Multiclass and Introduction to Structured Prediction

David S. Rosenberg

New York University

March 27, 2018
Introduction
Multiclass Setting

- Input space: $X$
- Output space: $Y = \{1, \ldots, k\}$

- Our approaches to multiclass problems so far:
  - multinomial / softmax logistic regression
- Soon: trees and random forests
- Today we consider linear methods specifically designed for multiclass.
- But the main takeaway will be an approach that generalizes to situations where $k$ is “exponentially large” – too large to enumerate.
Reduction to Binary Classification
One-vs-All / One-vs-Rest

Plot courtesy of David Sontag.
One-vs-All / One-vs-Rest

- Train $k$ binary classifiers, one for each class.
- Train $i$th classifier to distinguish class $i$ from rest
- Suppose $h_1, \ldots, h_k : \mathcal{X} \to \mathbb{R}$ are our binary classifiers.
  - Can output hard classifications in $\{-1, 1\}$ or scores in $\mathbb{R}$.
- Final prediction is
  \[ h(x) = \arg \max_{i \in \{1, \ldots, k\}} h_i(x) \]
- Ties can be broken arbitrarily.
Linear Classifiers: Binary and Multiclass
Linear Binary Classifier Review

- Input Space: $\mathcal{X} = \mathbb{R}^d$
- Output Space: $\mathcal{Y} = \{-1, 1\}$

Linear classifier score function:

$$f(x) = \langle w, x \rangle = w^T x$$

Final classification prediction: $\text{sign}(f(x))$

Geometrically, when are $\text{sign}(f(x)) = +1$ and $\text{sign}(f(x)) = -1$?
Suppose $\|w\| > 0$ and $\|x\| > 0$:

$$f(x) = \langle w, x \rangle = \|w\| \|x\| \cos \theta$$

$$f(x) > 0 \iff \cos \theta > 0 \iff \theta \in (-90^\circ, 90^\circ)$$

$$f(x) < 0 \iff \cos \theta < 0 \iff \theta \not\in [-90^\circ, 90^\circ]$$
Three Class Example

- Base hypothesis space $\mathcal{H} = \{ f(x) = w^T x \mid x \in \mathbb{R}^2 \}$.
- Note: Separating boundary always contains the origin.

Example based on Shalev-Schwartz and Ben-David’s *Understanding Machine Learning*, Section 17.1
Class 1 vs Rest:

\[ f_1(x) = w_1^T x \]
Three Class Example: One-vs-Rest

- Examine “Class 2 vs Rest”
  - Predicts everything to be “Not 2”.
  - If it predicted some “2”, then it would get many more “Not 2” incorrect.
One-vs-Rest: Predictions

- Score for class $i$ is

$$f_i(x) = \langle w_i, x \rangle = ||w_i|| ||x|| \cos \theta_i,$$

where $\theta_i$ is the angle between $x$ and $w_i$. 
For simplicity, we’ve assumed $\|w_1\| = \|w_2\| = \|w_3\|$.

Then $\|w_i\|$ and $\|x\|$ are equal for all scores.

$\implies x$ is classified by whichever has largest $\cos \theta_i$ (i.e. $\theta_i$ closest to 0)
One-vs-Rest: Class Boundaries

- This approach doesn't work well in this instance.
- How can we fix this?
The Linear Multiclass Hypothesis Space

- **Base Hypothesis Space**: \( \mathcal{H} = \{ x \mapsto w^T x \mid w \in \mathbb{R}^d \} \).
- **Linear Multiclass Hypothesis Space** (for \( k \) classes):
  \[
  \mathcal{F} = \left\{ x \mapsto \arg \max_i h_i(x) \mid h_1, \ldots, h_k \in \mathcal{H} \right\}
  \]

- What’s the action space here?
Recall: A learning algorithm chooses the hypothesis from the hypothesis space.

Is this a failure of the hypothesis space or the learning algorithm?
This works... so the problem is not with the hypothesis space.

How can we get a solution like this?
Multiclass Predictors
Multiclass Hypothesis Space

- **Base Hypothesis Space**: $\mathcal{H} = \{ h : \mathcal{X} \rightarrow \mathbb{R} \}$ (“score functions”).
- **Multiclass Hypothesis Space** (for $k$ classes):

  $\mathcal{F} = \left\{ x \mapsto \arg \max_i h_i(x) \mid h_1, \ldots, h_k \in \mathcal{H} \right\}$

  - $h_i(x)$ **scores** how likely $x$ is to be from class $i$.

Issue: Need to learn (and represent) $k$ functions. Doesn’t scale to very large $k$. 
General [Discrete] Output Space: \( Y \) (e.g. \( Y = \{1, \ldots, k\} \) for multiclass)

*New idea:* Rather than a score function for each class,
- use one function \( h(x, y) \) that gives a **compatibility score** between input \( x \) and output \( y \)

Final prediction is the \( y \in Y \) that is “most compatible” with \( x \):

\[
f(x) = \arg \max_{y \in Y} h(x, y)
\]

This subsumes the framework with class-specific score functions.

Given class-specific score functions \( h_1, \ldots, h_k \), we could define compatibility function as

\[
h(x, i) = h_i(x), \ i = 1, \ldots, k.
\]
General [Discrete] Output Space: \( Y \)

Base Hypothesis Space: \( \mathcal{H} = \{ h : X \times Y \rightarrow R \} \)
- \( h(x, y) \) gives compatibility score between input \( x \) and output \( y \)

Multiclass Hypothesis Space

\[
\mathcal{F} = \left\{ x \mapsto \arg \max_{y \in Y} h(x, y) \mid h \in \mathcal{H} \right\}
\]

- Final prediction function is an \( f \in \mathcal{F} \).
- For each \( f \in \mathcal{F} \) there is an underlying compatibility score function \( h \in \mathcal{H} \).
Base Hypothesis Space: $\mathcal{H} = \{ h : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R} \}$

Training data: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

Learning process chooses $h \in \mathcal{H}$.

Want compatibility $h(x, y)$ to be large when $x$ has label $y$, small otherwise.
Learning in a Multiclass Hypothesis Space: In Math

- $h(x, y)$ classifies $(x_i, y_i)$ correctly iff
  \[
h(x_i, y_i) > h(x_i, y) \forall y \neq y_i
  \]
- $h$ should give higher score for correct $y$ than for all other $y \in Y$.
- An equivalent condition is the following:
  \[
h(x_i, y_i) > \max_{y \neq y_i} h(x_i, y)
  \]
- If we define
  \[
m_i = h(x_i, y_i) - \max_{y \neq y_i} h(x_i, y),
  \]
  then classification is correct if $m_i > 0$. Generally want $m_i$ to be large.
- Sound familiar?
A Linear Multiclass Hypothesis Space
A linear class-sensitive score function is given by

\[ h(x, y) = \langle w, \Psi(x, y) \rangle, \]

where \( \Psi(x, y) : X \times Y \to \mathbb{R}^d \) is a class-sensitive feature map.

- \( \Psi(x, y) \) extracts features relevant to how compatible \( y \) is with \( x \).
- Final compatibility score is a linear function of \( \Psi(x, y) \).

**Linear Multiclass Hypothesis Space**

\[ \mathcal{F} = \left\{ x \mapsto \arg \max_{y \in Y} \langle w, \Psi(x, y) \rangle \mid w \in \mathbb{R}^d \right\} \]
Example: \( \mathcal{X} = \mathbb{R}^2, \mathcal{Y} = \{1, 2, 3\} \)

- \( w_1 = \left( -\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right), \ w_2 = (0, 1), \ w_3 = \left( \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right) \)

- Prediction function: \((x_1, x_2) \mapsto \arg \max_{i \in \{1, 2, 3\}} \langle w_i, (x_1, x_2) \rangle\).

- How can we get this into the form \( x \mapsto \arg \max_{y \in \mathcal{Y}} \langle w, \Psi(x, y) \rangle \)
The Multivector Construction

- What if we stack $w_i$'s together:

$$w = \begin{pmatrix} \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0, 1, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \\ w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

- And then do the following: $\Psi : \mathbb{R}^2 \times \{1, 2, 3\} \rightarrow \mathbb{R}^6$ defined by

$$\Psi(x, 1) := (x_1, x_2, 0, 0, 0, 0)$$
$$\Psi(x, 2) := (0, 0, x_1, x_2, 0, 0)$$
$$\Psi(x, 3) := (0, 0, 0, 0, x_1, x_2)$$

- Then $\langle w, \Psi(x, y) \rangle = \langle w_y, x \rangle$, which is what we want.
NLP Example: Part-of-speech classification

- $\mathcal{X} = \{\text{All possible words}\}$.
- $\mathcal{Y} = \{\text{NOUN, VERB, ADJECTIVE, ADVERB, ARTICLE, PREPOSITION}\}$.
- Features of $x \in \mathcal{X}$: [The word itself], ENDS_IN_ly, ENDS_IN_ness, ...
- $\Psi(x, y) = (\psi_1(x, y), \psi_2(x, y), \psi_3(x, y), \ldots, \psi_d(x, y))$:
  
  $\psi_1(x, y) = 1(x = \text{apple} \text{ AND } y = \text{NOUN})$
  $\psi_2(x, y) = 1(x = \text{run} \text{ AND } y = \text{NOUN})$
  $\psi_3(x, y) = 1(x = \text{run} \text{ AND } y = \text{VERB})$
  $\psi_4(x, y) = 1(x \text{ ENDS_IN_ly AND } y = \text{ADVERB})$
  $\vdots \quad \vdots \quad \vdots$

- e.g. $\Psi(x = \text{run}, y = \text{NOUN}) = (0, 1, 0, 0, \ldots)$
- After training, what would you guess corresponding $w_1, w_2, w_3, w_4$ to be?
NLP Example: How does it work?

\[ \Psi(x, y) = (\psi_1(x, y), \psi_2(x, y), \psi_3(x, y), \ldots, \psi_d(x, y)) \in \mathbb{R}^d: \]

\[ \psi_1(x, y) = 1(x = \text{apple} \ \text{AND} \ y = \text{NOUN}) \]
\[ \psi_2(x, y) = 1(x = \text{run} \ \text{AND} \ y = \text{NOUN}) \]
\[ \vdots \quad \vdots \quad \vdots \]

After training, we've learned \( w \in \mathbb{R}^d \). Say \( w = (5, -3, 1, 4, \ldots) \)

To predict label for \( x = \text{apple} \),

- we compute compatibility scores for each \( y \in Y \):
  \[ \langle w, \Psi(\text{apple, NOUN}) \rangle \]
  \[ \langle w, \Psi(\text{apple, VERB}) \rangle \]
  \[ \langle w, \Psi(\text{apple, ADVERB}) \rangle \]
  \[ \vdots \]

- Predict class that gives highest score.
Another Approach: Use Label Features

- What if we have a very large number of classes?
- Make features for the classes.
- Common in advertising
  - $\mathcal{X}$: User and user context
  - $\mathcal{Y}$: A large set of banner ads
- Suppose user $x$ is shown many banner ads.
- We want to predict which one the user will click on.
- Possible compatibility features:

  \[
  \psi_1(x, y) = 1(x \text{ interested in sports AND } y \text{ relevant to sports}) \\
  \psi_2(x, y) = 1(x \text{ is in target demographic group of } y) \\
  \psi_3(x, y) = 1(x \text{ previously clicked on ad from company sponsoring } y)
  \]
Linear Multiclass SVM
The Margin for Multiclass

- Let $h : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbf{R}$ be our compatibility score function.
- Define a "margin" between correct class and each other class:

**Definition**
The [class-specific] **margin** of score function $h$ on the $i$th example $(x_i, y_i)$ for class $y$ is

$$m_{i,y}(h) = h(x_i, y_i) - h(x_i, y).$$

- Want $m_{i,y}(h)$ to be large and positive for all $y \neq y_i$.
- For our linear hypothesis space, margin is

$$m_{i,y}(w) = \langle w, \Psi(x_i, y_i) \rangle - \langle w, \Psi(x_i, y) \rangle.$$
Recall binary SVM (without bias term):

$$\min_{w \in \mathbb{R}^d} \frac{1}{2} ||w||^2 + \frac{c}{n} \sum_{i=1}^{n} \max \left(0, 1 - y_i w^T x_i\right).$$

Multiclass SVM (Version 1):

$$\min_{w \in \mathbb{R}^d} \frac{1}{2} ||w||^2 + \frac{c}{n} \sum_{i=1}^{n} \max_{y \neq y_i} \left[\max \left(0, 1 - m_{i,y}(w)\right)\right]$$

where $m_{i,y}(w) = \langle w, \Psi(x_i, y_i) \rangle - \langle w, \Psi(x_i, y) \rangle.$

As in SVM, we’ve taken the value 1 as our “target margin” for each $i, y.$
Class-Sensitive Loss

- In multiclass, some misclassifications may be worse than others.
- Rather than 0/1 Loss, we may be interested in a more general loss

\[ \Delta : Y \times A \rightarrow [0, \infty) \]

- We can use this \( \Delta \) as our target margin for multiclass SVM.
- Multiclass SVM (Version 2):

\[
\min_{w \in \mathbb{R}^d} \frac{1}{2} ||w||^2 + \frac{c}{n} \sum_{i=1}^{n} \max_{y \neq y_i} \left[ \max(0, \Delta(y_i, y) - m_{i,y}(w)) \right]
\]

- If margin \( m_{i,y}(w) \) meets or exceeds its target \( \Delta(y_i, y) \) \( \forall y \neq y_i \), then no loss on example \( i \).
- Note: If \( \Delta(y, y) = 0 \ \forall y \in Y \), then we can replace \( \max_{y \neq y_i} \) with \( \max_y \).
Interlude: Is This Worth The Hassle Compared to One-vs-All?
Recap: What Have We Got?

- Problem: Multiclass classification \( Y = \{1, \ldots, k\} \)

- Solution 1: One-vs-All
  - Train \( k \) models: \( h_1(x), \ldots, h_k(x) : \mathcal{X} \rightarrow \mathbb{R} \).
  - Predict with \( \arg\max_{y \in Y} h_y(x) \).
  - Gave simple example where this fails for linear classifiers

- Solution 2: Multiclass
  - Train one model: \( h(x, y) : \mathcal{X} \times Y \rightarrow \mathbb{R} \).
  - Prediction involves solving \( \arg\max_{y \in Y} h(x, y) \).
Does it work better in practice?

  - Extensive experiments, carefully done
    - albeit on relatively small UCI datasets
  - Suggests one-vs-all works just as well in practice
    - (or at least, the advantages claimed by earlier papers for multiclass methods were not compelling)

- Compared
  - many multiclass frameworks (including the one we discuss)
  - one-vs-all for SVMs with RBF kernel
  - one-vs-all for square loss with RBF kernel (for classification!)

- All performed roughly the same
The framework we have developed for multiclass
- compatibility features / score functions
- multiclass margin
- target margin

Generalizes to situations where $k$ is very large and one-vs-all is intractable.

Key point is that we can generalize across outputs $y$ by using features of $y$. 
Introduction to Structured Prediction
Part-of-speech (POS) Tagging

- Given a sentence, give a part of speech tag for each word:

\[
\begin{array}{|c|c|c|c|}
\hline
x & [START] & He & eats & apples \\
& x_0 & x_1 & x_2 & x_3 \\
\hline
y & [START] & Pronoun & Verb & Noun \\
& y_0 & y_1 & y_2 & y_3 \\
\hline
\end{array}
\]

\[\mathcal{V} = \{\text{all English words}\} \cup \{[\text{START}],",",\} \]
\[\mathcal{P} = \{\text{START, Pronoun, Verb, Noun, Adjective}\} \]
\[\mathcal{X} = \mathcal{V}^n, \ n = 1, 2, 3, \ldots \ [\text{Word sequences of any length}] \]
\[\mathcal{Y} = \mathcal{P}^n, \ n = 1, 2, 3, \ldots [\text{Part of speech sequence of any length}] \]
A **structured prediction** problem is a multiclass problem in which \( \mathcal{Y} \) is very large, but has (or we assume it has) a certain structure.

For POS tagging, \( \mathcal{Y} \) grows exponentially in the length of the sentence.

**Typical structure** assumption: The POS labels form a Markov chain.

- i.e. \( y_{n+1} \mid y_n, y_{n-1}, \ldots, y_0 \) is the same as \( y_{n+1} \mid y_n \).
Local Feature Functions: Type 1

- A “type 1” **local feature** only depends on
- the label at a single position, say $y_i$ (label of the $i$th word) and
- $x$ at any position

**Example:**

$$\phi_1(i, x, y_i) = 1(x_i = \text{runs})1(y_i = \text{Verb})$$
$$\phi_2(i, x, y_i) = 1(x_i = \text{runs})1(y_i = \text{Noun})$$
$$\phi_3(i, x, y_i) = 1(x_{i-1} = \text{He})1(x_i = \text{runs})1(y_i = \text{Verb})$$
A “type 2” **local feature** only depends on
- the labels at 2 consecutive positions: $y_{i-1}$ and $y_i$
- $x$ at any position

**Example:**

$$\theta_1(i, x, y_{i-1}, y_i) = 1(y_{i-1} = \text{Pronoun})1(y_i = \text{Verb})$$
$$\theta_2(i, x, y_{i-1}, y_i) = 1(y_{i-1} = \text{Pronoun})1(y_i = \text{Noun})$$
Local Feature Vector and Compatibility Score

- At each position \( i \) in sequence, define the **local feature vector**: 

\[
\Psi_i(x, y_{i-1}, y_i) = (\phi_1(i, x, y_i), \phi_2(i, x, y_i), \ldots, \\
\theta_1(i, x, y_{i-1}, y_i), \theta_2(i, x, y_{i-1}, y_i), \ldots)
\]

- **Local compatibility score** for \((x, y)\) at position \(i\) is \(\langle w, \Psi_i(x, y_{i-1}, y_i)\rangle\).
The **compatibility score** for the pair of sequences \((x, y)\) is the sum of the local compatibility scores:

\[
\sum_i \langle w, \Psi_i(x, y_{i-1}, y_i) \rangle
\]

\[
= \langle w, \sum_i \Psi_i(x, y_{i-1}, y_i) \rangle
\]

\[
= \langle w, \Psi(x, y) \rangle,
\]

where we define the sequence feature vector by

\[
\Psi(x, y) = \sum_i \Psi_i(x, y_{i-1}, y_i).
\]

So we see this is a special case of linear multiclass prediction.
Sequence Target Loss

- How do we assess the loss for prediction sequence $y'$ for example $(x, y)$?

- **Hamming loss** is common:

  \[
  \Delta(y, y') = \frac{1}{|y|} \sum_{i=1}^{|y|} 1(y_i \neq y'_i)
  \]

- Could generalize this as

  \[
  \Delta(y, y') = \frac{1}{|y|} \sum_{i=1}^{|y|} \delta(y_i, y'_i)
  \]
What remains to be done?

- To compute predictions, we need to find
  \[
  \arg \max_{y \in Y} \langle w, \Psi(x, y) \rangle.
  \]
- This is straightforward for $|Y|$ small.
- Now $|Y|$ is exponentially large.
- Because $\Psi$ breaks down into local functions only depending on 2 adjacent labels,
  - we can solve this efficiently using dynamic programming.
  - (Similar to Viterbi decoding.)
- Learning can be done with SGD and a similar dynamic program.