Fundamentals of Artificial Intelligence

VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA

COMP307/AIML420

ML: kNN and K-fold CV

Dr. Heitor Murilo Gomes
heitor.gomes@vuw.ac.nz
http://www.heitorgomes.com

Outline

- **Evaluation**
  - More on test and train (holdout evaluation)
  - Metrics

- **K-Fold Cross Validation (CV)**
  - Why?
  - Leave-one out

- **K-Nearest Neighbor (kNN)**
  - <u>Classification</u> (Supervised Learning)
  - Basic NN (1-NN)
  - Distance metrics

Evaluation

In the last lecture…

- Discussed different **machine learning tasks and their characteristics:**
  - **Supervised:** predictive
  - **Unsupervised learning:** exploratory

- Attention to methods that you are unlikely to see elsewhere (e.g. Association Rules)

- Introduced the **basic evaluation strategy** (or approach) of splitting the data into two **disjoint** datasets: **train** and **test***

* Holdout method

Evaluation

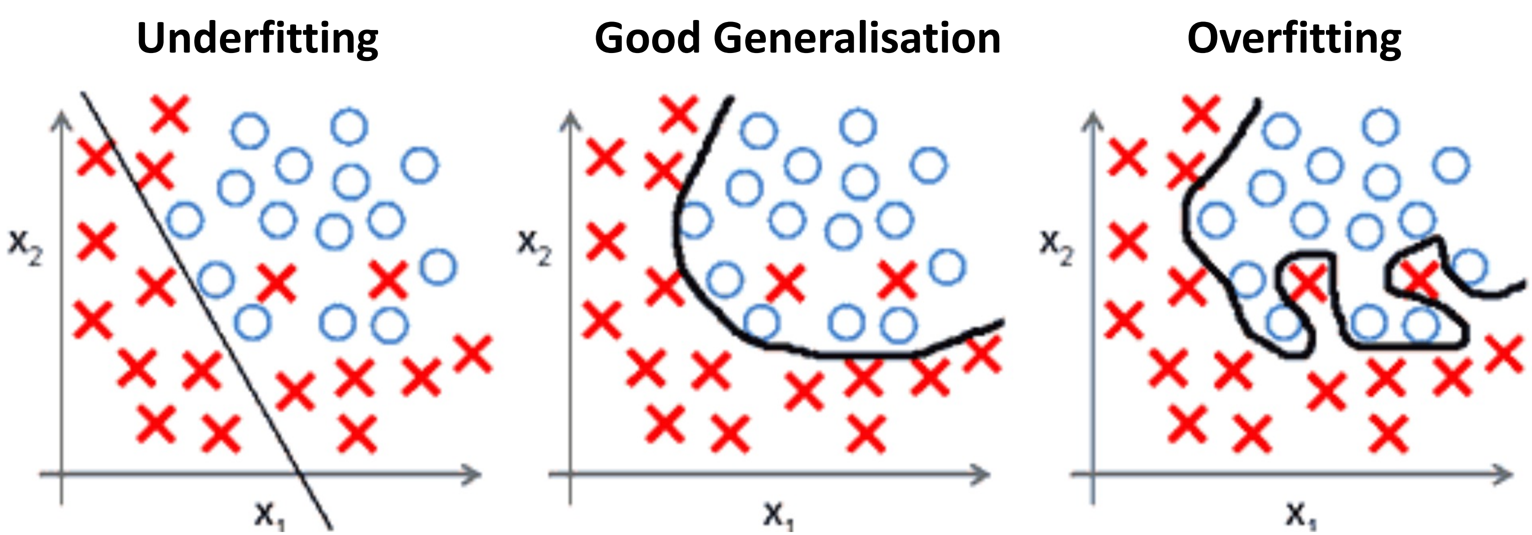
- Why? Assess how well the model works on data that it hasn't seen yet (test data)

- If a model **overfits** the **training** data, it is **likely** to **underperform** on **unseen data**

  - **Unseen data** is where we care that the model performs well
  - **Training data** is the one used to learn the model
  - **Test data** simulates unseen data

Underfit and Overfit

- **Overfitting**: too specific to the training data
- **Underfitting**: too simple and didn't learn essential patterns
- Overall, we want to avoid too complex or too simple models



Evaluation Metrics

- Evaluation strategy and Evaluation Metric
  - Examples:
    - Strategy: **holdout (test and train splits)**
    - Metric: **Accuracy**
  - The **strategy** is how we do it, the **metric** is what we measure

- Different metrics for different tasks and problems
  - Choosing the correct evaluation metric is very important
  - The task at hand (classification or regression?)
  - The problem characteristics (much more instances of one class?)
  - What is important for this problem? (depends on the domain)

Evaluation Metrics

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Confusion matrix

TP: True Positive; FP: False Positive;
FN: False Negative; TN: True Negative

There are several metrics…

- Accuracy
- Precision
- Recall
- F1-score
- Kappa
- …

Evaluation Metrics

Example

- Problem: classify instances as **SPAM (negative)** or **HAM (positive)**; test data contains 100 instances;
- Suppose an algorithm obtains 95% accuracy. Is this good?

Accuracy = (TP+TN) / (TP+TN+FN+FP)

Evaluation Metrics

Example

- Problem: classify instances as **SPAM (negative)** or **HAM (positive)**; test data contains 100 instances;
- Suppose an algorithm obtains 95% accuracy. Is this good?

What if 95 of the instances in the test data are **SPAM?**

Accuracy = (TP+TN) / (TP+TN+FN+FP)

Evaluation Metrics

Example

- Problem: classify instances as **SPAM (negative)** or **HAM (positive)**; test data contains 100 instances;
- Suppose an algorithm obtains 95% accuracy. Is this good?

What if 95 of the instances in the test data are **SPAM?**

It depends

Accuracy = (TP+TN) / (TP+TN+FN+FP)

Evaluation Metrics

- **Accuracy** does not tell us anything about how accurate the model is in predicting one specific class. In our example, we would like to know if the model can correctly predict **HAM** (positive class)

- Given that 95 of 100 test instances are **SPAM** and the accuracy is 95% it is likely that the model is simply always predicting **SPAM** (negative class)

- In these situations, we need to use more suitable metrics. In our example, we could use **recall***

* recall = TP / (TP+FN)

Evaluation Metrics

While a deep dive into selecting metrics for various machine learning problems is beyond the scope of COMP307, remember that choosing an appropriate metric is essential

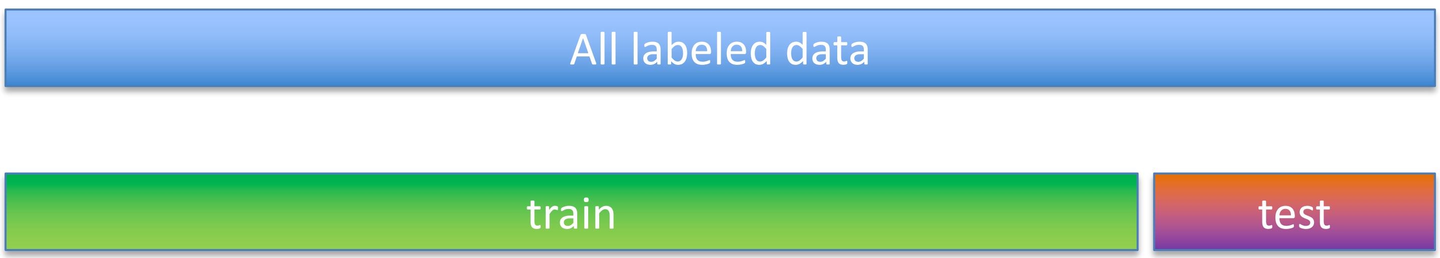
The **choice** of metric can significantly **influence** the **perception of a model's performance**, potentially **overstating** the capabilities of a weak model or **underestimating** the effectiveness of a strong one

Choose wisely!

Evaluation strategy

Holdout evaluation

- To address **overfitting**, we can evaluate using data that was not used for training (i.e. test data)



- The aim is to evaluate the model's ability to **generalize** to new and unseen data by testing it with previously unused data.

Is holdout evaluation enough?

- What are the main **limitations** of the *Holdout Evaluation*?

Is holdout evaluation enough?

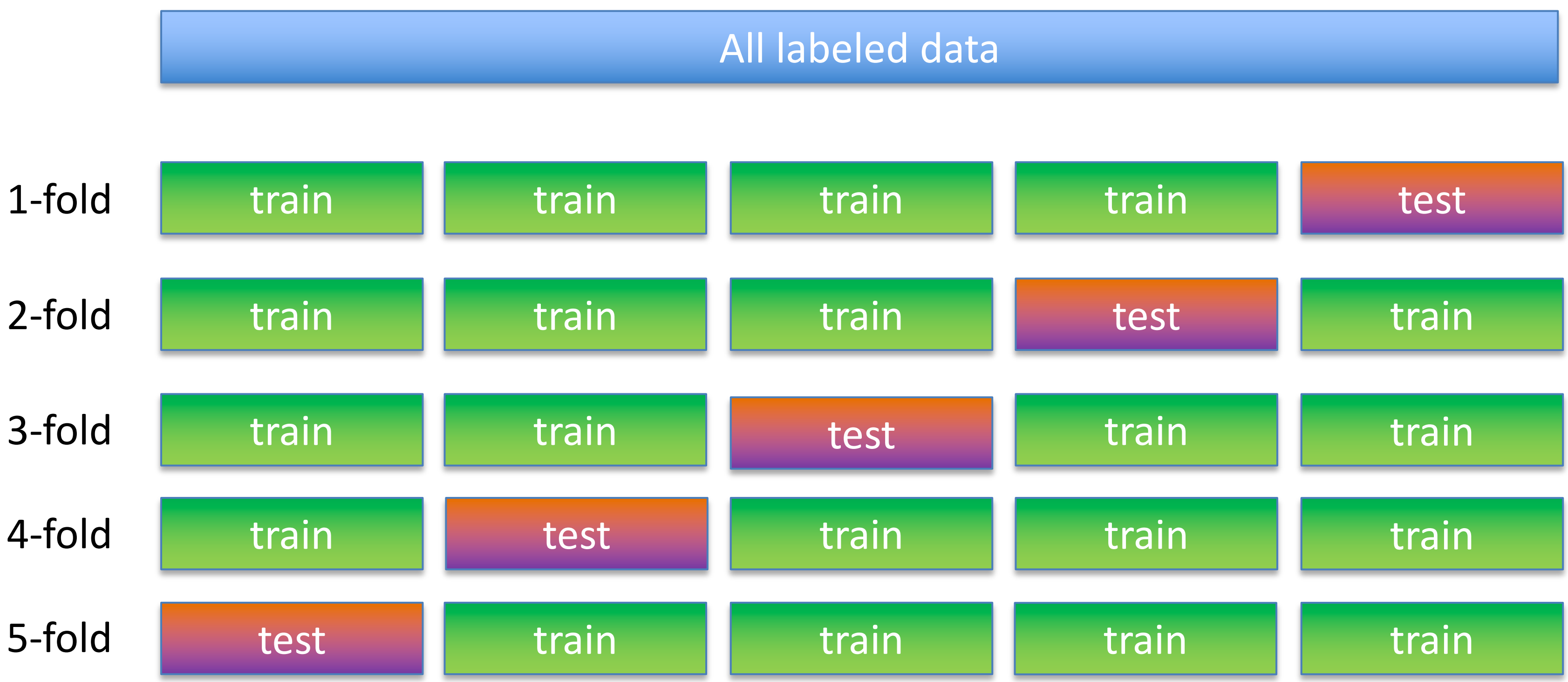
- What are the main **limitations** of the *Holdout Evaluation*?
  - Holdout evaluation **may be biased** if the testing dataset is **not representative of the overall population**
  - Having a **small dataset** results in a smaller test set, which can **lower our confidence** in the experimental results

K-fold cross validation can help us mitigate these limitations!

k-fold cross validation

k-fold cross validation

- Create k train and test sets, e.g. 5-fold CV



k-fold cross validation

- k-fold CV allows us to make **efficient use of the available data** by using each data point for both training and testing, leading to a better estimate of model performance

- **More reliable estimate of performance** especially when the dataset is small or there is a high variance in the data

- **Reduces the risk of overfitting**, as it tests the model on multiple independent test sets, preventing the model from becoming too specific to the training data

k-fold cross validation

- **<u>Leave-one out:</u>** Extreme case of k-fold CV
  - Create as many test sets as there are data points, each test set contains only one instance
  - It can be useful when the dataset is very small
  - Computationally intensive
  - Can lead to overfitting in some cases

- In overall, we use 5-fold CV or 10-fold CV
  - It depends on the computational resources available
  - More folds leads to more confidence on the results

- Stratified k-fold CV
  - take into account the distribution of the class labels

K-Nearest Neighbor

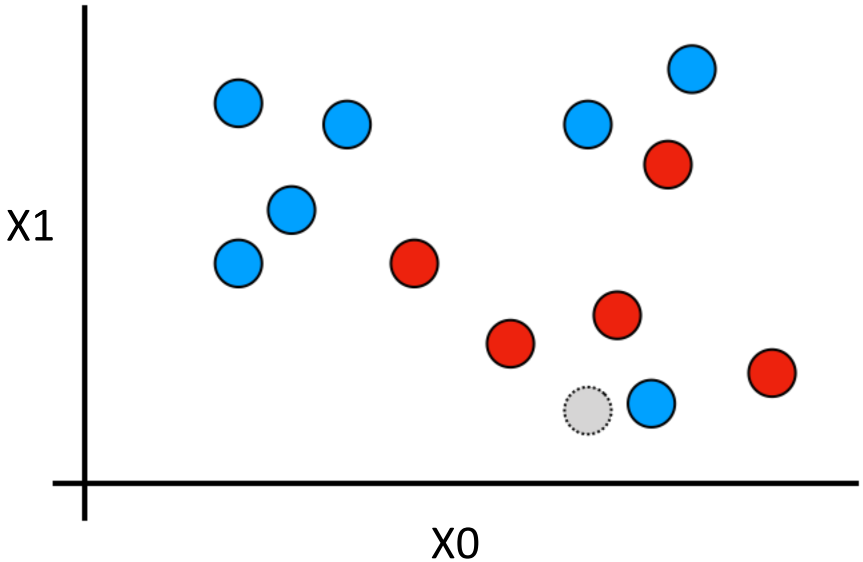
K-Nearest Neighbor

- One of the most intuitive ML algorithms
- Classic example of Lazy Learning (or case-based learning)
- **K** refers to the number of neighbors considered
- Requires a distance metric

The basic algorithm

- **Training:** store all training instances
- **Predicting:** The most **common** class label among the **k** instances **closer** to a **new instance** determines its label

1-NN



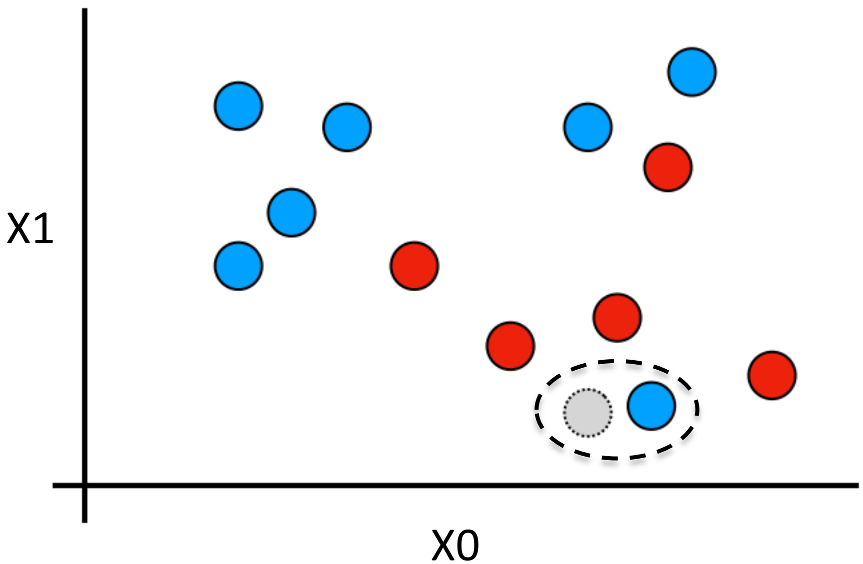
Assume a binary classification problem: blue and red circles

There are two input features X0 and X1

We want to classify the "unknown" gray instance

- **Training:** The training data is shown above (all the red and blue circles)
- We don't know the class for the gray instance.

1-NN



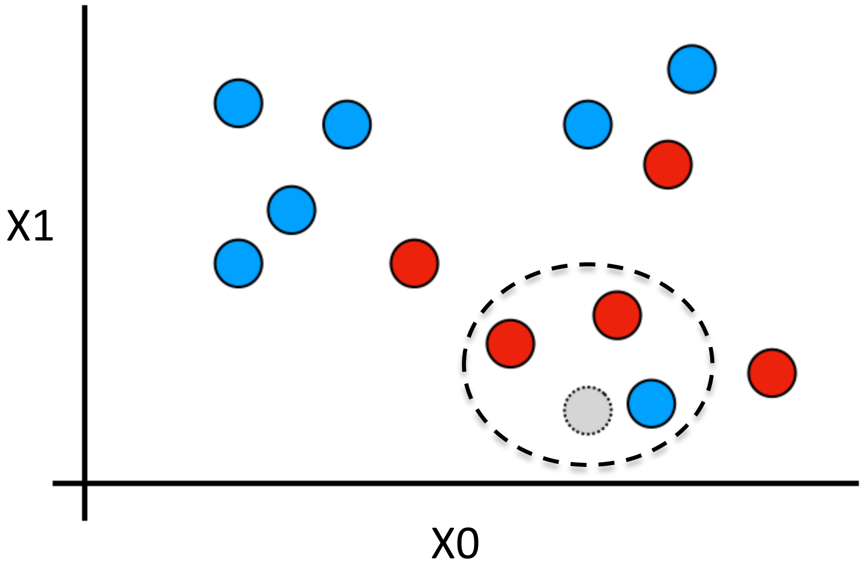
Assume a binary classification problem: blue and red circles

There are two input features X0 and X1

We want to classify the "unknown" gray instance

- **Training:** The training data is shown above (all the red and blue circles)
- We don't know the class for the gray instance.
- **Prediction:** Find the closest neighbor in the training set to the gray inst.
  - The blue instance is the closest neighbor (prediction = blue)

k-NN



Assume a binary classification problem: blue and red circles

There are two input features X0 and X1

We want to classify the "unknown" gray instance

- **Training:** The training data is shown above (all the red and blue circles)
- We don't know the class for the gray instance.
- **Prediction:** Find the closest neighbors in the training set to the gray inst.
  - Assuming **k = 3**, there are 1 blue and 2 red as nearest neighbors (prediction = red)

Distance

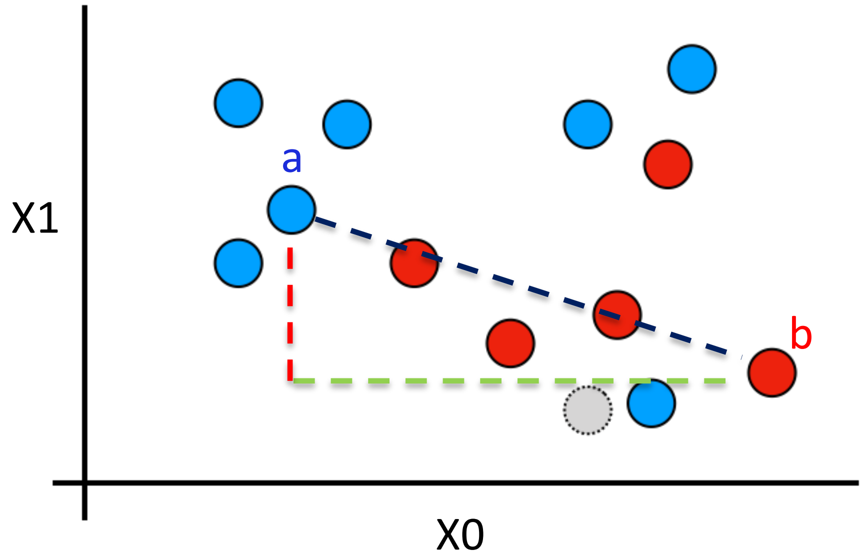
- The notion of "distance" is crucial for kNN (and other ML algorithms)

- Most popular one is the Euclidean distance
  - Given two feature vectors **a** and **b** with **continuous values**

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2}$$

Euclidean Distance Example

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2}$$



$$d(a,b) = \sqrt{(a_{X0} - b_{X0})^2 + (a_{X1} - b_{X1})^2}$$

More on distances

- We often "normalize" the input data, so that values from all features are within the same range (0,1)
- A common normalization technique is Min-Max normalization

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where x is a feature

- We can apply this directly inside the Euclidean distance calculation:

$$d(a,b) = \sqrt{\sum_{i=1}^{n} \frac{(a_i - b_i)^2}{R_i^2}}$$

assuming R_i as the range for feature X_i

i.e. x_{max} - x_{min}

Considerations about kNN

- **Training** and **Predicting**
  - In most ML algorithms training is costly, but predicting is efficient*
  - **What about kNN?**

- **Nominal features** can be difficult to handle

- **High-dimensional data**: distance may lose meaning

- How to choose **k**? Try to **avoid ties**

- The distances can be used to **weight neighbors**

- kNN can be easily adapted for **regression**

Wrap up

- **kNN** is a simple yet powerful **supervised learning** method
- **K-fold CV** is a widely used and accepted **evaluation strategy** in ML
- It is crucial to choose the **evaluation metric** appropriately

Coming up next…

- ML examples (Tutorial this week)
- Next lecture: Decision trees