boldsymbol{w}}(\hat{y}|\boldsymbol{x})$.

$$\mathbb{E}\left[(\hat{y} - y)^2\right] = \mathbb{E}\left[(\hat{y} - y_\star)^2\right] + \mathrm{Var}[y]$$

Bias and Variance

Expected Test Error

Assume model $(p_{\boldsymbol{w}})$ trained on $\mathcal{D} \sim p_{\mathcal{D}}(\boldsymbol{x}, y)$; compute predictions on $\boldsymbol{x}$.

Predictions generated as $\hat{y} \sim p_{\boldsymbol{w}}(\hat{y}|\boldsymbol{x})$.

$$\begin{aligned}
\mathbb{E}\left[(\hat{y} - y)^2\right] &= \mathbb{E}\left[(\hat{y} - y_\star)^2\right] + \mathrm{Var}[y] \\
&= \mathbb{E}\left[y_\star^2 - 2\hat{y}y_\star + \hat{y}^2\right] + \mathrm{Var}[y]
\end{aligned}$$

Bias and Variance

Expected Test Error

Assume model $(p_{\boldsymbol{w}})$ trained on $\mathcal{D} \sim p_{\mathcal{D}}(\boldsymbol{x}, y)$; compute predictions on $\boldsymbol{x}$.

Predictions generated as $\hat{y} \sim p_{\boldsymbol{w}}(\hat{y}|\boldsymbol{x})$.

$$\begin{aligned} \mathbb{E}\left[(\hat{y} - y)^2\right] &= \mathbb{E}\left[(\hat{y} - y_\star)^2\right] + \mathrm{Var}[y] \\ &= \mathbb{E}\left[y_\star^2 - 2\hat{y}y_\star + \hat{y}^2\right] + \mathrm{Var}[y] \\ &= y_\star^2 - 2y_\star\,\mathbb{E}[\hat{y}] + \mathbb{E}\left[\hat{y}^2\right] + \mathrm{Var}[y] && \text{(linearity of expectation)} \end{aligned}$$

Bias and Variance

Expected Test Error

Assume model ($p_{\boldsymbol{w}}$) trained on $\mathcal{D} \sim p_{\mathcal{D}}(\boldsymbol{x}, y)$; compute predictions on $\boldsymbol{x}$.

Predictions generated as $\hat{y} \sim p_{\boldsymbol{w}}(\hat{y}|\boldsymbol{x})$.

$$
\begin{aligned}
\mathbb{E}\left[(\hat{y} - y)^2\right] &= \mathbb{E}\left[(\hat{y} - y_\star)^2\right] + \mathrm{Var}[y] \\
&= \mathbb{E}\left[y_\star^2 - 2\hat{y}y_\star + \hat{y}^2\right] + \mathrm{Var}[y] \\
&= y_\star^2 - 2y_\star\,\mathbb{E}[\hat{y}] + \mathbb{E}\left[\hat{y}^2\right] + \mathrm{Var}[y] && \text{(linearity of expectation)} \\
&= y_\star^2 - 2y_\star\,\mathbb{E}[\hat{y}] + \mathbb{E}[\hat{y}]^2 + \mathrm{Var}[\hat{y}] + \mathrm{Var}[y] && \text{(expression for variance)}
\end{aligned}
$$

Bias and Variance

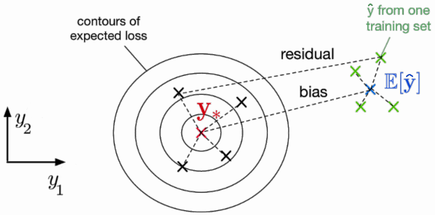
Expected Test Error

Assume model (p_w) trained on $\mathcal{D} \sim p_{\mathcal{D}}(\boldsymbol{x}, y)$; compute predictions on $\boldsymbol{x}$.

Predictions generated as $\hat{y} \sim p_w(\hat{y}|\boldsymbol{x})$.

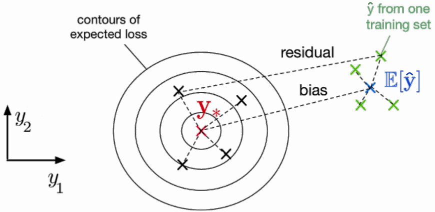
$$
\begin{aligned}
\mathbb{E}\left[(\hat{y} - y)^2\right] &= \mathbb{E}\left[(\hat{y} - y_\star)^2\right] + \mathrm{Var}[y] \\
&= \mathbb{E}\left[y_\star^2 - 2\hat{y}y_\star + \hat{y}^2\right] + \mathrm{Var}[y] \\
&= y_\star^2 - 2y_\star\,\mathbb{E}[\hat{y}] + \mathbb{E}\left[\hat{y}^2\right] + \mathrm{Var}[y] && \text{(linearity of expectation)} \\
&= y_\star^2 - 2y_\star\,\mathbb{E}[\hat{y}] + \mathbb{E}[\hat{y}]^2 + \mathrm{Var}[\hat{y}] + \mathrm{Var}[y] && \text{(expression for variance)} \\
&= \underbrace{(y_\star - \mathbb{E}[\hat{y}])^2}_{\text{bias}} + \underbrace{\mathrm{Var}[\hat{y}]}_{\text{variance}} + \underbrace{\mathrm{Var}[y]}_{\text{Bayes error}}
\end{aligned}
$$

Bias and Variance: Schematic



Figures: Roger Grosse - Generalization

Bias and Variance: Schematic



Generalisation Error:

average squared length of *residual* $\|\hat{y} - y\|^2$

Bias:

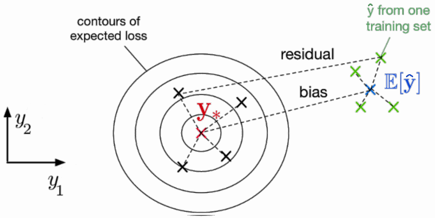
average squared length of *bias* $\|y_\star - \mathbb{E}[\hat{y}]\|^2$

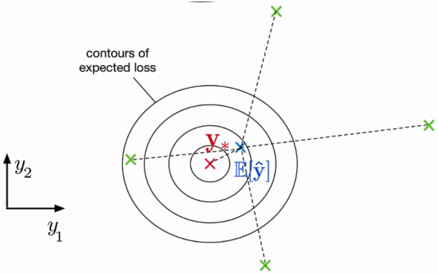
Variance: spread of green $\times$'s

Bayes error: spread of black $\times$'s

Figures: Roger Grosse - Generalization

Bias and Variance: Schematic





Generalisation Error:

average squared length of *residual* $\|\hat{y} - y\|^2$

Bias:

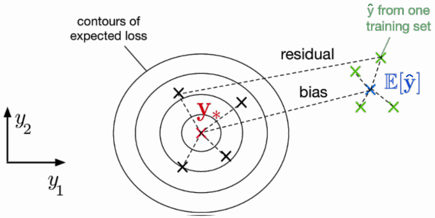
average squared length of *bias* $\|y_\star - \mathbb{E}[\hat{y}]\|^2$

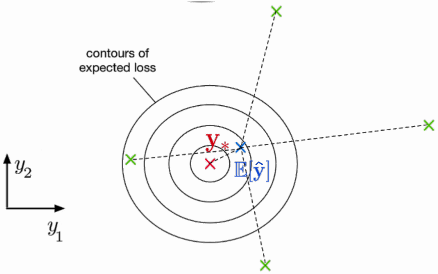
Variance: spread of green $\times$'s

Bayes error: spread of black $\times$'s

Figures: Roger Grosse - Generalization

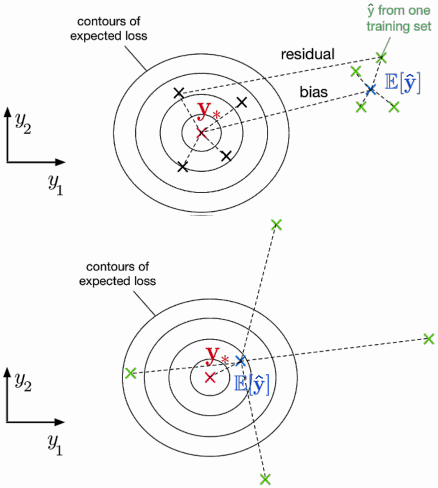
Bias and Variance: Schematic

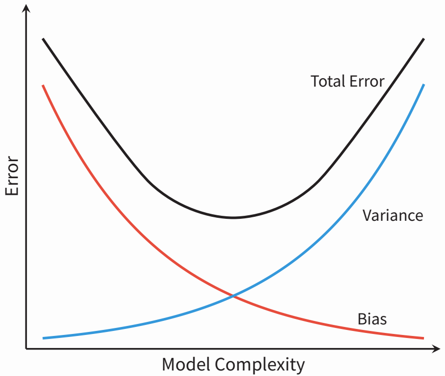




Figures: Roger Grosse - Generalization

Bias and Variance: Schematic





Figures: Roger Grosse - Generalization

Generalisation

Improving Generalisation

Strategies for Improving Generalisation

Primarily concerned with reducing overfitting.

Strategies for Improving Generalisation

Primarily concerned with reducing overfitting.

- Reducing capacity
- Early stopping
- Ensembles
- Regularisation

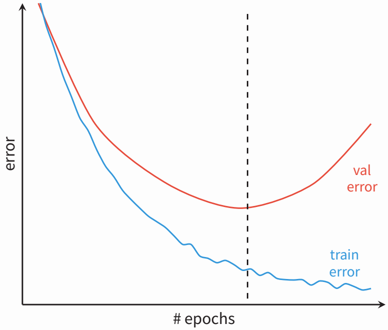
- Model capacity – hyper-parameter
- E.g. degree M of polynomial, # NN layers
- tune on a validation set

Note: Dangerous as can simplify model too much!

Strategies for Improving Generalisation

Primarily concerned with reducing overfitting.

- Reducing capacity
- Early stopping
- Ensembles
- Regularisation



Stop training when generalisation error starts to increase

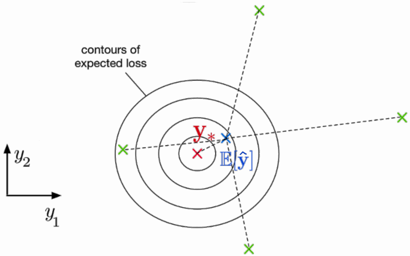
Strategies for Improving Generalisation

Primarily concerned with reducing overfitting.

- Reducing capacity
- Early stopping
- Ensembles
- Regularisation

- Train different models on random subsets of training data …similar to cross validation
- Averaging predictions from multiple models reduces variance

Ensemble: set of trained models whose predictions are combined



Strategies for Improving Generalisation

Primarily concerned with reducing overfitting.

- Reducing capacity
- Early stopping
- Ensembles
- Regularisation

Regularisation

Key Idea

Penalise parameters that may be pathological and unlikely to generalise well, by adding a “complexity” cost.

Regularisation

Key Idea

Penalise parameters that may be pathological and unlikely to generalise well, by adding a "complexity" cost.

$$\mathcal{J}(\boldsymbol{w}) = \underbrace{\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(f(\boldsymbol{x};\boldsymbol{w}),y)}_{\text{train loss}} + \underbrace{\mathcal{R}(\boldsymbol{w})}_{\text{regulariser}}$$

⋆ Requires model parameters to be *continuous*

Regularisation: Linear Regression

Intuition

Penalising polynomials with *large* coefficients, should get less “wiggly” solutions.

Regularisation: Linear Regression

Intuition

Penalising polynomials with *large* coefficients, should get less “wiggly” solutions.

L_2 regularisation

$$\mathcal{R}(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|^2$$

Caution: Don’t shrink the bias term w_0!

Regularisation: Linear Regression

Intuition

Penalising polynomials with *large* coefficients, should get less “wiggly” solutions.

L_2 regularisation

$$\mathcal{R}(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|^2$$

Caution: Don’t shrink the bias term w_0!

Solved w

$$\boldsymbol{w} = (\Phi^\top \Phi + \lambda I)^{-1} \Phi^\top \boldsymbol{y}$$

Regularisation: Linear Regression

Intuition

Penalising polynomials with *large* coefficients, should get less “wiggly” solutions.

L_2 regularisation

$$\mathcal{R}(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|^2$$

Caution: Don’t shrink the bias term w_0!

Solved $\boldsymbol{w}$

$$\boldsymbol{w} = (\Phi^\top \Phi + \lambda I)^{-1} \Phi^\top \boldsymbol{y}$$

Optimisation

$$\nabla_{\boldsymbol{w}} \mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{w}} \mathcal{L}^i + \nabla_{\boldsymbol{w}} \mathcal{R}$$

$$\begin{aligned} \boldsymbol{w} &= \boldsymbol{w} - \eta \left(\nabla_{\boldsymbol{w}} \mathcal{L}^i + \nabla_{\boldsymbol{w}} \mathcal{R} \right) \qquad \text{(SGD)} \\ &= \boldsymbol{w} - \eta \left(\nabla_{\boldsymbol{w}} \mathcal{L}^i + 2\lambda \boldsymbol{w} \right) \\ &= (1 - 2\eta\lambda)\boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}^i \end{aligned}$$

Regularisation: Linear Regression

Intuition

Penalising polynomials with *large* coefficients, should get less "wiggly" solutions.

L_2 regularisation

$$\mathcal{R}(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|^2$$

Caution: Don't shrink the bias term w_0!

Solved $\boldsymbol{w}$

$$\boldsymbol{w} = (\Phi^\top \Phi + \lambda I)^{-1} \Phi^\top \boldsymbol{y}$$

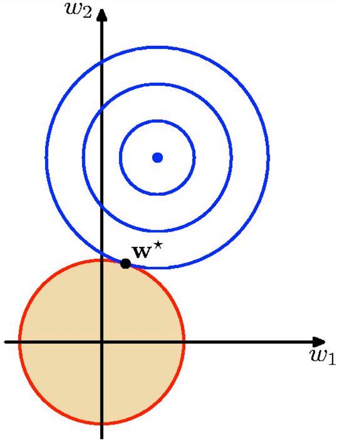
Optimisation

$$\nabla_{\boldsymbol{w}} \mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{w}} \mathcal{L}^i + \nabla_{\boldsymbol{w}} \mathcal{R}$$

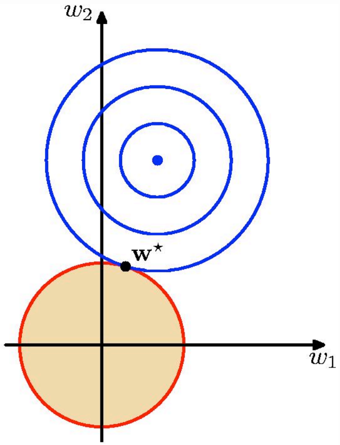
$$\begin{aligned} \boldsymbol{w} &= \boldsymbol{w} - \eta \left(\nabla_{\boldsymbol{w}} \mathcal{L}^i + \nabla_{\boldsymbol{w}} \mathcal{R} \right) \qquad \text{(SGD)} \\ &= \boldsymbol{w} - \eta \left(\nabla_{\boldsymbol{w}} \mathcal{L}^i + 2\lambda \boldsymbol{w} \right) \\ &= (1 - 2\eta\lambda) \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}^i \end{aligned}$$

Each iteration shrinks weights by factor $(1 - 2\eta\lambda)$: *weight decay*

Regularisation: Schematic



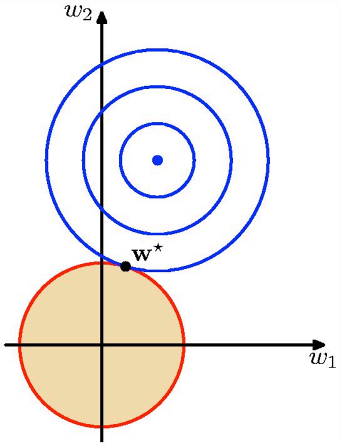
Regularisation: Schematic



- $\mathcal{J}(w)$ is the sum of two parabolic "bowls"

  ...also a parabolic "bowl"

Regularisation: Schematic



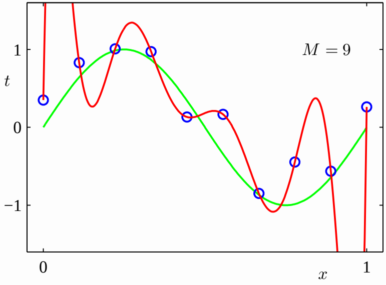
- $\mathcal{J}(w)$ is the sum of two parabolic "bowls"

  ...also a parabolic "bowl"

- Joint minimum on line between minimum of error and origin

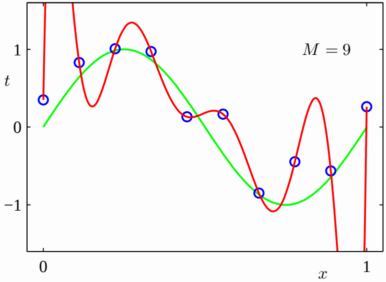
  ...also called *ridge* regression

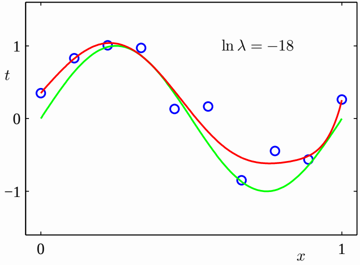
Regularisation: Example



Figures: C. Bishop - PRML

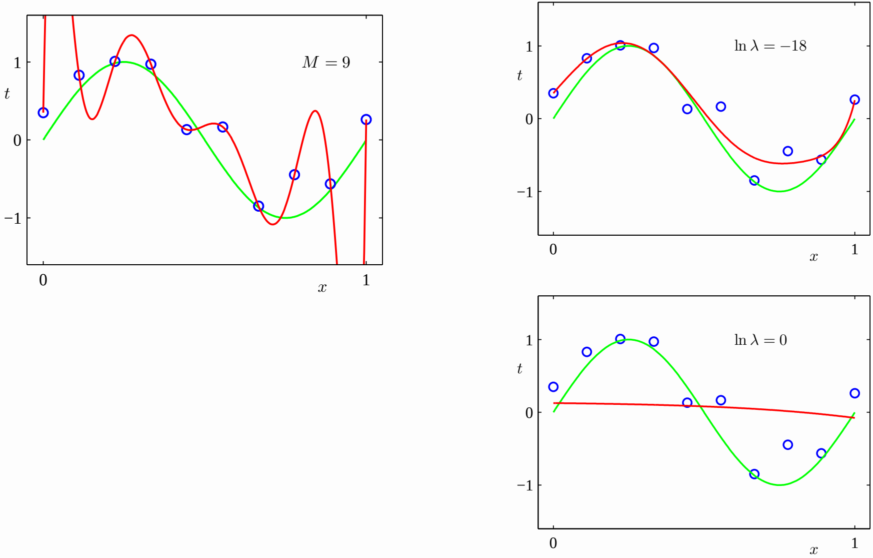
Regularisation: Example





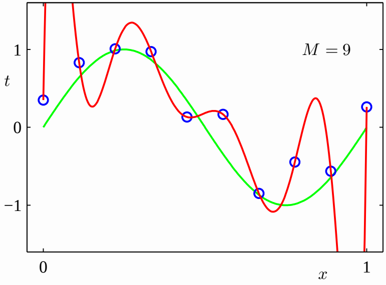
Figures: C. Bishop - PRML

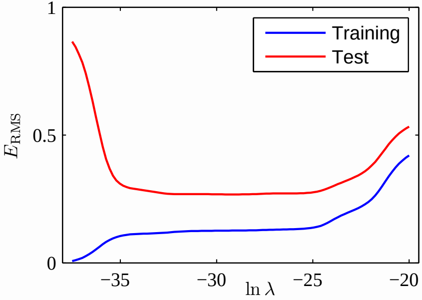
Regularisation: Example



Figures: C. Bishop - PRML

Regularisation: Example





Figures: C. Bishop - PRML

Summary

- What is Generalisation?

Materials credit: Roger Grosse - Generalization

Summary

- What is Generalisation?
    - Model’s ability to fit to *future*, *unseen* data

Materials credit: Roger Grosse - Generalization

Summary

- What is Generalisation?
  - Model's ability to fit to *future*, *unseen* data
  - Overfitting vs. Underfitting

Materials credit: Roger Grosse - Generalization

Summary

- What is Generalisation?
    - Model's ability to fit to *future*, *unseen* data
    - Overfitting vs. Underfitting
    - Train/Test error: # Training examples, # Model parameters

Materials credit: Roger Grosse - Generalization

Summary

- What is Generalisation?
  - Model's ability to fit to *future*, *unseen* data
  - Overfitting vs. Underfitting
  - Train/Test error: # Training examples, # Model parameters
  - Hyper-parameters

Materials credit: Roger Grosse - Generalization

Summary

- What is Generalisation?
  - Model's ability to fit to *future*, *unseen* data
  - Overfitting vs. Underfitting
  - Train/Test error: # Training examples, # Model parameters
  - Hyper-parameters

- How do we characterise/measure it?

Materials credit: Roger Grosse - Generalization

Summary

- What is Generalisation?
    - Model's ability to fit to *future*, *unseen* data
    - Overfitting vs. Underfitting
    - Train/Test error: # Training examples, # Model parameters
    - Hyper-parameters

- How do we characterise/measure it?
    - Test error: Data partitioning with cross validation, val/test splits

Materials credit: Roger Grosse - Generalization

Summary

- What is Generalisation?
  - Model’s ability to fit to *future*, *unseen* data
  - Overfitting vs. Underfitting
  - Train/Test error: # Training examples, # Model parameters
  - Hyper-parameters

- How do we characterise/measure it?
  - Test error: Data partitioning with cross validation, val/test splits
  - Bias vs. Variance: relation to overfitting / underfitting

Materials credit: Roger Grosse - Generalization

Summary

- What is Generalisation?
  - Model's ability to fit to *future*, *unseen* data
  - Overfitting vs. Underfitting
  - Train/Test error: # Training examples, # Model parameters
  - Hyper-parameters

- How do we characterise/measure it?
  - Test error: Data partitioning with cross validation, val/test splits
  - Bias vs. Variance: relation to overfitting / underfitting

- What can we do to improve it?

Materials credit: Roger Grosse - Generalization

Summary

- What is Generalisation?
  - Model's ability to fit to *future*, *unseen* data
  - Overfitting vs. Underfitting
  - Train/Test error: # Training examples, # Model parameters
  - Hyper-parameters

- How do we characterise/measure it?
  - Test error: Data partitioning with cross validation, val/test splits
  - Bias vs. Variance: relation to overfitting / underfitting

- What can we do to improve it?
  - Multiple options: reduce capacity, early stopping, ensemble, regularisation

Materials credit: Roger Grosse - Generalization

Summary

- What is Generalisation?
  - Model's ability to fit to *future*, *unseen* data
  - Overfitting vs. Underfitting
  - Train/Test error: # Training examples, # Model parameters
  - Hyper-parameters

- How do we characterise/measure it?
  - Test error: Data partitioning with cross validation, val/test splits
  - Bias vs. Variance: relation to overfitting / underfitting

- What can we do to improve it?
  - Multiple options: reduce capacity, early stopping, ensemble, regularisation
  - Regularisation: L_2 for linear regression—solution, optimisation

Materials credit: Roger Grosse - Generalization