Structured Prediction

Mark Dredze

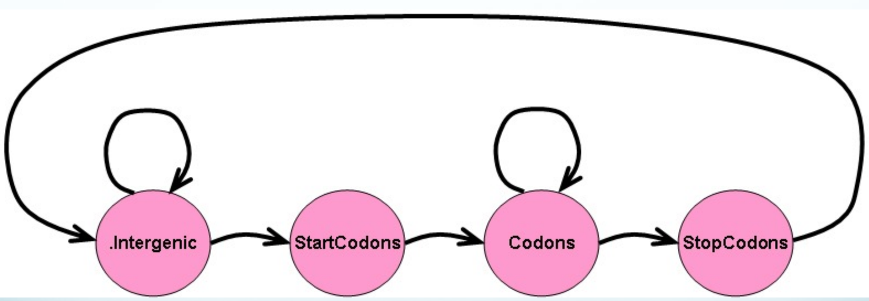
Machine Learning
CS 601.475

Binary Prediction

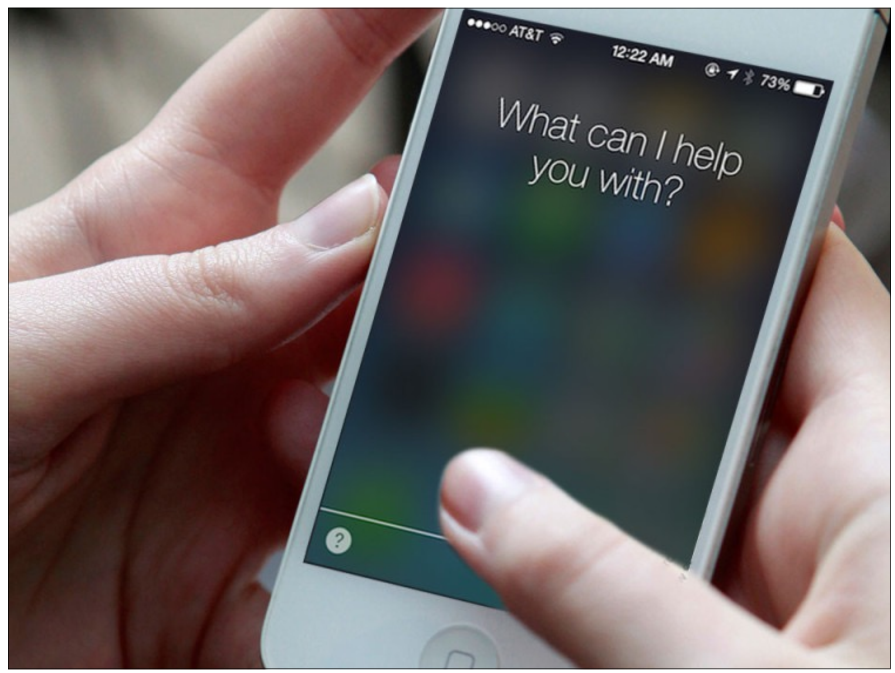
- So far we've focused on binary prediction
- Machine learning can do much more than that!

- Multi-class prediction
- Complex structured outputs

Sequence Prediction

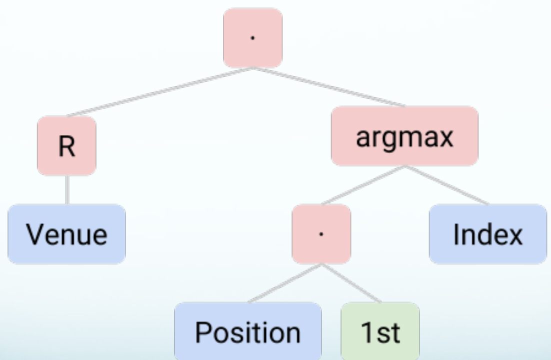


http://cs.wellesley.edu/~cs313/projects/project8/images/OneStrand.jpg



Trees

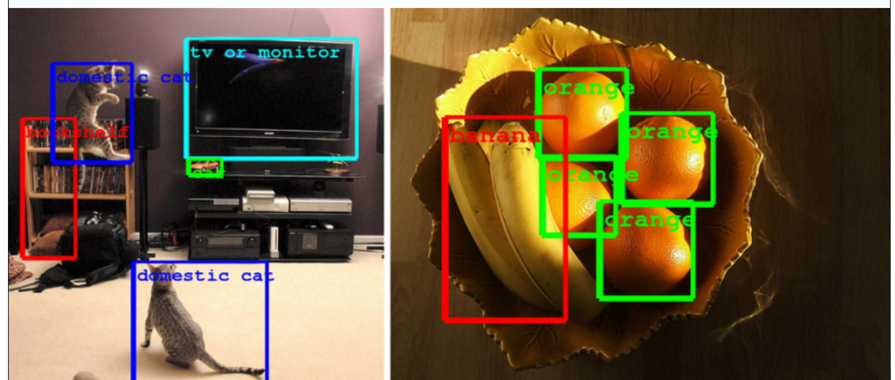
In what city did Piotr's last 1st place finish occur?



R[Venue].argmax(Position.1st, Index)

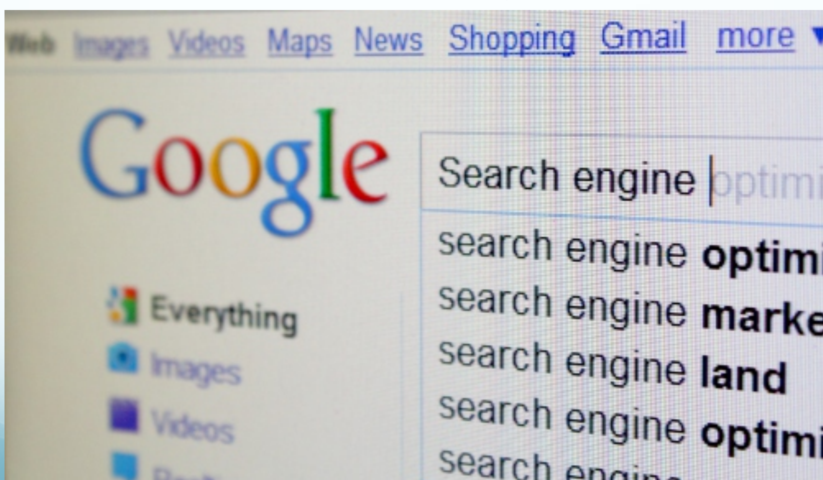
http://nlp.stanford.edu/software/sempre/wikitable/images/piotr.png/

Object Recognition



https://3c1703fe8d.site.internapcdn.net/newman/gfx/news/hires/2014/googleteamri.png

Ranking



What is Structured Prediction?

- Input: x
  - Typically a structured input
  - Maintain structure of input in x
    - Do not flatten into list of features in an instance
- Output: y
  - y is now from a large set of possible outputs
  - Outputs defined based on input
    - Often exponential in size of input

Previous Approaches

- Naturally multi-class algorithm
  - Neural networks
  - Decision Trees
- Reduction to binary
  - e.g. one classifier per class
- These methods don't work when
  - Exponential number of output
  - Outputs defined based on input

Structured Prediction Challenges

- Scoring
  - How do we assign a score/probability to a possible output structure
- Search/Inference
  - Find the best scoring output structure
  - How do we search through an exponential number of options?

Outline

- Graphical models for structured prediction
  - Sequences: HMMs and CRFs
- Score based linear models
  - Perceptron, SVM
- Deep networks

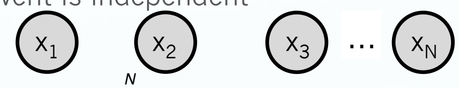
Graphical Models: Sequence Models

Sequential Events

- Many events happen in sequence
  - Weather on consecutive days
  - Words in a written sentence
  - Spoken sounds by a person
  - Movements in the stock market
  - DNA base pairs
- We want to model these sequences with a graphical model

Sequential Models

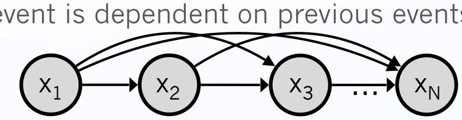
- Simple approach
  - Each event is independent



  - $p(x_1, x_2 \ldots x_N) = \prod_{n=1}^{N} p(x_n)$
  - Simple, but not very helpful

Sequential Models

- Complex approach
  - Each event is dependent on previous events



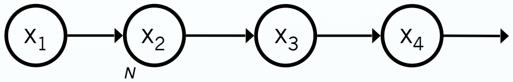
  - $p(x_1, x_2 \ldots x_N) = \prod_{n=1}^{N} p(x_n \mid x_1 \ldots x_{n-1})$
  - Captures dependencies, but way too complex

Markov Assumption

- The current state depends on a fixed number of previous states
  - The weather today depends on the past three days, but NOT two weeks ago
  - The next word in the sentence depends on the past three words, but nothing before
- Pro: makes for simple models
- Con: doesn't capture full history

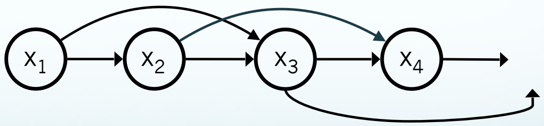
Markov Chains

- First order Markov chain



$$p(x_1, x_2 \dots x_N) = \prod_{n=1}^{N} p(x_n \mid x_{n-1})$$

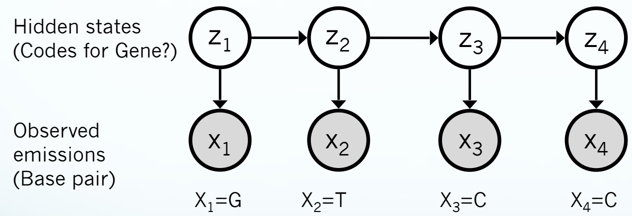
- Second order Markov chain



$$p(x_1, x_2 \dots x_N) = \prod_{n=1}^{N} p(x_n \mid x_{n-1}, x_{n-2})$$

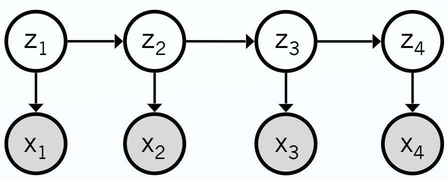
DNA Example

- Given a sequence of base pairs, find regions that encode for Genes



- What model is this?
- Hidden Markov model

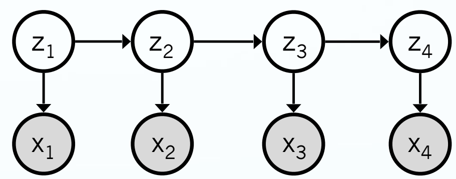
Markov Blankets



- The Markov blanket for z_n contains z_{n-1}, z_{n+1} and x_n
- The Markov blanket for x_n contains z_n
- Nodes are dependent on a small number of neighbors

Conditional Probability Tables

	P(z_n=1)	**...**
z_{n-1}=1		
⋮		



	P(x_n=1)	**...**
z_n=1		
⋮		

- Tables are the same for each node

Joint Probability of HMM

- The joint probability of an HMM

$$p(\mathbf{X},\mathbf{Z}\mid\theta) = p(z_1\mid\pi)\left[\prod_{n=2}^{N} p(z_n\mid z_{n-1},\mathbf{A})\right]\prod_{m=1}^{N} p(x_m\mid z_m,\phi)$$

  - **A**- transition probabilities (matrix)
    - A_{ij} is the probability of moving from state i to j
  - π- vector with starting probabilities
  - φ – emission probabilities (matrix)
    - φ_{ij} is the probability of state i and emitting observation j

Unsupervised Training

- How do we train a probabilistic model?
  - Maximum likelihood!

$$\max_\theta\ p(\mathbf{X},\mathbf{Z}\mid\theta)$$

  - Problem: we don't know **Z**
- Solution: EM
  - Step 1: Write the complete data likelihood

$$p(\mathbf{X}\mid\theta) = \sum_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z}\mid\theta)$$

$$p(\mathbf{X},\mathbf{Z}\mid\theta) = p(z_1\mid\pi)\left[\prod_{n=2}^{N} p(z_n\mid z_{n-1},\mathbf{A})\right]\prod_{m=1}^{N} p(x_m\mid z_m,\phi)$$

EM for HMMs

- E-Step
  - Find the expected values for the hidden variables **Z** given the model parameters
    - The most likely **Z** given **X** and current model parameters
- M-Step
  - Pretend to observe the values for **Z**
  - Update model parameters **A**, π, ϕ to maximize complete data likelihood

EM for HMMs

- E-Step
  - Given Q function, evaluate probabilities for **Z**

$$Q(\theta,\theta^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z}\mid\mathbf{X},\theta^{old})\log p(\mathbf{X},\mathbf{Z}\mid\theta)$$

  - For the HMM

$$Q(\theta,\theta^{old}) = \sum_{k=1}^{K}\gamma(z_{1k})\log\pi_k + \sum_{n=2}^{N}\sum_{j=1}^{K}\sum_{k=1}^{K}\xi(z_{n-1,j},z_{nk})\log A_{jk} + \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\log p(x_n\mid\phi_k)$$

$$\gamma(z_n) = p(z_n\mid X,\theta^{old})$$

π,Φ and **A** are model parameters

$$\xi(z_{n-1},z_n) = p(z_{n-1},z_n\mid X,\theta^{old})$$

EM for HMMs

- How can we get these values?

  $\gamma(z_n) = p(z_n \mid X, \theta^{old})$ $\xi(z_{n-1}, z_n) = p(z_{n-1}, z_n \mid X, \theta^{old})$

  - What is the probability of being in state z_n?
  - What is the probability of being in state z_n and z_{n+1}?
- Inference
  - Sum Product Algorithm
    - Forward Backward in HMMs

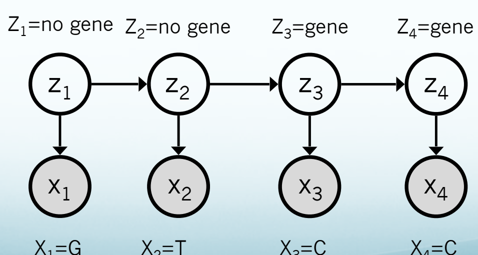
EM for HMMs

- M Step
  - Maximize Q function with respect to π,Φ and **A**
- Assuming values for **Z**, we get

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})} \qquad A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})} \qquad \phi_{ik} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) x_{ni}}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

Prediction

- Given a new sequence **X**, find the most likely set of states to have generated **X**
  - Find the sequence **Z** with the maximum probability given **X**



Prediction

- How do we find the most likely state?

  $\arg\max_{\mathbf{z}} p(\mathbf{z} \mid X, \theta)$

- Inference
  - Max Product Algorithm
  - Viterbi Decoding

Supervised Training

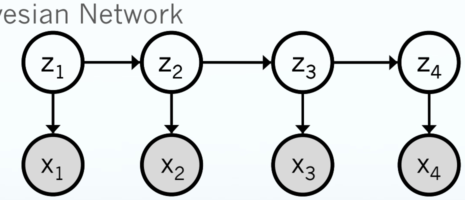
- We actually observe **Z**
  - Just compute M step a single time
  - Very fast and easy
- What if we observe only some **Z**
  - Case 1: only some examples are labeled with **Z**
  - Case 2: each example has only some labels for **Z**
- Semi-supervised Learning
  - Use EM algorithm but fix **Z** when known

Notes on HMMs

- An HMM can have continuous or discrete emissions
  - Discrete- base pair, word in sentence
  - Continuous- stock price, frequency of a sound
- An HMM has discrete hidden states
- A Linear Dynamical System has continuous hidden states

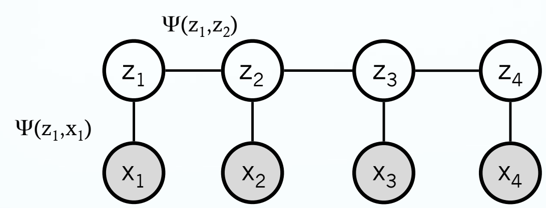
Sequence Models

- An HMM is a directed graphical model
  - Bayesian Network



  - What happens if we have an undirected graphical model?
    - Markov Random Field
    - Conditional Random Fields

Conditional Random Fields



- Replace conditional probabilities with potential functions
- The joint probability is a product of potential functions

$$p(\mathbf{X},\mathbf{Z}) = \frac{1}{Z}\prod_A \psi_A(\mathbf{x}_A)$$

Joint Probability

- The joint probability of the HMM can be written using potential functions
  - HMM

$$p(\mathbf{X},\mathbf{Z} \mid \theta) = p(z_1 \mid \pi)\left[\prod_{n=2}^{N} p(z_n \mid z_{n-1},\mathbf{A})\right]\prod_{m=1}^{N} p(x_m \mid z_m,\phi)$$

  - CRF

$$p(\mathbf{X},\mathbf{Z} \mid \theta) = \frac{1}{Z}\psi(z_1)\left[\prod_{n=2}^{N} \psi(z_n \mid z_{n-1})\right]\prod_{m=1}^{N} \psi(x_m \mid z_m)$$

$$p(\mathbf{X},\mathbf{Z} \mid \theta) = \frac{1}{Z}\prod_{A} \psi_A(\mathbf{x}_A,\mathbf{z}_A) \qquad \mathbf{Z} = \sum_{X,Z}\prod_{A} \psi_A(\mathbf{x}_A,\mathbf{z}_A)$$

Learning a CRF

- Given some data, we want to learn a CRF
- How should we learn the model?
  - Maximum likelihood!
    - Maximize the probability of the data given the model
- Questions
  - What are the parameters of our model?
    - The potential functions
  - What is the objective?
  - How do we compute model probabilities efficiently?

Model Parameters

- HMM: model parameters are CPTs
  - CRF: CPTs replaced with potential functions
- Parameterize the potential functions
  - Learn the parameters

$$\psi(\mathbf{x},\mathbf{z}) = \exp\left\{\sum_{k} \theta_k f_k(\mathbf{x},\mathbf{z})\right\}$$

  - Parameters θ determine value of potential function
  - f_k is a feature function
    - f_k = 1 if x is the base pair "G"
  - Notice: linear combination of features (linear model!)

Objective

- Let's maximize the likelihood of the data
  - For a single example

$$p(\mathbf{X},\mathbf{Z} \mid \theta) = \frac{1}{Z}\prod_{A} \psi_A(\mathbf{x}_A,\mathbf{z}_A)$$

  - What about Z?

$$\mathbf{Z} = \sum_{X,Z}\prod_{A} \psi_A(\mathbf{x}_A,\mathbf{z}_A)$$

  - Sum over all X!
    - All possible sequences of base pairs
  - It's too hard to learn X

Discriminative Training

- Solution: don't learn p(X)!
- Maximize the conditional likelihood of the data
  - For a single example

$$p(\mathbf{Z} \mid \mathbf{X}, \theta) = \frac{1}{Z} \prod_{A} \psi_A(\mathbf{x}_A, \mathbf{z}_A)$$

$$\mathbf{Z} = \sum_{Z} \prod_{A} \psi_A(\mathbf{x}_A, \mathbf{z}_A)$$

- CRF Conditional log likelihood of all examples

$$\log p(z \mid x) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k f_k(z_{it}, z_{it-1}, x_{it}) - \sum_{i=1}^{N} \log Z(x_i)$$

Problems with Objective

- Recall for logistic regression (discriminative training) maximum likelihood over-fit the data
- Solution: regularization

$$\log p(z \mid x) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k f_k(z_{it}, z_{it-1}, x_{it}) - \sum_{i=1}^{N} \log Z(x_i) - \sum_{k=1}^{K} \frac{\theta_k^2}{2\sigma^2}$$

  - Gaussian prior ($\mu=0$, $\Sigma=\sigma^2 I$)

Training a CRF

- The conditional log likelihood is convex
  - Take the derivative and solve for θ

$$\frac{\partial L}{\partial \theta_k} = \sum_{i=1}^{N} \sum_{t=11}^{T} f_k(z_{it}, z_{it-1}, x_{it}) - \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{z,z'} f_k(z, z', x_{it}) p(z, z' \mid x_{it}) - \sum_{k=1}^{K} \frac{\theta_k}{\sigma^2}$$

  - The derivative is 0 when
    - The last term (regularizer) is 0
    - The first term and the second term cancel each other
      - First term: the expected value for f_k under the empirical distribution (from the data)
      - Second term: expectation for f_k given model distribution

Computing Probabilities

- What do we need to compute the values in the derivative?
  - Marginal probability of $p(z, z' \mid x_{it})$
  - The normalization constant Z
    - Total score for all possible labelings of the sequence
    - Sum Product Algorithm (Forward-Backward)
- Prediction
  - Sequence of states with max probability
  - Prediction
    - How do we find the highest probability sequence?
      - Max Product Algorithm (Viterbi decoding)

CRF Summary

- CRFs are
  - Markov Random Fields
  - The MRF equivalent of a supervised HMM
  - Discriminatively trained using conditional log likelihood
  - Linear models (recall linear potential functions)
- CRF training contains versions of
  - Sum Product Algorithm
  - Max Product Algorithm
  - Convex optimization

Why CRFs?

- CRF training is much harder than HMM
  - Computing gradients, optimization vs. counting
  - 11 labels, 200k tokens: 2 hours / 45 labels, 1m tokens: 1 week
- Why bother?
  - HMMs require
    - Assumptions of causation / generative story
    - Independence assumptions for observations
  - These aren't problems for CRFs!
    - Can allow arbitrary dependencies
    - Transition can depend on x and z
    - Can condition on the whole sequence x
    - Recall:
      - Generative models limit the features
      - Discriminative models can have any types of features

HMMs and CRFs

- Generative/Discriminative pairs
  - A generative and discriminative parametric model family that can represent the same set of conditional probability distributions
  - Naïve Bayes/Logistic Regression
  - HMM/CRF
- HMM is a Naïve Bayes classifier at each node
- CRF is a Logistic Regression classifier at each node

Score Based Methods

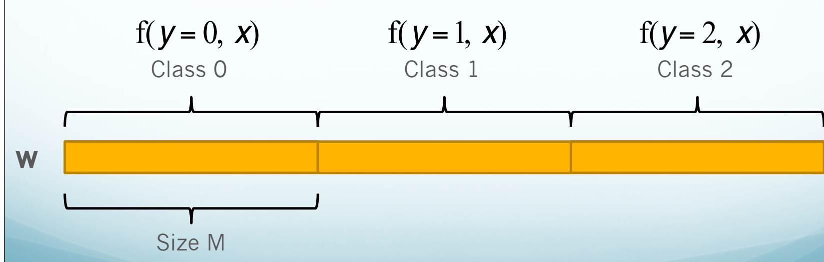
- We have several methods that produce a score for a label
  - Perceptron, SVM
  - Generalized linear models
  - Not probabilistic graphical models
- Can we use these for structured prediction?

Multi-class Encodings

- Encode multiple labels into a single weight vector
- Update the entire vector at once in a single model

1 of K Encoding

- Encode all K labels in a single linear classifier
  - Lengthen the weight vector to M*K
  - A segment of length M learns weights for class k



1 of K Encoding for Perceptron

- Learning:
  - Given example x, get score for each possible label
  - If highest scoring label != correct label
    - Update
      - Highest scoring incorrect label

$$w_i' = w_i - f_i(\hat{y}, x)$$

      - Correct label

$$w_i' = w_i + f_i(y, x)$$

  - This takes a gradient step
    - Increases the score of the correct label *and* decreases the score of the incorrect label

1 of K Encoding for Perceptron

- Prediction:
  - Given example x
  - For each label y:
    - Generate each feature $f_i(y,x)$
      - This returns a feature vector, where only the parameters corresponding to y are non-zero
    - Assign score = w•x
  - Return the label with the highest score

Beyond Multiclass

- We used a 1 of K encoding to extend Perceptron to multiclass
- What about for general structured problems?
  - Sequence
  - Ranking
  - Trees
- Can we learn this using a Perceptron style algorithm?
  - Yes!

Generalized Perceptron

- On each round
  - Receive example x
  - Predict $\hat{y} = \text{sign}(w \cdot x) \longrightarrow \hat{y} = \arg\max_{y \in L} w \cdot f(x, y)$
  - Receive correct label $y \in \{+1, -1\} \longrightarrow y \in L$
  - Suffer loss $\ell_{0/1}(y, \hat{y}) \longrightarrow \ell(y, \hat{y})$
  - Update w $w^{j+1} = w^j + y_i x_i \longrightarrow w^{j+1} = w^j + \text{f}(y_i, x_i) - \text{f}(\hat{y}, x_i)$

Feature Functions

- How do we get this encoding?
- Feature functions!
  - Define a function $f_i(y,x)$
  - Returns the value of the ith feature based on the label and instance
    - Ex. Does this document contain the word "sports" AND is y=="travel"
    - Create a duplicate of each feature for each label to obtain a unique position in w

Structured Perceptron

- On each round
  - Receive example x
  - Predict $\hat{y} = \arg\max_{y \in L} w \cdot f(x, y)$ Need efficient procedure since L may be exponential
  - Receive correct label $y \in L$
  - Suffer loss $\ell(y, \hat{y})$ Loss sensitive to comparing structured objects
  - Update w $w^{j+1} = w^j + \text{f}(y_i, x_i) - \text{f}(\hat{y}, x_i)$

General representations for each prediction

Example: Sequence Labeling

- Label a sequence of words with part of speech tags

John	saw	Mary	with	the	telescope
noun	verb	noun	preposition	article	noun

- We want to assign all tags for sentence at once

Requirements

- Procedure for finding best sequence using w
  - Yes! Viterbi decoding
    - We'll cover this in graphical models
- Loss sensitive to sequences
  - Yes! Hamming distance
    - How many labels disagree
- General representation
  - Features are based on sentence and label sequence

General Perceptron

- Perceptron updates can be used in any decision making process
- Requirements
  - Find best decision
  - Knowing incorrect decision
  - Featurize decisions

Other Types of Output

- Hierarchical Classification
  - Labels are organized in hierarchy
    - Mistaking "football" for "baseball" better than "football" for "NYSE"
- Sequences
  - Each label depends on neighboring labels
    - Part of speech
- Graphs
  - Output a graph or a tree given some input
    - Syntactic parse trees

One More Trick

- You can represent a structured Perceptron using a binary Perceptron
- Notice that:
  - Structured form $w^{i+1} = w^i + \mathrm{f}(y_i, x_i) - \mathrm{f}(\hat{y}, x_i)$

$$\mathrm{x}' = \mathrm{f}(y_i, x_i) - \mathrm{f}(\hat{y}, x_i)$$
$$y' = 1$$

  - Binary form $w^{i+1} = w^i + y' x'$
  - Prediction: Find highest scoring label according to w
    - It will be greater than all other labels