Statistical Learning Models for Text and Graph Data Sequence Labeling and Structured Output Learning: Conditional Models and Global Classifiers

Yangqiu Song

Hong Kong University of Science and Technology

yqsong@cse.ust.hk

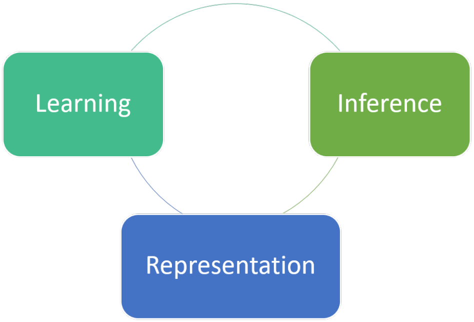
November 1, 2019

∗Contents are based on materials created by Vivek Srikumar, Dan Roth, Andrew Ng

Reference Content

- Vivek Srikumar. CS 6355 Structured Prediction. https://svivek.com/teaching/structured-prediction/spring2018/
- Dan Roth. CS546: Machine Learning and Natural Language . http://l2r.cs.uiuc.edu/~danr/Teaching/CS546-16/
- Andrew Ng. CS229: Machine Learning. http://cs229.stanford.edu/

Course Topics

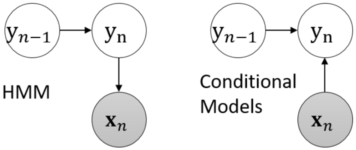


- Representation: language models, word embeddings, topic models, knowledge graphs
- Learning: supervised learning,unsupervised learning, semi-supervised learning, distant supervision, indirect supervision, sequence models, deep learning, optimization techniques
- Inference: constraint modeling, joint inference, search algorithms

Overview

The Next-state Model

$$P(y_n|y_{n-1}, y_{n-2}, \ldots, \mathbf{x}_n, \mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \ldots) = P(y_n|y_{n-1}, \mathbf{x}_n)$$



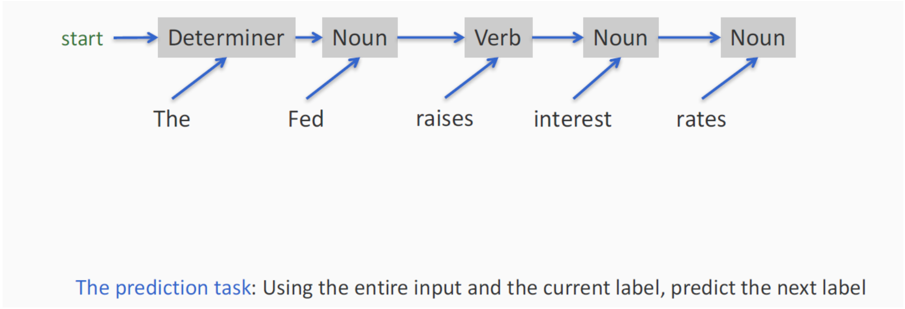
- This assumption lets us write the conditional probability of the output as

$$P(y_{1:N}|\mathbf{x}_{1:N}) = \prod_n P(y_n|y_{n-1}, \mathbf{x}_n)$$

Maximum Entropy Markov Model (MEMM)

Goal: Compute $P(y_{1:N}|\mathbf{x}_{1:N}, \mathbf{w}) = \prod_n P(y_n|y_{n-1}, \mathbf{x}_{1:N})$ where

$$P(y_n|y_{n-1}, \mathbf{x}_{1:N}) \propto \exp\left(\mathbf{w}^\top \phi(\mathbf{x}, n, y_n, y_{n-1})\right)$$

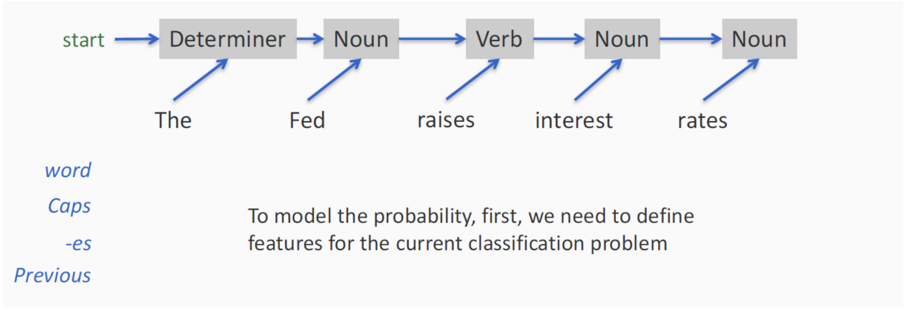


The prediction task: Using the entire input and the current label, predict the next label

Maximum Entropy Markov Model (MEMM)

Goal: Compute $P(y_{1:N}|\mathbf{x}_{1:N}, \mathbf{w}) = \prod_n P(y_n|y_{n-1}, \mathbf{x}_{1:N})$ where

$$P(y_n|y_{n-1}, \mathbf{x}_{1:N}) \propto \exp\left(\mathbf{w}^\top \phi(\mathbf{x}, n, y_n, y_{n-1})\right)$$

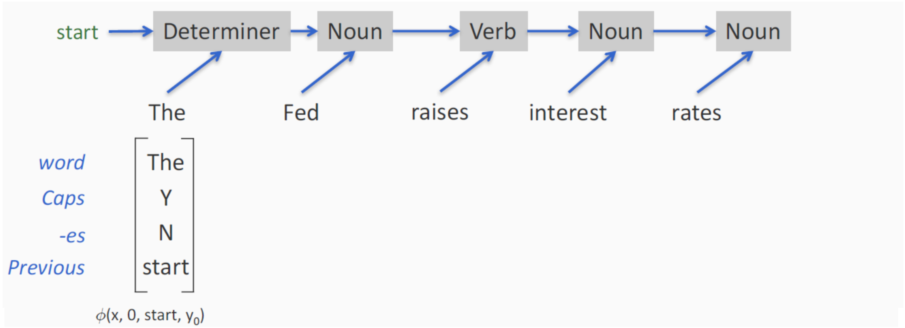


To model the probability, first, we need to define features for the current classification problem

Maximum Entropy Markov Model (MEMM)

Goal: Compute $P(y_{1:N}|\mathbf{x}_{1:N}, \mathbf{w}) = \prod_n P(y_n|y_{n-1}, \mathbf{x}_{1:N})$ where

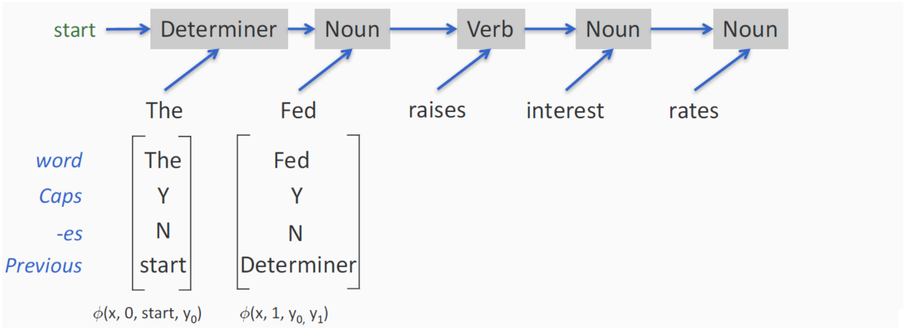
$$P(y_n|y_{n-1}, \mathbf{x}_{1:N}) \propto \exp\left(\mathbf{w}^\top \phi(\mathbf{x}, n, y_n, y_{n-1})\right)$$



Maximum Entropy Markov Model (MEMM)

Goal: Compute $P(y_{1:N}|\mathbf{x}_{1:N}, \mathbf{w}) = \prod_n P(y_n|y_{n-1}, \mathbf{x}_{1:N})$ where

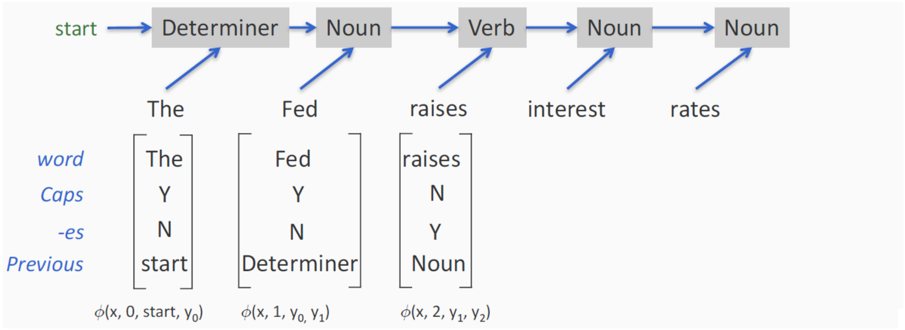
$$P(y_n|y_{n-1}, \mathbf{x}_{1:N}) \propto \exp\left(\mathbf{w}^\top \phi(\mathbf{x}, n, y_n, y_{n-1})\right)$$



Maximum Entropy Markov Model (MEMM)

Goal: Compute $P(y_{1:N}|\mathbf{x}_{1:N}, \mathbf{w}) = \prod_n P(y_n|y_{n-1}, \mathbf{x}_{1:N})$ where

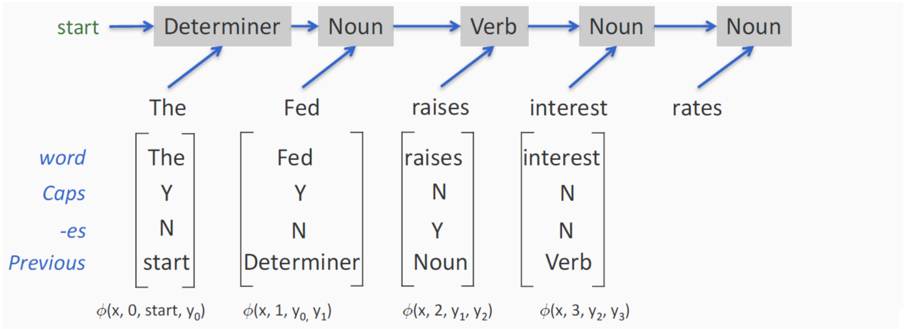
$$P(y_n|y_{n-1}, \mathbf{x}_{1:N}) \propto \exp\left(\mathbf{w}^\top \phi(\mathbf{x}, n, y_n, y_{n-1})\right)$$



Maximum Entropy Markov Model (MEMM)

Goal: Compute $P(y_{1:N}|\mathbf{x}_{1:N}, \mathbf{w}) = \prod_n P(y_n|y_{n-1}, \mathbf{x}_{1:N})$ where

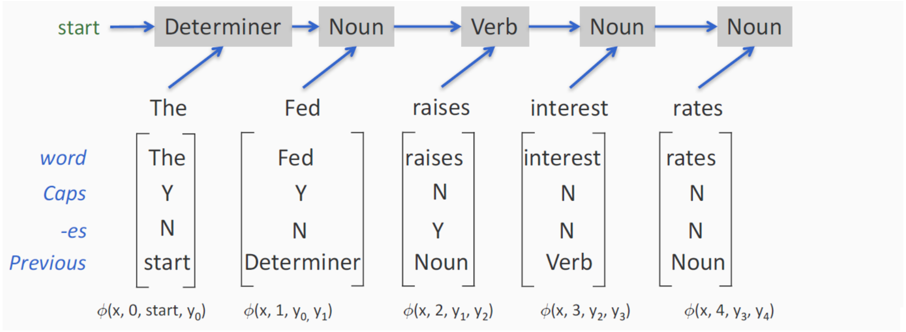
$$P(y_n|y_{n-1}, \mathbf{x}_{1:N}) \propto \exp\left(\mathbf{w}^\top \phi(\mathbf{x}, n, y_n, y_{n-1})\right)$$



Maximum Entropy Markov Model (MEMM)

Goal: Compute $P(y_{1:N}|\mathbf{x}_{1:N}, \mathbf{w}) = \prod_n P(y_n|y_{n-1}, \mathbf{x}_{1:N})$ where

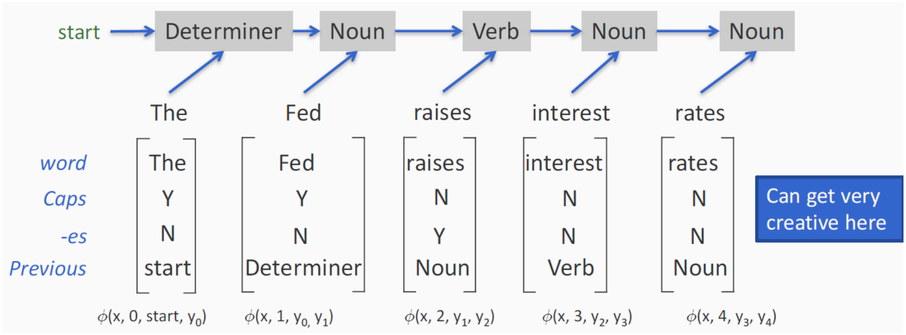
$$P(y_n|y_{n-1}, \mathbf{x}_{1:N}) \propto \exp\left(\mathbf{w}^\top \phi(\mathbf{x}, n, y_n, y_{n-1})\right)$$



Maximum Entropy Markov Model (MEMM)

Goal: Compute $P(y_{1:N}|\mathbf{x}_{1:N}, \mathbf{w}) = \prod_n P(y_n|y_{n-1}, \mathbf{x}_{1:N})$ where

$$P(y_n|y_{n-1}, \mathbf{x}_{1:N}) \propto \exp\left(\mathbf{w}^\top \phi(\mathbf{x}, n, y_n, y_{n-1})\right)$$

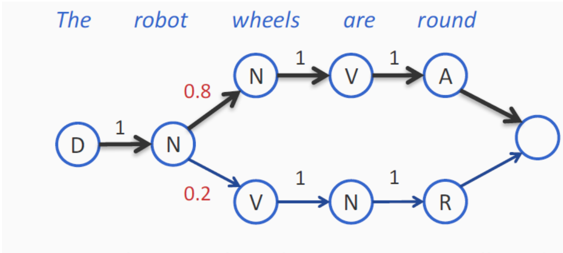


Overview

But ... Local Classifiers $\rightarrow$ Label Bias Problem

- Recall: the independence assumption ("Next-state" classifiers are locally normalized)

$$P(y_n|y_{n-1}, y_{n-2}, \ldots, \mathbf{x}_n, \mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \ldots) = P(y_n|y_{n-1}, \mathbf{x}_n)$$



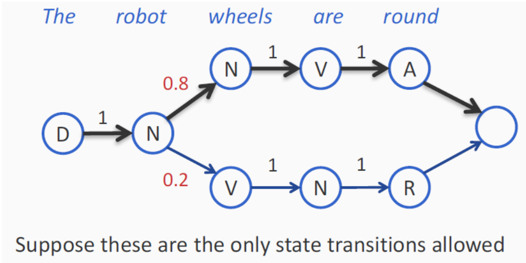
Suppose these are the only state transitions allowed

Option 1: P(D | The) ·
P(N | D, robot) ·
P(N | N, wheels) ·
P(V | N, are) ·
P(A | V, round)

Option 2: P(D | The) ·
P(N | D, robot) ·
P(V | N, wheels) ·
P(N | V, are) ·
P(R| N, round)

But ... Local Classifiers → Label Bias Problem

- The robot wheels Fred round



Suppose these are the only state transitions allowed

Option 1: P(D | The) ·
P(N | D, robot) ·
P(N | N, wheels) ·
~~P(V | N, are)~~ · P(V | N, Fred) ·
P(A | V, round)

Option 2: P(D | The) ·
P(N | D, robot) ·
P(V | N, wheels) ·
~~P(N | V, are)~~ · P(N | V, Fred) ·
P(R| N, round)

- The path scores are the same
- Even if the word Fred is never observed as a verb in the data, it will be predicted as one
- The input Fred does not influence the output at all

Label Bias

- States with a single outgoing transition effectively ignore their input
  - States with lower-entropy next states are less influenced by observations
- Why?
  - Because each the next-state classifiers are locally normalized
  - If a state has fewer next states, each of those will get a higher probability mass
    - and hence preferred
- Surprisingly doesn't affect some tasks
  - E.g., POS tagging

Summary: Local Models for Sequences

- Conditional models
- Use rich features in the mode
- Possibly suffer from label bias problem

Overview

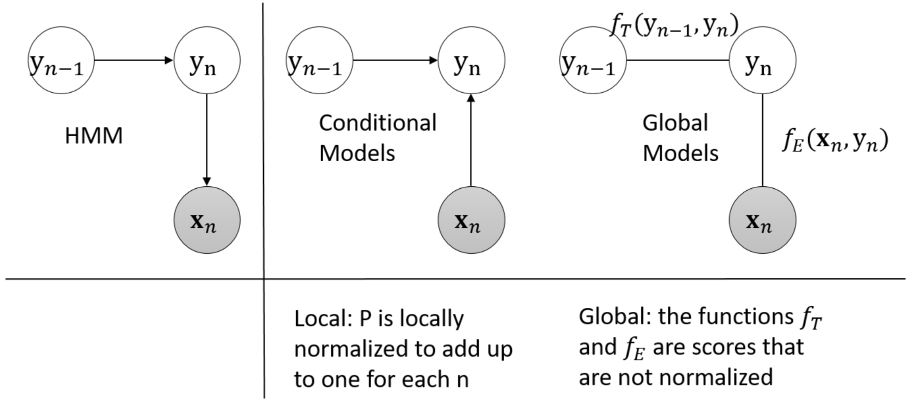
So Far...

- Hidden Markov models
  - Pros: Decomposition of total probability with tractable
  - Cons: Doesn't allow use of features for representing inputs
    - Also, generative model is not really a downside, but we may get better performance with conditional models if we care only about predictions
- Local, Conditional Markov Models
  - Pros: Conditional model, allows features to be used
  - Cons: Label bias problem

Global Models

- Train the predictor globally
    - Instead of training local decisions independently
- Normalize globally
    - Make each edge in the model undirected
    - Not associated with a probability, but just a “score”
- Recall the difference between local vs. global for multiclass

HMM vs. A Local Model vs. A Global Model

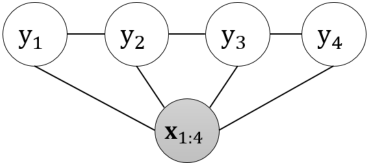


Local: P is locally normalized to add up to one for each n

Global: the functions f_T and f_E are scores that are not normalized

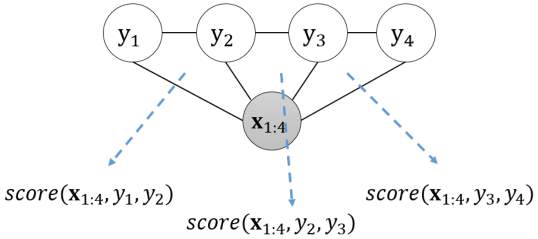
Conditional Random Field

- Each node is a random variable
- We observe some nodes and the rest are unobserved
- The goal: To characterize a probability distribution over the unobserved variables, given the observed



Conditional Random Field

- Each node is a random variable
- We observe some nodes and the rest are unobserved
- The goal: To characterize a probability distribution over the unobserved variables, given the observed


y_1
y_2
y_3
y_4
$\mathbf{x}_{1:4}$
$score(\mathbf{x}_{1:4}, y_1, y_2)$
$score(\mathbf{x}_{1:4}, y_2, y_3)$
$score(\mathbf{x}_{1:4}, y_3, y_4)$

- Each clique is associated with a score

Multi-class Classification as a Special Case

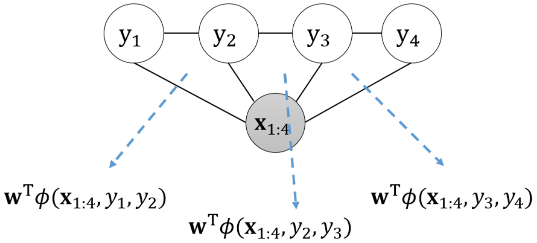
- Rewrite input features and weight vector
  - Define a feature vector for label i being associated to input $\mathbf{x}$
  - Stack all weight vectors into an nK-dimensional vector

$$\phi(\mathbf{x}, i) = \begin{bmatrix} \mathbf{0}_n \\ \vdots \\ \mathbf{x} \\ \vdots \\ \mathbf{0}_n \end{bmatrix}_{nK\times 1} \qquad \mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_K \end{bmatrix}_{nK\times 1}$$

This is called the Kesler construction

Conditional Random Field

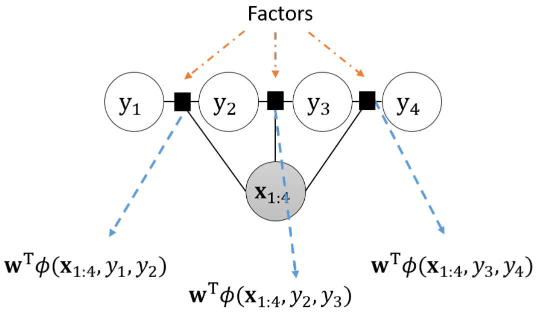
- Each node is a random variable
- We observe some nodes and the rest are unobserved
- The goal: To characterize a probability distribution over the unobserved variables, given the observed


y_1
y_2
y_3
y_4
$\mathbf{x}_{1:4}$
$\mathbf{w}^T\phi(\mathbf{x}_{1:4}, y_1, y_2)$
$\mathbf{w}^T\phi(\mathbf{x}_{1:4}, y_2, y_3)$
$\mathbf{w}^T\phi(\mathbf{x}_{1:4}, y_3, y_4)$

- Each clique is associated with a score
- We can put arbitrary features, as with local conditional models

Conditional Random Field

- Each node is a random variable
- We observe some nodes and the rest are unobserved
- The goal: To characterize a probability distribution over the unobserved variables, given the observed

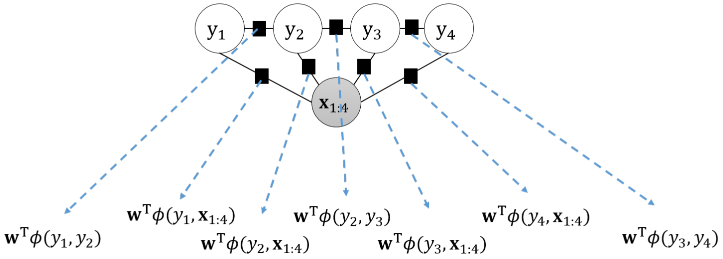


Factors
y_1
y_2
y_3
y_4
$\mathbf{x}_{1:4}$
$\mathbf{w}^T\phi(\mathbf{x}_{1:4}, y_1, y_2)$
$\mathbf{w}^T\phi(\mathbf{x}_{1:4}, y_2, y_3)$
$\mathbf{w}^T\phi(\mathbf{x}_{1:4}, y_3, y_4)$

- Each ~~clique~~ factor is associated with a score
- We can put arbitrary features, as with local conditional models

Conditional Random Field

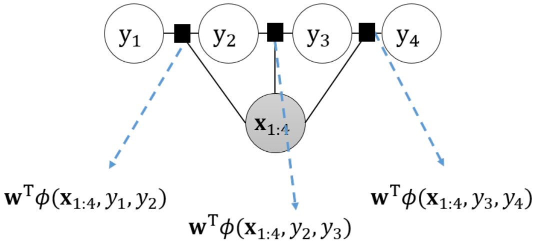
- Each node is a random variable
- We observe some nodes and the rest are unobserved
- The goal: To characterize a probability distribution over the unobserved variables, given the observed



y_1
y_2
y_3
y_4
$\mathbf{x}_{1:4}$
$\mathbf{w}^T\phi(y_1, y_2)$
$\mathbf{w}^T\phi(y_1, \mathbf{x}_{1:4})$
$\mathbf{w}^T\phi(y_2, \mathbf{x}_{1:4})$
$\mathbf{w}^T\phi(y_2, y_3)$
$\mathbf{w}^T\phi(y_3, \mathbf{x}_{1:4})$
$\mathbf{w}^T\phi(y_4, \mathbf{x}_{1:4})$
$\mathbf{w}^T\phi(y_3, y_4)$

- A different factorization: Recall decomposition of structures into parts. Same idea

Conditional Random Field for Sequences



y_1
y_2
y_3
y_4
$\mathbf{x}_{1:4}$
$\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_{1:4}, y_1, y_2)$
$\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_{1:4}, y_2, y_3)$
$\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_{1:4}, y_3, y_4)$

- The conditional probability is

$$P(y_{1:N}|\mathbf{x}_{1:N}) = \frac{1}{Z}\prod_n \exp(\mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_n, y_{n-1}))$$

  where Z is a normalization constant

$$Z = \sum_{y_{1:N}} \prod_n \exp(\mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_n, y_{n-1})$$

CRF: A Different View

- Input: $\mathbf{x}_{1:N}$, Output: $y_{1:N}$, both sequences (for now)
- Define a feature vector for the entire input and output sequence: $\phi(\mathbf{x}_{1:N}, y_{1:N})$
- Define a giant log-linear model, $P(y_{1:N}|\mathbf{x}_{1:N})$ parameterized by $\mathbf{w}$

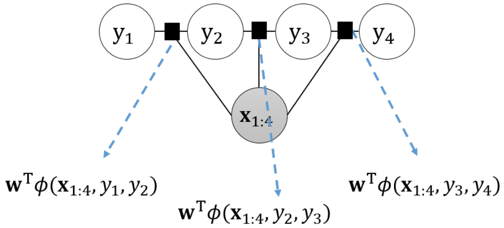
$$\begin{aligned} P(y_{1:N}|\mathbf{x}_{1:N}) &= \frac{1}{Z}\prod_n \exp(\mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_n, y_{n-1})) \\ &\propto \exp\left(\mathbf{w}^\top \sum_n \phi(\mathbf{x}_{1:N}, y_n, y_{n-1})\right) \end{aligned}$$

- Just like any other log-linear model, except
  - Space of y is the set of all possible sequences of the correct length
  - Normalization constant sums over all sequences
  - In an MEMM, probabilities were locally normalized

Global Features

- The feature function decomposes over the sequence

$$\phi(\mathbf{x}_{1:N}, y_{1:N}) = \sum_n \phi(\mathbf{x}_{1:N}, y_n, y_{n-1})$$



y_1
y_2
y_3
y_4
$\mathbf{x}_{1:4}$
$\mathbf{w}^\mathrm{T}\phi(\mathbf{x}_{1:4}, y_1, y_2)$
$\mathbf{w}^\mathrm{T}\phi(\mathbf{x}_{1:4}, y_2, y_3)$
$\mathbf{w}^\mathrm{T}\phi(\mathbf{x}_{1:4}, y_3, y_4)$

CRF Prediction

- Given

$$P(y_{1:N}|\mathbf{x}_{1:N}) = \frac{1}{Z}\exp\left(\mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_{1:N})\right)$$

- Goal: To predict most probable sequence $y_{1:N}$ given an input $\mathbf{x}_{1:N}$

$$\begin{aligned}\arg\max_{y_{1:N}} P(y_{1:N}|\mathbf{x}_{1:N}) &= \arg\max_{y_{1:N}} \exp \mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_{1:N}) \\ &= \arg\max_{y_{1:N}} \mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_{1:N})\end{aligned}$$

- The score decomposes as $\mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_{1:N}) = \mathbf{w}^\top \sum_n \phi(\mathbf{x}_{1:N}, y_n, y_{n-1})$
- So we do prediction via Viterbi (with sum instead of product)

$$\text{score}_1(y_1) = \mathbf{w}^\top \sum_n \phi(\mathbf{x}_{1:N}, y_1)$$

$$\text{score}_n(y_n) = \max_{y_{n-1}}(\mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_n, y_{n-1}) + \text{score}_{n-1}(y_{n-1}))$$

Training a Chain CRF

- Input:
  - Dataset with labeled sequences: $\mathcal{D} = \{\mathbf{x}_{1:N_i}^{(i)}, y_{1:N_i}^{(i)}\}$
  - A definition of the feature function
- How do we train?
  - Maximize the (regularized) log-likelihood

$$\max_{\mathbf{w}} -\frac{\lambda}{2}\mathbf{w}^\top \mathbf{w} + \sum_i \log P(y_{1:N_i}^{(i)} | \mathbf{x}_{1:N_i}^{(i)}, \mathbf{w})$$

Training with Inference

Given
$$\max_{\mathbf{w}} -\frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} + \sum_i \log P(y^{(i)}_{1:N_i}|\mathbf{x}^{(i)}_{1:N_i}, \mathbf{w})$$

and
$$P(y_{1:N}|\mathbf{x}_{1:N}) = \frac{1}{Z}\exp\left(\mathbf{w}^\top\phi(\mathbf{x}_{1:N}, y_{1:N})\right)$$

- Many methods for training
  - Numerical optimization
  - Can use a gradient or hessian based method
- Simple gradient ascent

$$\mathbf{w} \leftarrow \mathbf{w} + \sum_i \left(\phi(\mathbf{x}^{(i)}_{1:N_i}, y^{(i)}_{1:N_i}) - \sum_{\hat{y}_{1:N}} P(\hat{y}_{1:N}|\mathbf{x}^{(i)}_{1:N_i}, \mathbf{w})\phi(\mathbf{x}^{(i)}_{1:N_i}, \hat{y}_{1:N})\right)$$

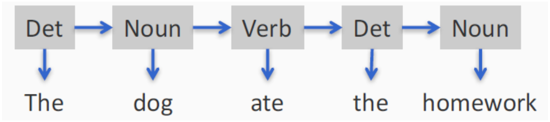
- Red part: Training involves inference!
  - A different kind than what we have seen so far
  - Summing over all sequences is just like Viterbi (with summation instead of maximization)

CRF Summary

- An undirected graphical model
  - Decompose the score over the structure into a collection of factors
  - Each factor assigns a score to assignment of the random variables it is connected to
- Training and prediction
  - Final prediction via $\arg\max_{y_{1:N}} \mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_{1:N})$
  - Train by maximum (regularized) likelihood
- Relation to other models
  - Effectively a linear classifier
  - A generalization of logistic regression to structures
  - An instance of Markov Random Field, with some random variables observed

Overview

HMM is also a Linear Classifier



- Consider $\log P(\mathbf{x}_{1:N}, y_{1:N}) = \sum_n \log P(y_n | y_{n-1}) + P(\mathbf{x}_n | y_n)$

$$\begin{bmatrix} \log P(Det \rightarrow Noun) \\ \log P(Noun \rightarrow Verb) \\ \log P(Verb \rightarrow Det) \\ \log P(The \mid Det) \\ \log P(dog \mid Noun) \\ \log P(ate \mid Verb) \\ \log P(the \mid Det \\ \log P(homework \mid Noun) \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- $\log P(\mathbf{x}_{1:N}, y_{1:N})$ is linear scoring function $\mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_{1:N})$
  - $\mathbf{w}$: parameters of the model
  - $\phi(\mathbf{x}_{1:N}, y_{1:N})$: properties of this output and the input

Towards Structured Perceptron

- HMM is a linear classifier
    - Can we treat it as any linear classifier for training?
    - If so, we could add additional features that are global properties
        - As long as the output can be decomposed for easy inference
- The Viterbi algorithm calculates $\max \mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_{1:N})$
    - Viterbi only cares about scores to structures (not necessarily normalized)
- We could push the learning algorithm to train for un-normalized scores
    - If we need normalization, we could always normalize by exponentiating and dividing by Z
    - That is, the learning algorithm can effectively just focus on the score of $y_{1:N}$ for a particular $\mathbf{x}_{1:N}$
    - Train a discriminative model instead of the generative one!

Structured Perceptron Algorithm

- Given a training set $\mathcal{D} = \{\mathbf{x}_{1:N}^{(i)}, y_{1:N}^{(i)}\}$
- (In practice, good to shuffle data before running)
- Initialize $\mathbf{w} = 0$
- For epoch $= 1, \dots, T$:
- (T is a hyperparameter to the algorithm)
    - For each training example $(\mathbf{x}_{1:N}, y_{1:N}) \in \mathcal{D}$
        - Predict $y'_{1:N} = \arg\max_{y'} \mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y'_{1:N})$
        - (Inference in the training loop)
        - If $y'_{1:N} \neq y_{1:N}$, update $\mathbf{w} \leftarrow \mathbf{w} + \eta(\phi(\mathbf{x}_{1:N}, y_{1:N}) - \phi(\mathbf{x}_{1:N}, y'_{1:N}))$
- Return $\mathbf{w}$

- Prediction: $\arg\max_y = \mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y_{1:N})$

Notes on Structured Perceptron

- Mistake bound for separable data, just like perceptron
- In practice, use averaging for better generalization
  - Initialize $\mathbf{a} = 0$
  - After each step, whether there is an update or not, $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{w}$
    - Note, we still check for mistake using $\mathbf{w}$ not $\mathbf{a}$
  - Return $\mathbf{a}$ at the end instead of $\mathbf{w}$
- Global update
  - One weight vector for entire sequence (not for each position)

CRF vs. Structured Perceptron

- Stochastic gradient descent update for CRF: expectation vs max

$$\mathbf{w} \leftarrow \mathbf{w} + \eta\Big(\phi(\mathbf{x}_{1:N_i}, y_{1:N_i}) - \sum_{\hat{y}_{1:N}} P(\hat{y}_{1:N}|\mathbf{x}_{1:N_i}, \mathbf{w})\phi(\mathbf{x}_{1:N_i}, \hat{y}_{1:N})\Big)$$

$$\mathbf{w} \leftarrow \mathbf{w} + \eta\big(\phi(\mathbf{x}_{1:N_i}, y_{1:N_i}) - \mathbb{E}_{P(\hat{y}_{1:N}|\mathbf{x}_{1:N_i},\mathbf{w})}[\phi(\mathbf{x}_{1:N_i}, \hat{y}_{1:N})]\big)$$

- Structured perceptron

$$\mathbf{w} \leftarrow \mathbf{w} + \eta\big(\phi(\mathbf{x}_{1:N}, y_{1:N}) - \phi(\mathbf{x}_{1:N}, y'_{1:N})\big)$$

  where $y'_{1:N} = \arg\max_{y'} \mathbf{w}^\top \phi(\mathbf{x}_{1:N}, y'_{1:N})$

- Caveat: Adding regularization will change the CRF update, averaging changes the perceptron update

The lay of the land

HMM: A generative model, assigns probabilities to sequences

Structured Perceptron/Structured SVM	**Conditional Random field**
Hidden Markov Models are actually just linear classifiers	Model probabilities via exponential functions. Gives us the log-linear representation
Dont really care whether we are predicting probabilities. We are assigning scores to a full output for a given input (like multiclass)	Log-probabilities for sequences for a given input
Generalize algorithms for linear classifiers. Sophisticated models that can use arbitrary features	Learn by maximizing likelihood. Sophisticated models that can use arbitrary features

Applicable beyond sequences. Eventually, similar objective minimized with different loss functions

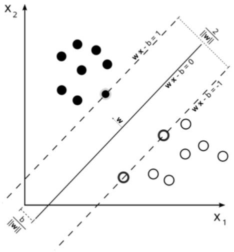
Overview

Recall: Binary SVM

- Functional margin $\gamma^{(i)} = y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)$
  - If $y^{(i)} = 1$, we want $\mathbf{w}^\top \mathbf{x}^{(i)} + b$ to be a large positive number
  - If $y^{(i)} = -1$, we want $\mathbf{w}^\top \mathbf{x}^{(i)} + b$ to be a large negative number
  - If $y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) > 0$, then our prediction on this example is correct
  - If we replace $(\mathbf{w}, b)$ with $(2\mathbf{w}, 2b)$, it does not change the decision
    - So it makes more sense to impose some sort of normalization condition such as that $||\mathbf{w}||_2 = 1$
  - The functional margin of a dataset is defined as the smallest of the function margins of individual exmaples

$$\gamma = \min_i \gamma^{(i)}$$

Recall: Binary SVM (Cont'd)



x_2
$\mathbf{w}\cdot\mathbf{x}-b=1$
$\frac{2}{\|\mathbf{w}\|}$
$\mathbf{w}\cdot\mathbf{x}-b=0$
$\mathbf{w}\cdot\mathbf{x}-b=-1$
$\mathbf{w}$
$\frac{b}{\|\mathbf{w}\|}$
x_1

- Geometric margin
  - The points lie on the decision boundary satisfy the equation $\mathbf{w}^\top \mathbf{x} + b = 0$
  - The points lie on the margin satisfy $\mathbf{w}^\top(\mathbf{x} - \gamma\frac{\mathbf{w}}{||\mathbf{w}||}) + b = 0$
  - Solving for γ yields

$$\gamma = \frac{\mathbf{w}^\top \mathbf{x} + b}{||\mathbf{w}||} = \left(\frac{\mathbf{w}}{||\mathbf{w}||}\right)^\top \mathbf{x} + \frac{b}{||\mathbf{w}||}$$

  - If $||\mathbf{w}|| = 1$, then the functional margin equals the geometric margin
  - The geometric margin is invariant to rescaling of the parameters
    - We replace $\mathbf{w}$ with $2\mathbf{w}$ and b with $2b$, then the geometric margin does not change

Recall: Binary SVM (Cont'd)

- To maximize geometric margin, we can pose the following optimization problem

$$\begin{array}{ll} \max_{\gamma,\mathbf{w},b} & \gamma \\ s.t. & y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq \gamma \\ & ||\mathbf{w}|| = 1 \end{array}$$

- The constraint $||\mathbf{w}|| = 1$ is non-convex. So we convert it as following nicer one

$$\begin{array}{ll} \max_{\gamma,\mathbf{w},b} & \frac{\gamma}{||\mathbf{w}||} \\ s.t. & y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq \gamma \end{array}$$

- Note that we can add an arbitrary scaling constraint on **w** and b without changing anything
- We introduce the scaling constraint that the functional margin of **w** and b with respect to the training set must be 1:

$$\begin{array}{ll} \min_{\gamma,\mathbf{w},b} & \frac{1}{2}||\mathbf{w}||^2 \\ s.t. & y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1 \end{array}$$

where $||\mathbf{w}||^2$ makes it easier (convex) for optimization and remain the same optimization problem

Recall: Binary SVM (Cont'd)

- SVM optimizes

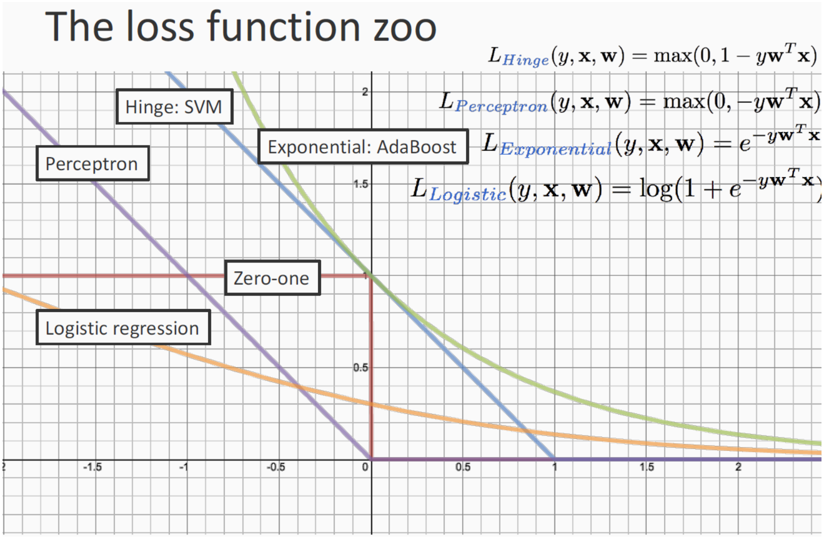
$$\begin{array}{ll} \min_{\gamma,\mathbf{w},b} & \frac{1}{2}||\mathbf{w}||^2 \\ s.t. & y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1 \end{array}$$

- With Lagrange multiplier, a equivalent formulation is

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_i \max(0, 1 - y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b))$$

  where $\max(0, 1 - y(\mathbf{w}^\top \mathbf{x} + b))$ is called hinge loss
- As a comparison, $\max(0, -y(\mathbf{w}^\top \mathbf{x} + b))$ is called perceptron loss

Compare losses


The loss function zoo

$$L_{Hinge}(y,\mathbf{x},\mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$

$$L_{Perceptron}(y,\mathbf{x},\mathbf{w}) = \max(0, -y\mathbf{w}^T\mathbf{x})$$

$$L_{Exponential}(y,\mathbf{x},\mathbf{w}) = e^{-y\mathbf{w}^T\mathbf{x}}$$

$$L_{Logistic}(y,\mathbf{x},\mathbf{w}) = \log(1 + e^{-y\mathbf{w}^T\mathbf{x}})$$

Multiclass SVM in the Separable Case

- For dataset with K classes, our optimization problem is

$$\begin{array}{ll} \min_{\gamma,\mathbf{w}} & \frac{1}{2}\sum_{k}^{K} ||\mathbf{w}_k||^2 \\ s.t. & \mathbf{w}_{y^{(i)}}^\top \mathbf{x}^{(i)} - \mathbf{w}_k^\top \mathbf{x}^{(i)} \geq 1 \;\; \forall i, k \neq y^{(i)} \end{array}$$

  The score for the true label is higher than the score for any other label by 1
- With Kesler construction, we have

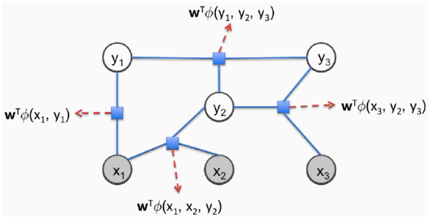
$$\begin{array}{ll} \min_{\gamma,\mathbf{w}} & \frac{1}{2}||\mathbf{w}||^2 \\ s.t. & \mathbf{w}^\top(\phi(\mathbf{x}^{(i)}, y^{(i)}) - \phi(\mathbf{x}^{(i)}, k)) \geq 1 \;\; \forall i, k \neq y^{(i)} \end{array}$$

$$\phi(\mathbf{x}, i) = \begin{bmatrix} \mathbf{0}_n \\ \vdots \\ \mathbf{x} \\ \vdots \\ \mathbf{0}_n \end{bmatrix}_{nK\times 1} \qquad \mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_K \end{bmatrix}_{nK\times 1}$$

Structural SVM: A First Attempt

- Suppose we have some definition of a structure (a factor graph)
  - And feature definitions for each "part" p as $\phi_p(\mathbf{x}, \mathbf{y}_p)$
  - Remember: we can talk about the feature vector for the entire structure
    - Features sum over the parts

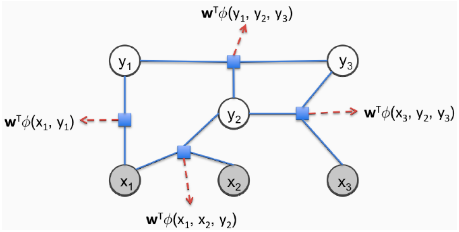
$$\phi(\mathbf{x}, \mathbf{y}) = \sum_{p \in parts(\mathbf{x})} \phi_p(\mathbf{x}, \mathbf{y}_p)$$



$\mathbf{w}^T\phi(y_1, y_2, y_3)$
y_1
y_3
$\mathbf{w}^T\phi(x_1, y_1)$
y_2
$\mathbf{w}^T\phi(x_3, y_2, y_3)$
x_1
x_2
x_3
$\mathbf{w}^T\phi(x_1, x_2, y_2)$

- We also have a dataset $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}$

Structural SVM: A First Attempt



$\mathbf{w}^\top\phi(y_1, y_2, y_3)$
$\mathbf{w}^\top\phi(x_1, y_1)$
$\mathbf{w}^\top\phi(x_3, y_2, y_3)$
$\mathbf{w}^\top\phi(x_1, x_2, y_2)$
y_1
y_2
y_3
x_1
x_2
x_3

What do we want from training (following the multiclass idea)?

- For each example,
  - The annotated structure **y** gets the highest score among all structures
  - The structure **y** gets a score of at least one more than any other

$$\mathbf{w}^\top\phi(\mathbf{x}, \mathbf{y}) \geq \mathbf{w}^\top\phi(\mathbf{x}, \mathbf{y}') + 1 \quad \forall \mathbf{y}' \neq \mathbf{y}$$

Structural SVM: A First Attempt

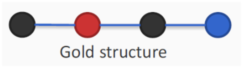
- Goal

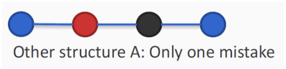
  maximize margin
  s.t. score for gold structure $\geq$ score for other structure $+ 1$
  for every training example

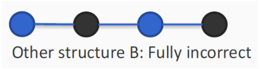
- Corresponding to

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}^\top \mathbf{w} \\ s.t. \quad & \mathbf{w}^\top \phi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \geq \mathbf{w}^\top \phi(\mathbf{x}^{(i)}, \mathbf{y}') + 1 \ \ \forall \mathbf{y}' \neq \mathbf{y}^{(i)} \end{aligned}$$

- Problem:







  - Structure B has is more wrong, but this formulation will be happy if both A ans B are scored one less than gold!

Structural SVM: Second Attempt

- First attempt

$$\begin{array}{ll} \min_{\mathbf{w}} & \frac{1}{2}\mathbf{w}^\top \mathbf{w} \\ s.t. & \mathbf{w}^\top \phi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \geq \mathbf{w}^\top \phi(\mathbf{x}^{(i)}, \mathbf{y}') + 1 \ \ \forall \mathbf{y}' \neq \mathbf{y}^{(i)} \end{array}$$

- Introduce Hamming distance between structures

$$\begin{array}{ll} \min_{\mathbf{w}} & \frac{1}{2}\mathbf{w}^\top \mathbf{w} \\ s.t. & \mathbf{w}^\top \phi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \geq \mathbf{w}^\top \phi(\mathbf{x}^{(i)}, \mathbf{y}') + \Delta(\mathbf{y}', \mathbf{y}^{(i)}) \ \ \forall \mathbf{y}' \neq \mathbf{y}^{(i)} \end{array}$$

  where $\Delta(\mathbf{y}', \mathbf{y}^{(i)})$ is defined as Hamming distance between structures
- The Hamming distance between:
  - "karolin" and "kathrin" is 3.
  - "karolin" and "kerstin" is 3.
  - 1011101 and 1001001 is 2.
  - 2173896 and 2233796 is 3.
- Intuition
  - Structures that are more different from the true structure should be scored much lower

Structural SVM: Third Attempt

- Problem? What if the data is not separable?
  - What if these constraints are not satisfied for any **w** for a given dataset?
- Slack variable for each example ξ_i, must be positive

$$\begin{array}{ll} \min_{\mathbf{w},\xi_i} & \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_i \xi_i \\ s.t. & \mathbf{w}^\top\phi(\mathbf{x}^{(i)},\mathbf{y}^{(i)}) \geq \mathbf{w}^\top\phi(\mathbf{x}^{(i)},\mathbf{y}') + \Delta(\mathbf{y}',\mathbf{y}^{(i)}) - \xi_i \;\; \forall \mathbf{y}' \neq \mathbf{y}^{(i)} \\ & \xi_i \geq 0 \end{array}$$

- Slack variables allow some examples to be misclassified
- Minimizing the slack forces this to happen as few times as possible
- An equivalent formulation

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_i \max_{\mathbf{y}'} \left(\mathbf{w}^\top\phi(\mathbf{x}^{(i)},\mathbf{y}') + \Delta(\mathbf{y}',\mathbf{y}^{(i)}) - \mathbf{w}^\top\phi(\mathbf{x}^{(i)},\mathbf{y}^{(i)})\right)$$

Comments

- Other slightly different formulations exist
  - Generally same principle
- Multiclass is a special case of structure
  - Structural SVM strictly generalizes multiclass SVM
- Can be seen as minimizing structured version of hinge loss
- Learning as optimization

Broader Picture: Learning as Loss Minimization

- Collect some annotated data. More is generally better
- Pick a hypothesis class (also called model)
  - Decide how the score decomposes over the parts of the output
- Choose a loss function
  - Decide on how to penalize incorrect decisions
- Learning = minimize empirical risk + regularizer
  - Typically an optimization procedure needed here

Structured Classifiers: Different Learning Objectives

- Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_i \max_{\mathbf{y}'}\left(\mathbf{w}^\top\phi(\mathbf{x}^{(i)},\mathbf{y}') + \Delta(\mathbf{y}',\mathbf{y}^{(i)}) - \mathbf{w}^\top\phi(\mathbf{x}^{(i)},\mathbf{y}^{(i)})\right)$$

- CRF: $\max_{\mathbf{w}} -\frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} + \sum_i \log P(y^{(i)}_{1:N_i}|\mathbf{x}^{(i)}_{1:N_i},\mathbf{w})$

$$\max_{\mathbf{w}} -\frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} + \sum_i\left(\mathbf{w}^\top\phi(\mathbf{x}^{(i)}_{1:N},y^{(i)}_{1:N}) - \log\sum_{y'_{1:N}}\left(\mathbf{w}^\top\phi(\mathbf{x}^{(i)}_{1:N},y'_{1:N})\right)\right)$$

$$\min_{\mathbf{w}} \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} + \sum_i\left(\log\sum_{y'_{1:N}}\left(\mathbf{w}^\top\phi(\mathbf{x}^{(i)}_{1:N},y'_{1:N})\right) - \mathbf{w}^\top\phi(\mathbf{x}^{(i)}_{1:N},y^{(i)}_{1:N})\right)$$

- Structured Perceptron

$$\min_{\mathbf{w}} C\sum_i \max_{\mathbf{y}'}\left(\mathbf{w}^\top\phi(\mathbf{x}^{(i)},\mathbf{y}') - \mathbf{w}^\top\phi(\mathbf{x}^{(i)},\mathbf{y}^{(i)})\right)$$

Further Reading

- Ng (2017). CS229 Lecture notes: Support Vector Machines. `http://cs229.stanford.edu/notes/cs229-notes3.pdf`
- Punyakanok and Roth (2000). The Use of Classifiers in Sequential Inference.
- McCallum et al. (2000). Maximum Entropy Markov Models for Information Extraction and Segmentation.
- Lafferty et al. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.
- Taskar et al. (2003). Max-Margin Markov Networks.
- Collins (2011). Course notes for COMS w4705: Log-linear models, MEMMs, and CRFs, 2011. `http://www.cs.columbia.edu/~mcollins/crf.pdf`
- Smith (2004). Log-linear models, 2004. `https://homes.cs.washington.edu/~nasmith/papers/smith.tut04.pdf`

References

Collins, M. (2011). Log-linear models, MEMMs, and CRFs. Technical report, Columbia University.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

McCallum, A., Freitag, D., and Pereira, F. C. N. (2000). Maximum entropy markov models for information extraction and segmentation. In *ICML*, pages 591–598.

Ng, A. (2017). Cs229 lecture notes: Support vector machines. Technical report, Stanford University.

Punyakanok, V. and Roth, D. (2000). The use of classifiers in sequential inference. In *NIPS*.

Smith, N. A. (2004). Log-linear models. Technical report, University of Washington.

Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin markov networks. In *NIPS*, pages 25–32.