CSE517A Machine Learning	**Spring 2020**

Practical ML: Multi-class Classification

Instructor: Marion Neumann — *Scribe: Jingyu Xin*

Reading:

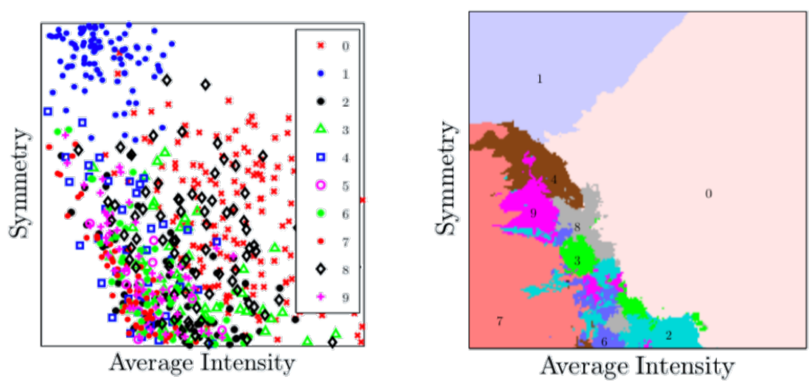
1 Introduction

Some ML applications/datasets have more than two classes and require us to perform multi-class classification. Some examples are handwritten digit classification (cf. Figure 1), where $\mathcal{C} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ or image-based plant disease classification, where the goal is to identify a *specific* disease (cf. Figure 2).

Given a data set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $y_i \in \{1, \dots, k\}$. Which model will you choose to do *multi-class* classification?

Recall: k-NN, Naive Bayes, Decision Trees, and Random Forests, and verify that these models can inherently deal with more than two classes.

Figure 1 (from [LFD CH6.2][1]) illustrates a k-NN solution for the multi-class handwritten digits problem using a simple set of $2d$ features.



(a) Multiclass digits data (b) 21-NN decision regions

Figure 1: 21-NN decision classifier for the multi-class digits data. [LFD CH6.2]

For **logistic regression** we can simply use the following *conditional likelihood* in a MLE or MAP approach (cf. Section 4 below):

$$p(y = c \mid \mathbf{x}, \mathbf{w}_1, \dots, \mathbf{w}_k) = \frac{e^{\mathbf{w}_c^\top \mathbf{x}}}{\sum_{c=1}^k e^{\mathbf{w}_c^\top \mathbf{x}}}.$$

However, some learning methods do not *easily* extend to multi-class models such a support vector machines (SVMs) and Gaussian processes (GPs). For these cases we can cast multi-class classification as **multiple binary classification problems**.

[1] [LFD CH6.2] Learning From Data, Abu-Mostafa et al., 2012, AMLBook

2 1-vs-all Multi-class Classification

Use K classification problems on $\tilde{y}_i = \begin{cases} +1 & \text{if } y_i = c \\ -1 & \text{otherwise} \end{cases}$ for $c \in \{1, \dots, k\}$.

$\Rightarrow$ we get $f_1, \dots, f_k$ models:

$$h(\mathbf{x}) = \arg\max_c f_c(\mathbf{x}) \tag{1}$$

assign the class label of the **most confident** model.

Note:

- only works if classifier **output is comparable** $\rightarrow$ for logistic regression and probabilistic classifiers such as GPs; does not work for SVMs
- for large K, we get class imbalanced

Platt Scaling for SVMs

Since for SVMs the output of different classifiers is not comparable (they are not normalized probabilities), we need to apply a scaling trick to transform the predictions into a probability distribution over classes. This method is commonly referred to as *Platt scaling*.

Let $z_i = f(\mathbf{x}_i) \quad \forall i = 1, \dots, n$ be the SVM predictions before squashing them through the sign function $h(\mathbf{x}) = \text{sign}(f(\mathbf{x})) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$.

Now, build k datasets $\tilde{D}^{(1)} = \{(z_i, \tilde{y}_i)\}_{i=1}^n, \dots, \tilde{D}^{(k)} = \{(z_i, \tilde{y}_i)\}_{i=1}^n$.
In $\tilde{D}^{(c)}$, $\tilde{y}_i = \begin{cases} +1 & \text{if } y_i = c \\ -1 & \text{otherwise} \end{cases}$ for $c \in \{1, \dots, k\}$. Use those datasets to train k **logistic regression models** $\tilde{h}_c(z) = \tilde{h}_c(f(\mathbf{x}))$ which rescale the original output $f(\mathbf{x})$. Then, use

$$H(\mathbf{x}) = \arg\max_c \tilde{h}_c(z) \tag{2}$$

3 1-vs-1 Multi-class Classification

Create $\frac{k(k-1)}{2}$ binary classifiers using i vs j $\forall i, j \in \{1, \dots, k\}$.

Assign final label based on (weighted) majority vote. For example:

	1	2	3	4	5
1	N/A	+	+	-	+
2	-	N/A	-	+	+
3	-	+	N/A	-	-
4	+	-	+	N/A	-
5	-	-	+	+	N/A

Table 1: Example of majority vote

The label assigned should be $c = 1$.

Figure 2 illustrates a multi-class plant disease classification approach. The classifier is a one-vs-one multi-class support vector machine (SVM) using a radial basis function (RBF) kernel [NEUMANN, ICPR 2014].[2]

[2] [NEUMANN, ICPR 2014] Erosion Band Features for Cell Phone Image Based Plant Disease Classification, M. Neumann, L. Hallau, B. Klatt, K. Kersting, C. Bauckhage, ICPR 2014

Figure 2: Image-based plant disease classification of sugar beet leaves. Leaf spots can be caused by Cercospora beticola (*cerc*), Ramularia beticola (*ram*), Pseudomonas syringae (*pseu*), Uromyces betae (*rust*), or Phoma betae (*phom*).

Advantages:

- each problem is class balanced
- each problem is small (few training data)

Disadvantages:

- run time is $\mathcal{O}(k^2)$
- less straight forward to implement

4 [optional] Multi-class Logistic Regression

For logistic regression we can implementation 1-vs-all directly as our usual MLE estimation by minimizing the negative log-likelihood. Our prediction model is:

$$p(y = c \mid \mathbf{x}, \mathbf{w}_1, \dots, \mathbf{w}_k) = \frac{e^{\mathbf{w}_c^\top \mathbf{x}}}{\sum_{l=1}^{k} e^{\mathbf{w}_l^\top \mathbf{x}}} \tag{3}$$

$$h(\mathbf{x}) = \arg\max_{c} \ e^{\mathbf{w}_c^\top \mathbf{x}} = \arg\max_{c} \ \mathbf{w}_c^\top \mathbf{x}, \tag{4}$$

where the $\mathbf{w}_c$ indicate k hyperplanes.

<u>**Training**</u>: minimize *negative log-likelihood*:

$$\begin{aligned}
\arg\min_{W} \ L(W) &= \arg\min_{\mathbf{w}_1,\dots,\mathbf{w}_k} \ L(\mathbf{w}_1, \dots, \mathbf{w}_k) \\
&= \arg\min_{\mathbf{w}_1,\dots,\mathbf{w}_k} \ -\log p(\mathbf{y} \mid X, \mathbf{w}_1, \dots, \mathbf{w}_k) \\
&= \arg\min_{\mathbf{w}_1,\dots,\mathbf{w}_k} \ -\log \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \mathbf{w}_1, \dots, \mathbf{w}_k)
\end{aligned} \tag{5}$$

Note: $p(y \mid \mathbf{x}, \mathbf{w})$ is a multinoulli distribution (one experiment with k categorical outcomes):

$$p(y = c \mid \mathbf{x}, \mathbf{w}) = \prod_{c=1}^{k} \theta_c^{I(y=c)}.$$

Pluging this into Eq. (5), we get

$$\begin{aligned}
\arg\min_{W} \; L(W) &= \arg\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k} \; -\log \prod_{i=1}^{n} \prod_{c=1}^{k} \left(\frac{e^{\mathbf{w}_c^\top \mathbf{x}_i}}{\sum_{l=1}^{k} e^{\mathbf{w}_l^\top \mathbf{x}_i}} \right)^{I(y_i=c)} \\
&= \arg\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k} \; \sum_{i=1}^{n} \left[\log \left(\sum_{l=1}^{k} e^{\mathbf{w}_l^\top \mathbf{x}_i} \right) - \sum_{c=1}^{k} I(y_i = c) \mathbf{w}_c^\top \mathbf{x}_i \right]
\end{aligned} \tag{6}$$

Now, we can take the derivatives w.r.t. $\mathbf{w}_1, \ldots, \mathbf{w}_k$ to get the gradient:

$$g(W) = \sum_{i=1}^{n} \left(\begin{bmatrix} \mu_{i1} \\ \vdots \\ \mu_{ik} \end{bmatrix} - \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \end{bmatrix} \right) \otimes \mathbf{x}_i \tag{7}$$

where

$$\mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_1 \mathbf{b} \\ a_2 \mathbf{b} \\ \vdots \\ a_k \mathbf{b} \end{bmatrix}$$

Just looking at the portion of $g(W)$ corresponding to $\mathbf{w}_c$, i.e. the derivative of $L(W)$ w.r.t. $\mathbf{w}_c$, we have:

$$\nabla_{\mathbf{w}_c} L(W) = \sum_i \left(\frac{e^{\mathbf{w}_c^\top \mathbf{x}_i}}{\sum_{l=1}^{k} e^{\mathbf{w}_l^\top \mathbf{x}_i}} - I(y_i = 0) \right) \mathbf{x}_i \tag{8}$$

The Hessian can be derived similarity and both expressions can be plugged into any gradient-based optimizer to get the MLE estimate.