Week4

Naive bayes classifier

- Simple probabilistic model.
- Naive because it assumes that each future from a given class, is conditionally independent from the others.
- The metric for train and evaluation don't have to be the same.
- Highly efficient learning and prediction.
- Generalisation performance may worse than more complex learning models.
- Works with high dimensional data.

- Simple efficient parameter estimation.
- Their confidence estimates for predictions are not very accurate.
- Useful as baselines

Types

- **Bernoulli (useful for text)**
  - Binary features (eg word presence/absence)
- **Multinomial (useful for text)**
  - Discrete features (eg word count)
- **Gaussian**
  - continous/real-valued features
  - During training, for each feature in each class, calculates mean, std.
  - For each sample, it recalculates mean and std and assigns the class that it more close to the training set class values.
  - Assumes that the data for each class was generated with a class specific guassian distribution.
  - Predicting the class, mathematically corresponds with estimating the prob. of the gaussian distribution that generated that sample.
  - For a binary case, the decision boundary is parabolic.
  - Useful for high dimensional sets.

Random forests

- Using ensembles is useful because by using different small models that tend to overfit in specific parts of the data. When averaging them all, the result is less overfitting.
- No need for preprocessing.
- Easily parallelized on CPUs.
- May be difficult to interpret by humans

- Not good for very high dimensional sparse data (like text).
- This idea used with trees, results in random forests.
- For classification: RandomForestClassifier
- For regression: RandomForestRegressor.
- Steps:
    1. The data for each tree is selected randomly (with replacement) → called bootstrap sample.
    2. The features for each tree are also selected randomly (max_features param).
    3. Select number of trees (n_estimator param).
    4. The the splits are calculated as with standard decision trees, only that for a small set of features (those in that tree).
    5. Pred:
        1. For regression, the prediction is the mean of the individual tree prediction.
        2. For classification: each tree gives a proba for each class. Probabilities are averaged across trees and the the class with the highest probability is assigned.
- Setting max_features = 1, leads to forests with diverse, complex trees. If max_features is close toe the true number of features, it will lead to similar forests with simple trees.

Params

- n_estimators: number of trees to use in ensamble (default:10).
    - Should be larger for larger datasets to reduce overfitting (but uses more
- max_features: has a strong effect on performance. Influences the diversity of trees in the forest.
    - Default works well in practice, but adjusting may lead to some further gains.

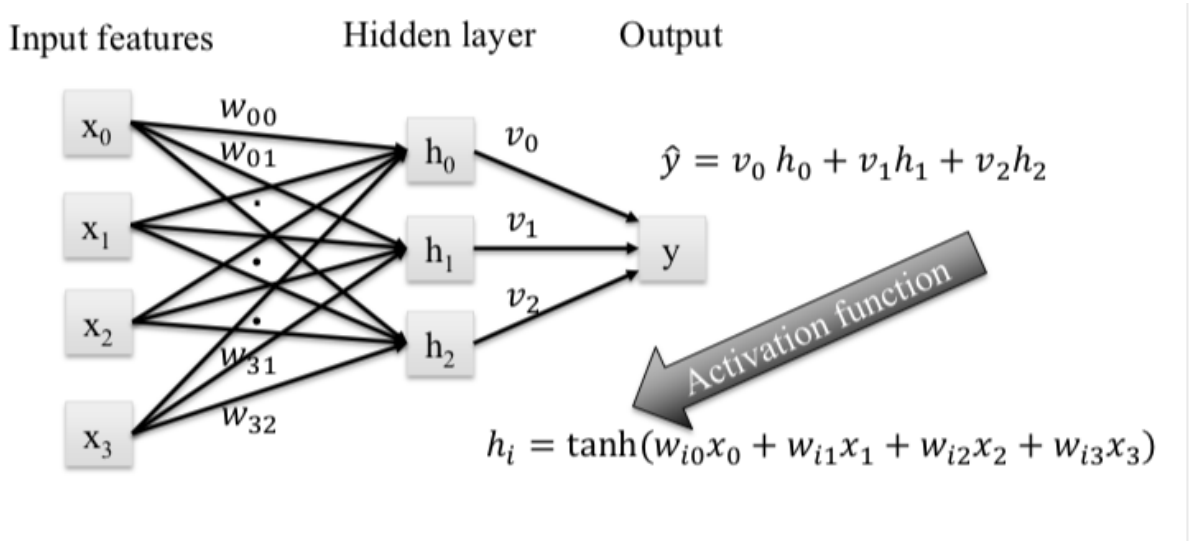
- max_depth: controls the depth of each tree(default:None.Splits until all leaves are pure).
- n_jobs: how many cores to use in parallel during training.
- Choose a fixed setting for the random_state parameter if you need reproducible results.

Gradient Boosted Decision Trees

- Creates small (shallow) trees
- Each tree attempts to correct errors from the previous stage (tree).
- n_estimators is also used (number of trees)
- The learning rate controls how hard each new tree tries to correct mistakes from previous tree.
  - High LR: more complex trees (more emphasis on correction).
  - Low LR: Less through in correcting mistakes from previous trees.
- It makes box-like decision boundaries (like most tree-based algos)
- Pros:
  - Often best off-the-shelf accuracy on many problems.
  - Using model for prediction requires only modest memory and is fast.
  - Doesn't require careful normalization of features to perform well.
  - Like decision trees, handles a mixture of feature types.
- Cons:
  - Like random forests, the models are often difficult for humans to interpret.
  - Requires careful tuning of the learning rate and other parameters.
  - Training can require significant computation.
  - Like decision trees, not recommended for text classification and other problems with very high dimensional sparse features, for accuracy and computational cost reasons.
- Parameters:

- n_estimators: sets # of small decision trees to use (weak learners) in the ensemble.
- learning_rate: controls emphasis on fixing errors from previous iteration.
- The above two are typically tuned together.
- n_estimators is adjusted first, to best exploit memory and CPUs during training, then other parameters.
- max_depth is typically set to a small value (e.g. 3-5) for most applications.

Neural networks



- There is regularization (alpha) like L2.
- Features must be normalized.
- Pros: They form the basis of state-of-the-art models and can be formed into advanced architectures that effectively capture complex features given enough data and computation.
- Cons:
  - Larger, more complex models require significant training time, data, and customization.

    - Careful preprocessing of the data is needed.
    - A good choice when the features are of similar types, but less so when features of very different types.
- Parameters:
    - Hidden_layer_sizes: sets the number of hidden layers (number of elements in list), and number of hidden units per layer (each list element). *Default: (100)*.
    - alpha: controls weight on the regularization penalty that shrinks weights to zero. *Default: alpha = 0.0001*.
    - activation: controls the nonlinear function used for the activation function, including: 'relu' (default), 'logistic', 'tanh'.

Data leakage

- Introducing information about the target during training that would not legitimately be available during
  actual use.
- When the data you're using to train contains information about what you're trying to predict.

Leakage in training data:

- Performing data preprocessing using parameters or results from analyzing the entire dataset: Normalizing and rescaling, detecting and removing outliers, estimating missing values, feature selection.
- Time-series datasets: using records from the future when computing features for the current prediction.
- Errors in data values/gathering or missing variable indicators (e.g. the special value 999) can encode information about missing data that reveals information about the future.

Leakage in features:

- Removing variables that are not legitimate without also removing variables that encode the same or related information (e.g. diagnosis info may still exist in patient ID).

- Reversing of intentional randomization or anonymization that reveals specific information about e.g. users not legitimately available in actual use.
- Any of the above could be present in any external data joined to the training set.

Detecting data leakage

B**efore building the model**

- *Exploratory data analysis to find surprises in the data*
- *Are there features very highly correlated with the target value?*

After building the model

- Look for surprising feature behavior in the fitted model.
- Are there features with very high weights, or high information gain?
- Simple rule-based models like decision trees can help with features like account numbers, patient IDs
- Is overall model performance surprisingly good compared to known results on the same dataset, or for similar problems on similar datasets?

L**imited real-world deployment of the trained model**

- Potentially expensive in terms of development time, but more realistic
- Is the trained model generalizing well to new data?

Minimising Data Leakage

Perform data preparation within each cross-validation fold separately

- *Scale/normalise data, perform feature selection, etc. within each fold separately, not using the entire dataset.*
- *For any such parameters estimated on the training data, you must use those same parameters to prepare data on the corresponding held-out test fold.*

With time series data, use a timestamp cutoff

- *The cutoff value is set to the specific time point where prediction is to occur using current and past records.*
- *Using a cutoff time will make sure you aren't accessing any data records that were gathered after the prediction time, i.e. in the future.*

Before any work with a new dataset, split off a final test validation dataset

- if you have enough data...
- Use this final test dataset as the very last step in your validation
- Helps to check the true generalisation performance of any trained models

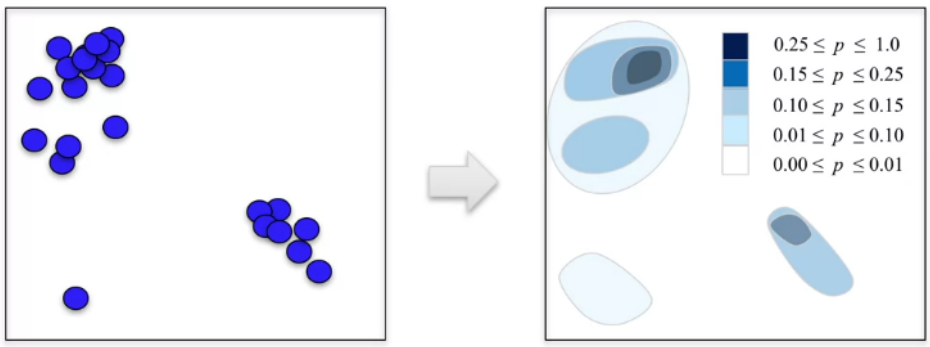
Unsupervised learning

Transformations

Processes that extract or compute information

Density estimation

- Calculates a continuous probability density over the feature space given a set of discrete samples in such space.
- An area is delimited and a probability is given to each observation, for belonging to such area.



Individual measurements

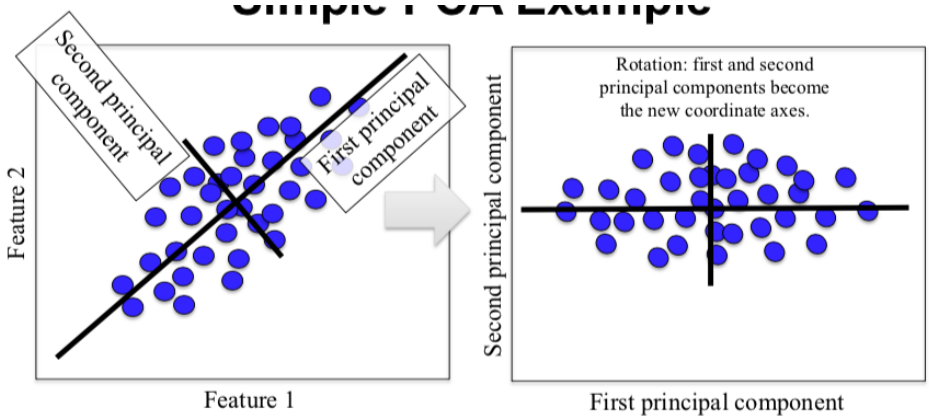
Density estimate
(Estimated probability p of observing a measurement at a given location)

Dimensionality reduction

- Finds an approximate version of your dataset using fewer features.
- Used for exploring and visualising a dataset to understand grouping or relationships. Often visualized using a 2-dimensional scatterplot.
- Also used for compression, finding features for supervised learning

PCA

- Take the data points and finds the rotation so the dimensions are statistically uncorrelated.
- Features should be normalised with standard scale.



- pca.components_ shows which feature was more correlated with each component.

Manifold learning algorithms

- Try to find low dimensional structures in a high dimensional feature space. Useful for visualization in 3d/2d.
- t-SNE: A powerful manifold learning method that finds a 2D projection trying to preserve information about neighbours in the original space.

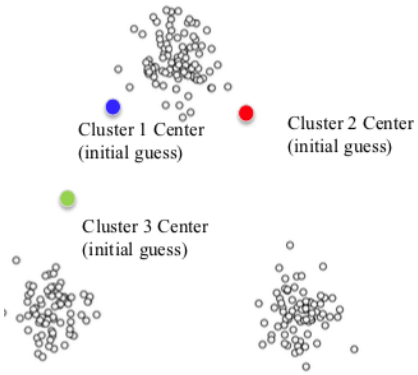
Clustering

- Find groups in the data. Data points within the same cluster should be 'close' or 'similar' in some way.

- Hard clustering: Each data point belongs to one cluster
- Soft or fuzzy clustering: Each data point os assigned a weight, score or probability of membership for each cluster.

K-means

1. Pick number of clusters k you want to find. Then pick k random points to serve as an initial guess for the cluster centers.
2. Step A: Assign each data point to the nearest cluster center.
3. Step B: Update each cluster center by replacing it with the mean of all points assigned to that cluster (in step A).
4. Repeat steps A and B until the centers converge to a stable solution.



Note: Min-Max scaling should be use on the features

- Works well for simple clusters that are same size, well-separated, globular shapes.
- Does not do well with irregular, complex clusters.
- Variants of k-means like k-medoids can work with categorical features.

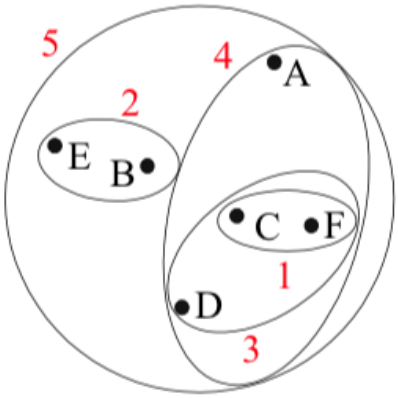
Agglomerative clustering

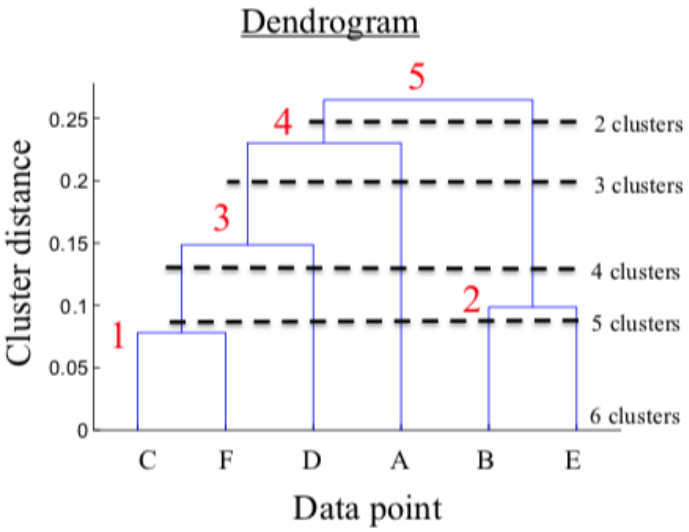
1. Each data point is in its own cluster.
2. The most similar two clusters are merged to form a new cluster.

   Linkage Criterias:

   1. Ward's method: Least increase in total variance (around cluster centroids)
   2. Average linkage: Average distance between clusters.
   3. Complete linkage: Max distance between clusters.

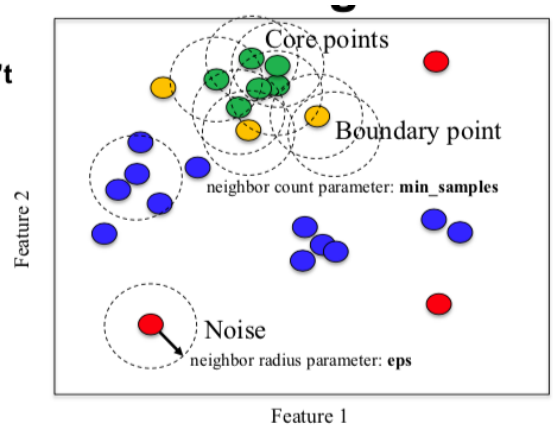
3. Step 2 is repeated until a condition is met. In sklearn the condition is the preselected number of clusters.





DBSCAN: Density Based Special Clustering of Applications with Noise

- Unlike k-means, you don't need to specify # of clusters
- Relatively efficient – can be used with large datasets
- Identifies likely noise points
- Two main parameters: **min_samples** & **eps**.
  - eps kinda controls the number of clusters, although ths is not explicit.



- The idea is to analize the density in different regions
  - More dense regions, are named as **core samples**. For a given data point, if there are min_samples other data points that lie within a

distance of eps, that given point is labeled as a core sample. Then all core samples that are with a distance of eps units apart, are put into the same cluster.

- Points that are not within any cluster, are considered as **noise**.
- Points that are within an eps unit of other points, but are not core points themselves, are termed **boundary points.**

- If the features are scaled using normalized scaling or min-max, finding the correct eps value is usually easier.
- A label of -1 in sklearn, means that the data point was classified as noise.

Clustering evaluation

- With ground truth, existing labels can be used to evaluate cluster quality.
- Without ground truth, evaluation can difficult: multiple clusterings may be plausible for a dataset.
- Consider task-based evaluation: Evaluate clustering according to performance on a task that does
  have an objective basis for comparison.
- Example: the effectiveness of clustering-based features for a supervised learning task.
- Some evaluation heuristics exist (e.g. silhouette) but these can be unreliable.