Multiclass Classification and Structured Prediction

He He
(adapted from David Rosenberg's and Dan Roth's slides)

CDS, NYU

April 7, 2019

Contents

Logistics

Section
- Review of lecture
- Averaged perceptron

Tutorial
- Multiclass concept check
- Review subgradient
- Review bias and variance of estimators

Overview

Motivation

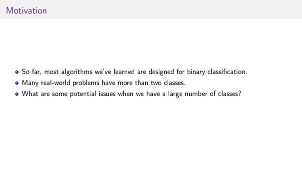
- So far, most algorithms we've learned are designed for binary classification.
- Many real-world problems have more than two classes.

- So far, most algorithms we've learned are designed for binary classification.
- Many real-world problems have more than two classes.

1. Which ones we've learned can handle more than 2 classes? Multinomial logistic regression, naive Bayes. Next, trees and random forests.
2. Examples? Text classification, object recognition (ImageNet has more than 20k classes).

Motivation

- So far, most algorithms we've learned are designed for binary classification.
- Many real-world problems have more than two classes.
- What are some potential issues when we have a large number of classes?



1. Which ones we've learned can handle more than 2 classes? Multinomial logistic regression, naive Bayes. Next, trees and random forests.
2. Examples? Text classification, object recognition (ImageNet has more than 20k classes).
3. Class imbalance, computation cost for both training and testing, different cost of errors etc.

Today's lecture

- Recap: how to reduce multiclass classification to binary classification?
- How do we generalize binary classification algorithm to the multiclass setting?
- Example of very large output space: structured prediction.

Today's lecture

- Recap: how to reduce multiclass classification to binary classification?
- How do we generalize binary classification algorithm to the multiclass setting?
- Example of very large output space: structured prediction.

1. Think of binary classification or linear regression as black-box predictors and start from there.
2. What needs to be changed here? The loss function.

Reduction to Binary Classification

One-vs-All / One-vs-Rest

Setting

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

One-vs-All / One-vs-Rest

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

Training
- Train k binary classifiers, one for each class: $h_1, \ldots, h_k : \mathcal{X} \to \mathbf{R}$.
- Classifier h_i distinguishes class i (+1) from the rest (-1).

One-vs-All / One-vs-Rest

Setting

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

Training

- Train k binary classifiers, one for each class: $h_1, \ldots, h_k : \mathcal{X} \to \mathbf{R}$.
- Classifier h_i distinguishes class i (+1) from the rest (-1).

Prediction

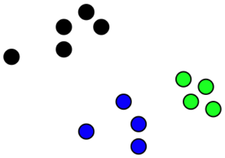
- Majority vote:

$$h(x) = \underset{i \in \{1, \ldots, k\}}{\arg\max}\, h_i(x)$$

- Ties can be broken arbitrarily.

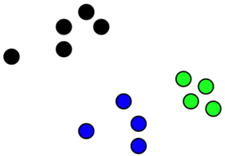
OvA: 3-class example

Consider a dataset with three classes:

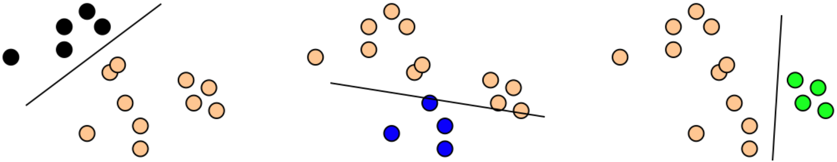


OvA: 3-class example

Consider a dataset with three classes:

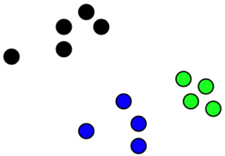


Train OvA classifiers:



OvA: 3-class example

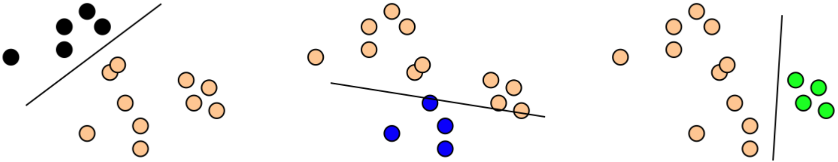
Consider a dataset with three classes:

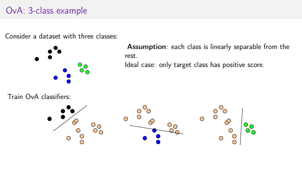


Assumption: each class is linearly separable from the rest.

Ideal case: only target class has positive score.

Train OvA classifiers:

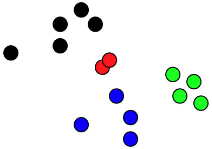




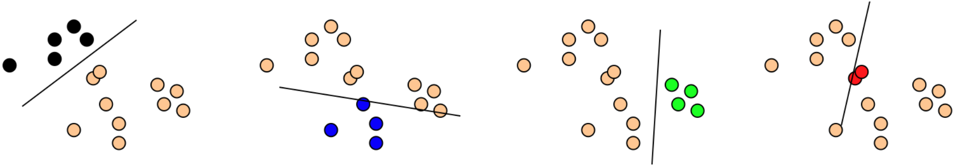
What’s a failure case for OvA?

OvA: 4-class non-separable example

Consider a dataset with four classes:

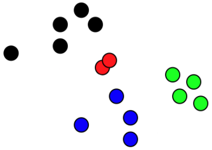


Train OvA classifiers:



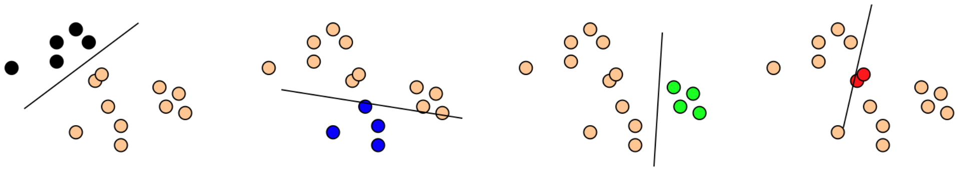
OvA: 4-class non-separable example

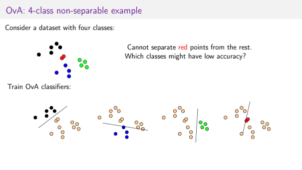
Consider a dataset with four classes:



Cannot separate red points from the rest.
Which classes might have low accuracy?

Train OvA classifiers:





1. Blue and green because they are getting positive score from the red classifier.
2. How can we fix this? Note that optimal linear classifiers exist in this example.

All vs All / One vs One / All pairs

Setting

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

All vs All / One vs One / All pairs

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

1. How many classifiers do we need to train?

All vs All / One vs One / All pairs

Setting

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

Training

- Train $\binom{k}{2}$ binary classifiers, one for each pair: $h_{ij} : \mathcal{X} \to \mathbf{R}$ for $i \in [1, k]$ and $j \in [i+1, k]$.
- Classifier h_{ij} distinguishes class i (+1) from class j (-1).

All vs All / One vs One / All pairs

Setting

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

Training

- Train $\binom{k}{2}$ binary classifiers, one for each pair: $h_{ij} : \mathcal{X} \to \mathbf{R}$ for $i \in [1, k]$ and $j \in [i+1, k]$.
- Classifier h_{ij} distinguishes class i (+1) from class j (-1).

Prediction

- Majority vote (each class gets $k-1$ votes)

$$h(x) = \underset{i \in \{1, \ldots, k\}}{\arg\max} \sum_{j \neq i} \underbrace{h_{ij}(x)\mathbb{I}\{i < j\}}_{\text{class } i \text{ is } +1} - \underbrace{h_{ji}(x)\mathbb{I}\{j < i\}}_{\text{class } i \text{ is -1}}$$

- Tournament
- Ties can be broken arbitrarily.

All vs All / One vs One / All pairs

Setting

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

Training

- Train $\binom{k}{2}$ binary classifiers, one for each pair: $h_{ij} : \mathcal{X} \to \mathbf{R}$ for $i \in [1, k]$ and $j \in [i+1, k]$.
- Classifier h_{ij} distinguishes class i (+1) from class j (-1).

Prediction

- Majority vote (each class gets $k-1$ votes)

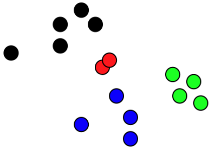
$$h(x) = \underset{i \in \{1,\ldots,k\}}{\arg\max} \sum_{j \neq i} \underbrace{h_{ij}(x)\mathbb{I}\{i < j\}}_{\text{class } i \text{ is } +1} - \underbrace{h_{ji}(x)\mathbb{I}\{j < i\}}_{\text{class } i \text{ is -1}}$$

- Tournament
- Ties can be broken arbitrarily.

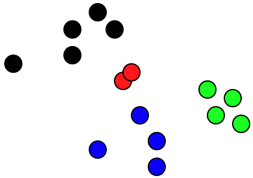
1. Majority vote: Class i gets a vote each time it is predicted.
2. Tournament: start with random pairs, only winners continue.

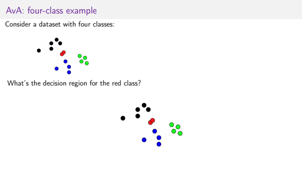
AvA: four-class example

Consider a dataset with four classes:



What's the decision region for the red class?

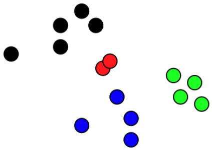




1. Draw lines separating red from each other classes. The intersection of the three regions get 3 votes (max).

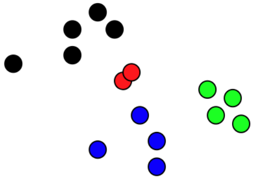
AvA: four-class example

Consider a dataset with four classes:



Assumption: each pair of classes are linearly separable.
More expressive than OvA.

What's the decision region for the red class?



OvA vs AvA

		OvA	AvA
computation	train	$O(k \quad\quad)$	$O(k^2 \quad\quad)$
	test	$O(k \quad)$	$O(k^2 \quad)$

OvA vs AvA

		OvA	AvA
computation	train	$O(k$ $)$	$O(k^2$ $)$
	test	$O(k$ $)$	$O(k^2$ $)$

If you're using SVM, would you prefer AvA or OvA to save computation? Dual form would prefer AvA. When number of examples much larger than feature dimensions, dual is more expensive.

OvA vs AvA

		OvA	AvA
computation	train	$O(kB_{\text{train}}(n))$	$O(k^2 B_{\text{train}}(n/k))$
	test	$O(kB_{\text{test}})$	$O(k^2 B_{\text{test}})$

challenges

		OvA	AvA
computation	train	$O(kB_{\text{train}}(n))$	$O(k^2 B_{\text{train}}(n/k))$
	test	$O(kB_{\text{test}})$	$O(k^2 B_{\text{test}})$

challenges

If you're using SVM, would you prefer AvA or OvA to save computation? Dual form would prefer AvA. When number of examples much larger than feature dimensions, dual is more expensive.

OvA vs AvA

		OvA	AvA
computation	train	$O(kB_{\text{train}}(n))$	$O(k^2 B_{\text{train}}(n/k))$
	test	$O(kB_{\text{test}})$	$O(k^2 B_{\text{test}})$
challenges	train	class imbalance	small training set
	test	calibration / scale tie breaking	

Lack theoretical justification but simple to implement and works well in practice (when # classes is small).

Question: When would you prefer AvA / OvA?

OvA vs AvA

		OvA	AvA
computation	train	$O(kB_{\text{train}}(n))$	$O(k^2B_{\text{train}}(n/k))$
	test	$O(kB_{\text{test}})$	$O(k^2B_{\text{test}})$
challenges	train	class imbalance	small training set
	test	calibration / scale tie breaking	

Lack theoretical justification but simple to implement and works well in practice (when # classes is small).

Question: When would you prefer AvA / OvA?

If you're using SVM, would you prefer AvA or OvA to save computation? Dual form would prefer AvA. When number of examples much larger than feature dimensions, dual is more expensive.

Code word for labels

Using the reduction approach, can you train fewer than k binary classifiers?

Code word for labels

Using the reduction approach, can you train fewer than k binary classifiers?

Key idea: Encode labels as binary codes and predict the code bits directly.

Code word for labels

Using the reduction approach, can you train fewer than k binary classifiers?

Key idea: Encode labels as binary codes and predict the code bits directly.

OvA encoding:

class	h_1	h_2	h_3	h_4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Code word for labels

Using the reduction approach, can you train fewer than k binary classifiers?

Key idea: Encode labels as binary codes and predict the code bits directly.

OvA encoding:

class	h_1	h_2	h_3	h_4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

OvA uses k bits to encode each label, what's the minimal number of bits you can use?

Error correcting output codes (ECOC)

Example: 8 classes, 6-bit code

class	h_1	h_2	h_3	h_4	h_5	h_6
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	1	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	1	0	1	0	0

Training Binary classifier h_i:

- +1: classes whose i-th bit is 1
- -1: classes whose i-th bit is 0

Error correcting output codes (ECOC)

Example: 8 classes, 6-bit code

class	h_1	h_2	h_3	h_4	h_5	h_6
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	1	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	1	0	1	0	0

Training Binary classifier h_i:

- +1: classes whose i-th bit is 1
- -1: classes whose i-th bit is 0

Prediction Closest label in terms of Hamming distance.

h_1	h_2	h_3	h_4	h_5	h_6
0	1	1	0	1	1

Error correcting output codes (ECOC)

Example: 8 classes, 6-bit code

class	h_1	h_2	h_3	h_4	h_5	h_6
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	1	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	1	0	1	0	0

Training Binary classifier h_i:

- $+1$: classes whose i-th bit is 1
- -1: classes whose i-th bit is 0

Prediction Closest label in terms of Hamming distance.

h_1	h_2	h_3	h_4	h_5	h_6
0	1	1	0	1	1

Code design Want good binary classifiers.

Error correcting output codes (ECOC)

Example: 8 classes, 6-bit code

class	h_1	h_2	h_3	h_4	h_5	h_6
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	1	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	1	0	1	0	0

Training Binary classifier h_i:
- +1: classes whose i-th bit is 1
- -1: classes whose i-th bit is 0

Prediction Closest label in terms of Hamming distance.

h_1	h_2	h_3	h_4	h_5	h_6
0	1	1	0	1	1

Code design Want good binary classifiers.

1. Random or depending on domain knowledge

Error correcting output codes: summary

- Computationally more efficient than OvA (a special case of ECOC). Better for large k.
- Why not use the minimal number of bits ($\log_2 k$)?

Error correcting output codes: summary

- Computationally more efficient than OvA (a special case of ECOC). Better for large k.
- Why not use the minimal number of bits ($\log_2 k$)?
  - If the minimum Hamming distance between any pair of code word is d, then it can correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.
  - In plain words, if rows are far from each other, ECOC is robust to errors.

Error correcting output codes: summary

- Computationally more efficient than OvA (a special case of ECOC). Better for large k.
- Why not use the minimal number of bits ($\log_2 k$)?
  - If the minimum Hamming distance between any pair of code word is d, then it can correct $\lfloor \frac{d-1}{2} \rfloor$ errors.
  - In plain words, if rows are far from each other, ECOC is robust to errors.

1. Larger distance -> more binary problems -> more likely to have hard binary problems.

Error correcting output codes: summary

- Computationally more efficient than OvA (a special case of ECOC). Better for large k.
- Why not use the minimal number of bits ($\log_2 k$)?
  - If the minimum Hamming distance between any pair of code word is d, then it can correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.
  - In plain words, if rows are far from each other, ECOC is robust to errors.
- Trade-off between code distance and binary classification performance.
- Nice theoretical results [Allwein et al., 2000] (also incoporates AvA).

Review

Reduction-based approaches:

- Reducing multiclass classification to binary classification: OvA, AvA, ECOC.
- Key is to design "natural" binary classification problems without large computation cost.

Review

Reduction-based approaches:

- Reducing multiclass classification to binary classification: OvA, AvA, ECOC.
- Key is to design "natural" binary classification problems without large computation cost.

But,

- Unclear how to generalize to extremely large # of classes.
- ImageNet: >20k labels; Wikipedia: >1M categories.

Next, generalize previous algorithms to multiclass settings.

Review

Reduction-based approaches:

- Reducing multiclass classification to binary classification: OvA, AvA, ECOC.
- Key is to design "natural" binary classification problems without large computation cost.

But,

- Unclear how to generalize to extremely large # of classes.
- ImageNet: >20k labels; Wikipedia: >1M categories.

Next, generalize previous algorithms to multiclass settings.

1. What needs to be changed? The loss function.

Linear Multiclass Predictors

OvA revisit

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

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

OvA revisit

- **Base Hypothesis Space**: $\mathcal{H} = \{h : \mathcal{X} \to \mathbf{R}\}$ (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.
- OvA objective: $h_i(x) > 0$ for x with label i and $h_i(x) < 0$ for x with all other labels.

OvA revisit

- **Base Hypothesis Space**: $\mathcal{H} = \{h : \mathcal{X} \to \mathbf{R}\}$ (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.
- OvA objective: $h_i(x) > 0$ for x with label i and $h_i(x) < 0$ for x with all other labels.
- At test time, for (x, i) we only need

$$h_i(x) > h_j(x) \qquad \forall j \neq i. \tag{1}$$

Multiclass perceptron

- Base linear predictors: $h_i(x) = w_i^T x$ $(w \in \mathbf{R}^d)$.
- Multiclass perceptron:

  Given a multiclass dataset $\mathcal{D} = \{(x, y)\}$;
  Initialize $w \leftarrow 0$;
  for $\text{iter} = 1, 2, \ldots, T$ **do**
      for $(x, y) \in \mathcal{D}$ **do**
          $\hat{y} = \arg\max_{y' \in \mathcal{Y}} w_{y'}^T x$;
          if $\hat{y} \neq y$ **then** `// We've made a mistake`
              $w_y \leftarrow w_y + x$; `// Move the target-class scorer towards` x
              $w_{\hat{y}} \leftarrow w_{\hat{y}} - x$; `// Move the wrong-class scorer away from` x
          end
      end
  end

- (Geometric interpretation)

Side note: Linear Binary Classifier Review

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

- Linear classifier score function:

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

Side note: Linear Binary Classifier Review

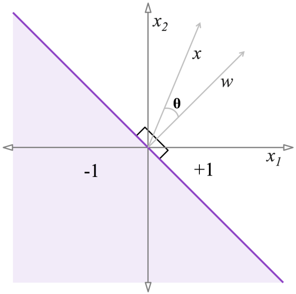
- Input Space: $\mathcal{X} = \mathbf{R}^d$
- Output Space: $\mathcal{Y} = \{-1, 1\}$

- Linear classifier score function:

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

- Final classification prediction: $\operatorname{sign}(f(x))$
- Geometrically, when are $\operatorname{sign}(f(x)) = +1$ and $\operatorname{sign}(f(x)) = -1$?

Side note: Linear Binary Classifier Review



Suppose $\|w\| > 0$ and $\|x\| > 0$:

$$
\begin{aligned}
f(x) &= \langle w, x \rangle = \|w\| \|x\| \cos\theta \\
f(x) > 0 &\iff \cos\theta > 0 \iff \theta \in (-90^\circ, 90^\circ) \\
f(x) < 0 &\iff \cos\theta < 0 \iff \theta \notin [-90^\circ, 90^\circ]
\end{aligned}
$$

Rewrite the scoring function

- Remember that we want to scale to very large # of classes and reuse algorithms and analysis for binary classification
  - $\Longrightarrow$ a single weight vector is desired
- How to rewrite the equation such that we have one w instead of k?

Rewrite the scoring function

- Remember that we want to scale to very large # of classes and reuse algorithms and analysis for binary classification
  - $\implies$ a single weight vector is desired
- How to rewrite the equation such that we have one w instead of k?

$$w_i^T x = w^T \psi(x, i) \tag{2}$$

$$h_i(x) = h(x, i) \tag{3}$$

  - Encode labels in the feature space.
  - Score for each label $\to$ score for the "compatibility" of a label and an input.

The Multivector Construction

How to construct the feature map ψ?

The Multivector Construction

How to construct the feature map ψ?

- What if we stack w_i's together (e.g., $x \in \mathbf{R}^2, \mathcal{Y} = \{1,2,3\}$)

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

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

$$\begin{aligned} \Psi(x,1) &:= (x_1, x_2, 0, 0, 0, 0) \\ \Psi(x,2) &:= (0, 0, x_1, x_2, 0, 0) \\ \Psi(x,3) &:= (0, 0, 0, 0, x_1, x_2) \end{aligned}$$

- Then $\langle w, \Psi(x,y) \rangle = \langle w_y, x \rangle$, which is what we want.

Rewrite multiclass perceptron

Multiclass perceptron using the multivector construction.

Given a multiclass dataset $\mathcal{D} = \{(x, y)\}$;
Initialize $w \leftarrow 0$;
for $\text{iter} = 1, 2, \ldots, T$ **do**
    for $(x, y) \in \mathcal{D}$ **do**
        $\hat{y} = \arg\max_{y' \in \mathcal{Y}} w^T \psi(x, y')$; `// Equivalent to` $\arg\max_{y' \in \mathcal{Y}} w_{y'}^T x$
        if $\hat{y} \neq y$ **then** `// We've made a mistake`
            $w \leftarrow w + \psi(x, y)$; `// Move the scorer towards` $\psi(x, y)$
            $w \leftarrow w - \psi(x, \hat{y})$; `// Move the scorer away from` $\psi(x, \hat{y})$
        end
    end
end

Exercise: What is the base binary classification problem in multiclass perceptron?

Rewrite multiclass perceptron

Multiclass perceptron using the multivector construction.

Given a multiclass dataset $\mathcal{D} = \{(x, y)\}$;
Initialize $w \leftarrow 0$;
for iter $= 1, 2, \ldots, T$ **do**
  for $(x, y) \in \mathcal{D}$ **do**
    $\hat{y} = \arg\max_{y' \in \mathcal{Y}} w^T \psi(x, y')$; // Equivalent to $\arg\max_{y' \in \mathcal{Y}} w_{y'}^T x$
    if $\hat{y} \neq y$ **then** // We've made a mistake
      $w \leftarrow w + \psi(x, y)$; // Move the scorer towards $\psi(x, y)$
      $w \leftarrow w - \psi(x, \hat{y})$; // Move the scorer away from $\psi(x, \hat{y})$
    end
  end
end

Exercise: What is the base binary classification problem in multiclass perceptron?

$$w^T(\phi(x,i) - \phi(x,j)) > 0.$$

Geometric interpretation

Feature templates

Toy multiclass example: Part-of-speech classification

- $\mathcal{X} = \{\text{All possible words}\}$
- $\mathcal{Y} = \{\text{NOUN,VERB,ADJECTIVE,}\ldots\}$.

Feature templates

Toy multiclass example: Part-of-speech classification

- $\mathcal{X} = \{\text{All possible words}\}$
- $\mathcal{Y} = \{\text{NOUN,VERB,ADJECTIVE}, \ldots\}$.

What are useful features?

Feature templates

Toy multiclass example: Part-of-speech classification

- $\mathcal{X} = \{\text{All possible words}\}$
- $\mathcal{Y} = \{\text{NOUN,VERB,ADJECTIVE,}\ldots\}$.
- Features of $x \in \mathcal{X}$: [The word itself], ENDS_IN_ly, ENDS_IN_ness, ...

How to construct the feature vector?

- Multivector construction: $w \in \mathbf{R}^{d \times k}$—doesn't scale.

Feature templates

Toy multiclass example: Part-of-speech classification

- $\mathcal{X} = \{\text{All possible words}\}$
- $\mathcal{Y} = \{\text{NOUN,VERB,ADJECTIVE,}\ldots\}$.
- Features of $x \in \mathcal{X}$: [The word itself], ENDS_IN_ly, ENDS_IN_ness, ...

How to construct the feature vector?

- Multivector construction: $w \in \mathbf{R}^{d \times k}$—doesn't scale.
- **Feature templates**: directly design features for each class.

$$\Psi(x,y) = (\psi_1(x,y), \psi_2(x,y), \psi_3(x,y), \ldots, \psi_d(x,y)) \tag{4}$$

  - Size can be bounded by d.

Feature templates

Sample training data:

```
The boy grabbed the apple and ran away quickly .
```

Feature templates

Sample training data:

```
The boy grabbed the apple and ran away quickly .
```

Feature templates:

$$
\begin{aligned}
\psi_1(x,y) &= 1(x = \text{apple AND } y = \text{NOUN}) \\
\psi_2(x,y) &= 1(x = \text{run AND } y = \text{NOUN}) \\
\psi_3(x,y) &= 1(x = \text{run AND } y = \text{VERB}) \\
\psi_4(x,y) &= 1(x \text{ ENDS\_IN\_ly AND } y = \text{ADVERB}) \\
&\dots
\end{aligned}
$$

Feature templates

Sample training data:

```
The boy grabbed the apple and ran away quickly .
```

Feature templates:

$$
\begin{aligned}
\psi_1(x,y) &= 1(x = \text{apple AND } y = \text{NOUN}) \\
\psi_2(x,y) &= 1(x = \text{run AND } y = \text{NOUN}) \\
\psi_3(x,y) &= 1(x = \text{run AND } y = \text{VERB}) \\
\psi_4(x,y) &= 1(x \text{ ENDS\_IN\_ly AND } y = \text{ADVERB}) \\
&\dots
\end{aligned}
$$

- E.g., $\Psi(x = \text{run}, y = \text{NOUN}) = (0, 1, 0, 0, \dots)$

Feature templates

Sample training data:

```
The boy grabbed the apple and ran away quickly .
```

Feature templates:

$$
\begin{aligned}
\psi_1(x,y) &= 1(x = \text{apple AND } y = \text{NOUN}) \\
\psi_2(x,y) &= 1(x = \text{run AND } y = \text{NOUN}) \\
\psi_3(x,y) &= 1(x = \text{run AND } y = \text{VERB}) \\
\psi_4(x,y) &= 1(x \text{ ENDS\_IN\_ly AND } y = \text{ADVERB}) \\
&\cdots
\end{aligned}
$$

- E.g., $\Psi(x = \text{run}, y = \text{NOUN}) = (0, 1, 0, 0, \ldots)$
- After training, what's w_1, w_2, w_3, w_4?

Feature templates

Sample training data:

```
The boy grabbed the apple and ran away quickly .
```

Feature templates:

$$\begin{aligned}
\psi_1(x,y) &= 1(x = \text{apple AND } y = \text{NOUN}) \\
\psi_2(x,y) &= 1(x = \text{run AND } y = \text{NOUN}) \\
\psi_3(x,y) &= 1(x = \text{run AND } y = \text{VERB}) \\
\psi_4(x,y) &= 1(x \text{ ENDS_IN_ly AND } y = \text{ADVERB}) \\
&\ldots
\end{aligned}$$

- E.g., $\Psi(x = \text{run}, y = \text{NOUN}) = (0, 1, 0, 0, \ldots)$
- After training, what's w_1, w_2, w_3, w_4?
- No need to include feature templates unseen in training data.

Feature templates: implementation

- Flexible, e.g., neighboring words, suffix/prefix.
- "Read off" features from the training data.
- Often sparse—efficient in practice, e.g., NLP problems.
- Can use a hash function: template $\rightarrow \{1, 2, \ldots, d\}$.

Review

Ingredients in multiclass classification:

- Scoring functions for each class (similar to ranking).
- Represent labels in the input space $\implies$ single weight vector.

Review

Ingredients in multiclass classification:

- Scoring functions for each class (similar to ranking).
- Represent labels in the input space $\implies$ single weight vector.

Uniqueness of solution, online learning / efficiency, inductive bias (margin)

Review

Ingredients in multiclass classification:

- Scoring functions for each class (similar to ranking).
- Represent labels in the input space $\implies$ single weight vector.

We've seen

- How to generalize the perceptron algorithm to multiclass setting.
- Very simple idea. Was popular in NLP for structured prediction (e.g., tagging, parsing).

Review

Ingredients in multiclass classification:

- Scoring functions for each class (similar to ranking).
- Represent labels in the input space $\implies$ single weight vector.

We've seen

- How to generalize the perceptron algorithm to multiclass setting.
- Very simple idea. Was popular in NLP for structured prediction (e.g., tagging, parsing).

Uniqueness of solution, online learning / efficiency, inductive bias (margin)

Review

Ingredients in multiclass classification:

- Scoring functions for each class (similar to ranking).
- Represent labels in the input space $\Longrightarrow$ single weight vector.

We've seen

- How to generalize the perceptron algorithm to multiclass setting.
- Very simple idea. Was popular in NLP for structured prediction (e.g., tagging, parsing).

Next,

- How to generalize SVM to the multiclass setting.
- Concept check: Why might one prefer SVM / perceptron?

2020-04-07

Review

Ingredients in multiclass classification:

- Scoring functions for each class (similar to ranking).
- Represent labels in the input space $\implies$ single weight vector.

We've seen

- How to generalize the perceptron algorithm to multiclass setting.
- Very simple idea. Was popular in NLP for structured prediction (e.g., tagging, parsing).

Next,

- How to generalize SVM to the multiclass setting.
- Concept check: Why might one prefer SVM / perceptron?

Uniqueness of solution, online learning / efficiency, inductive bias (margin)

Margin for Multiclass

Binary

- Margin for $(x^{(n)}, y^{(n)})$:

$$y^{(n)} w^T x^{(n)} \tag{5}$$

- Want margin to be large and positive ($w^T x^{(n)}$ has same sign as $y^{(n)}$)

Margin for Multiclass

Binary

- Margin for $(x^{(n)}, y^{(n)})$:

$$y^{(n)} w^T x^{(n)} \tag{5}$$

- Want margin to be large and positive ($w^T x^{(n)}$ has same sign as $y^{(n)}$)

Multiclass

- Class-specific margin for $(x^{(n)}, y^{(n)})$:

$$h(x^{(n)}, y^{(n)}) - h(x^{(n)}, y). \tag{6}$$

- Difference between scores of the correct class and each other class
- Want margin to be large and positive for all $y \neq y^{(n)}$.

Multiclass SVM: separable case

Binary

$$\min_{w} \quad \frac{1}{2}\|w\|^2 \tag{7}$$

$$\text{s.t.} \quad \underbrace{y^{(n)}w^T x^{(n)}}_{\text{margin}} \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D} \tag{8}$$

Multiclass SVM: separable case

Binary

$$
\begin{aligned}
\min_{w} \quad & \frac{1}{2}\|w\|^2 && (7)\\
\text{s.t.} \quad & \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}} \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D} && (8)
\end{aligned}
$$

Multiclass As in the binary case, take 1 as our target margin.

$$
m_{n,y}(w) \stackrel{\text{def}}{=} \underbrace{\left\langle w, \Psi(x^{(n)}, y^{(n)}) \right\rangle}_{\text{score of correct class}} - \underbrace{\left\langle w, \Psi(x^{(n)}, y) \right\rangle}_{\text{score of other class}} \tag{9}
$$

$$
\begin{aligned}
\min_{w} \quad & \frac{1}{2}\|w\|^2 && (10)\\
\text{s.t.} \quad & m_{n,y}(w) \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D}, y \neq y^{(n)} && (11)
\end{aligned}
$$

Multiclass SVM: separable case

Binary

$$\min_{w} \quad \frac{1}{2}\|w\|^2 \tag{7}$$

$$\text{s.t.} \quad \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}} \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D} \tag{8}$$

Multiclass As in the binary case, take 1 as our target margin.

$$m_{n,y}(w) \stackrel{\text{def}}{=} \underbrace{\left\langle w, \Psi(x^{(n)}, y^{(n)}) \right\rangle}_{\text{score of correct class}} - \underbrace{\left\langle w, \Psi(x^{(n)}, y) \right\rangle}_{\text{score of other class}} \tag{9}$$

$$\min_{w} \quad \frac{1}{2}\|w\|^2 \tag{10}$$

$$\text{s.t.} \quad m_{n,y}(w) \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D}, y \neq y^{(n)} \tag{11}$$

Exercise: write the objective for the non-separable case

Multiclass SVM: separable case

Binary

$$\min_w \quad \frac{1}{2}\|w\|^2 \tag{7}$$

$$\text{s.t.} \quad \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}} \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D} \tag{8}$$

Multiclass As in the binary case, take 1 as our target margin.

$$m_{n,y}(w) \stackrel{\text{def}}{=} \underbrace{\left\langle w, \Psi(x^{(n)}, y^{(n)}) \right\rangle}_{\text{score of correct class}} - \underbrace{\left\langle w, \Psi(x^{(n)}, y) \right\rangle}_{\text{score of other class}} \tag{9}$$

$$\min_w \quad \frac{1}{2}\|w\|^2 \tag{10}$$

$$\text{s.t.} \quad m_{n,y}(w) \geqslant 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D}, y \neq y^{(n)} \tag{11}$$

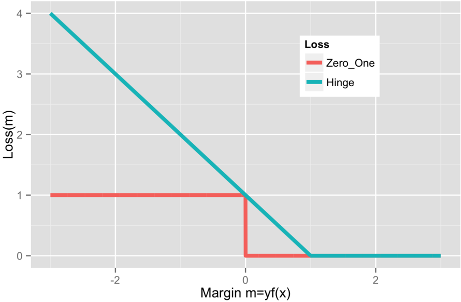
Exercise: write the objective for the non-separable case

Next, let's think about how to write the objective using hinge loss.

Recap: hingle loss for binary classification

- Hinge loss: a convex upperbound on the 0-1 loss

$$\ell_{\text{hinge}}(y, \hat{y}) = \max(0, 1 - yh(x)) \tag{12}$$



Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$ \tag{13} $$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y = y'\} \tag{13}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y = y'\} \tag{13}$$

- In general, can also have different cost for each class.

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y = y'\} \tag{13}$$

- In general, can also have different cost for each class.
- Upper bound on $\Delta(y, y')$.

$$\hat{y} \stackrel{\text{def}}{=} \operatorname*{arg\,max}_{y' \in \mathcal{Y}} \langle w, \Psi(x, y') \rangle \tag{14}$$

$$\tag{16}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y = y'\} \tag{13}$$

- In general, can also have different cost for each class.
- Upper bound on $\Delta(y, y')$.

$$\hat{y} \stackrel{\text{def}}{=} \underset{y' \in \mathcal{Y}}{\arg\max} \left\langle w, \Psi(x, y') \right\rangle \tag{14}$$

$$\implies \langle w, \Psi(x, y) \rangle \leqslant \langle w, \Psi(x, \hat{y}) \rangle \tag{15}$$

$$\tag{16}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y = y'\} \tag{13}$$

- In general, can also have different cost for each class.
- Upper bound on $\Delta(y, y')$.

$$\hat{y} \stackrel{\text{def}}{=} \underset{y' \in \mathcal{Y}}{\arg\max} \left\langle w, \Psi(x, y') \right\rangle \tag{14}$$

$$\Longrightarrow \langle w, \Psi(x, y) \rangle \leqslant \langle w, \Psi(x, \hat{y}) \rangle \tag{15}$$

$$\Longrightarrow \Delta(y, \hat{y}) \leqslant \Delta(y, \hat{y}) + \langle w, (\Psi(x, \hat{y}) - \Psi(x, y)) \rangle \qquad \text{When are they equal?} \tag{16}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y = y'\} \tag{13}$$

- In general, can also have different cost for each class.
- Upper bound on $\Delta(y, y')$.

$$\hat{y} \stackrel{\text{def}}{=} \underset{y' \in \mathcal{Y}}{\arg\max} \left\langle w, \Psi(x, y') \right\rangle \tag{14}$$

$$\Longrightarrow \langle w, \Psi(x, y) \rangle \leqslant \langle w, \Psi(x, \hat{y}) \rangle \tag{15}$$

$$\Longrightarrow \Delta(y, \hat{y}) \leqslant \Delta(y, \hat{y}) + \langle w, (\Psi(x, \hat{y}) - \Psi(x, y)) \rangle \qquad \text{When are they equal?} \tag{16}$$

- Generalized hinge loss:

$$\ell_{\text{hinge}}(y, \hat{y}) \stackrel{\text{def}}{=} \max_{y' \in \mathcal{Y}} \left(\Delta(y, y') + \left\langle w, \left(\Psi(x, y') - \Psi(x, y) \right) \right\rangle \right) \tag{17}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y = y'\} \tag{13}$$

- In general, can also have different cost for each class.
- Upper bound on $\Delta(y, y')$.

$$\hat{y} \stackrel{\text{def}}{=} \operatorname*{arg\,max}_{y' \in \mathcal{Y}} \left\langle w, \Psi(x, y') \right\rangle \tag{14}$$

$$\implies \langle w, \Psi(x, y) \rangle \leqslant \langle w, \Psi(x, \hat{y}) \rangle \tag{15}$$

$$\implies \Delta(y, \hat{y}) \leqslant \Delta(y, \hat{y}) + \langle w, (\Psi(x, \hat{y}) - \Psi(x, y)) \rangle \qquad \text{When are they equal?} \tag{16}$$

- Generalized hinge loss:

$$\ell_{\text{hinge}}(y, \hat{y}) \stackrel{\text{def}}{=} \max_{y' \in \mathcal{Y}} \left(\Delta(y, y') + \left\langle w, \left(\Psi(x, y') - \Psi(x, y) \right) \right\rangle \right) \tag{17}$$

Generalized hinge loss

- What's the zero-one loss for multiclass classification?

$$\Delta(y, y') = \mathbb{I}\{y = y'\} \tag{13}$$

- In general, can also have different cost for each class.
- Upper bound on $\Delta(y, y')$.

$$\hat{y} \stackrel{\text{def}}{=} \operatorname*{arg\,max}_{y' \in \mathcal{Y}} \langle w, \Psi(x, y') \rangle \tag{14}$$

$$\implies \langle w, \Psi(x, y) \rangle \leqslant \langle w, \Psi(x, \hat{y}) \rangle \tag{15}$$

$$\implies \Delta(y, \hat{y}) \leqslant \Delta(y, \hat{y}) + \langle w, (\Psi(x, \hat{y}) - \Psi(x, y)) \rangle \qquad \text{When are they equal?} \tag{16}$$

- Generalized hinge loss:

$$\ell_{\text{hinge}}(y, \hat{y}) \stackrel{\text{def}}{=} \max_{y' \in \mathcal{Y}} \left(\Delta(y, y') + \langle w, (\Psi(x, y') - \Psi(x, y)) \rangle \right) \tag{17}$$

When $y' = y$, $\ell_{\text{hinge}} = 0$.

Multiclass SVM with Hinge Loss

- Recall the hinge loss formulation for binary SVM (without the bias term):

$$\min_{w \in \mathbf{R}^d} \frac{1}{2}||w||^2 + C \sum_{n=1}^{N} \max\left(0, 1 - \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}}\right).$$

Multiclass SVM with Hinge Loss

- Recall the hinge loss formulation for binary SVM (without the bias term):

$$\min_{w \in \mathbf{R}^d} \frac{1}{2}||w||^2 + C \sum_{n=1}^{N} \max\left(0, 1 - \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}}\right).$$

- The multiclass objective:

$$\min_{w \in \mathbf{R}^d} \frac{1}{2}||w||^2 + C \sum_{n=1}^{N} \max_{y' \in \mathcal{Y}}\left(\Delta(y, y') - \underbrace{\left\langle w, \left(\Psi(x, y) - \Psi(x, y')\right)\right\rangle}_{\text{margin}}\right)$$

  - $\Delta(y, y')$ as target margin for each class.
  - If margin $m_{n,y'}(w)$ meets or exceeds its target $\Delta(y^{(n)}, y')$ $\forall y \in \mathcal{Y}$, then no loss on example n.

Multiclass SVM with Hinge Loss

- Recall the hinge loss formulation for binary SVM (without the bias term):

$$\min_{w \in \mathbf{R}^d} \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{N} \max\left(0, 1 - \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}}\right).$$

- The multiclass objective:

$$\min_{w \in \mathbf{R}^d} \frac{1}{2}\|w\|^2 + C \sum_{n=1}^{N} \max_{y' \in \mathcal{Y}} \left(\Delta(y, y') - \underbrace{\langle w, (\Psi(x,y) - \Psi(x,y')) \rangle}_{\text{margin}}\right)$$

  - $\Delta(y, y')$ as target margin for each class.
  - If margin $m_{n,y'}(w)$ meets or exceeds its target $\Delta(y^{(n)}, y')$ $\forall y \in \mathcal{Y}$, then no loss on example n.

If exceeds margin, loss is negative, which must be smaller than loss of the true class, which is zero.

Recap: What Have We Got?

- Problem: Multiclass classification $\mathcal{Y} = \{1, \ldots, k\}$

Recap: What Have We Got?

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

Recap: What Have We Got?

- Problem: Multiclass classification $\mathcal{Y} = \{1, \ldots, k\}$
- Solution 1: One-vs-All
  - Train k models: $h_1(x), \ldots, h_k(x) : \mathcal{X} \to \mathbf{R}$.
  - Predict with $\arg\max_{y \in \mathcal{Y}} h_y(x)$.
  - Gave simple example where this fails for linear classifiers
- Solution 2: Multiclass loss
  - Train one model: $h(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}$.
  - Prediction involves solving $\arg\max_{y \in \mathcal{Y}} h(x, y)$.

Does it work better in practice?

- Paper by Rifkin & Klautau: "In Defense of One-Vs-All Classification" (2004)
  - 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)

Does it work better in practice?

- Paper by Rifkin & Klautau: "In Defense of One-Vs-All Classification" (2004)
  - 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

Why Are We Bothering with Multiclass?

- The framework we have developed for multiclass
  - compatibility features / scoring functions
  - multiclass margin
  - target margin / multiclass loss

Why Are We Bothering with Multiclass?

- The framework we have developed for multiclass
  - compatibility features / scoring functions
  - multiclass margin
  - target margin / multiclass loss
- Generalizes to situations where k is very large and one-vs-all is intractable.

Why Are We Bothering with Multiclass?

- The framework we have developed for multiclass
  - compatibility features / scoring functions
  - multiclass margin
  - target margin / multiclass loss
- Generalizes to situations where k is very large and one-vs-all is intractable.
- Key idea is that we can generalize across outputs y by using features of y.

Introduction to Structured Prediction

Example: Part-of-speech (POS) Tagging

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

x	$\underbrace{\text{[START]}}_{x_0}$	$\underbrace{\text{He}}_{x_1}$	$\underbrace{\text{eats}}_{x_2}$	$\underbrace{\text{apples}}_{x_3}$
y	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

Example: Part-of-speech (POS) Tagging

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

x	$\underbrace{\text{[START]}}_{x_0}$	$\underbrace{\text{He}}_{x_1}$	$\underbrace{\text{eats}}_{x_2}$	$\underbrace{\text{apples}}_{x_3}$
y	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

- $\mathcal{V} = \{\text{all English words}\} \cup \{\text{[START]}, \text{"."}\}$
- $\mathcal{X} = \mathcal{V}^n$, $n = 1, 2, 3, \ldots$ [Word sequences of any length]

Example: Part-of-speech (POS) Tagging

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

x	$\underbrace{\text{[START]}}_{x_0}$	$\underbrace{\text{He}}_{x_1}$	$\underbrace{\text{eats}}_{x_2}$	$\underbrace{\text{apples}}_{x_3}$
y	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

- $\mathcal{V} = \{\text{all English words}\} \cup \{\text{[START]}, \text{"."}\}$
- $\mathcal{X} = \mathcal{V}^n$, $n = 1, 2, 3, \ldots$ [Word sequences of any length]
- $\mathcal{P} = \{\text{START}, \text{Pronoun}, \text{Verb}, \text{Noun}, \text{Adjective}\}$
- $\mathcal{Y} = \mathcal{P}^n$, $n = 1, 2, 3, \ldots$[Part of speech sequence of any length]

Multiclass Hypothesis Space

- Discrete output space: $\mathcal{Y}(x)$
  - Very large but has structure, e.g., linear chain (sequence labeling), tree (parsing)
  - Size depends on input x

Multiclass Hypothesis Space

- Discrete output space: $\mathcal{Y}(x)$
  - Very large but has structure, e.g., linear chain (sequence labeling), tree (parsing)
  - Size depends on input x
- Base Hypothesis Space: $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}\}$
  - $h(x, y)$ gives compatibility score between input x and output y

Multiclass Hypothesis Space

- Discrete output space: $\mathcal{Y}(x)$
  - Very large but has structure, e.g., linear chain (sequence labeling), tree (parsing)
  - Size depends on input x
- Base Hypothesis Space: $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}\}$
  - $h(x, y)$ gives compatibility score between input x and output y
- Multiclass hypothesis space

$$\mathcal{F} = \left\{ x \mapsto \underset{y \in \mathcal{Y}}{\arg\max}\, h(x, y) \mid h \in \mathcal{H} \right\}$$

- Final prediction function is an $f \in \mathcal{F}$.
- For each $f \in \mathcal{F}$ there is an underlying compatibility score function $h \in \mathcal{H}$.

Structured Prediction

- Part-of-speech tagging

$x:$	he	eats	apples
$y:$	pronoun	verb	noun

- Multiclass hypothesis space:

$$h(x,y) = w^T \Psi(x,y) \tag{18}$$

$$\mathcal{F} = \left\{ x \mapsto \arg\max_{y \in \mathcal{Y}} h(x,y) \mid h \in \mathcal{H} \right\} \tag{19}$$

- A special case of multiclass classification
- How to design the feature map Ψ? What are the considerations?

Structured Prediction

- Part-of-speech tagging

x:	he	eats	apples
y:	pronoun	verb	noun

- Multiclass hypothesis space:

$$h(x,y) = w^T \Psi(x,y) \tag{18}$$

$$\mathcal{F} = \left\{ x \mapsto \underset{y \in \mathcal{Y}}{\arg\max}\, h(x,y) \mid h \in \mathcal{H} \right\} \tag{19}$$

- A special case of multiclass classification
- How to design the feature map Ψ? What are the considerations?

contextual dependence, efficient argmax

Unary features

- A **unary feature** only depends on
  - the label at a single position, y_i, and x
- Example:

$$
\begin{aligned}
\phi_1(x, y_i) &= 1(x_i = \text{runs})1(y_i = \text{Verb}) \\
\phi_2(x, y_i) &= 1(x_i = \text{runs})1(y_i = \text{Noun}) \\
\phi_3(x, y_i) &= 1(x_{i-1} = \text{He})1(x_i = \text{runs})1(y_i = \text{Verb})
\end{aligned}
$$

Markov features

- A **markov feature** only depends on
  - two adjacent labels, y_{i-1} and y_i, and x
- Example:

$$
\begin{aligned}
\theta_1(x, y_{i-1}, y_i) &= 1(y_{i-1} = \text{Pronoun})1(y_i = \text{Verb}) \\
\theta_2(x, y_{i-1}, y_i) &= 1(y_{i-1} = \text{Pronoun})1(y_i = \text{Noun})
\end{aligned}
$$

- Reminiscent of Markov models in the output space
- Possible to have higher-order features

Local Feature Vector and Compatibility Score

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

$$\begin{aligned}\Psi_i(x, y_{i-1}, y_i) &= (\phi_1(x, y_i), \phi_2(x, y_i), \ldots, \\ &\quad \theta_1(x, y_{i-1}, y_i), \theta_2(x, y_{i-1}, y_i), \ldots)\end{aligned}$$

Local Feature Vector and Compatibility Score

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

$$\begin{aligned}\Psi_i(x, y_{i-1}, y_i) \quad = \quad & (\phi_1(x, y_i), \phi_2(x, y_i), \ldots, \\ & \theta_1(x, y_{i-1}, y_i), \theta_2(x, y_{i-1}, y_i), \ldots)\end{aligned}$$

- And **local compatibility score** at position i: $\langle w, \Psi_i(x, y_{i-1}, y_i)\rangle$.

Local Feature Vector and Compatibility Score

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

$$\begin{aligned}\Psi_i(x, y_{i-1}, y_i) &= (\phi_1(x, y_i), \phi_2(x, y_i), \ldots, \\ &\quad \theta_1(x, y_{i-1}, y_i), \theta_2(x, y_{i-1}, y_i), \ldots)\end{aligned}$$

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

$$\sum_i \langle w, \Psi_i(x, y_{i-1}, y_i)\rangle = \left\langle w, \sum_i \Psi_i(x, y_{i-1}, y_i)\right\rangle = \langle w, \Psi(x, y)\rangle, \tag{20}$$

where we define the **sequence feature vector** by

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

Structured perceptron

Given a dataset $\mathcal{D} = \{(x, y)\}$;
Initialize $w \leftarrow 0$;
for $\text{iter} = 1, 2, \ldots, T$ **do**
  for $(x, y) \in \mathcal{D}$ **do**
    $\hat{y} = \arg\max_{y' \in \mathcal{Y}(x)} w^T \psi(x, y')$;
    if $\hat{y} \neq y$ **then** `// We've made a mistake`
      $w \leftarrow w + \Psi(x, y)$; `// Move the scorer towards` $\psi(x, y)$
      $w \leftarrow w - \Psi(x, \hat{y})$; `// Move the scorer away from` $\psi(x, \hat{y})$
    end
  end
end

Structured perceptron

Given a dataset $\mathcal{D} = \{(x, y)\}$;
Initialize $w \leftarrow 0$;
for iter $= 1, 2, \ldots, T$ **do**
  for $(x, y) \in \mathcal{D}$ **do**
    $\hat{y} = \arg\max_{y' \in \mathcal{Y}(x)} w^T \psi(x, y')$;
    if $\hat{y} \neq y$ **then** `// We've made a mistake`
      $w \leftarrow w + \Psi(x, y)$; `// Move the scorer towards` $\psi(x, y)$
      $w \leftarrow w - \Psi(x, \hat{y})$; `// Move the scorer away from` $\psi(x, \hat{y})$
    end
  end
end

Identical to the multiclass perceptron algorithm except the $\arg\max$ is now over the structured output space $\mathcal{Y}(x)$.

Structured hinge loss

- Recall the generalized hinge loss

$$\ell_{\text{hinge}}(y,\hat{y}) \stackrel{\text{def}}{=} \max_{y' \in \mathcal{Y}(x)} \left(\Delta(y,y') + \left\langle w, \left(\Psi(x,y') - \Psi(x,y)\right)\right\rangle\right) \tag{21}$$

- What is $\Delta(y,y')$ for two sequences?

Structured hinge loss

- Recall the generalized hinge loss

$$\ell_{\text{hinge}}(y, \hat{y}) \stackrel{\text{def}}{=} \max_{y' \in \mathcal{Y}(x)} \left(\Delta(y, y') + \left\langle w, \left(\Psi(x, y') - \Psi(x, y) \right) \right\rangle \right) \tag{21}$$

- What is $\Delta(y, y')$ for two sequences?
- **Hamming loss** is common:

$$\Delta(y, y') = \frac{1}{L} \sum_{i=1}^{L} 1(y_i \neq y'_i)$$

  where L is the sequence length.
- Can generalize to the cost-sensitive version using $\delta(y_i, y'_i)$

Structured SVM

Exercise:

- Write down the objective of structured SVM using the structured hinge loss.
- Stochastic sub-gradient descent for structured SVM (similar to HW3 P3)
- Compare with the structured perceptron algorithm

The argmax problem for sequences

Figure by Daumé III. A course in machine learning. Figure 17.1.

The argmax problem for sequences

Problem To compute predictions, we need to find $\arg\max_{y \in \mathcal{Y}(x)} \langle w, \Psi(x, y) \rangle$, and $|\mathcal{Y}(x)|$ is exponentially large.

Figure by Daumé III. A course in machine learning. Figure 17.1.

The argmax problem for sequences

Problem To compute predictions, we need to find $\arg\max_{y \in \mathcal{Y}(x)} \langle w, \Psi(x, y) \rangle$, and $|\mathcal{Y}(x)|$ is exponentially large.

Observation $\Psi(x, y)$ decomposes to $\sum_i \Psi_i(x, y)$.

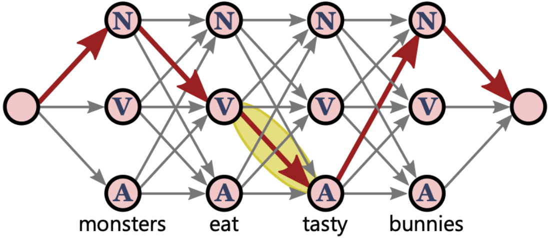
Figure by Daumé III. A course in machine learning. Figure 17.1.

The argmax problem for sequences

Problem To compute predictions, we need to find $\arg\max_{y \in \mathcal{Y}(x)} \langle w, \Psi(x,y) \rangle$, and $|\mathcal{Y}(x)|$ is exponentially large.

Observation $\Psi(x,y)$ decomposes to $\sum_i \Psi_i(x,y)$.

Solution Dynamic programming (similar to the Viterbi algorithm)



What's the running time?

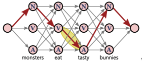
Figure by Daumé III. A course in machine learning. Figure 17.1.

The argmax problem for sequences

Problem To compute predictions, we need to find $\arg\max_{y \in \mathcal{Y}(x)} \langle w, \Psi(x,y) \rangle$, and $|\mathcal{Y}(x)|$ is exponentially large.

Observation $\Psi(x,y)$ decomposes to $\sum_i \Psi_i(x,y)$.

Solution Dynamic programming (similar to the Viterbi algorithm)



What's the running time?

Figure by Daumé III. A course in machine learning. Figure 17.1.

Let $K = |\mathcal{Y}|$, DP runtime $O(K^2 L)$, mth order Markov feature has runtime $O(K^m L)$, naive runtime $O(K^L)$.

The argmax problem in general

Efficient problem-specific algorithms:

problem	structure	algorithm
constituent parsing	binary trees with context-free features	CYK
dependency parsing	spanning trees with edge features	Chu-Liu-Edmonds
image segmentation	2d with adjacent-pixel features	graph cuts

The argmax problem in general

Efficient problem-specific algorithms:

problem	structure	algorithm
constituent parsing	binary trees with context-free features	CYK
dependency parsing	spanning trees with edge features	Chu-Liu-Edmonds
image segmentation	2d with adjacent-pixel features	graph cuts

General algorithm:

- Integer linear programming (ILP)

$$\max_z a^T z \quad \text{s.t. linear constraints on } z \tag{22}$$

- z: indicator of substructures, e.g., $\mathbb{I}\{y_i = \text{article and } y_{i+1} = \text{noun}\}$
- constraints: z must correspond to a valid structure

Conclusion

Multiclass algorithms

- Reduce to binary classification, e.g., OvA, AvA, ECCO
  - Good enough for simple multiclass problems
- Generalize binary classification algorithms using multiclass loss
  - Useful for problems with extremely large output space, e.g., structured prediction
  - Related problems: ranking, multi-label classification