Lecture-13: Multi-class classification: algorithms

1 Uncombined multi-class algorithms

1.1 Multi-class SVMs

Consider the family of hypotheses $\mathcal{H}_{K,2} = \left\{(x,y) \mapsto \langle \mathbf{w}_y, \Phi(x) \rangle : \sum_{i=1}^{k} \|\mathbf{w}_i\|^2 \leqslant \Lambda^2 \right\}$, with the margin error $\rho_h(x_i,y_i) = \langle \mathbf{w}_{y_i}, \Phi(x_i) \rangle - \max_{y \neq y_i} \langle \mathbf{w}_y, \Phi(x_i) \rangle$, and the empirical error for each sample point with margin unity as $\xi_i = \max\{1 - \max\{\rho_h(x_i,y_i), 0\}, 0\}$. Then, with probability at least $1-\delta$, we have

$$R(h) \leqslant \frac{1}{m}\sum_{i=1}^{m} \xi_i + 4k\sqrt{\frac{r^2\Lambda^2}{m}} + \sqrt{\frac{\ln\frac{1}{\delta}}{2m}}.$$

Problem 1 (Primal multi-class SVM algorithm). An algorithm that minimizes the right hand side of the generalization bound is the multi-class SVM algorithm given by

$$\begin{aligned}
&\min_{\mathbf{W},\xi} \frac{1}{2}\sum_{y \in \mathcal{Y}} \|\mathbf{w}_y\|^2 + C\sum_{i=1}^{m} \xi_i,\\
\text{subject to: } &\xi_i \geqslant 0, \text{ for all } i \in [m]\\
&\langle \mathbf{w}_{y_i}, \Phi(x_i) \rangle \geqslant \langle \mathbf{w}_y, \Phi(x_i) \rangle + 1 - \xi_i, \text{ for all } i \in [m], y \in \mathcal{Y} \setminus \{y_i\}.
\end{aligned}$$

Remark 1. The decision function is of the form $x \mapsto \arg\max_{y \in \mathcal{Y}} \langle \mathbf{w}_y, \Phi(x) \rangle$, and the above optimization problem is convex differentiable with affine constraints. Therefore, KKT conditions hold at the optimum.

Problem 2 (Dual multi-class SVM algorithm). We can write the dual optimization problem in terms of Kernel function in terms of matrix $\alpha \in \mathbb{R}^{m \times k}$ with unit vectors $e_y \in \mathbb{R}^{\mathcal{Y}}$ in yth dimension.

$$\begin{aligned}
&\max_{\alpha \in \mathbb{R}^{m\times k}} \sum_{i=1}^{m} \langle \alpha_i, e_{y_i} \rangle - \frac{1}{2}\sum_{i=1}^{m} \langle \alpha_i, \alpha_j \rangle K(x_i, x_j),\\
\text{subject to: } &\text{for all } i \in [m], (0 \leqslant \alpha_{i,y_i} \leqslant C) \wedge (\alpha_{i,y} \leqslant 0 \text{ for all } y \neq y_i) \wedge (\langle \alpha_i, 1 \rangle = 0).
\end{aligned}$$

1.2 Decision trees

Definition 1.1 (Binary decision tree). A binary decision tree is a tree representation of a partition of the feature space $\mathcal{X}$. Each interior node of a decision tree corresponds to a question related to the features. It can be a numerical question of the form $X_i \leqslant a$ for feature vector $X \in \mathbb{R}^N$, and some threshold $a \in \mathbb{R}$, or a categorical question such as $X_i \in A$ for some set $A \subseteq \mathcal{X}$, when feature X_i takes a categorical value. Each leaf is labeled with a label $y \in \mathcal{Y}$.

Example 1.2 (Binary space partition (BSP) trees). These trees partition the space with convex polyhedral regions, based on questions of the form $\langle \alpha, X \rangle \leqslant a$, where $X \in \mathcal{X}$ is a feature vector, and $\alpha \in \mathcal{X}, a \in \mathbb{R}_+$.

Example 1.3 (Sphere trees). These trees partition with pieces of spheres based on questions of the form $\|X - a_0\| \leqslant a$, where $X \in \mathcal{X}$ is a feature vector, and $a_0 \in \mathcal{X}$ a fixed vector.

Remark 2. More complex tree questions lead to richer partitions and thus hypothesis sets, which can cause overfitting in the absence of a sufficiently large training sample. They also increase the computational complexity of prediction and training. Decision trees can also be generalized to branching factors greater than two, but binary trees are most commonly used due their more limited computational cost.

1.2.1 Prediction/partitioning

To predict the label of any point $x \in \mathcal{X}$ we start at the root node of the decision tree and go down the tree until a leaf is found, by moving to the right child of a node when the response to the node question is positive, and to the left child otherwise. When we reach a leaf, we associate x with the label of this leaf.

Thus, each leaf defines a region of $\mathcal{X}$ formed by the set of points corresponding exactly to the same node responses and thus the same traversal of the tree. By definition, no two regions intersect and all points belong to exactly one region. Thus, leaf regions define a partition of $\mathcal{X}$.

In multi-class classification, the label of a leaf is determined using the training sample: the class with the majority representation among the training points falling in a leaf region defines the label of that leaf, with ties broken arbitrarily.

1.2.2 Learning a decision tree using labeled sample

The general problem of finding a decision tree with the smallest multi-class classification error is NP-hard.

Definition 1.4. For any node $\mathbf{n}$, let $p_y(\mathbf{n}) = \frac{\sum_{i=1}^m \mathbb{1}_{\{x_i \in \mathbf{n}, y_i = y\}}}{\sum_{i=1}^m \mathbb{1}_{\{x_i \in \mathbf{n}\}}}$ denote the fraction of points at $\mathbf{n}$ that belong to class $y \in \mathcal{Y}$.

Remark 3. The three most common measures of node impurity F are

$$F(\mathbf{n}) = \begin{cases} 1 - \max_{y \in \mathcal{Y}} p_y(\mathbf{n}), & \textit{misclassification}, \\ \sum_{y \in \mathcal{Y}} p_y(\mathbf{n})(1 - p_y(\mathbf{n})), & \textit{Gini index}, \\ -\sum_{y \in \mathcal{Y}} p_y(\mathbf{n}) \log_2 p_y(\mathbf{n}), & \textit{entropy}. \end{cases}$$

Remark 4. Gini index and entropy are largest for uniform distribution $p_y(\mathbf{n}) = \frac{1}{k}$ for all $y \in \mathcal{Y}$. All three measures are zero if $p_y(\mathbf{n}) = 1$ for some $y \in \mathcal{Y}$.

Remark 5. We observe that $1 - \max_{y \in \mathcal{Y}} p_y(\mathbf{n}) \leqslant (1 - p_y(\mathbf{n})) \leqslant -\log_2 p_y(\mathbf{n})$, and hence misclassification impurity is upper bounded by Gini index impurity, and that is upper bounded by entropy impurity. All three impurities are concave and hence $\widetilde{F}(\mathbf{n}, \mathbf{q}) \geqslant 0$.

Remark 6. Misclassification is linear and hence impurity may not decrease by splitting. Gini index and entropy are strictly concave, and hence lead to strict decrease in impurity by splitting. Further, the latter two are differentiable and hence amenable to optimization.

1.2.3 Greedy technique

Definition 1.5. We denote by $F(\mathbf{n})$ the impurity of $\mathbf{n}$. The decrease in node impurity after a split of node $\mathbf{n}$ based on question $\mathbf{q}$ is defined as follows. Let $\mathbf{n}_+(\mathbf{n}, \mathbf{q})$ denote the right child of $\mathbf{n}$ after the split, $\mathbf{n}_-(\mathbf{n}, \mathbf{q})$ the left child, and $\eta(\mathbf{n}, \mathbf{q}) \triangleq \frac{\mathbf{n}_-(\mathbf{n}, \mathbf{q})}{\mathbf{n}}$ the fraction of the points in the region defined by $\mathbf{n}$ that are moved to $\mathbf{n}_-(\mathbf{n}, \mathbf{q})$. The total impurity of the leaves $\mathbf{n}_-(\mathbf{n}, \mathbf{q})$ and $\mathbf{n}_+(\mathbf{n}, \mathbf{q})$ is therefore $\eta(\mathbf{n}, \mathbf{q}) F(\mathbf{n}_-(\mathbf{n}, \mathbf{q})) + (1 - \eta(\mathbf{n}, \mathbf{q})) F(\mathbf{n}_+(\mathbf{n}, \mathbf{q}))$. Thus, the decrease in impurity $F(\mathbf{n}, \mathbf{q})$ by that split is given by

$$\widetilde{F}(\mathbf{n}, \mathbf{q}) = F(\mathbf{n}) - [\eta(\mathbf{n}, \mathbf{q}) F(\mathbf{n}_-(\mathbf{n}, \mathbf{q})) + (1 - \eta(\mathbf{n}, \mathbf{q})) F(\mathbf{n}_+(\mathbf{n}, \mathbf{q}))].$$

The method consists of starting with a tree reduced to a single (root) node, which is a leaf whose label is the class that has majority over the entire sample. Next, at each round, a node $\mathbf{n}_t$ is split based on some question $\mathbf{q}_t$. The pair $(\mathbf{n}_t, \mathbf{q}_t)$ is chosen so that the *node impurity* is maximally decreased according to some measure of impurity F. The algorithm is stopped when the nodes have sufficiently low impurity or the number of examples at each leaf node are sufficiently low.

Remark 7. Issues with greedy splitting.

1. One bad split may dominate subsequent useful splits, leading to local minimum of impurity. A possible solution is to use a look-ahead d depth for splitting decisions. However, this is computationally costly.
2. To achieve a desired level of impurity, large depth of trees maybe needed. However, these trees define complex hypotheses of large VC-dimensions and could be overfitting.

Algorithm 1 Greedy decision trees

```
1: procedure GreedyDecisionTrees($S = ((x_1, y_1), \dots, (x_m, y_m))$)
2:     tree $\leftarrow \{\mathbf{n}_0\}$                                ▷ root node
3:     for $t \leftarrow 1 : T$ do
4:         $(\mathbf{n}_t, \mathbf{q}_t) \leftarrow \arg\max_{\mathbf{n},\mathbf{q}} \widetilde{F}(\mathbf{n}, \mathbf{q})$
5:         SPLIT(tree,$\mathbf{n}_t, \mathbf{q}_t$)
6:     return tree
```

1.2.4 Grow-then-prune strategy

First a very large tree is grown until it fully fits the training sample or until no more than a very small number of points are left at each leaf. Then, the resulting tree, denoted as **tree**, is pruned back to minimize an objective function defined (based on generalization bounds) as the sum of an empirical error and a complexity term. The complexity can be expressed in terms of the size of $\widetilde{\mathbf{tree}}$, the set of leaves of tree. The resulting objective is

$$G_\lambda(\mathbf{tree}) = \sum_{n \in \widetilde{\mathbf{tree}}} |\mathbf{n}| \, F(\mathbf{n}) + \lambda \left| \widetilde{\mathbf{tree}} \right|,$$

where $\lambda \geqslant 0$ is a regularization parameter determining the trade-off between misclassification, or more generally impurity, versus tree complexity. For any tree $\mathbf{tree}'$, we denote by $\hat{R}(\mathbf{tree}')$ the total empirical error $\sum_{\mathbf{n} \in \widetilde{\mathbf{tree}'}} |\mathbf{n}| \, F(\mathbf{n})$. We seek a sub-tree $\mathbf{tree}_\lambda$ of **tree** that minimizes G_λ and that has the smallest size, which can be shown to be unique. To determine $\mathbf{tree}_\lambda$, the following pruning method is used, which defines a finite sequence of nested sub-trees $\mathbf{tree}^{(0)}, \dots, \mathbf{tree}^{(n)}$. We start with the full tree $\mathbf{tree}^{(0)} = \mathbf{tree}$ and for any $i \in \{0, \dots, n-1\}$, define $\mathbf{tree}^{(i+1)}$ from $\mathbf{tree}^{(i)}$ by collapsing an internal node $\mathbf{n}'$ of $\mathbf{tree}^{(i)}$, that is by replacing the sub-tree rooted at $\mathbf{n}'$ with a leaf, or equivalently by combining the regions of all the leaves dominated by $\mathbf{n}'$, chosen so that collapsing it causes the smallest per node increase in $\hat{R}(\mathbf{tree})$, that is the smallest $r(\mathbf{tree}^{(i)}, \mathbf{n}')$ is defined by

$$r(\mathbf{tree}^{(i)}, \mathbf{n}') \triangleq \frac{|\mathbf{n}'| \, F(\mathbf{n}') - \hat{R}(\mathbf{tree}')}{\left| \widetilde{\mathbf{tree}'} \right| - 1},$$

where $\mathbf{n}'$ is an internal node of $\mathbf{tree}^{(i)}$. If several nodes $\mathbf{n}'$ in $\mathbf{tree}^{(i)}$ cause the same smallest increase per node $r(\mathbf{tree}^{(i)}, \mathbf{n}')$, then all of them are pruned to define $\mathbf{tree}^{(i+1)}$ from $\mathbf{tree}^{(i)}$. This procedure continues until the tree $\mathbf{tree}^{(n)}$ obtained has a single node. The sub-tree $\mathbf{tree}_\lambda$ can be shown to be among the elements of the sequence $\mathbf{tree}^{(0)}, \dots, \mathbf{tree}^{(n)}$. The parameter λ is determined via n-fold cross-validation.

Remark 8. Decision trees seem relatively easy to interpret, and this is often underlined as one of their most useful features.

Remark 9. Decision trees are unstable: small changes in the training data may lead to very different splits and thus entirely different trees, as a result of their hierarchical nature.

1.3 Random Forest

A forest is a collection of trees. Trees are trimmed in a random fashion.

1. For a sample $S \in (\mathcal{X} \times \mathcal{Y})^m$, choose a sub-sample S' randomly from S
2. For each example $(x_i, y_i) \in S'$ such that $\mathcal{X} \subseteq \mathbb{R}^d$, pick $d' \leqslant \sqrt{d}$ features randomly
3. Form a decision tree T_1 using sample S' and features d' using greedy method
4. Repeat the process and form new decision trees $T_1, T_2, \dots, T_\ell$
5. Classify new example x using all ℓ decision trees, and overall classification using majority decision

Remark 10. Random forest can provide good performance on average. Choice of $d' \leqslant \sqrt{d}$ ensures that different trees are weakly dependent.

Remark 11. Very limited theoretical analysis of random forests, partly due to use of greedy algorithms. Easier analysis using random decision trees instead of greedy decision trees.

2 Aggregated multi-class algorithms

An alternative approach is to reduce the multi-class classification to multiple binary classification tasks. A binary classification algorithm is then trained for each of these tasks independently, and the multi-class predictor is defined as a combination of the hypotheses returned by each of these algorithms. We first discuss two standard techniques for the reduction of multi-class classification to binary classification, and then show that they are both special instances of a more general framework.

2.1 One-versus-all (OVA)

We assume $\mathcal{Y} = [k]$ and we have k binary classifiers $h_y : \mathcal{X} \to \{-1,1\}$ for each $y \in \mathcal{Y}$. A SVM based hypothesis is $h_y(x) = \text{sign}(f_y(x))$ where scoring function $f_y(x) \triangleq \langle \mathbf{w}_y, \Phi(x) \rangle$ for all $x \in \mathcal{X}$.

Definition 2.1. Then, the OVA multi-class hypothesis $h : \mathcal{X} \to \mathcal{Y}$ is defined for each $x \in \mathcal{X}$ as

$$h(x) \triangleq \arg\max_{y \in \mathcal{Y}} f_y(x).$$

Remark 12. For uncombined algorithms, scoring functions $(f_y : y \in \mathcal{Y})$ are learned together, and are learned independently for aggregated algorithms.

Remark 13. Scoring functions $f_y(x)$ can be interpreted as estimates of $P(\{X = x\} \mid \{Y = y\})$.

Remark 14. In general, the scores given by functions f_y are not comparable for labels $y \in \mathcal{Y}$ and the OVA technique admits no principled justification. This is sometimes referred to as a *calibration problem*.

Remark 15. When it is justifiable, the OVA technique is simple and its computational cost is k times that of training a binary classification algorithm, which is similar to the computation costs for many uncombined algorithms.

2.2 One-versus-one (OVO)

Another approach is to use the training data to independently learn, for each pair of distinct classes $(y,y') \in \mathcal{Y} \times \mathcal{Y}$, a binary classifier $h_{yy'} : \mathcal{X} \to \{-1,1\}$ discriminating between classes y and y'. For any pair of labels y,y', the binary classifier $h_{yy'}$ is obtained by training a binary classification algorithm on the sub-sample containing exactly the points labeled with y or y', with the value $+1$ returned for class y' and -1 for class y. This requires training $\binom{k}{2} = k(k-1)/2$ classifiers, which are combined to define a multi-class classification hypothesis $h : \mathcal{X} \to \mathcal{Y}$ via majority vote

$$h(x) \triangleq \arg\max_{y' \in \mathcal{Y}} \left| \left\{ y \in \mathcal{Y} : h_{yy'}(x) = 1 \right\} \right|.$$

Remark 16. For a fixed example, the binary classifier $h_{yy'}(x)$ indicates winning of y' over y, and h predicts the label with largest number of wins.

Remark 17. Let $x \in \mathcal{X}$ be a point belonging to class y'. By definition of the OVO, if $h_{yy'}(x) = 1$ for all $y \neq y'$, then the class associated to x by OVO is the correct class y' since $\left| \left\{ y : h_{yy'}(x) = 1 \right\} \right| = k - 1$ and no other class can reach $(k-1)$ wins. By contraposition, if the OVO hypothesis misclassifies x, then at least one of the $(k-1)$ binary classifiers $h_{yy'}$ incorrectly classifies x. If the generalization error of all binary classifiers $h_{yy'}$ used by OVO is at most r, then the generalization error of the hypothesis returned by OVO is at most $(k-1)r$.

Remark 18. The OVO approach does not suffer from the calibration problem that arises for OVA. However, when the size of the sub-sample containing members of the classes y and y' is relatively small, $h_{yy'}$ may be learned without sufficient data or with increased risk of overfitting. Another concern often raised for the use of OVO is the computational cost of training $k(k-1)/2$ binary classifiers versus $k/2$ in OVA.

Remark 19. However, training the binary classifiers for OVO maybe less time consuming due to smaller sample size.

2.3 Error-correcting output codes

For each label $y \in \mathcal{Y}$ assign a generator matrix $\mathbf{M}_y \in \{-1,1\}^c$ which is yth row of matrix $\mathbf{M} \in \{-1,1\}^{k\times c}$. For each column $j \in [c]$, we can define set $A_j \triangleq \{y \in \mathcal{Y} : \mathbf{M}_{yj} = 1\}$, and a binary classifier $h_j : \mathcal{X} \to \{-1,1\}$ is learnt using sample S such that all states $y \in A_j$ are labeled 1 and in A_j^c are labeled -1. For any $x \in \mathcal{X}$, we denote $\mathbf{h}(x) \triangleq \begin{bmatrix} h_1(x) & \cdots & h_c(x) \end{bmatrix}$, and the multi-class hypothesis is defined as

$$h(x) \triangleq \arg\min_{y\in\mathcal{Y}} d_H(\mathbf{M}_y, \mathbf{h}(x))$$

Remark 20. Success of ECOC depends on the minimum Hamming distance between class code words, defined as $d \triangleq \min\left\{d_H(\mathbf{M}_y, \mathbf{M}_{y'}) : y \neq y' \in \mathcal{Y}\right\}$.

Remark 21. Up to $r_0 = \left\lfloor \frac{d-1}{2} \right\rfloor$ binary classification errors can be corrected by ECOC. By definition of distance d, even if $r < r_0$ binary classifiers h_j misclassify $x \in \mathcal{X}$, $h(x)$ is closest to the code word of the correct class of x. For a fixed codeword length c, the design of error-correction matrix $\mathbf{M}$ is subject to a trade-off, since larger d values may imply substantially more difficult binary classification tasks. In practice, each column may correspond to a class feature determined based on domain knowledge.

2.3.1