

1 Introduction

In this lecture, we look at the multiclass learning problem. We learn a simple approach to solve it by turning multiclass classification to binary classification and discuss related issues. We also study direct approaches for learning multiclass predictors, and expand it by applying Stochastic Gradient Descent learning framework. At last, we discuss the structured output prediction problem.

2 Multiclass Learning

2.1 One-versus-All and All-Pairs

The simplest approach to handle the multiclass prediction problems is by reduction to binary classification. There are two algorithms in the approach, One-versus-All and All-Pairs.

2.1.1 One-versus-All

In the One-versus-All method (a.k.a One-versus-Rest, One-against-all) we train binary classifiers for each class. With those binary classifiers, we construct a multiclass predictor using the rule, $h(\mathbf{x}) \in \operatorname{argmax}_{i \in [k]} h_i(\mathbf{x})$. It means that we predict the class of \mathbf{x} as k when all values of binary classifiers are zero except for the value of k -th classifier, one. A pseudo-code of the One-versus-All approach is given below.

```

One-versus-All

input:
  training set  $S = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$ 
  algorithm for binary classification  $A$ 
foreach  $i \in \mathcal{Y}$ 
  let  $S_i = (\mathbf{x}_1, (-1)^{\mathbb{1}_{[y_1 \neq i]}}, \dots, (\mathbf{x}_m, (-1)^{\mathbb{1}_{[y_m \neq i]}})$ 
  let  $h_i = A(S_i)$ 
output:
  the multiclass hypothesis defined by  $h(\mathbf{x}) \in \operatorname{argmax}_{i \in \mathcal{Y}} h_i(\mathbf{x})$ 

```

Figure 1: One-versus-All Algorithm

There are problems in the One-versus-All approach.

- What if there are more than two binary classifiers returning one?
- What if all the binary classifier returns zero?

For these issue, we can decide the label as the one between the two classification classes, or we make our binary classifiers return real values, not the zero/one. However, it's still not robust because a single binary classification error can result in multiclass classification error.

2.1.2 All-Pairs

Another approach is the All-Pair method (a.k.a All-versus-All, All-against-All). In here, we compare all pairs of classes to each other like a basketball league. With k classes, for every $1 \leq i < j \leq k$, we construct a binary training sequence, $S_{i,j}$, containing all examples from S whose label is either i or j , and it will be classified as $+1$ or -1 . We train a binary classification algorithm based on every $S_{i,j}$ to get $h_{i,j}$, and in the end, we construct multiclass classifier by predicting the class that had the highest number of *wins*. Here is the pseudo-code of the All-Pairs method.

All-Pairs

input:
 training set $S = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$
 algorithm for binary classification A

foreach $i, j \in \mathcal{Y}$ s.t. $i < j$
 initialize $S_{i,j}$ to be the empty sequence
for $t = 1, \dots, m$
 If $y_t = i$ add $(\mathbf{x}_t, 1)$ to $S_{i,j}$
 If $y_t = j$ add $(\mathbf{x}_t, -1)$ to $S_{i,j}$
 let $h_{i,j} = A(S_{i,j})$

output:
 the multiclass hypothesis defined by
 $h(\mathbf{x}) \in \operatorname{argmax}_{i \in \mathcal{Y}} \left(\sum_{j \in \mathcal{Y}} \operatorname{sign}(j - i) h_{i,j}(\mathbf{x}) \right)$

Figure 2: All-Pairs Algorithm

Even if the All-Pair approach is better than the One-versus-All approach, there comes a question if it's appropriate to use binary learners in both methods to construct a multiclass predictor.

2.1.3 Problem of Reduction Methods

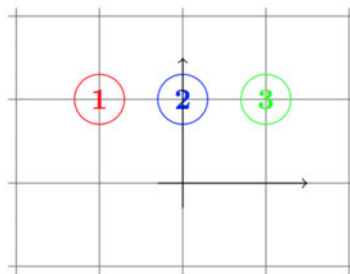


Figure 3: Multiclass Categorization Problem

For instance, consider a multiclass prediction problem where the instance space is $\mathcal{X} = \mathbb{R}^2$ and the labels are 1, 2, 3. Suppose instances of each class is located as depicted in Figure 5, and the probability masses of classes 1, 2, 3 are 40%, 20%, 40%. Once we use the One-versus-All method, our multiclass predictor might error on all the samples from class 2 because when we distinguish the class 2 from the rest of the classes, the optimal half-space would be the all negative classifier. Thus, the method cannot find a optimal predictor from this class.

2.2 Linear Multiclass Predictors

To address the issues in reduction methods, we take direct approach for learning multiclass predictors. We define multiclass predictor, $h : \mathcal{X} \rightarrow \mathcal{Y}$, as below. In the definition, we think of the elements of $\Psi(\mathbf{x}, y)$ as a score function that assess how well the label y fits the instance \mathbf{x} .

$$h(\mathbf{x}) = \operatorname{argmax}_{y \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle$$

2.2.1 Score Function

We can think, naturally, designing a nice score function is similar to the problem of designing a nice feature mapping. One useful construction method is *The Multivector Construction*. Let $\mathcal{Y} = 1, \dots, k$ and let $\xi = \mathbb{R}^n$. We define $\Psi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$, where $d = nk$, as follows:

$$\Psi(\mathbf{x}, y) = [\underbrace{0, \dots, 0}_{\in \mathbb{R}^{(y-1)n}}, \underbrace{x_1, \dots, x_n}_{\mathbb{R}^n}, \underbrace{0, \dots, 0}_{\in \mathbb{R}^{(k-y)n}}].$$

It means, the score function is composed of k vectors, each of which is of dimension n , where we set all the vectors to be the all zeros vector except the y 'th vector, which is set to be \mathbf{x} . It follows that we can think of $\mathbf{w} \in \mathbb{R}^{nk}$ as being composed of k weight vectors in \mathbb{R}^n , that is, $\mathbf{w} = [\mathbf{w}_1 ; \dots ; \mathbf{w}_k]$. By the construction we have that $\langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle = \langle \mathbf{w}_y, \mathbf{x} \rangle$, and therefore the multiclass prediction becomes

$$h(\mathbf{x}) = \operatorname{argmax}_{y \in \mathcal{Y}} \langle \mathbf{w}_y, \mathbf{x} \rangle \tag{1}$$

A geometric illustration of the multiclass prediction over $\mathcal{X} = \mathbb{R}^2$ is given in the following.

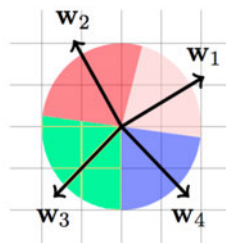


Figure 4: Geometric Intuition of Multiclass Prediction

The illustration implies:

- We're finding a vector maximizes $\langle \mathbf{w}, \mathbf{x} \rangle$.
- However, the illustration depicts a homogeneous case. Non-homogeneous case will be different.

Moreover, it's sometimes useful in the Multivector construction to add the common features to our vector. For example, we can think the problem of classifying cats, dogs, and iguana. Instead of using different features for each class, we can use, for instance, common features between cats and dogs during classification. It results in smaller norm of the vector \mathbf{w} .

2.2.2 Generalized Loss

Let $S(\mathbf{x}, y)$ is our score function. What we want to do is the score with the truth is always bigger than the score with any label by small amount:

$$\begin{aligned} S(\mathbf{x}, \underbrace{y}_{\text{truth}}) &\geq S(\mathbf{x}, \underbrace{y'}_{\text{any } y' \neq y}) + \text{small amount} \\ &\geq S(\mathbf{x}, y') + \Delta(y, y') \\ &\geq \operatorname{argmax}_{y'} S(\mathbf{x}, y') + \Delta(y, y') \end{aligned}$$

It means that we know, instead of beating all score function, it's better to beat the best score function with a little amount in terms of computations. A geometric intuition of this is given in the following.

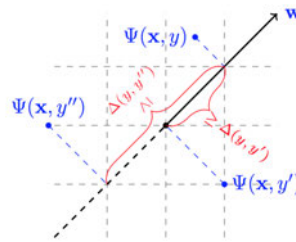


Figure 5: Geometric Intuition of Score Functions

This figure clearly shows what we want to do. For each $y' \neq y$, we project the value of $\langle \mathbf{w}, \Psi(\mathbf{x}, y') \rangle$ to \mathbf{w} . We want to make sure the projection with true label y , $\langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle$, to \mathbf{w} can defeat all the other projection values by $\Delta(y, y')$.

2.3 Multiclass SGD

We can also apply the SGD learning framework. Here is the pseudo-code for SGD for multiclass learning.

```

SGD for Multiclass Learning

parameters:
  Scalar  $\eta > 0$ , integer  $T > 0$ 
  loss function  $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$ 
  class-sensitive feature mapping  $\Psi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ 
initialize:  $\mathbf{w}^{(1)} = \mathbf{0} \in \mathbb{R}^d$ 
for  $t = 1, 2, \dots, T$ 
  sample  $(\mathbf{x}, y) \sim \mathcal{D}$ 
  find  $\hat{y} \in \operatorname{argmax}_{y' \in \mathcal{Y}} (\Delta(y', y) + \langle \mathbf{w}^{(t)}, \Psi(\mathbf{x}, y') - \Psi(\mathbf{x}, y) \rangle)$ 
  set  $\mathbf{v}_t = \Psi(\mathbf{x}, \hat{y}) - \Psi(\mathbf{x}, y)$ 
  update  $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \mathbf{v}_t$ 
output  $\bar{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^T \mathbf{w}^{(t)}$ 
    
```

Figure 6: SGD for Multiclass Learning

3 Structured Prediction

Features for structured prediction

➤ Allowed to encode *anything* you want

Pro	Md	Vb	Dt	Nn
I	can	can	a	can

$\phi(x, y) =$

I_Pro	: 1	<s>-Pro	: 1	has_verb	: 1
can_Md	: 1	Pro-Md	: 1	has_nn_lft	: 0
can_Vb	: 1	Md-Vb	: 1	has_n_lft	: 1
a_Dt	: 1	Vb-Dt	: 1	has_nn_rgt	: 1
can_Nn	: 1	Dt-Nn	: 1	has_n_rgt	: 1
...		Nn-</s>	: 1	...	

➤ Output features, Markov features, other features

- Now the input is a sentence I can can a can, and we try to put label to each word of the sentence. It's a kind of multi-class classification problem, but now we have tons of possible classes. For example, for speech tag problem, for each word there is 45 tags. For 10 words, it becomes 45^{10} possible classes. The number of possible classes grows exponentially to the number of words.
- The generalization bound does not depend on the number of classes.
- (x, y) pair: x is the sentence I can can a can. y is the structured output Pro Md ... Nn.
- We want to extract features about this (x, y) pair. Features should tell us is y a good labeling of x ?
- Output feature: Blue part in the slides mean we tag the word I by a Pro one time and so on so forth. Also known as node feature.
- Markov feature: Green part in the slides. Tells us the relation between the neighboring word. For example, we do not expect Dt follows by another Dt, but we expect Dt follows by a Nn.
- Other feature: Brown part in the slides. Sometimes also called global features. For example, most English sentence has a Vb.

Structured perceptron

Enumeration over 1..K

Enumeration over all outputs

➤ For $n=1..N$:

➤ Predict:

$$\hat{y} = \arg \max_k \mathbf{w} \cdot \phi(x_n, k)$$

➤ If $\hat{y} \neq y_n$:

$$\mathbf{w} = \mathbf{w} - \phi(x_n, \hat{y}) + \phi(x_n, y_n)$$

➤ For $n=1..N$:

➤ Predict:

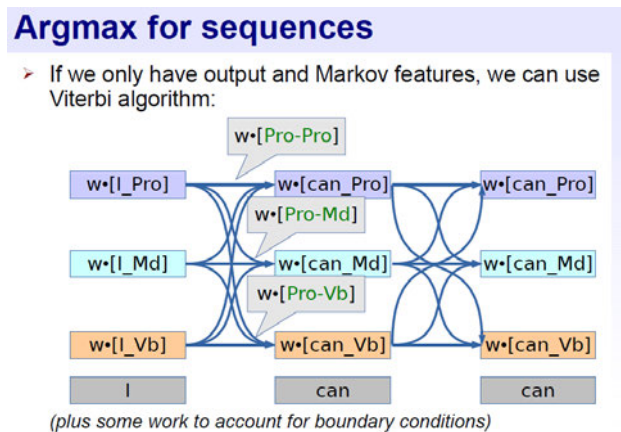
$$\hat{y} = \arg \max_k \mathbf{w} \cdot \phi(x_n, k)$$

➤ If $\hat{y} \neq y_n$:

$$\mathbf{w} = \mathbf{w} - \phi(x_n, \hat{y}) + \phi(x_n, y_n)$$

[Collins, EMNLP02]

- We make prediction and get \hat{y} . If the prediction is wrong, i.e. $\hat{y} \neq y_n$, then we make an update.
- What's the difference between the perceptron version and SGD version? The main difference is *argmax* in perceptron just finds the highest score; however, in SGD, *argmax* finds the highest score among the postulates after postulates boosting their scores.
- Although the algorithm is same as before, the complexity for the calculation of the *argmax* is quite different. For example, previous multi-class classification the number of class might just 20; however, for the structured prediction problem the number might be 45^{10} for just 10 words sentence.



- The answer for the calculation of the *argmax* for the sentence is dynamic programming.
- We assume the features we have are node feature and Markov feature. The following are similar to Viterbi algorithm.
- Among y-axis are the possible labels. Each combination of possible labeling to I can can corresponds to a path through the tellis. Target is to find out the maximum weight path among the trellis. We put the weight on the node corresponding to the node feature and put the weight to the edges between the nodes corresponds to the Markov feature.
- For example, if we put I as Pro then we expect the dot product gives a higher score than we put I as a Md. Another example is Pro follows by a Pro. We expect to get lower score then Pro follows by a Md.
- Complexity is $O(nk^2)$. n is the length of the sequence. k is the number of labels. k^2 accounts for the number of edges.

Structured perceptron as ranking

- For $n=1..N$:
 - Run Viterbi: $\hat{y} = \arg \max_k w \cdot \phi(x_n, k)$
 - If $\hat{y} \neq y_n^*$ $w = w - \phi(x_n, \hat{y}) + \phi(x_n, y_n^*)$

➤ When does this make an update?

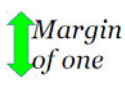
Pro	Md	Vb	Dt	Nn
Pro	Md	Md	Dt	Vb
Pro	Md	Md	Dt	Nn
Pro	Md	Nn	Dt	Md
Pro	Md	Nn	Dt	Nn
Pro	Md	Vb	Dt	Md
Pro	Md	Vb	Dt	Vb
I	can	can	a	can

- When the top rank labeling is not correct, we make an update. The update meant to increase the score of the truth.
- We also need to perform loss augmentation. Try to make sure the truth beat everything has one mistake as the slides below. Try to make bad output ranks very low.
- Viterbi algorithm gives us the highest score path. To incorporate the loss augmentation, we need to do some modification to the original Viterbi algorithm.

Ranking margins

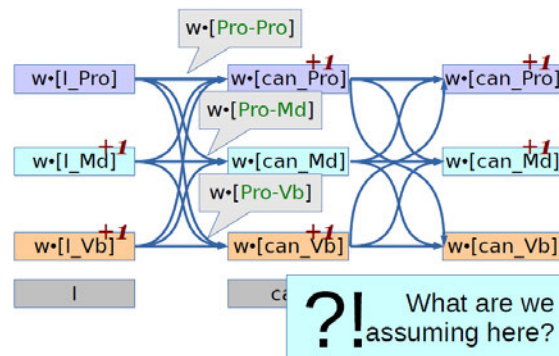
➤ Some errors are worse than others...

Pro	Md	Vb	Dt	Nn
Pro	Md	Md	Dt	Vb
Pro	Md	Md	Dt	Nn
Pro	Md	Nn	Dt	Md
Pro	Md	Nn	Dt	Nn
Pro	Md	Vb	Dt	Md
Pro	Md	Vb	Dt	Vb
I	can	can	a	can



Augmented argmax for sequences

- Add "loss" to each wrong node!



- Why add +1? We want to make bad pass look artificially better. Even we make them looks better, we still find the right path.

4 Problem 17.2

Multiclass Perceptron: Consider the following algorithm:

Multiclass Batch Perceptron

Input:
 A training set $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$
 A class-sensitive feature mapping $\Psi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$

Initialize: $\mathbf{w}^{(1)} = (0, \dots, 0) \in \mathbb{R}^d$

For $t = 1, 2, \dots$
 If $(\exists i$ and $y \neq y_i$ s.t. $\langle \mathbf{w}^{(t)}, \Psi(\mathbf{x}_i, y_i) \rangle \leq \langle \mathbf{w}^{(t)}, \Psi(\mathbf{x}_i, y) \rangle)$ then
 $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \Psi(\mathbf{x}_i, y_i) - \Psi(\mathbf{x}_i, y)$
 else
 output $\mathbf{w}^{(t)}$

Prove the following:

THEOREM 17.5 Assume that there exists \mathbf{w}^* such that for all i and for all $y \neq y_i$ it holds that $\langle \mathbf{w}^*, \Psi(x_i, y_i) \rangle \geq \langle \mathbf{w}^*, \Psi(x_i, y) \rangle + 1$. Let $R = \max_{i,y} \|\Psi(\mathbf{x}_i, y_i) - \Psi(\mathbf{x}_i, y)\|$. Then, the multiclass Perceptron algorithm stops after at most $(R\|\mathbf{w}^*\|)^2$ iterations, and when it stops it holds that $\forall i \in [m], y_i = \operatorname{argmax}_y \langle \mathbf{w}^{(t)}, \Psi(\mathbf{x}_i, y) \rangle$.

Solution:

The idea of the proof is to show that after performing T iterations, the cosine of the angle between \mathbf{w}^* and $w^{(T+1)}$ is at least $\frac{\sqrt{T}}{R\|\mathbf{w}^*\|}$. That is, we will show that

$$\frac{\langle \mathbf{w}^*, w^{(T+1)} \rangle}{\|\mathbf{w}^*\| \cdot \|w^{(T+1)}\|} \geq \frac{\sqrt{T}}{R\|\mathbf{w}^*\|} \quad (2)$$

By the Cauchy-Schwartz inequality, the left-hand side of Equation (2) is at most 1. Therefore, Equation (2) would imply that

$$1 \geq \frac{\sqrt{T}}{R\|\mathbf{w}^*\|} \Rightarrow T \leq (R\|\mathbf{w}^*\|)^2$$

which will conclude our proof.

To show that Equation (2) holds, we first show that $\langle \mathbf{w}^*, w^{(T+1)} \rangle \geq T$. Indeed, at the first iteration, $w^{(1)} = (0, \dots, 0)$ and therefore $\langle \mathbf{w}^*, w^{(1)} \rangle = 0$, while on iteration t , if we update using example (x_i, y_i) we have that

$$\begin{aligned} \langle \mathbf{w}^*, w^{(t+1)} \rangle - \langle \mathbf{w}^*, w^{(t)} \rangle &= \langle \mathbf{w}^*, w^{(t+1)} - w^{(t)} \rangle \\ &= \langle \mathbf{w}^*, \Psi(x_i, y_i) - \Psi(x_i, y) \rangle \\ &= \langle \mathbf{w}^*, \Psi(x_i, y_i) \rangle - \langle \mathbf{w}^*, \Psi(x_i, y) \rangle \\ &\geq 1. \end{aligned}$$

Therefore, after performing T iterations, we get:

$$\langle \mathbf{w}^*, w^{(T+1)} \rangle = \sum_{t=1}^T (\langle \mathbf{w}^*, w^{(t+1)} \rangle - \langle \mathbf{w}^*, w^{(t)} \rangle) \geq T \quad (3)$$

as required.

Next, we upper bound $\|w^{(T+1)}\|$. For each iteration t we have that

$$\begin{aligned}
\|w^{(t+1)}\|^2 &= \|w^{(t)} + (\Psi(x_i, y_i) - \Psi(x_i, y))\|^2 \\
&= \|w^{(t)}\|^2 + 2 \langle w^{(t)}, \Psi(x_i, y_i) - \Psi(x_i, y) \rangle + \|\Psi(x_i, y_i) - \Psi(x_i, y)\|^2 \\
&= \|w^{(t)}\|^2 + 2(\langle w^{(t)}, \Psi(x_i, y_i) \rangle - \langle w^{(t)}, \Psi(x_i, y) \rangle) + \|\Psi(x_i, y_i) - \Psi(x_i, y)\|^2 \\
&\leq \|w^{(t)}\|^2 + R^2
\end{aligned} \tag{4}$$

where the last inequality is due to the fact that example i is necessarily such that $\langle w^{(t)}, \Psi(x_i, y_i) \rangle \leq \langle w^{(t)}, \Psi(x_i, y) \rangle$, and the norm of $\Psi(x_i, y_i) - \Psi(x_i, y)$ is at most R . Now, since $\|w^{(1)}\|^2 = 0$, if we use Equation (4) recursively for T iterations, we obtain that

$$\|w^{(T+1)}\|^2 \leq TR^2 \Rightarrow \|w^{(T+1)}\| \leq \sqrt{TR} \tag{5}$$

Combining Equation (3) with Equation (5), we obtain that

$$\frac{\langle w^*, w^{(T+1)} \rangle}{\|w^*\| \cdot \|w^{(T+1)}\|} \geq \frac{T}{\|w^*\| \sqrt{TR}} = \frac{\sqrt{T}}{R \|w^*\|}$$

We have thus shown that Equation (2) holds, and this concludes our proof.