Performance Measures

**Classification F-Measure:**
(careful: similar but not the same F-measure as the F-measure we saw for clustering!)

Tradeoff between classifying correctly all datapoints of the same class and making sure that each class contains points of only one class.

True Positives\((TP)\): nm of datapoints of class 1 that are correctly classified
False Negative \((FN)\): nm of datapoints of class 1 that are incorrectly classified
False Positives\((FP)\): nm of datapoints of class 2 that are incorrectly classified

Recall: \[
\frac{TP}{TP + FN}
\]

Precision: \[
\frac{TP}{TP + FP}
\]

\[F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}\]

*Precision*: proportion of datapoints of class 1 correctly classified over all datapoints classified in class 1
Performance Measures

Classification F-Measure:
(careful: similar but not the same F-measure as the F-measure we saw for clustering!)

Tradeoff between classifying correctly all datapoints of the same class and making sure that each class contains points of only one class.

True Positives (TP): nm of datapoints of class 1 that are correctly classified
False Negative (FN): nm of datapoints of class 1 that are incorrectly classified
False Positives (FP): nm of datapoints of class 2 that are incorrectly classified

Recall: \[ \frac{TP}{TP + FN} \]

Precision: \[ \frac{TP}{TP + FP} \]

\[ F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \]

Proportion of datapoints correctly classified in Class 1

Precision: proportion of datapoints of class 1 correctly classified over all datapoints classified in class 1
The ROC curve plots the fraction of true positives and false positives over the total number of samples of class $y=+1$ in the dataset. Each point on the curve corresponds to a different value of the classifier’s parameter (usually a threshold; e.g. a threshold on Bayes’ classification).
Gaussian Mixture Models (GMM) + Bayes

How does it work

- Fit a GMM model on each class
- Compare the pdf of each class

\[ p_+ (x) > p_- (x) \]

What does it learn

- a density model for each class

Why is it good

- Small number of parameters for good generalization
- Learns importance for each dimension
Multi-Layer Perceptron (with Back-Propagation)

How does it work

- Each neuron “cuts a plane”
- We combine $n$ neurons together to get a non-linear classifier

What does it learn

- Cuts the space into hyperplanes that are “combined” together

Why is it good

- Fixed size of the model (size of the hidden layer)
- Extremely fast at testing time
K-Nearest Neighbors

How does it work

- Find the $k$ nearest samples
- If more samples belong to the positive class, the output is positive

$$y = sgn\left(\sum_{i=1}^{k} C(x_i)\right)$$

Why is it good

- Extremely simple to implement
- Training = copying the data
Gaussian Process Classification

How does it work

- A “smooth” version of KNN
- Measure of how close you are to the known samples (training)

What does it learn

- A conditional density of a latent function modeled on the training data
- The latent function is then used for classification

Why is it good

- Very high precision
- Very good density model for each class
A Comparison

**Training**
- Boost
- RVM
- RANSAC
- Bagging
- GP
- KNN
- GMM
- SVM
- MLP

**Testing**
- Bagging
- RVM
- SVM
- GMM
- Boost
- MLP
- RANSAC
- GP
- KNN

**WARNING:** most of these algorithms require a certain amount of *tweaking* of the hyperparameters to get optimal results
A Comparison

Training

Testing

WARNING: most of these algorithms require a certain amount of tweaking of the hyperparameters to get optimal results
A Comparison

**Training**
- SVM
- MLP
- Bagging

**Testing**
- GMM
- Boost
- MLP
- RANSAC
- Bagging
- SVM
- RVM
- GMM
- KNN

**WARNING:** most of these algorithms require a certain amount of *tweaking* of the hyperparameters to get optimal results
Other factors to be aware of

**Accuracy**
- Given sufficient tweaking of hyper-parameters, all or most of the algorithms will provide similarly good accuracies

**Hyper-parameters**
- Methods such as KNN or GMM have relatively few hyper-parameters
- Methods like GP or Boosting have a larger number of parameters to choose

**Implementations**
- Often enough, finding an implementation of the algorithm that provides access to all the customization is not trivial
- Many researchers end up using the ‘vanilla version’ of the algorithms for lack of better implementations
What these classifiers have in common

\[ f(x) = \sum_{i} \alpha_i d(x, \theta^i) + b \]

To get the class label we then use \( \text{sgn}(), \text{argmax}(), \ldots \)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>\textbf{alpha}</th>
<th>\textbf{d()}</th>
<th>\textbf{theta}</th>
<th>\textbf{b}</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM / RVM</td>
<td>Lagrange multipliers</td>
<td>kernel function</td>
<td>support vectors</td>
<td>bias</td>
</tr>
<tr>
<td>Gaussian Process</td>
<td>scaling factor</td>
<td>kernel function</td>
<td>all datapoints</td>
<td>-</td>
</tr>
<tr>
<td>GMM</td>
<td>prior</td>
<td>gaussian function</td>
<td>means, variances</td>
<td>-</td>
</tr>
<tr>
<td>Boosting</td>
<td>weights</td>
<td>weak learner</td>
<td>params of weaks</td>
<td>bias</td>
</tr>
<tr>
<td>MLP</td>
<td>weights</td>
<td>activation function</td>
<td>params of act. fct.</td>
<td>bias</td>
</tr>
</tbody>
</table>