Lecture-12: Multiclass Classification & Generative Models

1 Multiclass Classification

1.1 Decision Tree (contd.)

- Criterion for splitting at each node of the decision tree:
  Selecting which feature (attribute) of input space to use and what should be the threshold value. Let's say each data instance $x = [x_1, ..., x_d]^T$, $x \in R^d$. We need to select a feature x_i and the threshold value t_i for $x_i < t_i$. Usually those features and threshold values are used, which reduce the classification error. However, it is not differentiable. Therefore, *entropy* which upper bounds the classification error and is strictly convex and differentiable is preferred. *Entropy measures the randomness in the system*. Entropy E of the system is computed as follows:

$$E = -\sum_{i=1}^{K} p_i \log p_i \tag{1}$$

  where, p_i is the fraction of samples with class-label i. The goal is to select that particular feature (and threshold) which reduces the sum of entropy in children nodes.
  Another measure to use in determining splitting criteria is the *Gini index G*, defined as:

$$G = \sum_{i=1}^{K} p_i(1 - p_i) \tag{2}$$

  This also upper bounds the classification error and is strictly convex and differentiable.

- Following are some of the drawbacks of Decision Tree:
  Decision Trees are *unstable*.
  - There is no margin defined, as compared to Support Vector Machines. The margin provides some *tolerance* to noise.
  - Decision Trees are more prone to overfitting.
- Random Forests have been found to be doing much better than Decision Trees and are less prone to overfitting. Small perturbation in the threshold values or feature selection for splitting is taken care by random selection of features. Thus, these are also stable.

1.2 K-Nearest Neighbours (K-NN)

The K-nearest neighbor (K-NN) algorithm is one of the simplest machine learning algorithms for classification. It works on the idea of *closeness* of data points as defined in the *metric space*. Very often we take the Euclidean metric space.

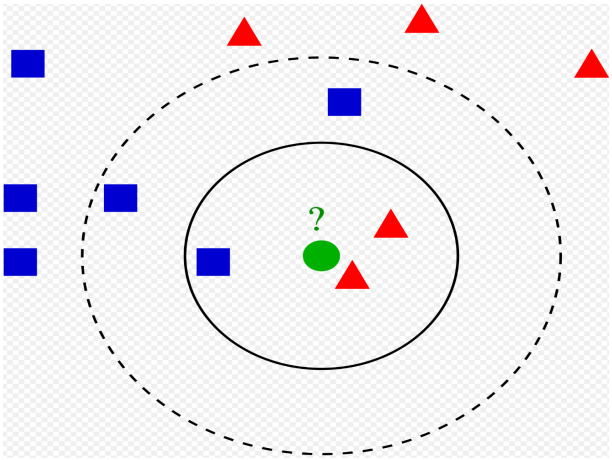

Figure 1: Example of K-NN classification. The test sample (green dot) should be classified either to blue squares or to red triangles. If $K = 3$ (solid line circle) it is assigned to the red triangles because there are 2 triangles and only 1 square inside the inner circle. If $K = 5$ (dashed line circle) it is assigned to the blue squares (3 squares vs. 2 triangles inside the outer circle). Figure taken from Wikipedia.

- Let's say S = $\{(x_1, y_1), ..., (x_m, y_m)\}$ is the sample of *m i.i.d. (independent and identically distributed)* data instances which are drawn from distribution D.
- $x_i \in X$, $X \subseteq R^d$, $y_i \in \{1, 2, ..., L\}$
- (X, d) is a metric space and $d(x_i, x_j)$ is its distance measure. For example:

$$d(x_i, x_j) = ||x_i - x_j||_2 \tag{3}$$

We predict the class-label of a new data instance, x_{new} by taking a majority vote among the classes of K-points in the training sample S which are closest to this new instance x_{new}. The *closeness* is the inverse of distance d in metric space (X, d). Figure 1 explains how the classification is done using K-NN algorithm.

Performance Analysis:

- As sample size goes to infinity, the expected error of 1-NN converges to twice the Bayes risk. For K-NN, it converges to $(1 + \sqrt{8/K})$ times the error of the Bayes classifier.
- K-NN suffers from curse of dimensionality. The sample size might increase exponentially with the dimensions. Refer [Shai], Pages 219-227.

2 Generative Models

- In *generative models*, we compute (infer) the class-conditional densities $p(x|H_k)$ for each class H_k, as well as the class priors $p(H_k)$, and then use them to compute posterior class probabilities $p(H_k|x)$ through Bayes theorem.

$$p(H_k|x) = \frac{p(x|H_k)p(H_k)}{p(x)} \tag{4}$$

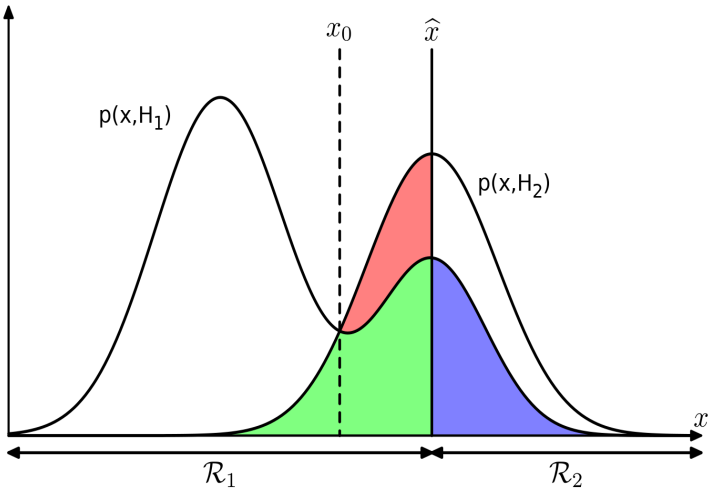

Figure 2: Figure illustrates the schematic description of random decision boundary $x = \hat{x}$ and optimal decision boundary $x = x_o$ over the two overlapping joint probability densities, $p(x,H_1)$ and $p(x,H_2)$. Figure taken from C. Bishop's *Pattern Recogition & Machine Learning*.

where, $p(x) = \sum_k p(x|H_k)p(H_k)$.

Using the posterior (class) probabilities, we can determine class membership for each new input x. Such classifiers are called *Bayesian Classifiers*. This approach explicitly or implicitly models the distribution of inputs as well as outputs.

- In *discriminative models*, we directly compute the posterior (class) probabilities from the sample training data and use it for classification of new data instance x. Logistic Regression is one such algorithm which explicitly computes the posterior (class) probabilities only.
- Let's consider *Bayesian Classifier*, which is a generative model:
  - A sample S of m data instances drawn *iid* from distribution D, $(x_1,y_1),.....,(x_m,y_m) \sim^{iid} D$
  - $x_i \in R^d$, $y_i \in \{1,2,....,K\}$.
  - $\pi_j = p(H_j)$ is the prior probability of class j.
  - c_{ij} is the cost of choosing class-label i while the sample comes from class j.
  - R_i is the region in which if x falls, a classifier will declare it from class i.
  - $p(x \in R_i|H_j)$ is the probability that sample x is in region R_i given that x is from class j.

  Total cost for this classifier

$$= \sum_{i=1}^{K}\sum_{j=1}^{K} c_{ij}\pi_j p(x \in R_i|H_j)$$
$$= \sum_{i=1}^{K}\sum_{j=1}^{K} c_{ij}\pi_j \int_{R_i} p(x|H_j)dx$$

$$= \sum_{i=1}^{K} \int_{R_i} \sum_{j=1}^{K} c_{ij} \pi_j p(x|H_j) dx$$
$$= \sum_{i=1}^{K} \int_{R_i} \sum_{j=1}^{K} c_{ij} p(x, H_j) dx$$
$$= \sum_{i=1}^{K} \int_{R_i} \sum_{j=1}^{K} c_{ij} p(H_j|x) p(x) dx$$
$$= \sum_{i=1}^{K} \int_{R_i} c_i(x) p(x) dx$$
where, $c_i(x) = \sum_{j=1}^{K} c_{ij} p(H_j|x)$

The goal is to minimize the above cost. The optimal classifier is called *Bayes Classifier*. For this classifier,

$$R_i = \{x : i = \arg\min_i c_i(x)\} \tag{5}$$

– *Uniform cost*: If we select $c_{ij} = 0$, when $i = j$ and 1 when $i \neq j$. Then,
$c_i(x) = \sum_{j \neq i}^{K} p(H_j|x) = 1 - p(H_i|x)$

Figure 2 shows the error regions in colors marked with blue, green and red. For the decision boundary $x = \hat{x}$, in region R_1, the errors are due to points from class H_2 being misclassified as H_1 which is represented by the sum of red and green regions, and in region R_2, the errors are due to points from class H_1 being misclassified as H_2 (represented by the blue region). Decision boundary $x = x_0$ represents the optimal one. It must be noted that error represented by red region vanishes when we select the optimal decision boundary and this is indeed the goal.

Therefore, the class-label for data instance x is:

$$\begin{aligned} \hat{y}_{MAP} = \arg\min_i c_i(x) &= \arg\min_i (1 - p(H_i|x)) \\ &= \arg\max_i p(H_i|x) \end{aligned} \tag{6}$$

where, $p(H_i|x)$ is the *posterior probability* of class i.

This is also called MAP (*Maximum Aposteriori Probability*) classifier.

– If all π_i are same and equal to $1/K$ then in the *MAP classifier*, the class-label for data instance x is:

$$\begin{aligned} \hat{y}_{ML} = \arg\max_i p(H_i|x) &= \arg\max_i \frac{p(x|H_i) 1/K}{p(x)} \\ &= \arg\max_i p(x|H_i) \end{aligned} \tag{7}$$

This classifier is called *Maximum Likelihood Classifier*.

- **Naive Bayes' Classifier**:
MAP classifier classifies the data instance x as $\arg\max_i p(H_i|x)$ which is equal to $\arg\max_i p(x|H_i)\pi_i$.

Very often, we don't know about the class-conditional densities $p(x|H_i)$ and the prior distribution of classes π_i. We only have sample of data from which we need to estimate the prior distribution of each class and the class-conditional probability distributions. Hence, they are called *plug-in classifiers*. Let's consider the following scenario:

- Sample $S = \{(x_1, y_1), ..., (x_m, y_m)\}$. Each $(x_i, y_i) \sim^{iid} D, i \in [m]$.
- Let's consider for simplicity $x_i \in \{0,1\}^d, y_i \in \{1,2,.....,K\}, i \in \{1,2,.....,m\}$.

We need to estimate the class-conditional probability distributions:

$$p(x_i|H_j) = p(x_{i,1}, x_{i,2}, ..., x_{i,d}|H_j). \tag{8}$$

This requires 2^d number of probabilities to estimate, which is infeasible. Therefore, we use a simplifying assumption of conditional independence on the componenents of each x_i given the class information:

$$p(x_{i,1}, x_{i,2}, ..., x_{i,d}|H_j) = \prod_{l=1}^{d} p(x_{i,l}|H_j). \tag{9}$$

This needs only d number of probabilities to be estimated. Each $p(x_{i,l}|H_j)$ is estimated from the sample.

$$\hat{p}(x_{i,l} = 1|H_j) = \frac{1}{m_j} \sum_i 1_{(x_{i,l}=1)} \tag{10}$$

where, m_j is the number of samples belonging to class H_j

We can estimate class prior probabilities as:

$$\hat{\pi}_j = \frac{m_j}{m} \tag{11}$$

Naive Bayes classifier classifies the new data instance x_{new} as $\hat{y}_{nb}$:

$$\hat{y}_{nb} = \arg\max_j p(H_j|x_{new}) = \arg\max_j \prod_{l=1}^{d} \hat{p}(x_{new,l}|H_j)\hat{\pi}_j \tag{12}$$

- Performance Analysis: *Discriminative vs Generative models*
  - Discriminative model classifiers are asymptotically better for large sample size.
  - Convergence rate of parameter estimation for Generative models is $O(\log d)$ while Discriminative models need $O(d)$. [d is the number of features of input space].

References

[Shai] Shalev-Shwartz, Shai, and Shai Ben-David. Understanding machine learning: From theory to algorithms. Cambridge university press, 2014.