Multiclass Classification

Tirgul 8
MNIST Example

• Binary Problem: 1 vs 8
MNIST Example

- Multiclass Problem – 1 vs. 7 vs. 8
Multiclass Classification

• Given:
  • A training set: \( S = \{(x_1, y_1), \ldots (x_m, y_m)\}, \)
    • Where:
      • \( x_i \in \mathbb{R}^d \) (i.e., the examples are vectors of length \( d \))
      • \( y_i \in \mathcal{Y} = \{1, 2, \ldots, k\} \) (i.e., the labels are from a set of \( k \) possible classes)

• Each training example belongs to one of \( k \) different classes.
• Goal: construct a function that, given a new data point, will correctly predict the class to which the new point belongs.
Multiclass Classification – cont.

• Two general techniques to build a multiclass classifier:

  1. **Binary Reduction:**
     Reduction of the multiclass problem to several binary classification problems.

  2. **Multiclass:**
     Design of a multiclass classifier.
Binary Reductions
One Against All - Intuition
One Against All

• The One-Against-All method:
  • Based on a reduction of the multiclass problem into $k$ binary problems
  • Each problem discriminates between one class to all the rest.
One Against All

• The Method:
  • We now have \( k \) binary problems:
    • Binary problem #1:
      • Assign label +1 to all examples labeled 1
      • Assign label -1 to all examples labeled 2, 3, ..., k.
        • The resulting **weight vector**: called \( w^1 \)
    • Binary problem #2:
      • Assign label +1 to all examples labeled 2
      • Assign label -1 to all examples labeled 1, 3, 4, ..., k.
        • The resulting **weight vector**: called \( w^2 \)
    • And so on...
One Against All

• ... we end up with $k$ weight vectors.

• Given a new instance $x$, predict the label that gets the highest confidence:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} w^y \cdot x$$
Example
All Pairs - Intuition
All Pairs

• In this approach, all pairs of classes are compared to each other.

• The Method:
  • We now have $\binom{k}{2} = \frac{(k - 1)k}{2}$ binary problems:
    • Binary problem #1: 1 vs 7
      • Assign label +1 to all examples labeled 1
      • Assign label -1 to all examples labeled 7
      • The resulting weight vector: called $\mathbf{w}^{17}$
    • Binary problem #2: 1 vs 8
      • Assign label +1 to all examples labeled 1
      • Assign label -1 to all examples labeled 8
      • The resulting weight vector: called $\mathbf{w}^{18}$
    • And so on...
All Pairs

• ... we end up with \((k - 1)k/2\) weight vectors.

• Given a new instance \(x\), predict the label that gets the highest confidence:

\[
\hat{y} = \arg \max_{y \in Y} \sum_{r \neq y} w_{yr} \cdot x
\]
Error Correction Output Codes (ECOC)

• The Method:
  • Define a coding matrix $M \in \{-1, 0, 1\}^{k \times l}$
    • $k$ is the number of classes
    • $l$ is the number of hypotheses (/binary problems).
    • (Entries $M(r, s) = 0$ indicate that we don’t care how hypothesis $f_s$ classifies examples labeled $r$.)
  • Given a learning algorithm A:
    • for $s = 1, \ldots, l$, algorithm A is provided with labeled data of the form: $(x_i, M(y_i, s))$ for all examples in the training set (except for those with $M(y_i, s) = 0$).
    • Algorithm A uses this data to generate $l$ hypotheses $f_s$.
  • Classify an input $x$ to be the class whose representation by matrix $M$ is closest to the output of the binary classifier $\text{sign}(f(x))$. 
ECOC

• Suppose we choose a $k \times k$ matrix $M$:
  • All diagonal elements are +1
  • All other elements are -1

\[
M = \begin{bmatrix}
  +1 & -1 & -1 \\
  -1 & +1 & -1 \\
  -1 & -1 & +1
\end{bmatrix}
\]

• ... this is the One-Against-All approach.
ECOC

• The two approaches of One-Against-All and All-Pairs have been unified under the framework of ECOC.

• All-Pairs Approach:
  • \( M \) is a \( k \times \binom{k}{2} \) matrix
    • Each column corresponds to a distinct pair \((r_1, r_2)\).
    • For this column, \( M \) has:
      • +1 in row \( r_1 \)
      • -1 in row \( r_2 \)
      • Zeros in all other rows
ECOC Example

• Given:
  • An input example $x$
  • $l$ binary classifiers

• Suppose the outputs of the binary classifiers are as follows:
  • $f_1(x) = 0.5$
  • $f_2(x) = -7$
  • $f_3(x) = -1$
  • $f_4(x) = -2$
  • $f_5(x) = -10$
  • $f_6(x) = -12$
  • $f_7(x) = -9$

$\Rightarrow$

In short, the vector of predictions is:

$$f(x) = [0.5, -7, -1, -2, -10, -12, 9]$$
ECOC Example

• Let the coding matrix $M$ be:

$$M = \begin{bmatrix}
-1 & 0 & -1 & -1 & +1 & -1 & -1 \\
+1 & -1 & 0 & +1 & +1 & +1 & -1 \\
+1 & 0 & -1 & -1 & -1 & +1 & +1 \\
-1 & -1 & +1 & 0 & -1 & -1 & +1 
\end{bmatrix}$$

  class #1  class #2  class #3  class #4

• The vector of predictions was:

$$f(x) = [0.5, -7, -1, -2, -10, -12, 9]$$

$$\text{sign}(f(x)) = [+1, -1, -1, -1, -1, -1, +1]$$
ECOC Example

• Given the predictions of $f(x)$ on test point $x$:
  • Which of the $k$ labels in $\mathcal{Y}$ should be predicted?

• The basic idea:
  • Predict the label $r$ whose row $M(r)$ is “closest” to the predictions $f(x)$.
  • In other words:
    • Predict the label $r$ that minimizes: $d(M(r), f(x))$, for some distance function $d$. 
ECOC Example

• How do we measure the distance between the two vectors?

• **Method #1:**
  
  • Count the number of positions $s$ in which the sign of the prediction $f_s(x)$ is different than the matrix entry $M(r, s)$.
  
  • Called: *Hamming Decoding*
ECOC Example

• Hamming Decoding:

\[
d_H(M(r), f(x)) = \sum_{s=1}^{l} \left( \frac{1 - \text{sign}(M(r,s) \cdot f_s(x))}{2} \right)
\]

• Simply put:

\[
d_H(M(r), f(x)) = \sum_{s=1}^{l} \left\{ \begin{array}{ll}
0 & \text{if } M(r) = f(x) \land M(r) \neq 0 \land f(x) \neq 0 \\
1 & \text{if } M(r) \neq f(x) \land M(r) \neq 0 \land f(x) \neq 0 \\
1/2 & \text{if } M(r) = 0 \lor f(x) = 0
\end{array} \right.
\]

\[
\text{sign}(z) = \begin{cases}
+1 & \text{if } z > 0 \\
-0 & \text{if } z < 0 \\
0 & \text{if } z = 0
\end{cases}
\]
ECOC Example

• For an instance $x$ and a matrix $M$, the predicted label is:

$$\hat{y} = \arg \min_{y \in \mathcal{Y}} d_H(M(r), f(x))$$

• A disadvantage:
  • Hamming decoding ignores the magnitude of the predictions, which can often be an indication of a level of “confidence” of the classifier.
ECOC Example

• Back to our example...

\[ M = \begin{bmatrix}
-1 & 0 & -1 & -1 & +1 & -1 & -1 \\
+1 & -1 & 0 & +1 & +1 & +1 & -1 \\
+1 & 0 & -1 & -1 & -1 & +1 & +1 \\
-1 & -1 & +1 & 0 & -1 & -1 & +1 \\
\end{bmatrix} \quad d_H = 3.5 \]

\[ d_H(M(r), f(x)) = \sum_{s=1}^{l} \left( \frac{1 - \text{sign}(M(r, s) \cdot f_s(x))}{2} \right) \]

\[ f(x) = [0.5, -7, -1, -2, -10, -12, 9] \]

• Hence, our prediction here is **class 3** (the minimal Hamming distance).
ECOC Example

• How do we measure the distance between the two vectors?
  • Recall Hamming’s disadvantage:
    • Ignores the magnitude of the predictions, which can often be an indication of a level of “confidence” of the classifier.

• Method #2:
  • Takes into account:
    • The magnitude of predictions
    • A relevant loss function $L$
  • The idea:
    • Choose the label $r$ that minimizes the loss on example $x$.
    • Called loss-based decoding
ECOC Example

• Loss-based decoding:

\[ d_L(M(r), f(x)) = \sum_{s=1}^{l} L(M(r, s) \cdot f_s(x)) \]

• The loss:

\[ L(z) = \begin{cases} 
\max \{0, 1 - z\} \\
\exp(-z) \\
\log(1 + \exp(-z)) 
\end{cases} \]
ECOC Example

• For an instance $x$ and a matrix $M$, the predicted label is:

\[
\hat{y} = \arg \min_{y \in \mathcal{Y}} d_L(M(r), f(x))
\]
ECOC Example

• Back to our example...

\[
M = \begin{bmatrix}
-1 & 0 & -1 & -1 & +1 & -1 & -1 \\
+1 & -1 & 0 & +1 & +1 & +1 & -1 \\
+1 & 0 & -1 & -1 & -1 & +1 & +1 \\
-1 & -1 & +1 & 0 & -1 & -1 & +1
\end{bmatrix}
\]

\[
f(x) = [0.5, -7, -1, -2, -10, -12, 9]
\]

\[
d_L(M(r), f(x)) = \sum_{s=1}^{l} L(M(r, s) \cdot f_s(x))
\]

\[
L(z) = \exp(-z)
\]

\[
d_L = \exp(-(-1 * 0.5)) + \exp(-(0 * -7)) + ... = 30,132
\]

\[
d_L = 192,893
\]

\[
d_L = 162,757
\]

\[
d_L = 5.4
\]

• Hence, our prediction here is **class 4** (the minimal loss-based distance).
Multiclass
Multiclass Perceptron

• A direct extension of the binary Perceptron.

Binary Perceptron:
\[
\hat{y} = \text{sign} (w \cdot x_i)
\]
if \( \hat{y} \neq y_i \):
\[
w \leftarrow w + y_i x_i
\]

Extended Binary Perceptron:
\[
\hat{y} = \arg \max_{y \in \{-1,+1\}} w^y \cdot x_i
\]
if \( \hat{y} \neq y_i \):
\[
w^y \leftarrow w^y + x_i
\]
\[
w^{\hat{y}} \leftarrow w^{\hat{y}} - x_i
\]

Both algorithms are mathematically the same, where \( w^y = - w^{\hat{y}} \)
Multiclass Perceptron

• Extended Binary Perceptron:
  • The update rule

\[ \mathbf{w}^+ + \mathbf{x} - \mathbf{w}^- - \mathbf{x} = \mathbf{w}^+ + \mathbf{x} - \mathbf{w}^- - \mathbf{x} \]
Multiclass Perceptron

• Extension of Perceptron to the multiclass case:

\[
\hat{y} = \arg \max_{y \in \{1, \ldots, k\}} w^y \cdot x_i
\]

if \(\hat{y} \neq y_i\):

\[w^r \leftarrow w^r + \tau_r x_i\]  \(\text{for every class } r\)

Where:

\[
\tau_r = \begin{cases} 
+1 & \text{if } r = y_i \\
-1 & \text{if } r = \hat{y} \\
0 & \text{otherwise}
\end{cases}
\]

Here, we update only two classes:
• The correct class
• The predicted class
Multiclass Perceptron

• A different update rule:

\[ \hat{y} = \arg \max_{y \in \{1, \ldots, k\}} w^y \cdot x_i \]

if \( \hat{y} \neq y_i \):

\[ w^r \leftarrow w^r + \tau_r x_i \quad \text{(for every class } r) \]

Where:

\[ E_y = \{ r \neq y \mid w^r \cdot x > w^y \cdot x \} \]

\[ \tau_r = \begin{cases} +1 & \text{if } r = y_i \\ \frac{1}{|E_y|} & \text{if } r \in E_{y_i} \\ 0 & \text{otherwise} \end{cases} \]

Here, we update all classes that have a higher score than the correct one.
Multiclass SVM

• The model is of the following term:

\[ \hat{y} = \arg \max_{y \in \{1, \ldots, k\}} w^y \cdot x \]
Multiclass SVM

• We want to find the weight vectors \( \{w_1, \ldots, w_k\} \) such that the probability of an error \( \hat{y} \neq y \) is minimized:

\[
\{w^r\} = \arg \min_{\{w^r\}} P(\hat{y} \neq y)
\]

\[
= \arg \min_{\{w^r\}} E[\delta(\hat{y} \neq y)]
\]

• Replace the expectation with average, and add regularization:

\[
\{w^r\} = \arg \min_{\{w^r\}} \frac{1}{m} \sum_{i=1}^{m} \delta(\hat{y} \neq y) + \frac{\lambda}{2} \sum_{r=1}^{k} ||w^r||^2
\]
Multiclass SVM

• SVM works by upper-bounding the loss $\delta(\hat{y} \neq y)$ by a surrogate hinge loss:

\[
\delta(\hat{y} \neq y) = \delta(\hat{y} \neq y) - w^y \cdot x + w^y \cdot x \\
\leq \delta(\hat{y} \neq y) - w^y \cdot x + \max_{\hat{y}} [w^{\hat{y}} \cdot x] \\
\leq \max_{\hat{y}'} [\delta(\hat{y}' \neq y) - w^y \cdot x + w^{\hat{y}'} \cdot x]
\]
Multiclass SVM

• Multiclass SVM optimization problem:

\[
\{w^r\} = \arg\min_{\{w^r\}} \frac{1}{m} \sum_{i=1}^{m} \max_{\hat{y}'} [\delta(\hat{y}' \neq y_i) - w^{y_i} \cdot x_i + w^{\hat{y}'} \cdot x_i] + \frac{\lambda}{2} \sum_{r=1}^{k} ||w^r||^2
\]
Multiclass SVM

• The Algorithm (with Stochastic Gradient Descent):

\[
\hat{y}' = \underset{\hat{y}\in\{1,\ldots,k\}}{\arg\max} \left[ \delta(\hat{y}' \neq y_i) + w^{\hat{y}'} \cdot x_i \right]
\]

if \( \delta(\hat{y}' \neq y_i) - w^{y_i} \cdot x_i + w^{\hat{y}'} \cdot x_i \) > 0:

\[
w^{y_i} \leftarrow (1 - \lambda\eta_t) w^{y_i} + \eta_t x_i
\]

\[
w^{\hat{y}'} \leftarrow (1 - \lambda\eta_t) w^{\hat{y}'} - \eta_t x_i
\]

\[
w^{y} \leftarrow (1 - \lambda\eta_t) w^{y} \quad \text{for} \ y \neq \hat{y}', y_i
\]
Summary

• Binary reductions:
  • One against all
  • All-Pairs
  • ECOC
• Multiclass
  • Perceptron
  • SVM