# COMP30027 Machine Learning
# Interpreting/Visualising Data & Models

Semester 1, 2018

Jeremy Nicholson & Tim Baldwin & Karin Verspoor


THE UNIVERSITY OF MELBOURNE

# Lecture Outline

# Interpreting Models

- While we have talked a lot about different models and their mathematical basis/assumptions, we have largely ignored the question of how to interpret them
- This can be done in two primary ways (with close interactions between them):
  (1) why is it that a given model has *mis*classified an instance in the way it has? (= **error analysis**)
  (2) why is it that a given model has classified an instance in the way it has? (= **model interpretability**)

# What's Error Analysis?

- **Error analysis** is "simply" what it says: (manual) analysis of the sorts of errors that a given model makes ... "simply" because it is all looking over a representative set out of outputs and painstakingly doing the following:
  - identifying different "classes" of error that the system makes (relative to the predicted vs. actual labels)
  - hypothesising as to what has caused the different errors, and testing those hypotheses against the actual data
  - often feeding those hypotheses back into feature/model engineering to see if the model can be improved any (in which case we want to be careful to do error analysis over only the dev data ... why?)
  - quantifying whether (for different classes) it is a question of data quantity/sparsity, or something more fundamental than that

# Mechanics of Error Analysis I

- The starting point for error analysis is usually: (a) a confusion matrix; and (b) a random subsample of misclassified (= "off-diagonal") instances:

| | | *Predicted* | | |
|---|---|---|---|---|
| | | ***A*** | ***B*** | ***C*** |
| ***Actual*** | ***A*** | 10 | 30 | 5 |
| | ***B*** | 5 | 15 | 3 |
| | ***C*** | 2 | 7 | 20 |

- A good starting assumption is that a given "cell" in the confusion matrix forms a single error class, but more often than not, there are different things going on in a given cell ... and multiple cells (esp. across rows/down columns) can also form classes

# Mechanics of Error Analysis II

- Always be sure to test hypotheses against your data — things which intuitively make sense may not actually have any bearing on the classification
- Where possible, use the model to guide the error analysis (in terms of particular traits in the instance that are leading to the misclassification)
- A nice example of error analysis (in the context of question answering): Moldovan et al. [2003]

  `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.441.6742&rep=rep1&type=pdf`

# What's Model Interpretability?

- **Model interpretability** is the means by which we can interpret the basis of a given model classifying an instance the way it does
- In order to talk about model interpretability, we first need to understand what a model is, in terms of its parameters and hyperparameters

# Hyperparameters and Parameters

- The **hyperparameters** of a model are parameters (i.e. tunable "knobs") which define/bias/constrain the learning process
- The **parameters** of a classifier are what is learned when a given learner with a given set of hyperparameters is applied to a particular training dataset, and is then used to classify test instances
- A model (trained with a given set of hyperparameters) can then be interpreted relative to the parameters associated with a given test instance

# Hyperparameters and Parameters

- The **hyperparameters** of a model are parameters (i.e. tunable "knobs") which define/bias/constrain the learning process
- The **parameters** of a classifier are what is learned when a given learner with a given set of hyperparameters is applied to a particular training dataset, and is then used to classify test instances
- A model (trained with a given set of hyperparameters) can then be interpreted relative to the parameters associated with a given test instance
- ... this is perhaps most easily understood with respect to the learners we have seen to date ...

# Nearest Neighbour Classifiers

- Hyperparameters:
- Parameters:
- Interpretation:

# Nearest Neighbour Classifiers

- Hyperparameters:
  - $k$ (neighbourhood size)
  - distance/similarity metric
  - feature weighting/selection ...
- Parameters:

- Interpretation:

# Nearest Neighbour Classifiers

- Hyperparameters:
  - $k$ (neighbourhood size)
  - distance/similarity metric
  - feature weighting/selection ...
- Parameters:
  - there are none, as the model is "lazy" and doesn't abstract away from the training instances in any way
- Interpretation:

# Nearest Neighbour Classifiers

- Hyperparameters:
  - $k$ (neighbourhood size)
  - distance/similarity metric
  - feature weighting/selection ...
- Parameters:
  - there are none, as the model is "lazy" and doesn't abstract away from the training instances in any way
- Interpretation:
  - relative to the training instances that give rise to a given classification, and their geometric distribution

# Nearest Prototype Classifiers

- Hyperparameters:
- Parameters:
- Interpretation:

# Nearest Prototype Classifiers

- Hyperparameters:
  - distance/similarity metric used to calculate the prototype, and distance to each prototype in classification
  - feature weighting/selection ...
- Parameters:

- Interpretation:

# Nearest Prototype Classifiers

- Hyperparameters:
  - distance/similarity metric used to calculate the prototype, and distance to each prototype in classification
  - feature weighting/selection ...
- Parameters:
  - the prototype for each class
  - size $= \mathcal{O}(|C||F|)$
    - $C$ = set of classes
    - $F$ = set of features
- Interpretation:

# Nearest Prototype Classifiers

- Hyperparameters:
  - distance/similarity metric used to calculate the prototype, and distance to each prototype in classification
  - feature weighting/selection ...
- Parameters:
  - the prototype for each class
  - size $= \mathcal{O}(|C||F|)$
    - $C$ = set of classes
    - $F$ = set of features
- Interpretation:
  - relative to the geometric distribution of the prototypes, and distance to each for a given test instance

# Naive Bayes

- Hyperparameters:
- Parameters:
- Interpretation:

# Naive Bayes

- Hyperparameters:
  - the choice of smoothing method
  - optionally the choice of distribution used to model the features (e.g. binomial per feature, or multinomial over all features)
- Parameters:

- Interpretation:

# Naive Bayes

- Hyperparameters:
  - the choice of smoothing method
  - optionally the choice of distribution used to model the features (e.g. binomial per feature, or multinomial over all features)
- Parameters:
  - the class priors and conditional probability for each feature–value–class combination
  - size $= \mathcal{O}(|C| + |C||FV|)$
    - $C$ = set of classes
    - $FV$ = set of feature–value pairs
- Interpretation:

# Naive Bayes

- Hyperparameters:
  - the choice of smoothing method
  - optionally the choice of distribution used to model the features (e.g. binomial per feature, or multinomial over all features)
- Parameters:
  - the class priors and conditional probability for each feature–value–class combination
  - size $= \mathcal{O}(|C| + |C||FV|)$
    - $C$ = set of classes
    - $FV$ = set of feature–value pairs
- Interpretation:
  - usually based on the most *positively*-weighted features associated with a given instance

# Decision Trees

- Hyperparameters:
- Parameters:
- Interpretation:

# Decision Trees

- Hyperparameters:
  - the choice of function used for attribute selection
  - the convergence criterion
- Parameters:

- Interpretation:

# Decision Trees

- Hyperparameters:
  - the choice of function used for attribute selection
  - the convergence criterion
- Parameters:
  - the decision tree itself
  - worst-case size $= \mathcal{O}(\mathcal{V}|Tr|)$
    typical size $= \mathcal{O}(|FV|)$
    - $\mathcal{V}$ = average branching factor
    - $Tr$ = set of training instances
    - $FV$ = set of feature–value pairs
- Interpretation:

# Decision Trees

- Hyperparameters:
  - the choice of function used for attribute selection
  - the convergence criterion
- Parameters:
  - the decision tree itself
  - worst-case size $= \mathcal{O}(\mathcal{V}|Tr|)$
    typical size $= \mathcal{O}(|FV|)$
    - $\mathcal{V}$ = average branching factor
    - $Tr$ = set of training instances
    - $FV$ = set of feature–value pairs
- Interpretation:
  - based directly on the path through the decision tree

# Support Vector Machines

- Hyperparameters:
- Parameters:
- Interpretation:

# Support Vector Machines

- Hyperparameters:
  - the penalty term $C/\epsilon$ for soft-margin SVMs
  - feature value scaling
  - the choice of kernel (and any hyperparameters associated with it)
- Parameters:

- Interpretation:

# Support Vector Machines

- Hyperparameters:
  - the penalty term $C/\epsilon$ for soft-margin SVMs
  - feature value scaling
  - the choice of kernel (and any hyperparameters associated with it)
- Parameters:
  - vector of feature weights + bias term
  - size = $\mathcal{O}(|C||F|)$ (assuming one-vs-rest SVM)
    - $C$ = set of classes
    - $F$ = set of features
- Interpretation:

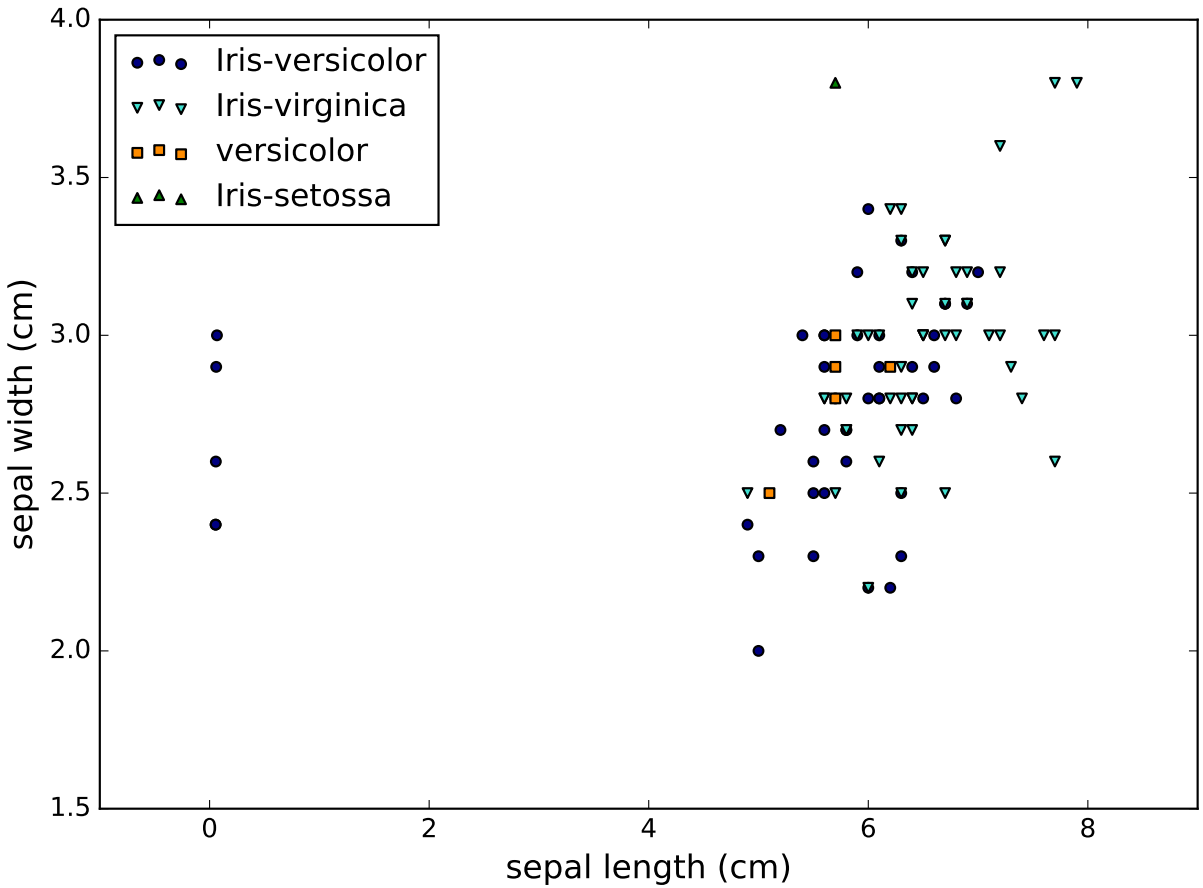
# Support Vector Machines

- Hyperparameters:
  - the penalty term $C/\epsilon$ for soft-margin SVMs
  - feature value scaling
  - the choice of kernel (and any hyperparameters associated with it)
- Parameters:
  - vector of feature weights + bias term
  - size = $\mathcal{O}(|C||F|)$ (assuming one-vs-rest SVM)
    - $C$ = set of classes
    - $F$ = set of features
- Interpretation:
  - the absolute value of the weight associated with each non-zero feature in a given instance provides an indication of its relative importance in classification (modulo kernel projection)

# Lecture Outline

# Visualising Data

- As we saw in the lab back in Week 3, visualising your data can be a valuable way of "getting to know it", in addition to visually detecting any anomalies in the data



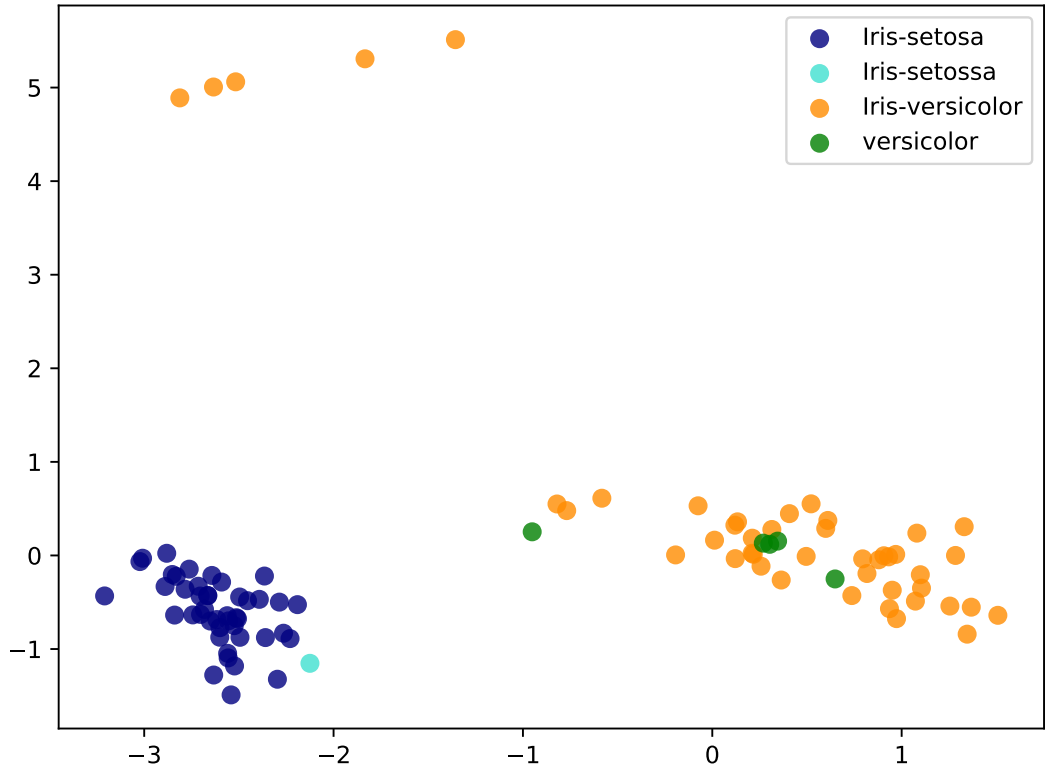
# Dimensionality Reduction

- But what if there are more than 2 attributes (as there usually are)?

  $\Rightarrow$ use **dimensionality reduction** to reduce feature space down to two (or three) dimensions, to be more easily graphable
- Any dimensionality reduction method is going to be lossy, in that it is generally not possible to faithfully reproduce the original data from the reduced version
- Feature selection is one form of dimensionality reduction you have seen already, but ideally we want to be able to reduce the dimensionality in a way which is more faithful to the full feature set

# Principal Component Analysis I

- One popular form of dimensionality reduction is **principal component analysis** ("PCA"), e.g. 2d rendering of Iris:



# Principal Component Analysis II

- *The central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables.* [Jolliffe, 2002]

# Principal Component Analysis III

- Let:
  - $X$ be a set of instances $\boldsymbol{x}$ of dimensionality $p$
  - $\alpha_k$ be a constant-valued vector of size $p$
  - $\alpha'_k \cdot x = \sum_{j=1}^{p} \alpha_{kj} x_j$

  PCA operates as follows:
  - For $i = 1...p$ find $\alpha'_i$ with maximum variance uncorrelated with $\alpha'_1 ... \alpha'_{i-1}$
- In general, at $m \ll p$, most of the variation in $X$ will be accounted for
- PCA is generally performed using an eigenvalue solver (e.g. based on singular value decomposition) ... but the details are beyond the scope of this subject

# Other Commonly-used Dimensionality Reduction Methods

- Kernel PCA [Schölkopf et al., 1997]
- t-distributed stochastic neighbor embedding ("t-SNE") [Van der Maaten and Hinton, 2008]

# Lecture Outline

# Summary

- What is error analysis, and how is it generally carried out?
- What are model hyperparameters and parameters?
- For each of the primary machine learning algorithms we have seen so far, what are the common hyperparameters, how many parameters are there, and how can the model be interpreted?
- What is dimensionality reduction, and (intuitively) how does PCA operate?

# References I

Ian Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2002.

Dan Moldovan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)*, 21(2):133–154, 2003.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588, Lausanne, Switzerland, 1997.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579–2605):85, 2008.