Australian National University

COMP4650/6490 Document Analysis

Machine Learning Basics - Part II

ANU School of Computing

Administrative matters

- Assignment 1
  - Due: 5pm Wednesday 16 August, AEST (UTC +10)
- Assignment 2
  - Will be released later this week

Australian National University

Outline

- Data splitting
  - Generalisation
  - Validation set & test set
  - Data splitting for cross validation
- Standardisation
  - Standardising features
  - Standardising sparse vectors
- Regularisation
  - Overfitting
  - Types of regularisation
- Hyper-parameter tuning
  - Grid search, random search, Bayesian hyper-parameter optimisation
  - Practical advice

Australian National University

Outline

- Data splitting
  - Generalisation
  - Validation set & test set
  - Data splitting for cross validation
- Standardisation
  - Standardising features
  - Standardising sparse vectors
- Regularisation
  - Overfitting
  - Types of regularisation
- Hyper-parameter tuning
  - Grid search, random search, Bayesian hyper-parameter optimisation
  - Practical advice

Data Splitting

Generalisation

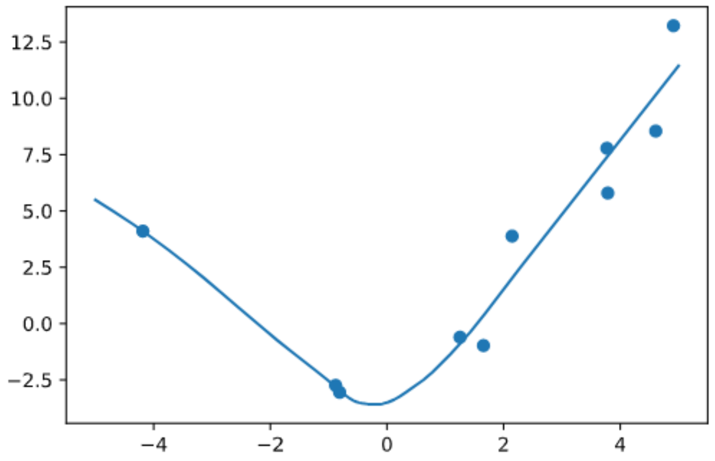
- We want to know how good our model is at making predictions for *new*, *unseen* data points.
- Training error
  - We can use the loss (or another error metric) on training data to measure how well predictions match targets
  - But even if the loss is low we cannot be sure that the model will work well on *new* data points
  - The model could just be *memorising* the training data (i.e. overfitting)
- Generalisation error (i.e. test error)
  - The expected error on a *new*, *unseen* data point
  - This is typically estimated by the performance on a *test dataset* not used to train the model

Australian National University

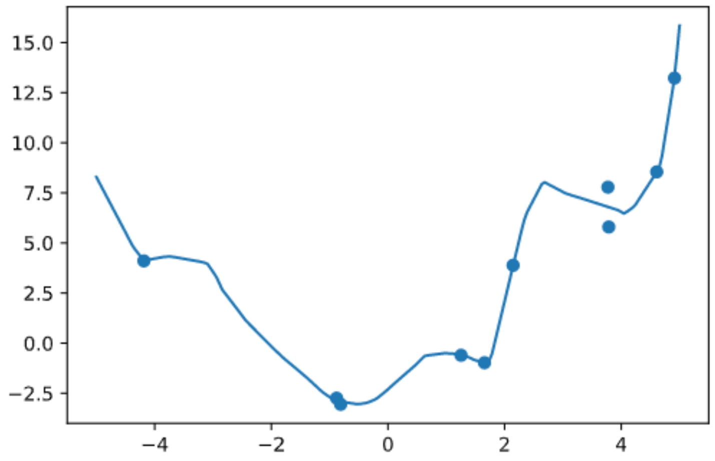
Data Splitting

Generalisation

Good model (Regression)



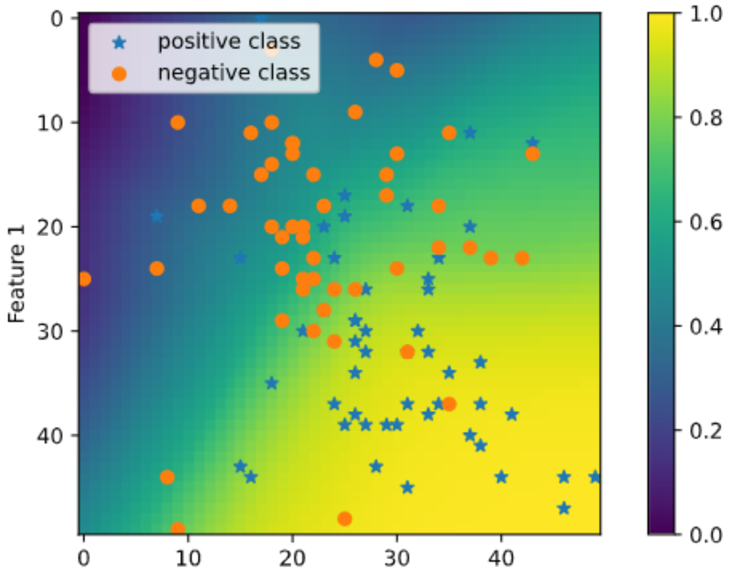
Overfitting (Regression)



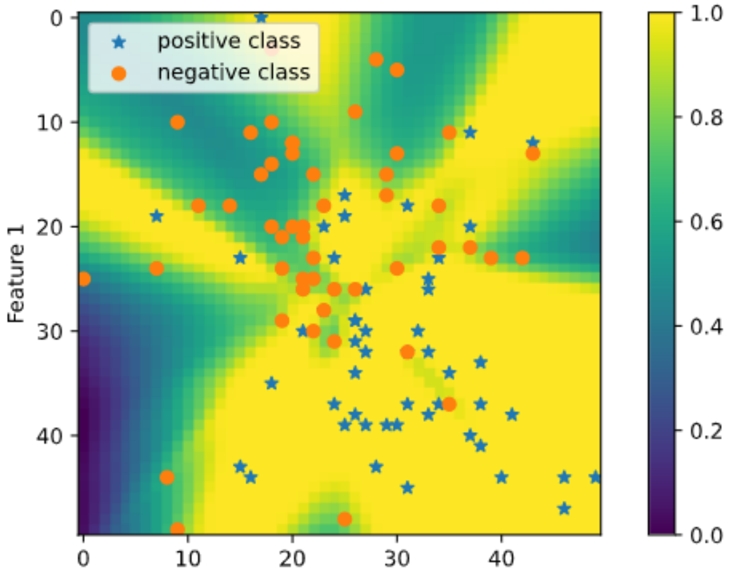
Data Splitting

Generalisation

Good model (Classification)



Overfitting (Classification)



NOTE

- We cannot know if a model has overfit just by looking at the training data or decision boundaries, we must consider the performance on new data points.
- In the above example, the data is synthetic and generated from two Gaussian blobs so we can say that the figure on the right has overfit.

Data Splitting

Test set

- It is crucial that we reserve some of our data for estimating the generalisation performance, this subset is called the test set
- The model is NOT trained on test set
- After training is finished, the model's performance on the *test set* will give us a good idea of how well it can handle new data.

Data Splitting

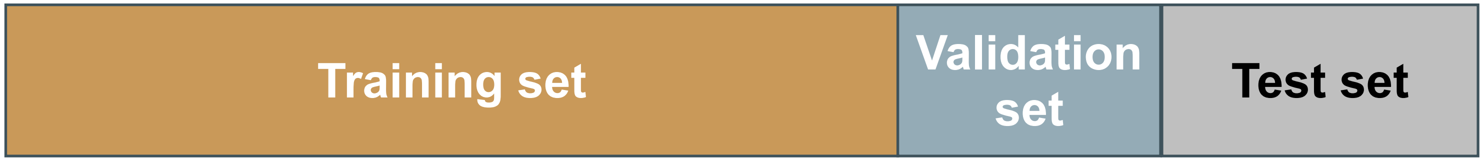
Validation set

- Model selection
  - We often want to compare different models (e.g. models with different hyper-parameters) to choose the best one
  - We want to compare their performance on data not used for training
- The best model is NOT selected using the test set
  - If we choose the model with the best performance on the test set, then its test set performance is no longer a good indication of its performance on new data.
- Validation set
  - The subset of data reserved for model selection (e.g. tuning the hyper-parameters)
  - Also called the development set (or dev set)

Data Splitting

Fixed data splitting

- It is common to split the data into three mutually exclusive sets: Training set, validation set, and test set
  - Training set is used to train models
  - Validation set is used to compare models during development
  - Testing set is used to evaluate models
  - Typical split ratio is 70:10:20
- Data points are often randomly assigned to training, validation, and test sets, but you might want to:
  - Split by Time (e.g. if publication labels are available)
  - Split by Document (e.g. if classifying sentences)
  - Stratify by Class (keep class ratios the same between splits)



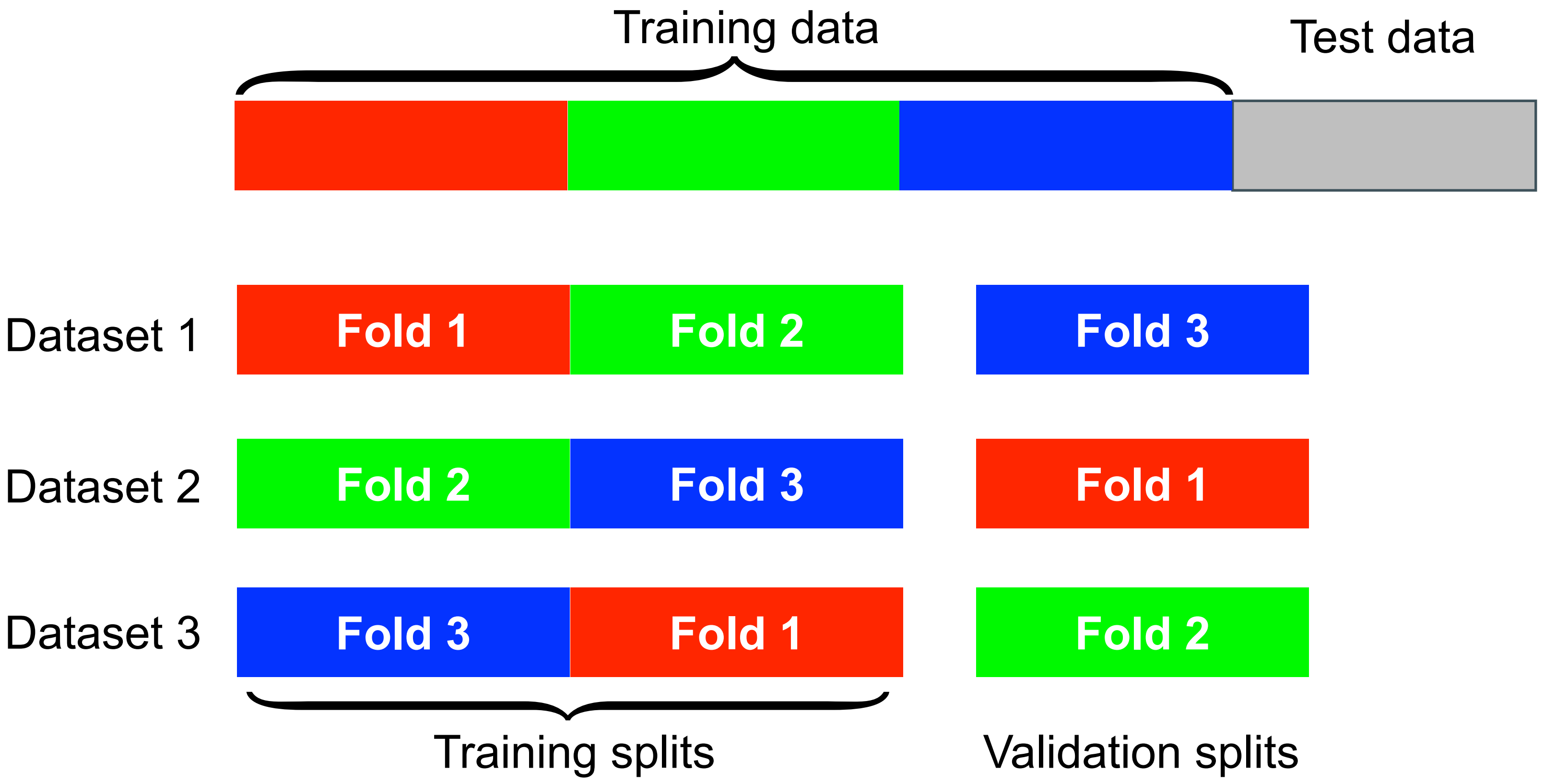
Data splitting for cross validation

- K-fold cross validation for tuning hyper-parameters
  - Split training data into K folds (i.e. subsets)
  - Train a model using K-1 folds, use the remaining fold to evaluate the performance of the trained model
  - Repeat K times to train K models
  - Report the average performance (of the K models)
  - The above steps are repeated for each configuration of hyper-parameters, then choose the configuration with the best average performance
- Better than fixed data splits if limited data and/or model is fast to train and evaluate
  - Do not need to hold a validation set for model selection
  - But can be computationally expensive
  - Rarely used for deep learning

Data Splitting

Data splitting for cross validation

Example of 3-fold cross validation



Australian National University

Data Splitting

Batches

- A batch of examples is a (small) subset of the dataset
- Many machine learning models are trained on batches of examples
- Batches are usually sampled without replacement from the training data. When the training data is empty, all points are added back in and the process is repeated.
- Each iteration through the entire dataset is called one **epoch** of training

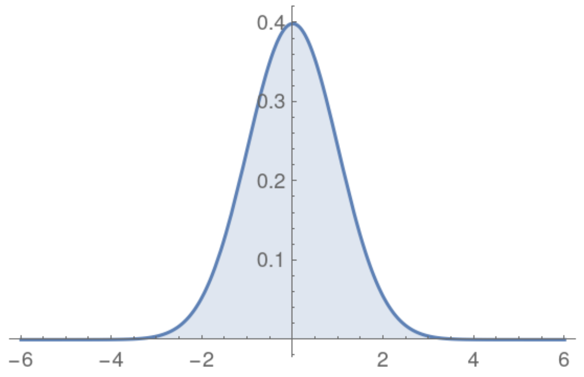
Outline

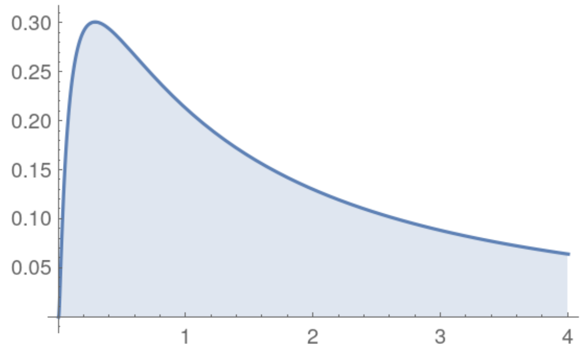
- Data splitting
  - Generalisation
  - Validation set & test set
  - Data splitting for cross validation
- Standardisation
  - Standardising features
  - Standardising sparse vectors
- Regularisation
  - Overfitting
  - Types of regularisation
- Hyper-parameter tuning
  - Grid search, random search, Bayesian hyper-parameter optimisation
  - Practical advice

Standardisation

Standardising features

- Many models perform badly if the features are not approximately normal with mean 0
- We typically try to make this true by
  - Subtracting the mean of each feature
  - Then dividing by the standard deviation of the feature
- Sometimes you might also do a log transform if your data is very skewed
- Why standardising features
  - Can improve performance significantly
  - Ensures each feature has similar effect on the loss
  - Makes regularisation techniques applied to all parameters equally





Standardisation

Standardising features

Data points	Feature 1	Feature 2	Feature 3	Feature 4
$\boldsymbol{x}^{(1)}$	0.5	1.1	99.1	1000
$\boldsymbol{x}^{(2)}$	3.0	4.5	108.1	4123
$\boldsymbol{x}^{(3)}$	2.1	1.1	99.2	-2201

To standardise feature 1:

$$\mu_1 = \frac{1}{3}(0.5 + 3.0 + 2.1) = 1.867$$

$$\sigma_1 = \sqrt{\frac{1}{3}\left((0.5 - \mu_1)^2 + (3.0 - \mu_1)^2 + (2.1 - \mu_1)^2\right)} = 1.034$$

$$\frac{0.5 - \mu_1}{\sigma_1} = -1.322 \qquad \frac{3.0 - \mu_1}{\sigma_1} = 1.096 \qquad \frac{2.1 - \mu_1}{\sigma_1} = 0.226$$

Standardising sparse vectors

- Subtracting the mean of a sparse vector is usually not a good idea
  - If the mean is non-zero it would make most vectors non-sparse
- Alternatives
  - Just divide by the standard deviation
  - Just divide by another value, e.g. TF max normalisation

Outline

- Data splitting
  - Generalisation
  - Validation set & test set
  - Data splitting for cross validation
- Standardisation
  - Standardising features
  - Standardising sparse vectors
- Regularisation
  - Overfitting
  - Types of regularisation
- Hyper-parameter tuning
  - Grid search, random search, Bayesian hyper-parameter optimisation
  - Practical advice

Regularisation

Why regularisation

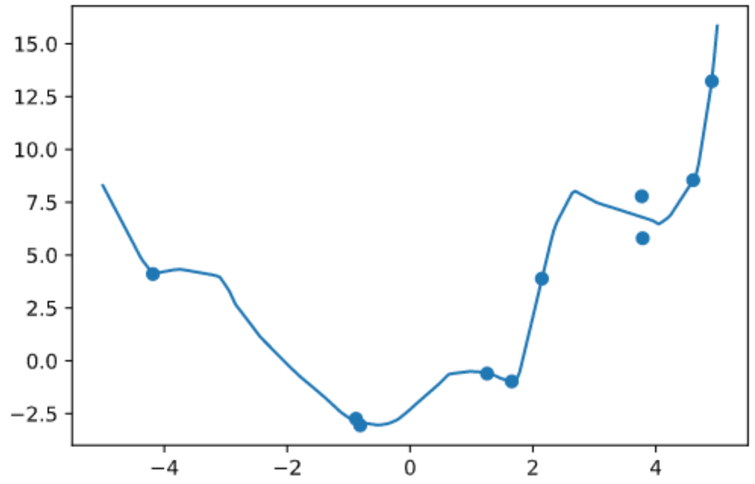
- We want our model to generalise well to examples it was not trained on
- **Occam's razor**: the simplest explanation is usually the best one
- The simplest model that fits the data is usually the one that generalises best
- Bias variance trade-off: There is a trade-off between fitting the training data well and having a simple model

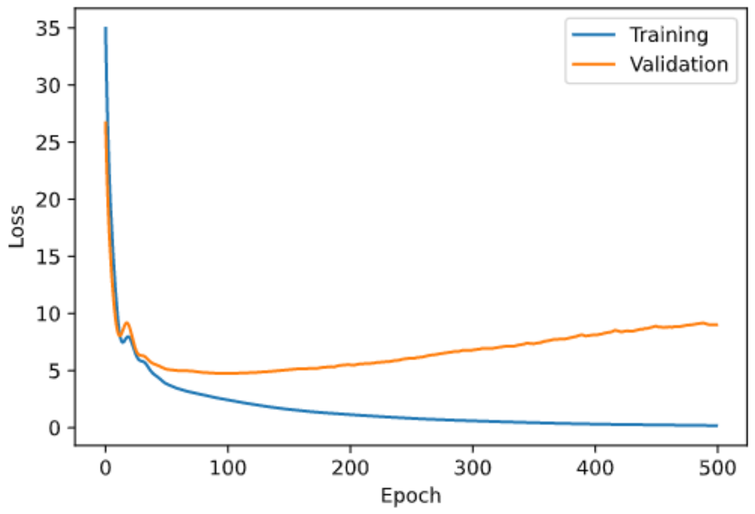
Australian National University

Regularisation

Overfitting

- When a model gets so good at fitting the training data (and the noise) that its performance on the test data suffers





Australian National University

Regularisation

Types of regularisation

- Explicitly constrain parameters as part of the loss / cost function, e.g. L2 regularised MSE for linear regression

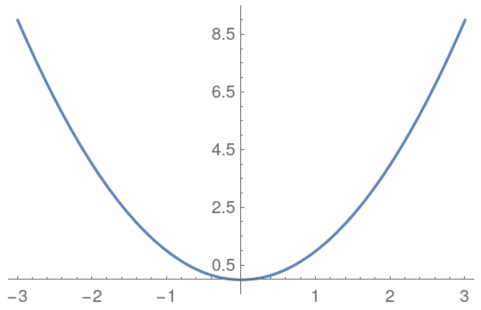
$$J(\mathbf{w}, b) = \boxed{\lambda \mathbf{w}^\top \mathbf{w}} + \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2$$

- Implicit regularisation
  - Stochastic gradient descent (SGD)
  - Early stopping
  - Dropout
  - Batch normalisation
  - Layer normalisation

Regularised loss function

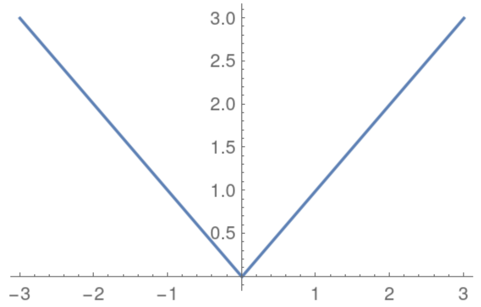
- Most common types:
  - L2 regularisation (Ridge), e.g. for linear regression

$$J(\mathbf{w}, b) = \boxed{\lambda \mathbf{w}^\top \mathbf{w}} + \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2$$



  - L1 regularisation (Lasso), e.g. for linear regression

$$J(\mathbf{w}, b) = \boxed{\lambda\left(\sum_{j=1}^{d} |w_j|\right)} + \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2$$



Regularisation

Regularised loss function

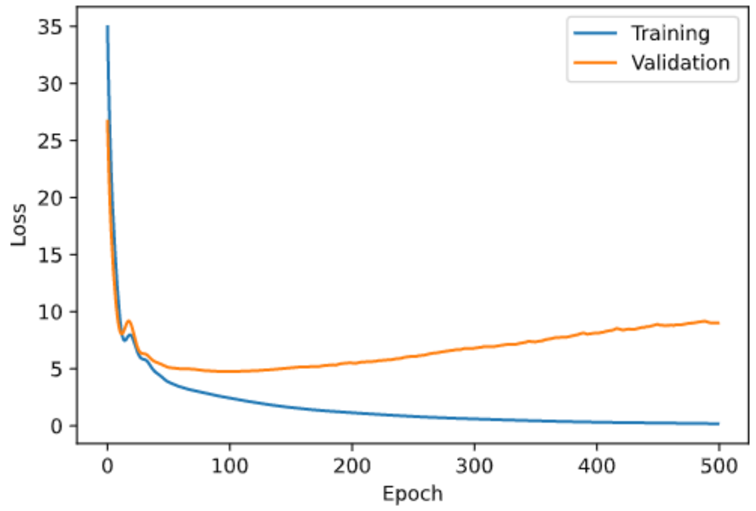
- Typically encourages all the parameters to be closer to 0
  - Limits the space of reasonable parameters
  - Makes linear models have a shallower slope (encodes the prior belief that there is no trend)
  - Leads to sparser models where unimportant features are ignored (L1 regularisation)
- Normally do not regularise the bias

Australian National University

Regularisation

Early stopping

- After each training epoch evaluate the model on the validation set
- If validation performance does not increase, then stop training.



Regularisation

Regularisation is very important

- Particularly important when you have many parameters and/or a small amount of training data
- You might use multiple types of regularisation to get the desired effect

Outline

- Data splitting
  - Generalisation
  - Validation set & test set
  - Data splitting for cross validation
- Standardisation
  - Standardising features
  - Standardising sparse vectors
- Regularisation
  - Overfitting
  - Types of regularisation
- Hyper-parameter tuning
  - Grid search, random search, Bayesian hyper-parameter optimisation
  - Practical advice

Hyper-parameters

- Many machine learning models (e.g. deep neural networks) have many hyper-parameters
  - Architecture (e.g. number of layers and units per layer)
  - Learning rate
  - Regularisation parameters
- How do you choose good settings?

Hyper-param. Tuning

Grid search

- Specify possible values for each hyper-parameter, e.g.
  - learning rate: [0.001, 0.01, 0.1, 1]
  - regularisation parameter: [1, 3, 10, 30, 100]
- Train a model for every possible combination of settings
- Pick the combination with best performance on validation set
- Can take a very long time!
- It is usually good enough to select hyper parameters one at a time
  - Initialise hyper parameters to some reasonable defaults
  - Select one of them and tweak it to be optimal with all others held constant
  - Optimise the next hyper parameter, and so on.

Hyper-param. Tuning

Random search

- It is often more efficient to just randomly search the space of hyper-parameters
    (1) Select a random configuration/combination of hyper-parameters within the grid

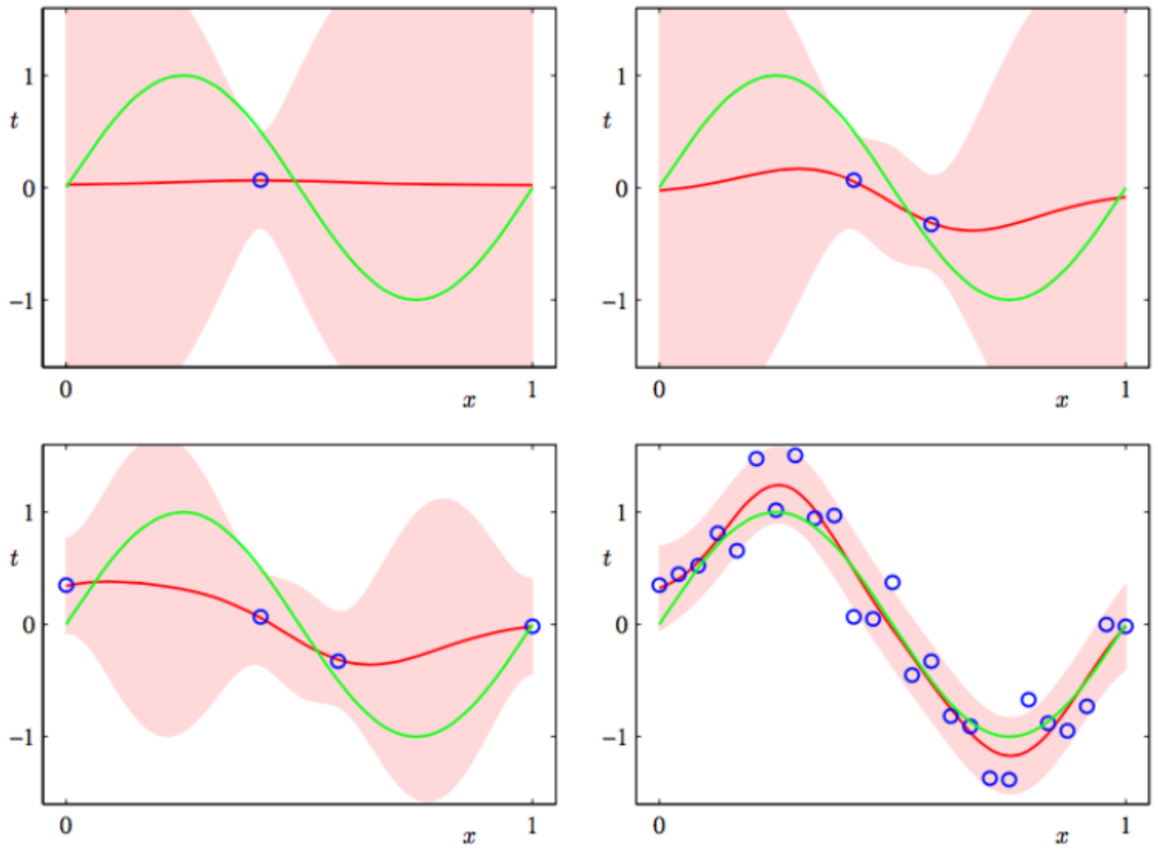
    (2) Train and evaluate the model

    (3) If the computation budget is not exhausted goto step (1)

    (4) Pick the combination of hyper-parameters with best performance on the validation set

Bayesian hyper-parameter optimisation

- The x axis is the hyper-parameter.
- The y axis is the performance or error on the validation set.
- Each point is an evaluation of the model with different value of the hyper-parameter.
- We can guide the search to try hyper-parameters that have a greater probability of being good.



Hyper-param. Tuning

Practical advice

- If your method is fast to train or you have a lot of compute resources, then use one of the presented methods.
- Otherwise, it is better to guide the search by tuning a single parameter at a time and using intuition about the problem.
- The best parameters to start tuning are typically the regularisation parameters and the learning rate.
- For deep neural networks, you should generally make the model as large as possible given your training and test constraints and then use regularisation to prevent overfitting.

Summary

- Data splitting
  - Generalisation
  - Validation set & test set
  - Data splitting for cross validation
- Standardisation
  - Standardising features
  - Standardising sparse vectors
- Regularisation
  - Overfitting
  - Types of regularisation
- Hyper-parameter tuning
  - Grid search, random search, Bayesian hyper-parameter optimisation
  - Practical advice