

Multiclass Classification

Mengye Ren

NYU

Nov 7, 2023

Overview

Motivation

- So far, most algorithms we've learned are designed for binary classification.
 - Sentiment analysis (positive vs. negative)
 - Spam filter (spam vs. non-spam)
- Many real-world problems have more than two classes.
 - Document classification (over 10 classes)
 - Object recognition (over 20k classes)
 - Face recognition (millions of classes)
- What are some potential issues when we have a large number of classes?
 - Computation cost
 - Class imbalance
 - Different cost of errors

Today's lecture

- How to *reduce* multiclass classification to binary classification?
 - We can think of binary classifier or linear regression as a black box. Naive ways:
 - E.g. multiple binary classifiers produce a binary code for each class (000, 001, 010)
 - E.g. a linear regression produces a numerical value for each class (1.0, 2.0, 3.0)
- How do we *generalize* binary classification algorithm to the multiclass setting?
 - We also need to think about the loss function.
- Example of very large output space: structured prediction.
 - Multi-class: Mutually exclusive class structure.
 - Text: Temporal relational structure.

Reduction to Binary Classification

One-vs-All / One-vs-Rest

Setting

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

Training

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

Prediction

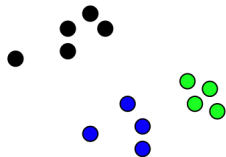
- Majority vote:

$$h(x) = \arg \max_{i \in \{1, \dots, k\}} h_i(x)$$

- Ties can be broken arbitrarily.

OvA: 3-class example (linear classifier)

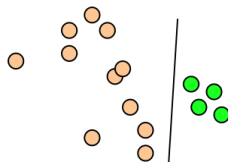
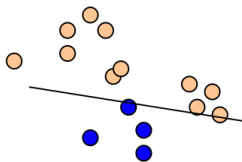
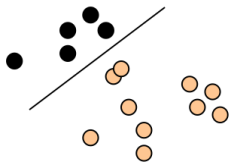
Consider a dataset with three classes:



Assumption: each class is linearly separable from the rest.

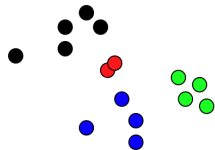
Ideal case: only target class has positive score.

Train OvA classifiers:



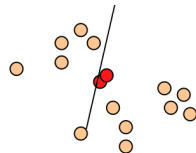
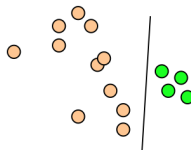
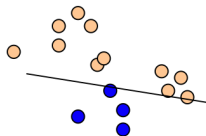
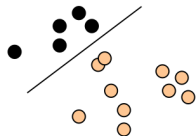
OvA: 4-class non linearly separable example

Consider a dataset with four classes:



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

Train OvA classifiers:



All vs All / One vs One / All pairs

Setting

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

Training

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

Prediction

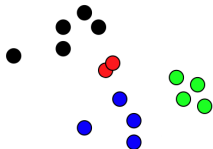
- Majority vote (each class gets $k-1$ votes)

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

- Tournament
- Ties can be broken arbitrarily.

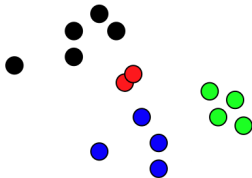
AvA: four-class example

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	$O(kB_{\text{train}}(n))$	$O(k^2B_{\text{train}}(n/k))$
	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).

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

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: 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.
- Trade-off between code distance and binary classification performance.
- Nice theoretical results [Allwein et al., 2000] (also incorporates AvA).

Reduction-based approaches:

- Reducing multiclass classification to binary classification: OvA, AvA
- 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.

Multiclass Loss

Binary Logistic Regression

- Given an input x , we would like to output a classification between $(0,1)$.

$$f(x) = \textit{sigmoid}(z) = \frac{1}{1 + \exp(-z)} = \frac{1}{1 + \exp(-w^\top x - b)}. \quad (1)$$

- The other class is represented in $1 - f(x)$:

$$1 - f(x) = \frac{\exp(-w^\top x - b)}{1 + \exp(-w^\top x - b)} = \frac{1}{1 + \exp(w^\top x + b)} = \textit{sigmoid}(-z). \quad (2)$$

- Another way to view: one class has $(+w, +b)$ and the other class has $(-w, -b)$.

Multi-class Logistic Regression

- Now what if we have one w_c for each class c ?

$$f_c(x) = \frac{\exp(w_c^\top x) + b_c}{\sum_c \exp(w_c^\top x + b_c)} \quad (3)$$

- Also called “softmax” in neural networks.
- Loss function: $L = \sum_i -y_c^{(i)} \log f_c(x^{(i)})$
- Gradient: $\frac{\partial L}{\partial z} = f - y$. Recall: MSE loss.

Comparison to OvA

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

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

- Intuitively, $h_i(x)$ 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, to predict (x, i) correctly we only need

$$h_i(x) > h_j(x) \quad \forall j \neq i. \quad (4)$$

Multiclass Perceptron

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

- Multiclass perceptron:

Given a multiclass dataset $\mathcal{D} = \{(x, y)\}$;

Initialize $w \leftarrow 0$;

for $iter = 1, 2, \dots, T$ **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

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 ?

$$w_i^T x = w^T \psi(x, i) \quad (5)$$

$$h_i(x) = h(x, i) \quad (6)$$

- Encode labels in the feature space.
- Score for each label \rightarrow score for the “*compatibility*” of a label and an input.

The Multivector Construction

How to construct the feature map ψ ?

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

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

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

Rewrite multiclass perceptron

Multiclass perceptron using the multivector construction.

Given a multiclass dataset $\mathcal{D} = \{(x, y)\}$;

Initialize $w \leftarrow 0$;

for $iter = 1, 2, \dots, T$ **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?

Features

Toy multiclass example: Part-of-speech classification

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

How to construct the feature vector?

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

$$\Psi(x, y) = (\psi_1(x, y), \psi_2(x, y), \psi_3(x, y), \dots, \psi_d(x, y)) \quad (7)$$

- Size can be bounded by d .

Features

Sample training data:

The boy grabbed the apple and ran away quickly .

Feature:

$$\psi_1(x, y) = \mathbb{1}[x = \text{apple AND } y = \text{NOUN}]$$

$$\psi_2(x, y) = \mathbb{1}[x = \text{run AND } y = \text{NOUN}]$$

$$\psi_3(x, y) = \mathbb{1}[x = \text{run AND } y = \text{VERB}]$$

$$\psi_4(x, y) = \mathbb{1}[x \text{ ENDS_IN_ly AND } y = \text{ADVERB}]$$

...

- E.g., $\Psi(x = \text{run}, y = \text{NOUN}) = (0, 1, 0, 0, \dots)$
- After training, what's w_1, w_2, w_3, w_4 ?
- No need to include features 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: $\text{template} \rightarrow \{1, 2, \dots, d\}$.

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?

Margin for Multiclass

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

$$y^{(n)} w^T x^{(n)} \quad (8)$$

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

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

$$h(x^{(n)}, y^{(n)}) - h(x^{(n)}, y). \quad (9)$$

- 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

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

$$\text{s.t.} \quad \underbrace{y^{(n)} w^T x^{(n)}}_{\text{margin}} \geq 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D} \quad (11)$$

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

$$m_{n,y}(w) \stackrel{\text{def}}{=} \underbrace{\left\langle w, \Psi(x^{(n)}, y^{(n)}) \right\rangle}_{\text{score of correct class}} - \underbrace{\left\langle w, \Psi(x^{(n)}, y) \right\rangle}_{\text{score of other class}} \quad (12)$$

$$\min_w \quad \frac{1}{2} \|w\|^2 \quad (13)$$

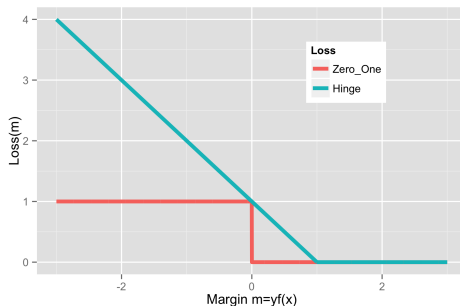
$$\text{s.t.} \quad m_{n,y}(w) \geq 1 \quad \forall (x^{(n)}, y^{(n)}) \in \mathcal{D}, y \neq y^{(n)} \quad (14)$$

Exercise: write the objective for the non-separable case

Recap: hinge loss for binary classification

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

$$\ell_{\text{hinge}}(y, \hat{y}) = \max(0, 1 - yf(x)) \quad (15)$$



Generalized hinge loss

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

$$\Delta(y, y') = \mathbb{I}\{y \neq y'\} \quad (16)$$

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

$$\hat{y} \stackrel{\text{def}}{=} \arg \max_{y' \in \mathcal{Y}} \langle w, \Psi(x, y') \rangle \quad (17)$$

$$\implies \langle w, \Psi(x, y) \rangle \leq \langle w, \Psi(x, \hat{y}) \rangle \quad (18)$$

$$\implies \Delta(y, \hat{y}) \leq \Delta(y, \hat{y}) - \langle w, (\Psi(x, y) - \Psi(x, \hat{y})) \rangle \quad \text{When are they equal?} \quad (19)$$

- Generalized hinge loss:

$$\ell_{\text{hinge}}(y, x, w) \stackrel{\text{def}}{=} \max_{y' \in \mathcal{Y}} (\Delta(y, y') - \langle w, (\Psi(x, y) - \Psi(x, y')) \rangle) \quad (20)$$

Multiclass SVM with Hinge Loss

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

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

- The multiclass objective:

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

- $\Delta(y, y')$ 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 .

Recap: What Have We Got?

- Problem: Multiclass classification $\mathcal{Y} = \{1, \dots, k\}$
- Solution 1: One-vs-All
 - Train k models: $h_1(x), \dots, h_k(x) : \mathcal{X} \rightarrow \mathbb{R}$.
 - 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: $h(x, y) : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.
 - 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)
- 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
- 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	$\underbrace{[\text{START}]}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

- $\mathcal{V} = \{\text{all English words}\} \cup \{[\text{START}], ", ."]\}$
- $\mathcal{X} = \mathcal{V}^n, n = 1, 2, 3, \dots$ [Word sequences of any length]
- $\mathcal{P} = \{\text{START, Pronoun, Verb, Noun, Adjective}\}$
- $\mathcal{Y} = \mathcal{P}^n, n = 1, 2, 3, \dots$ [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
- Base Hypothesis Space: $\mathcal{H} = \{h: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}\}$
 - $h(x, y)$ gives **compatibility score** between input x and output y
- Multiclass hypothesis space

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

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

Structured Prediction

- Part-of-speech tagging

x :	he	eats	apples
y :	pronoun	verb	noun

- Multiclass hypothesis space:

$$h(x, y) = w^T \Psi(x, y) \quad (21)$$

$$\mathcal{F} = \left\{ x \mapsto \arg \max_{y \in \mathcal{Y}} h(x, y) \mid h \in \mathcal{H} \right\} \quad (22)$$

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

Unary features

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

$$\phi_1(x, y_i) = \mathbb{1}[x_i = \text{runs}] \mathbb{1}[y_i = \text{Verb}]$$

$$\phi_2(x, y_i) = \mathbb{1}[x_i = \text{runs}] \mathbb{1}[y_i = \text{Noun}]$$

$$\phi_3(x, y_i) = \mathbb{1}[x_{i-1} = \text{He}] \mathbb{1}[x_i = \text{runs}] \mathbb{1}[y_i = \text{Verb}]$$

Markov features

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

$$\theta_1(x, y_{i-1}, y_i) = \mathbb{1}[y_{i-1} = \text{Pronoun}] \mathbb{1}[y_i = \text{Verb}]$$

$$\theta_2(x, y_{i-1}, y_i) = \mathbb{1}[y_{i-1} = \text{Pronoun}] \mathbb{1}[y_i = \text{Noun}]$$

- 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):

$$\Psi_i(x, y_{i-1}, y_i) = (\phi_1(x, y_i), \phi_2(x, y_i), \dots, \theta_1(x, y_{i-1}, y_i), \theta_2(x, y_{i-1}, y_i), \dots)$$

- 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:

$$\sum_i \langle w, \Psi_i(x, y_{i-1}, y_i) \rangle = \left\langle w, \sum_i \Psi_i(x, y_{i-1}, y_i) \right\rangle = \langle w, \Psi(x, y) \rangle, \quad (23)$$

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

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

Structured perceptron

Given a dataset $\mathcal{D} = \{(x, y)\}$;

Initialize $w \leftarrow 0$;

for $iter = 1, 2, \dots, T$ **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

$$\ell_{\text{hinge}}(y, \hat{y}) \stackrel{\text{def}}{=} \max_{y' \in \mathcal{Y}(\mathbf{x})} (\Delta(y, y') + \langle w, (\Psi(\mathbf{x}, y') - \Psi(\mathbf{x}, y)) \rangle) \quad (24)$$

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

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

where L is the sequence length.

Exercise:

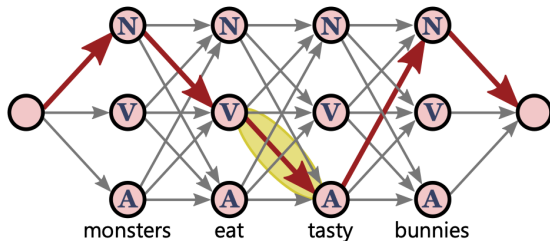
- Write down the objective of structured SVM using the structured hinge loss.
- Stochastic sub-gradient descent for structured SVM (similar to HW3 P3)
- Compare with the structured perceptron algorithm

The argmax problem for sequences

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)



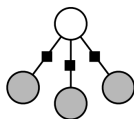
What's the running time?

Conditional random field (CRF)

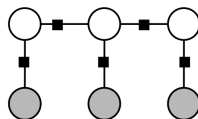
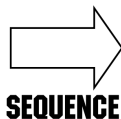
- Recall that we can write logistic regression in a general form:

$$p(y|x) = \frac{1}{Z(x)} \exp(w^\top \psi(x, y)).$$

- Z is normalization constant: $Z(x) = \sum_{y \in Y} \exp(w^\top \psi(x, y))$.
- Example: linear chain $\{y_t\}$
- We can incorporate unary and Markov features: $p(y|x) = \frac{1}{Z(x)} \exp(\sum_t w^\top \psi(x, y_t, y_{t-1}))$



Logistic Regression



Linear-chain CRFs

Conditional random field (CRF)

- Compared to Structured SVM, CRF has a probabilistic interpretation.
- We can draw samples in the output space.
- How do we learn w ? Maximum log likelihood, and regularization term: $\lambda \|w\|^2$
- Loss function:

$$\begin{aligned} l(w) &= -\frac{1}{N} \sum_{i=1}^N \log p(y^{(i)} | x^{(i)}) + \frac{1}{2} \lambda \|w\|^2 \\ &= -\frac{1}{N} \sum_i \sum_t \sum_k w_k \psi_k(y_t^{(i)}, y_{t-1}^{(i)}) + \frac{1}{N} \sum_i \log Z(x^{(i)}) + \frac{1}{2} \sum_k \lambda w_k^2 \end{aligned}$$

Conditional random field (CRF)

- Loss function:

$$l(w) = -\frac{1}{N} \sum_i \sum_t \sum_k w_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)}) + \frac{1}{N} \sum_i \log Z(x^{(i)}) + \frac{1}{2} \sum_k \lambda w_k^2$$

- Gradient:

$$\frac{\partial l(w)}{\partial w_k} = -\frac{1}{N} \sum_i \sum_t \sum_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)}) \quad (25)$$

$$+ \frac{1}{N} \sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp\left(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1})\right) + \sum_k \lambda w_k \quad (26)$$

Conditional random field (CRF)

- What is $\frac{1}{N} \sum_i \sum_t \sum_k \psi_k(x^{(i)}, y_t^{(i)}, y_{t-1}^{(i)})$?
- It is the expectation $\psi_k(x^{(i)}, y_t, y_{t-1})$ under the empirical distribution $\tilde{p}(x, y) = \frac{1}{N} \sum_i \mathbb{1}[x = x^{(i)}] \mathbb{1}[y = y^{(i)}]$.

Conditional random field (CRF)

- What is $\frac{1}{N} \sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1}))$?

$$\frac{1}{N} \sum_i \frac{\partial}{\partial w_k} \log \sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1})) \quad (27)$$

$$= \frac{1}{N} \sum_i \left[\sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1})) \right]^{-1} \quad (28)$$

$$\left[\sum_{y' \in Y} \exp(\sum_t \sum_{k'} w_{k'} \psi_{k'}(x^{(i)}, y'_t, y'_{t-1})) \sum_t \psi_k(x^{(i)}, y'_t, y'_{t-1}) \right] \quad (29)$$

$$= \frac{1}{N} \sum_i \sum_t \sum_{y' \in Y} p(y'_t, y'_{t-1} | x) \psi_k(x^{(i)}, y'_t, y'_{t-1}) \quad (30)$$

- It is the expectation of $\psi_k(x^{(i)}, y'_t, y'_{t-1})$ under the model distribution $p(y'_t, y'_{t-1} | x)$.

Conditional random field (CRF)

- To compute the gradient, we need to infer expectation under the model distribution $p(y|x)$.
- Compare the learning algorithms: in structured SVM we need to compute the argmax, whereas in CRF we need to compute the model expectation.
- Both problems are NP-hard for general graphs.

CRF Inference

- In the linear chain structure, we can use the forward-backward algorithm for inference, similar to Viterbi.
- Initiate $\alpha_j(1) = \exp(w^\top \psi(y_1 = j, x_1))$
- Recursion: $\alpha_j(t) = \sum_i \alpha_i(t-1) \exp(w^\top \psi(y_t = j, y_{t-1} = i, x_t))$
- Result: $Z(x) = \sum_j \alpha_j(T)$
- Similar for the backward direction.
- Test time, again use Viterbi algorithm to infer argmax.
- The inference algorithm can be generalized to belief propagation (BP) in a tree structure (exact inference).
- In general graphs, we rely on approximate inference (e.g. loopy belief propagation).

- POS tag Relationship between constituents, e.g. NP is likely to be followed by a VP.
- Semantic segmentation
Relationship between pixels, e.g. a grass pixel is likely to be next to another grass pixel, and a sky pixel is likely to be above a grass pixel.
- Multi-label learning
An image may contain multiple class labels, e.g. a bus is likely to co-occur with a car.

Multiclass algorithms

- Reduce to binary classification, e.g., OvA, AvA
 - Good enough for simple multiclass problems
 - They don't scale and have simplified assumptions
- Generalize binary classification algorithms using multiclass loss
 - Multi-class perceptron, multi-class logistics regression, multi-class SVM
- Structured prediction: Structured SVM, CRF. Data containing structure. Extremely large output space. Text and image applications.