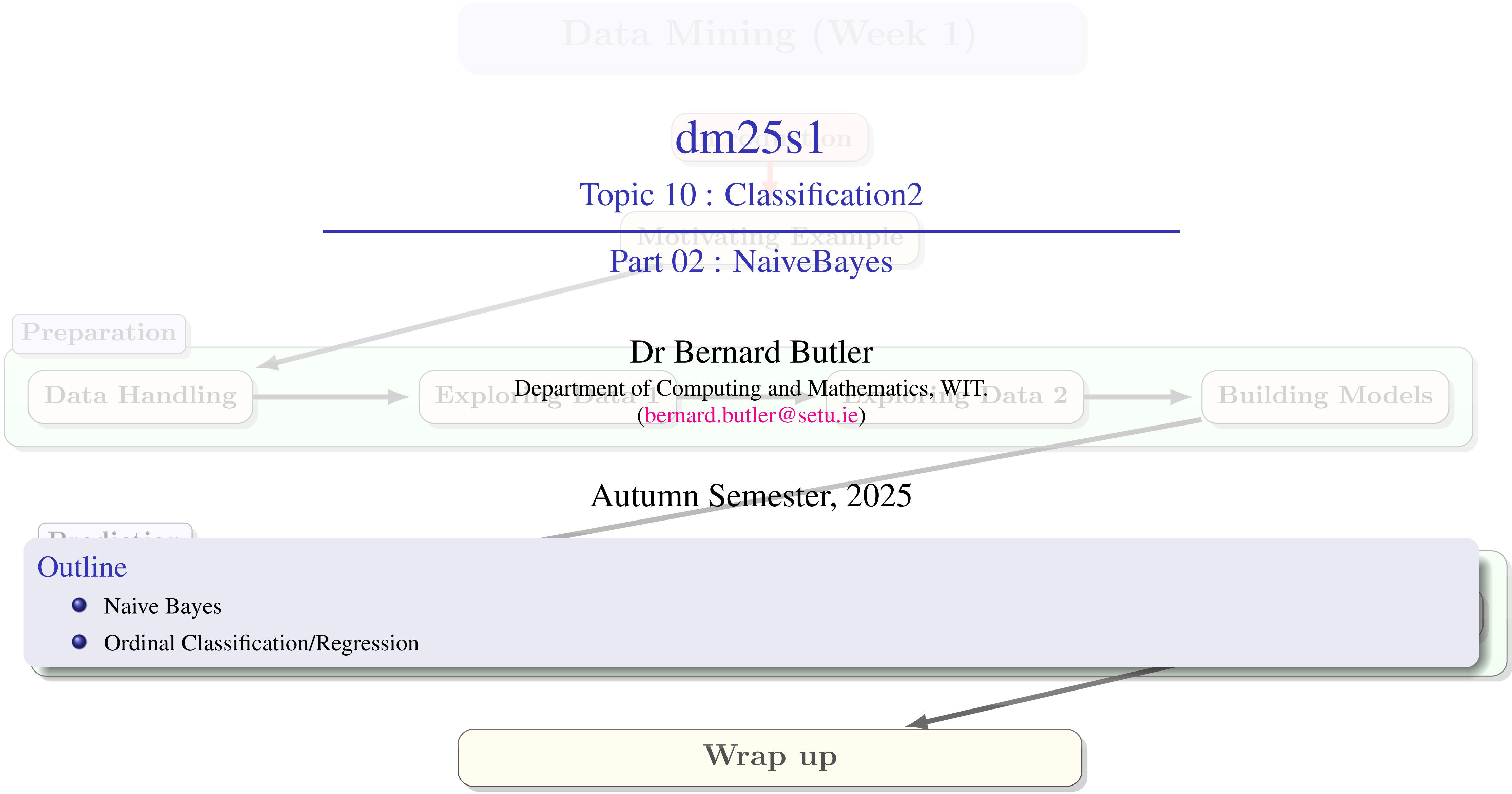


dm25s1

Topic 10 : Classification2

Part 02 : NaiveBayes

Dr Bernard Butler

Department of Computing and Mathematics, WIT.

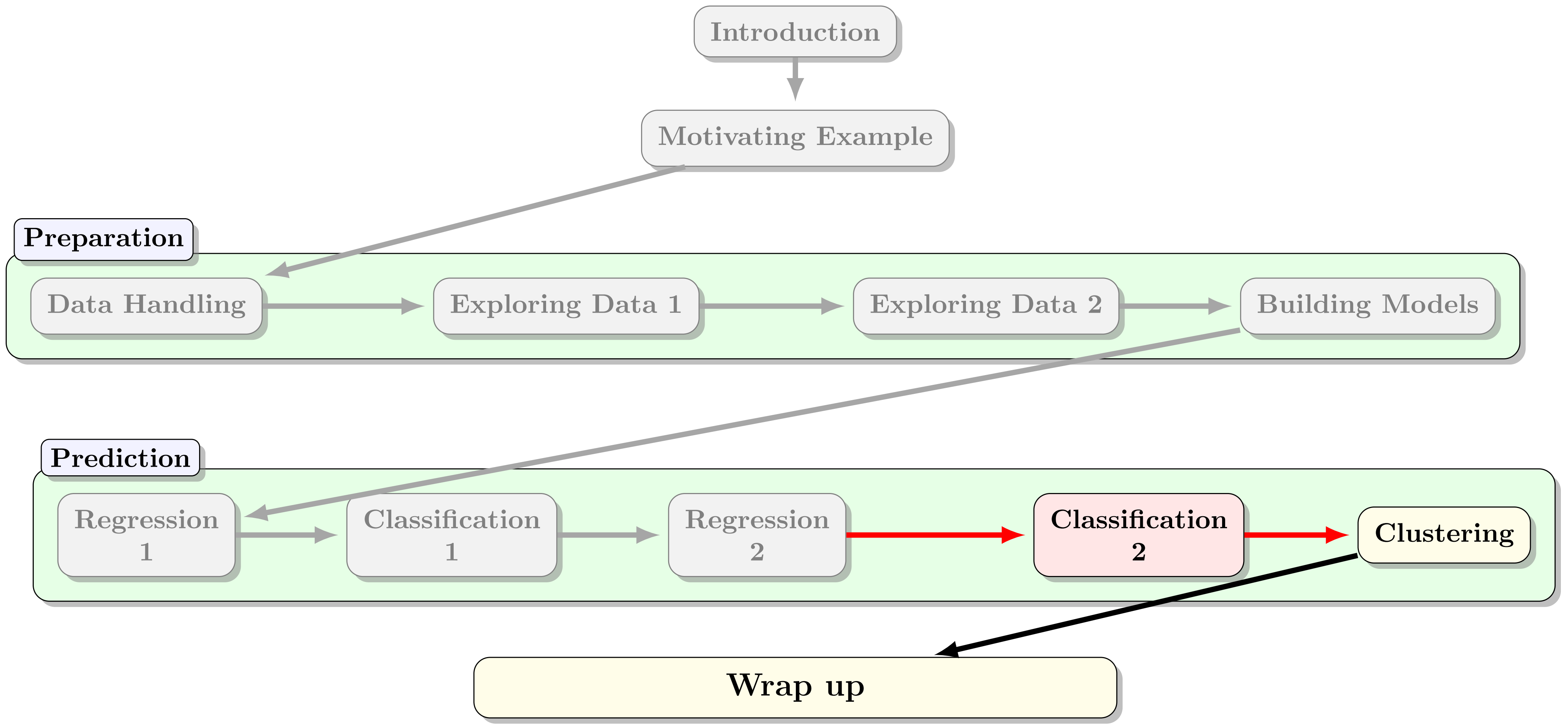
(bernard.butler@setu.ie)

Autumn Semester, 2025

Outline

- Naive Bayes
- Ordinal Classification/Regression

Data Mining (Week 10)



Outline

Rev. Bayes and his theorem

Rev. Thomas Bayes, 1702–1761

Usage

Given $P(E|H)$ (Probability of Evidence (attributes) given the Hypothesis (the known classes) in the *training* set), Bayes theorem shows how to invert this relationship to get $P(H|E)$ (Probability of the Hypothesis (class) given the evidence (attributes) with an (unseen) *test case*).

Bayes' Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

where $P(X|Y)$ is the conditional probability of X given that we know Y is true.

Application to classification

By convention, $A = H$ and $B = E$, where H is the hypothesis (observation has specific class) and E is the evidence form the training data in support of that hypothesis.
With this interpretation, the Bayes identity can be used to predict class probabilities (hypothesis) from features (evidence).

Conditional probabilities and Bayes terminology

Definition 1 (Conditional Probability)

If A and B are events, the Probability of A, given that B is true (has happened), written $P(A|B)$ is defined as follows:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{2}$$

where $P(A \cap B)$ is the probability that both A and B are true.

Given Bayes Theorem 1, we have

Term	Description
$P(A)$	Class prior; Prior probability
$P(B)$	Predictor prior; evidence
$P(A\|B)$	Posterior probability; Updated Class Prior
$P(B\|A)$	Likelihood

- In courtrooms, B might represent the available evidence in favour of guilt, and A might represent the verdict “is guilty”.
- In data mining, B might represent the features (derived from the Data) for a given instance and A might represent the *predicted label* for these features.
- If A and B are independent events, $P(A \cap B) \equiv P(A)P(B)$, so $P(A|B) = P(A)$ and $P(B|A) = P(B)$.

Extended Naive Bayes

In practice, there could be multiple features/evidence so $B = \{B_1, B_2, \ldots, B_n\}$

.

Definition 2 (Extended Bayes Theorem)

The extended form, when $\{B_j\}$ partition B, so $B = \cup_j B_j$ and $B_p \cap B_q \equiv \emptyset$ unless $p = q$, is

$$P(A|\{B_i\}) = \frac{P(\{B_i\}|A)P(A)}{P(\{B_i\})} \tag{3}$$

which is the component-wise version of the standard Bayes Theorem.

Side note: Prosecutor's Fallacy

Note that $P(A|B) \neq P(B|A)$ in general. If the ratio $\frac{P(A)}{P(B)}$ is not close to 1, lawyers can mislead jurors regarding guilt or innocence. *Probability of Guilt given the evidence is not the same as the probability of the evidence assuming the defendant is guilty.* "Since the defendant is probably guilty, and we have proved he had the opportunity, he must have committed the crime."

Naive Extended Bayes

Definition 3 (Naive Bayes)

If the features B are assumed to be independent of each other, it can be shown that

$$P(B) = P(B_1 \cap B_2 \cap \ldots B_n) = \prod_i P(B_i) \tag{4}$$

$$P(B|A_j) = \prod_k P(B_i|A_j) \tag{5}$$

The naïve form of Bayes theorem becomes

$$P(A_j|B) = \frac{\prod_i P(B_i|A_j)P(A_j)}{\prod_i P(B_i)} \tag{6}$$

Naive Bayes classifier

Definition 4 (Naive Bayes classifier)

- The Naive Bayes algorithm predicts a class, given a set of set of features, using probability principles.
- It is naive because it assumes the features are statistically independent
- Even though the requirement that *all* feature-feature correlations are negligible is a strong assumption, Naive Bayes often works well in practice.
- The training data is used to estimate the probabilities and likelihood, which are the “input parameters” used in Bayes theorem.
- Bayes theorem provides the expression used to predict a new observation’s membership of each class (associated with a label).
- The observation is then assigned to the class for which its conditional probability is greatest.

Naive Bayes Interpretation

- Both the simple and extended Bayes Theorem provide a formula to flip between “B given A” (data given labels) to “A given B” (labels given data).
- From the Training set, we can calculate
  - $P(A_j)$ (prior probability of each class label),
  - $P(B_i)$ (the predictor prior) and
  - $P(B_i|A_j)$ (the evidence that the feature valued B_i predicts the class label A_j).
- From this, we can use the Naive version of the extended Bayes Theorem 6 to predict $P(A_j|B)$, the posterior probability of class label A_j given all the evidence from the features B.

Overview of Naïve Bayes algorithm

- In the *Training Phase*, the method precomputes various probabilities $P(A_j)$ and conditional probabilities $P(B_k|A_j)$.
- In the *Prediction Phase*
  - the extended form of Bayes' Theorem 6 is used to compute the posterior class probabilities $P(A_j|B)$ for the given observation.
  - according to the *Maximum A Posteriori* (MAP) criterion, the observation is assigned to the class with the highest probability from $P(A_j|B)$.
- One aspect of Naïve Bayes (with $P(A|B)$), like decision trees (with $P(A \cap B)$), is the direct role played by probability
- When training Naïve Bayes, it is convenient to compute a table of *marginal counts*, as seen in the next slide, and to use these for prediction.

Fruit classification example

Example: Fruit classification

Type	Long	¬Long	Sweet	¬Sweet	Yellow	¬Yellow	*Total*
Banana	400	100	350	150	450	50	*500*
Orange	0	300	150	150	300	0	*300*
Other	100	100	150	50	50	150	*200*
Total	*500*	*500*	*650*	*350*	*800*	*200*	*1000*

Source: stackoverflow

Fruit classification : Precalculations

P(<Fruit>) = Total_<Fruit> / Total_*	$\rightarrow$ P(Other) = 200/1000 = 0.2
P(<Feature>) = Total_<Feature> / Total_*	$\rightarrow$ P(Sweet) = 650/1000 = 0.65
P(<Feature> | <Fruit>) = <Fruit,Feature> / Total_<Fruit>	$\rightarrow$ P(Sweet | Other) = 150/200 = 0.75

Fruit classification example: prediction

Given observation: Long=L, Sweet=S, Yellow=Y fruit, what type of fruit is it?

Banana - B

$P(\mathrm{B}|L,S,Y) =$

$$\frac{P(\mathrm{L}|\mathrm{B})P(\mathrm{S}|\mathrm{B})P(\mathrm{Y}|\mathrm{B})P(\mathrm{B})}{P(\mathrm{L})P(\mathrm{S})P(\mathrm{Y})}$$

$$= \frac{0.8 \times 0.7 \times 0.9 \times 0.5}{0.5 \times 0.65 \times 0.8}$$

$= 0.97$

Orange - O

$P(\mathrm{O}|L,S,Y) =$

$$\frac{P(\mathrm{L}|\mathrm{O})P(\mathrm{S}|\mathrm{O})P(\mathrm{Y}|\mathrm{O})P(\mathrm{O})}{P(\mathrm{L})P(\mathrm{S})P(\mathrm{Y})}$$

$$= \frac{0.0 \times 0.5 \times 1.0 \times 0.3}{0.5 \times 0.65 \times 0.8}$$

$= 0$

Other = R

$P(\mathrm{R}|L,S,Y) =$

$$\frac{P(\mathrm{L}|\mathrm{R})P(\mathrm{S}|\mathrm{R})P(\mathrm{Y}|\mathrm{R})P(\mathrm{R})}{P(\mathrm{L})P(\mathrm{S})P(\mathrm{Y})}$$

$$= \frac{0.5 \times 0.75 \times 0.25 \times 0.2}{0.5 \times 0.65 \times 0.8}$$

$= 0.07$

According to the MAP criterion, the observation (mystery fruit) is a banana!

Given the 3 binary-valued attributes, there are $2^3 = 8$ possible combinations - Naïve Bayes will classify each of these 8 combinations as one of the 3 fruit classes.

Naïve Bayes using scikit-learn

Setup

```python
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Fit and Predict

```python
gnb = GaussianNB()
gnb.fit(Xtrain, ytrain)
y_gnbTest = gnb.predict(Xtest)
print(accuracy_score(ytest, y_gnbTest))
print(confusion_matrix(ytest, y_gnbTest))
print(classification_report(ytest, y_gnbTest, digits=3, target_names=target_names))
```

Note that `GaussianNB` rarely has arguments and numeric features do not need to be scaled before use

Why is it called Gaussian Naive Bayes?

- Categorical features are used to group counts of obaservations when computing probabilities
- But what about numerical features?
- Assume each numeric feature has a Gaussian distribution, characterised by its mean (μ) and standard deviation (σ) parameters
- `GaussianNB` fits this prior distribution to each numeric feature, when computing the marginal counts for the categorical features
- For a given test instance, its z-score for each numeric feature is computed from the fitted μ and σ for that feature (scaling is implicit).
- Hence, its likelihood for that probability distribution can be obtained, and substituted in the Naive Bayes (NB) expression over all the features, for each class value.
- The predicted class value is just the class value with the largest NB prediction over the features.

Naïve Bayes Classifier summary

- Advantages
  - Conceptually simple and easy to implement (could be programmed by hand!)
  - Fast and scalable (compute counts and ratios, many terms can be precomputed. . .)
  - Variants exist for numeric (Gaussian-), binary (Bernoulli-) and multi-class (mulinomial-) featured Naïve Bayes
  - Particularly good for text classification and email spam detection
- Disadvantages
  - Ignores feature relationships, so #(feature) + #(¬feature) is the same for all features
  - Be careful of zero-valued conditional probabilities (numerical underflow if numerator, divide-by-zero if denominator)
  - Prone to underfitting (high bias) because so much aggregation happens that observation-specific information is lost
- For prediction $F(X) \to Y$
  - Most classifiers are discriminative—they learn $P(Y|X)$: “Given data X, what class Y is it?”
  - Naïve Bayes is generative—it learns $P(X|Y)$ and $P(Y)$: “Given classes Y, what data X fits each class, and how often do those classes occur?”
- Implementations exist in `sklearn`: `from sklearn.naive_bayes import GaussianNB`, etc.

Ordinal regression vs Ordinal Classification

- Should ordinal targets be predicted using regression?
  - Yes, because like numbers, they have a natural order…
  - No, because differences don't work the same way…
- Should ordinal targets be predicted using classification?
  - Yes, because the targets are categories, not numbers…
  - No, because the difference between two categories depends on their order, and classification ignores this
- scikit-learn does not offer Ordinal Regression/Ordinal Classification directly
- But there are proposals to wrap existing classifiers and to solve an extended problem that predicts the target while considering ordinal target values.

In the meantime, either Regression or Classification is used, with caveats…

Summary

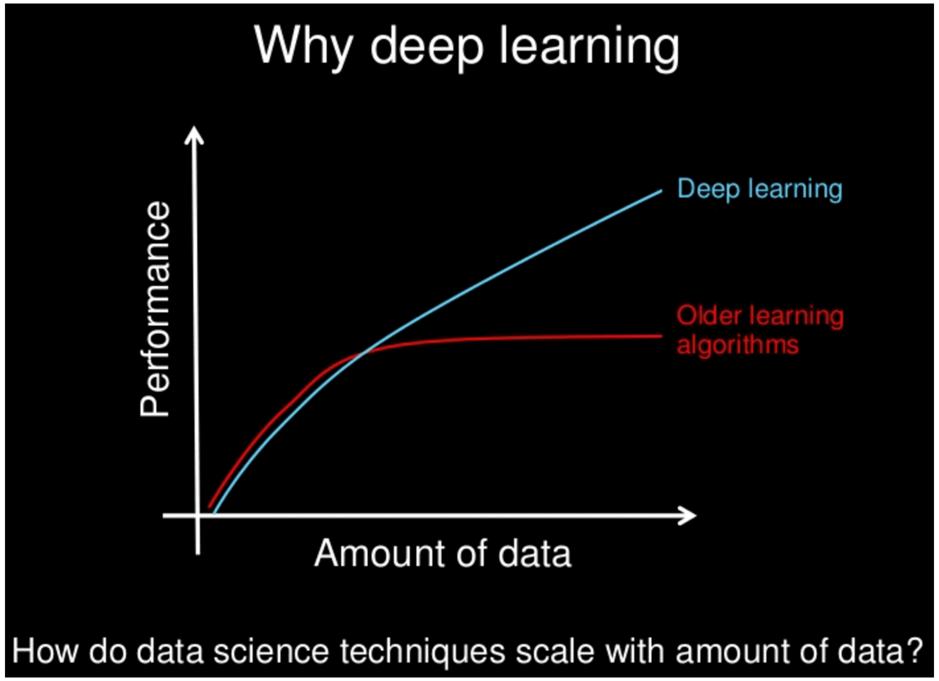
- Classification is one of the main tasks in data mining, and is a mature and well-studied field
  - k-nearest-neighbours is conceptually simple (based on voting) and the lack of a model (lazy learning) means it responds better to data drift
  - Logistic regression is an extension of linear regression and benefits from its strengths
  - Decision trees learn a representation that is often easily interpretable, but works better with linear boundaries
  - Naïve Bayes offers a probability-based generative model, able to work from data summaries, ideal for text and email classification
- More advanced classifiers have their own advantages, especially in relation to high dimensional data:
  - Ensemble methods (such as RandomForest and AdaBoost) were state of the art (2000-2012, say) and sacrifice interpretability for good performance with high dimensional data
  - Support Vector Machines (SVM) were state of the art (1985-2000, say) and are still extremely effective for very high dimensional problems like document classification: a small number of support vectors define the decision boundary, so the classification decision collapses to 1D

Other considerations

- KNN uses *lazy* learning, all other techniques above use *eager* learning (derive model from training data)
- Naive Bayes uses *generative* learning to learn how the data was generated, all other techniques above use *discriminative* learning to derive the function that assigns class labels
- For KNN and Decision Trees, the representation grows with the size of the data - that is not generally true in all other techniques above

Classification is sometimes confused with clustering - will cover *clustering* next week.

But is that the last word on Classification?



Source: Andrew Ng, *Why Deep Learning*

Learning from big data

- Traditional classification algorithms eventually run out of steam as data size increases
- Shallow neural networks had been discounted in the 1980s and 1990s when trained with small data
- Deep learning to the rescue!
- Kernel SVM and logistic regression lead nicely to perceptron models, hence ANNs, hence deep learning
- Deep learning requires lots of data but the models can scale better to take account of extra data

Deep Learning will probably be covered in semester 2…

General References