2nd Exam - Machine Learning - Notes

p1 10 - linear models:

▷ **Logistic regression:** is a linear classifier that is based on maximum likelihood principles to jointly model the classification boundary and the certainty of the fit.

▷ **Support Vector Machines (SVM):** is a prototypical example of discriminative learning.

↳ SVM classifier finds the boundary with maximum distance/margin to both classes.

better in the back ↑

↳ **C parameter:** controls the trade-off between achieving a smooth decision boundary and classifying the training points correctly. (max the margin and min misclassifications)
roughly balances margin and misclassification rate (soft-margin SVM)
Smaller C -> simpler decision boundary with misclassifications
Larger C -> more accurate classification, more complex boundary. (min misclassifications) ↓ (max margin)

↳ **Support Vectors:** the data points closer to the decision boundary (hyperplane). These points are crucial in determing the optimal hyperplane and the margin.

p1 11 - Kernels & PDF: map data points into a higher-dimensional space.

▷ **Kernel types:**

linear -> $K(x_i, x_j) = x_i^T x_j$

polynomial -> $K(x_i, x_j) = (x_i^T x_j + 1)^d$

RBF -> gaussian -> $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right) = e^{-\gamma\|x_i - x_j\|^2}$

↳ **Bad Kernel:** Mostly diagonal, most points are orthogonal to each other, no clusters, no structure

↳ **Good Kernel:** The matrix has structure and show clusters

↳ **Ideal Kernel:** $K_{ideal} = YY^T$

↳ **Kernel trick:** allows algorithms to compute the dot product (or similarity) of data-points in the higher-dimensional space without explicitly calculating the transformed data points.

↳ **Example:** Kernel PCA

▷ This is useful when the transformation would be computationally expensive or even imposible to compute explicitly

Definition of a Kernel: $K(x, y)$
Is a function that finds similarities between pairs of data points and map them into a higher-dimensional spaces. It calculates the similarity of the data points without explicitly transforming them into this higher-dimensional space.

- The notion of kernel allows to transform any linear classifier into a non-linear one
- Kernels are measurs of similarity in a certain transformed space (usually high dimensional)
- The kernel trick allows to kernalize any algorithm by replacing $\langle x, y \rangle$ by $K(x, y)$
- We can go beyond the kernel trick considering RKHS equivalents

p11112_Ensemble learning:

▲ **Decision trees:** classification strategy

↳ The idea is to partition the space in patches and fit a model in that patch.

What's great about Decision trees:

* Easy to interpet. Set of rules. Each path ~~of~~ from root to one leaf is an AND combination of the thresholded features

* Given a finite dataset, D.T. can express any function of the input attributes.

* There can be more than one tree that fits the same data. From them we would like a tree with minimum number of nodes. But the problem is NP.

Trees can easily overfit:

To prevent:

1. Stop growing the tree when the split is not statistically significant
2. Grow a full tree and post-prune.

↳ Post-pruning -> "reduced error pruning":

1. Split the data into train & validation
2. Create a candidate tree on the training set
3. Do until further pruning is harmful:
   A. Evaluate impact on validation set of removing each possible node
   B. Remove the node that improves the performance the most.

▷ **Ensemble learning:** seek additional opinions before making a major decision

↳ Derided in 2 steps:
- Train a set of classifiers
- Aggregate their results

↳ Why using ensemble learning:

1. Statistical reasons
2. Large scale data sets
3. Divide and conquer (Difficultly)
4. Data fusion: (heterogenous sources)

Ensembe models:

↳ Bootstrapping aggregation / Bagging:

- Designed to improve the stability and accuracy of ML algorithms, especially Decision trees.
- The idea is train multiple instances of the same learning ~~rule~~ algorithm on different subsets of the training data and then combine their predictions.
  ↳ for the final decision:
  Repression tasks: predictions may be averaged
  Classification tasks: Voting or averaging of class probabilities

↳ Helps reduce overfitting & variance by introducing diversity in the training process

* Example: Random Forest: Introduces a randomization over the feature selected for building each tree in the ensemble in order to improve diversity

▷ **Diversity:** refers to the differences or variations among the individual models that constitute the ensemble. With diverse classifiers that make different errors improve the overall performance of the ensemble

How to achieve diversity:
- Different training sets. Use resampling strategies to obtain different optimal classifiers
- Using different training parameteres for different classifiers
- Combining different architectures (svm, decission trees, ...)
- Training on different features (random subspaces, random projections)

Better in the back
- -> Bootstrap sampling (Bagging)
- -> Random feature selection (Random Forest)
- -> Boosting with Weighted samples: AdaBoost
  ↓
  adding weights to train instances based on their performance
  train series of weak learners and using weights to each instance

Methods for the multi-class problem:

- Decision trees
- Nearest Neighbors
- Error correcting output (one-vs-all, one-vs-one)

pill13_ Neural networks and deep learning: check Jupyter

pill14a_ Manifold Learning:

Manifold learning: set of techniques to uncover the underlying structure or geometry of high-dimensional data.

"Manifold": refers to a lower-dimensional, smooth and usually curved subspace within the hight-dimensional space.
↳ represent the data in lower-dimensional spaces

Techniques:

- **Multi Dimensional Scaling (MDS):** similar to PCA dimensionality reduction
- **Isomap:** preserve the geodesic distances (shortest paths along the curved surface) between all pairs of data points.
  ↳ Uses MDS to geodesic rather than straight line distances to find a low-dimensional mapping that preserves these pairwise distances.
  1. Construct a neighborhood graph
  2. Compute shortest path (geodesic distances) between all points
  3. Embed the data via MDS
- **Locally Linear Embedding (LLE):** seeks to reconstruct each data point as a linear combination of it's nearest neighors and then finds a low-dimensional representation that preserve that local relationships.
  ↳ Effective for non-linear structures
- **T-Distributed Stochastic Neighbor Embedding:** is a non-linear dimensionality reduction technique that focuses on preserving local relationships between data points
  ↳ visualise high-dimensional data to 2 or 3 dimensions

p1111S_PCA:

Principal Component Analysis: (unsupervised technique)

- Is used to reduce dimensionality
- The goal is to project the original data onto a subset of the eigenvectors of the covariance matrix
- The eigenvectors represent the directions in the original feature space where the data varies the most.

Define the amount of dimensions:

- fixed value set beforehand
- User specify the amount of dimensions based on the percentage of variance (energy) they want to preserve (Larger variance -> more information)

Fisher's linear discriminant analysis (LDA):

- Dimensionality reduction
- Unlike PCA which focuses on maximizing variance, LDA focuses on maximazing the separation between classes

  - Sb: between class scatter: spread between the class means => How well-separated the classes are
  - Sw: within-class scatter: spread within each class, => How much each class varies internally

Sequential Forward Floating Search:

Feature selection algorithm designed to improve the quality of the selected feature subset.
3 appoaches: Start with an empty set

1. Forward search: · Iterate through each feature
   · Evaluate the performance of the model with the addition of each feature
   · Select feature with the best performance, add in the set
2. Backward search: The same but with removing features each time
3. Floating search: Combine the previous two. First we add and then try to remove. Is repeated while the performance increases.

p1116 - Clustering:

K-means algorithm: (unsupervised ML)

clustering algorithm: split a dataset into k distinct, non-overlapping clusters.
the goal: minimize the sum of squared distances between data points and centers of their respective clusters

The algorith iteratively refines the cluster assignments and updates the centroids until convergence

K-means has some limitations:

- Sensitive to initializations: may converge to different solutions depending on the initial placement
- Assumption of spherical clusters: assumes that the clusters are spherical & equal sized, may not hold for all types of data
- Fixed number of clusters(k): the user has to give the number of k
- Sensitive to outliers

Soft K-means: Unlike K-means which assigns each data point to a single cluster with "hard" assignment, soft k-means allows for "soft" or probabilistic assignments, indicating the likelihood of a data point in each cluster
(good when clusters have overlapping regions)

↳ mixture of gaussians

Hierarchical Clustering:

Builds a hierarchy of clusters. This hierarchy is represented as a tree-like structure called a dendogram.
Does not require specifying the number of clusters in advance.
Two types:

1. Bottom-Up agglomerative: start by taking each data point as single cluster and the merge the two closest into a one cluster until only one cluster remains.
2. Top-Down divisive: the entire set as one cluster and then split he cluster into smaller ones, until each data point is a separate cluster

▷ Density-based clustering - DBSCAN:

Identify clusters based on the local density of data points in the feature space.
They don't assume that clusters have a specific shape/size.
They discover clusters of arbitrary shapes.

Advantages:
- Robust noise and outliers
- Does not require specifying the number of clusters in advance

Challenges:
- Sensitive to the choice of parameters
- Performance may degrade in high-dimensional spaces

↳ **DBSCAN:** certain knowledge about the domain for parameter ε and minPts
=> Using k-th nearest neighbor

↳ **Pros:** · Handles noise implicitly
· Can find clusters of with different shapes

↳ **Cons:** · Difficult to find the correct parameterization (small values of ε tend to oversegment data)
· Can not adapt to clusters with different densities

PDF:

Question 2: Soft-margin SVM paired with RBF Kernel: C, γ
Describe the role of each one and their influence in the obtained solution.

C: controls the trade-off between the margin and the misclassifications. Trade-off between achieving a smooth decision boundary and classifying training errors correctly

Smaller C: Simplier decision boundary with misclassifications

Larger C: More complex decision boundary but more accurate classification

γ: determines the influence of a single training example on the SVM's decision boundary. It defines the reach of the kernel function, controlling how far the influence of a single training point extends. (Scope of the data influenced)

Smaller γ: Simplier decision boundary, less complex. Create a more generalized model, missed details.

Larger γ: Making the decision boundary more intricate and tailored to the training data. More complex, potentially overfit model

↳ $e^{-\gamma d(x_i, x_j)}$ ↳ distance between points

Combined influence: ↓C ↑γ: more tolerant, smoother decision boundary

↑C ↓γ: tighter, more intricate decision boundary

Question 3: What does diversity mean? Three different way to achieve diversity.

Bootstrapped Sampling (Bagging): Each classifier in the ensemble is trained on a different subset of the original training data, sampled with replacement

Random feature selection (Random Forest): Random subsets of features are considered for training each classifier. In R.F. each tree is trained on a different subset of features, and the final prediction is made by aggregating the outputs of all trees.

Boosting with Weighted Samples (AdaBoost): Classifier are trained sequentially and each subsequent classifier focuses on correcting the errors made by the previous ones.