Interpretation & Visualisation

Semester 1, 2021

Ling Luo

Outline

- Interpreting Models
  - Error Analysis
  - Model Interpretability
- Visualising Data
  - Different types of plots
  - Dimensionality reduction

Interpreting Models

- How to interpret models?
- This can be done in two primary ways :
  - why a given model has misclassified an instance in the way it has (= **error analysis**) why it doesn't work ?
  - why a given model has classified an instance in the way it has? (= **model interpretability**) → why it works ?

Error Analysis (1)

- Analysis of the sorts of errors that a given model makes
  - **identifying** different "classes" of error that the system makes (predicted vs. actual labels)

    通过比较，找出不同类型的错误.
  - **hypothesising** as to what has caused the different errors, and testing those hypotheses against the actual data
  - **quantifying** whether (for different classes) it is a question of data quantity/sparsity, or something more fundamental than that

    noise, sparsity, assumption
  - **feeding** those hypotheses **back** into feature/model engineering to see if the model can be improved

Error Analysis (2)

- Starting point: a confusion matrix & a random subsample of misclassified instances (off-diagonal)
- A good starting assumption is that a given "cell" in the confusion matrix forms a single error class

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

太多A被判成B

⇓

分析原因.

assumption 不成立

training 没见过太多A.

feature 不能区分 A&B (feature不够好)

Error Analysis (3)

Tips:

if we only analyse errors on a particular test set. overfit and not generalise.

- It is possible that different things going on in a given cell and multiple cells (e.g. across rows/down columns) can also form a single class of errors
- Always be sure to test hypotheses against your data
- Where possible, use the model assumption to guide the error analysis (in terms of particular traits in the instance that are leading to the misclassification)

Model Interpretability

- Interpret the basis of a given model classifying an instance the way it does
- What is a model?
    - Hyperparameters and parameters

Hyperparameters and Parameters

- **Hyperparameters**: parameters which **define and constrain the learning process**
- **Parameters**: what are **learned** when a given learner with a given set of hyperparameters is applied to a particular training dataset, and are then used to classify test instances eg SVM weight, bias are parameters (support vectors)
- A model trained with a given set of hyperparameters can be interpreted relative to the parameters associated with a given test instance

Hyperparameters and Parameters

sklearn.neighbors.KNeighborsClassifier

class `sklearn.neighbors.KNeighborsClassifier`(*n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs*) [source]

Classifier implementing the k-nearest neighbors vote.

Read more in the User Guide.

Hyperparameters for the model

Parameters:

n_neighbors : *int, optional (default = 5)*
Number of neighbors to use by default for `kneighbors` queries.

weights : *str or callable, optional (default = 'uniform')*
weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

algorithm : *{'auto', 'ball_tree', 'kd_tree', 'brute'}, optional*
Algorithm used to compute the nearest neighbors:

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

KNN Classifiers

0-R O(1).

- Hyperparameters
  - neighbourhood size K
  - distance/similarity metric
  - weighting strategy

size = data itself

- Parameters
  - 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 distribution in the feature space.

如果top k 的距离都很远，不满足 assumption

Nearest Prototype Classifiers

- Hyperparameters
  - distance/similarity metric
  - feature weighting → normalise feature to avoid feature domination
- Parameters
  - prototype for each class
  - size: $\mathcal{O}(|C||F|)$ |C| classes → |C| prototype → |C||F|
    |F| feature

    C = set of classes, F = set of features
- Interpretation
  - relative to the distribution of the prototypes in the space, and distance to each for a given test instance

Naïve Bayes

- Hyperparameters
  - smoothing method
  - optionally the choice of distribution used to model the features (e.g. Gaussian for continuous features)
- Parameters
  - class priors and conditional probability for each feature-value-class combination
  - size: $\mathcal{O}(\underbrace{|C| + |C||FV|}) \checkmark$
    C = set of classes, FV = set of feature-value pairs

$|C|$ prior

$|C|$ $\quad |C| \cdot \sum_{i=1}^{k} fv_i$

$p(c_i) + \prod_{i=1}^{N} P(x_i | c_i)$

fv_i each feature value.

$\sum_{i=1}^{k} fv_i = |FV|$,

k feature

- Interpretation
  - usually based on the most *positively-weighted* features associated with a given instance

Decision Trees

- Hyperparameters
  - attribute selection: e.g. information gain, gain ratio
  - stopping criterion

    max_depth to control overfitting
    max_number of leaf nodes.
- Parameters
  - decision tree itself
  - typical size: $\mathcal{O}(|FV|)$
    FV = set of feature-value pairs
- Interpretation
  - based on the path through the decision tree

SVM

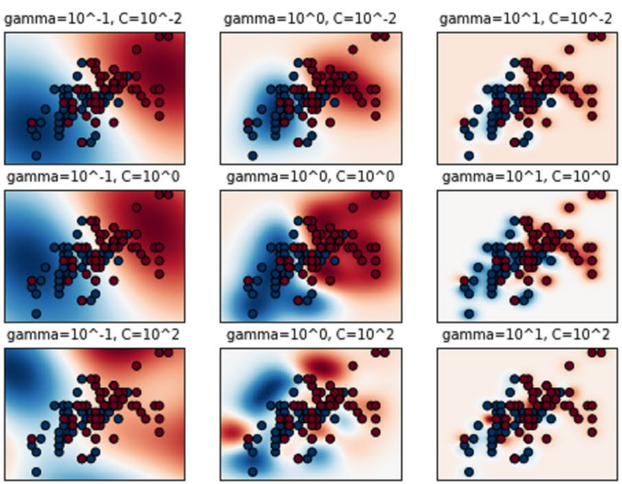
- Hyperparameters
  - penalty term C for soft-margin SVM
  - choice of kernel and any hyperparameters associated with it
  - how to deal with multi-class problem ovo ovr
- Parameters
  - hyperplane: normal vector + bias   bias are constant, just ignore mainly consider normal vector $O(F)$.
  - size: $\mathcal{O}(|C||F|)$

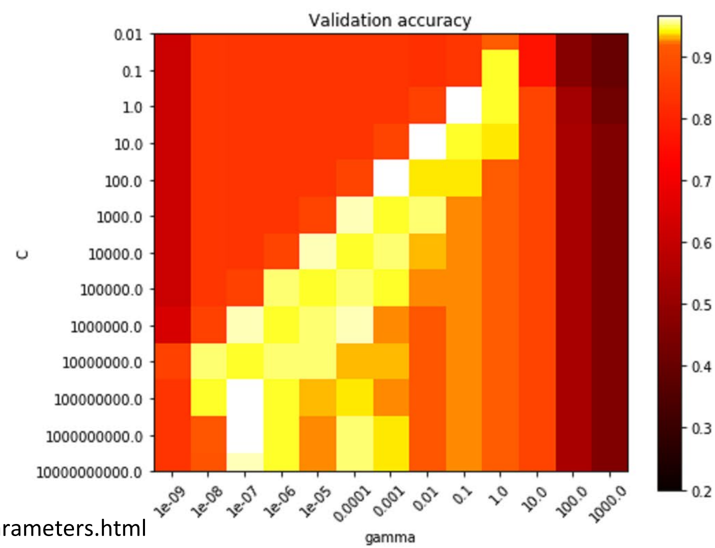
    C = set of classes, F = set of features, assuming one-vs-all SVM

    one-vs-one $O(C^2F)$.
- Interpretation   weight ~ importance
  - 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

Tune Hyperparameters

- Understand the meaning of a hyperparameter
- Try different settings (manual tuning, grid search etc.)
- Compare the performance on validation set.



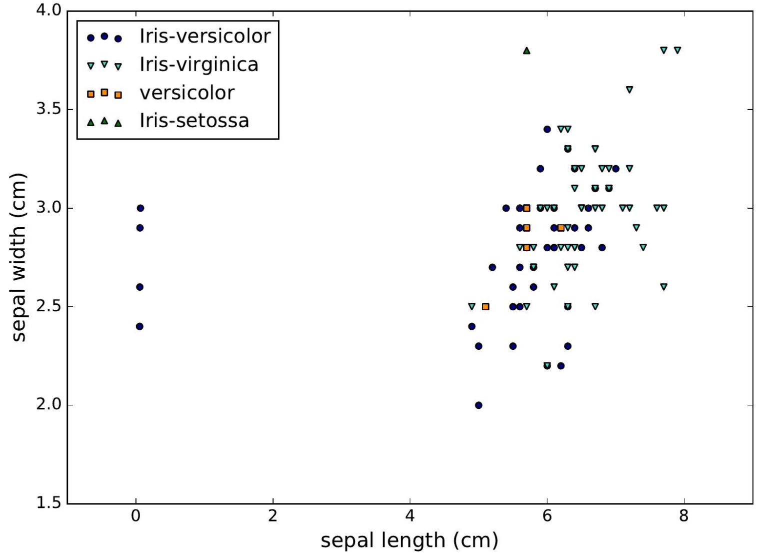


https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Visualising Data

Visualising Data

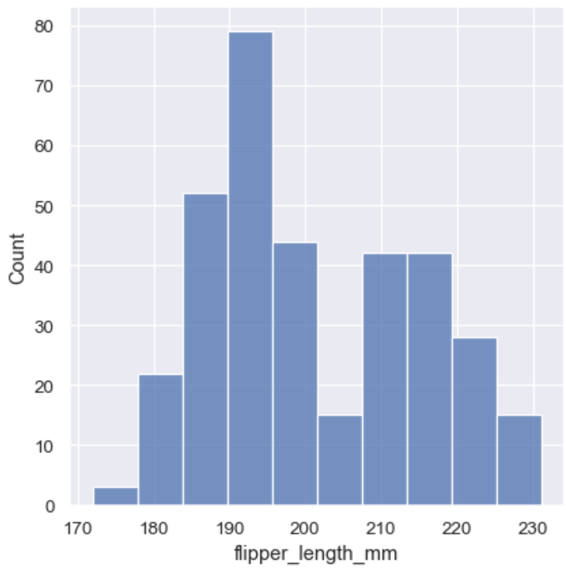
- Visualising your data can be a valuable way of getting to know it
- Example: visually detect any anomalies in the data

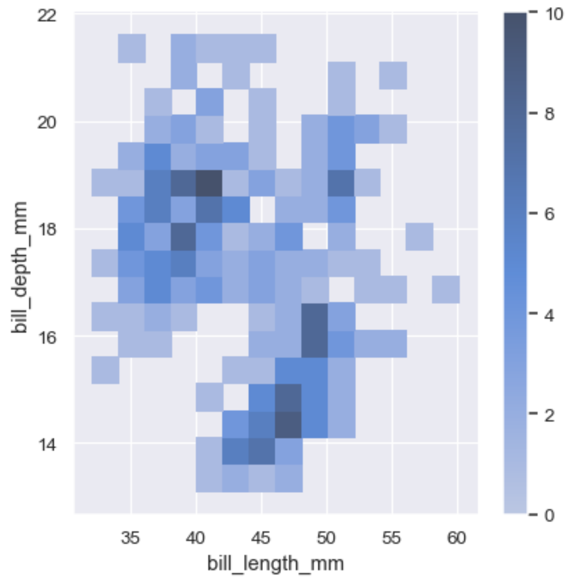


scatter plot
⇓
overview of dataset
linear seperable

More Types of Plots

- Check the distribution of data

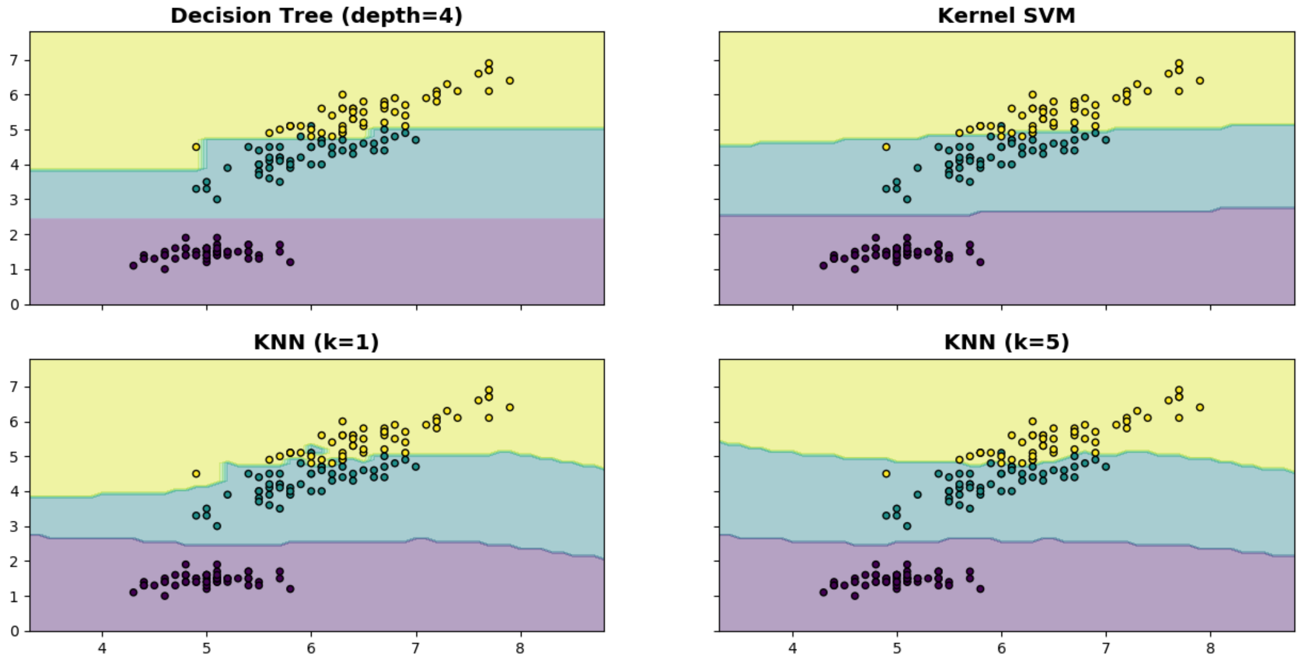




source: https://seaborn.pydata.org/tutorial/distributions.html

More Types of Plots

- Check decision boundary



adapted from: https://scikit-learn.org/stable/auto_examples/ensemble/plot_voting_decision_regions.html

Dimensionality Reduction

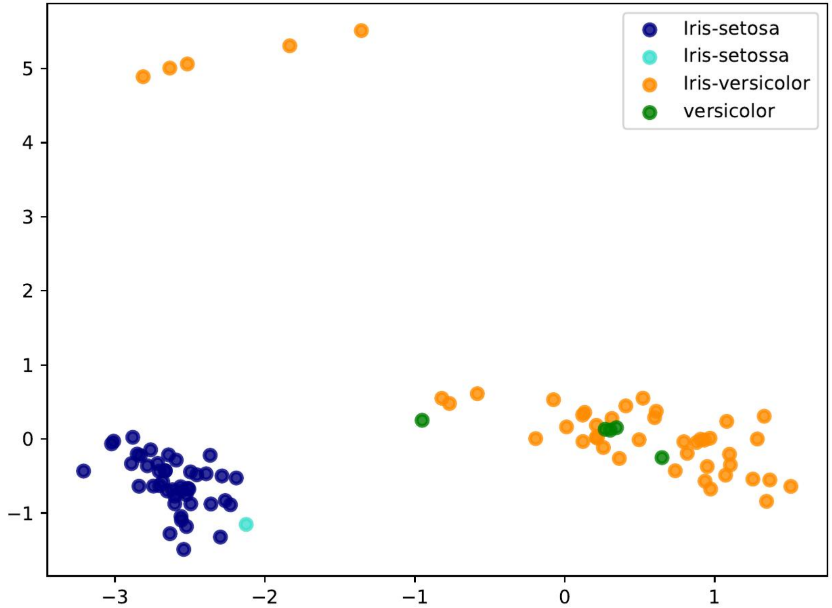
- What if there are more than 3 attributes?
  → reduce feature space down to 2 or 3 dimensions
- Remove some features?
  combination of old features to new feature
  Feature selection vs. dimensionality reduction
  subset of features
- Any dimensionality reduction method is going to be lossy, and it is generally not possible to faithfully reproduce the original data from the reduced version

cons: lose information

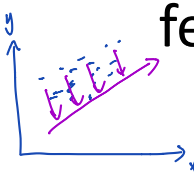
pros: fast

Principal Component Analysis

- A popular form of dimensionality reduction
- Example: 2D rendering of Iris



Principal Component Analysis



- Central idea: the principle components (new features)
  - are linear combinations of the original features
  - are orthogonal to each other
  - capture the maximum amount of variation in the data

2nd put all component orthogonal to the 1st one, select the max variation

SVD

- 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

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 are dimensionality reduction and PCA?

References

- An 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

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