Feature-based Discriminative Models

More Sequence Models

Yoav Goldberg

Bar Ilan University

Reminder

PP-attachment

He saw a mouse with a telescope.

saw, mouse, with, telescope $\rightarrow$ V

POS-tagging

Holly/NNP came/VBD from/IN Miami/NNP ,/, F.L.A/NNP ,/,
hitch-hiked/VBD her/PRP way/NN across/IN the/DT USA/NNP

Both were solved with a probabilistic model and MLE estimates, based on counting and dividing.

Today

Discriminative Training

Sentence-boundary-detection revisited

Georg "Mr. George" Svendsen (19 March 1894 – 1966) was a Norwegian journalist and crime novelist.

He was born in Eidanger, and started his journalistic career in Bratsberg-Demokraten before moving on to Demokraten where he was a subeditor. In 1921 he was hired in Fremtidenand replaced in Demokraten by Evald O. Solbakken. In 1931 he was hired in Arbeiderbladet. Under the pen name "Mr. George" he became known for his humorous articles in the newspaper. At his death he was also called "the last of the three great criminal and police reporters in Oslo", together with Fridtjof Knutsen and Axel Kielland. He was also known for practising journalism as a trade in itself, and not as a part of a political career. He retired in 1964, and died in 1966.

He released the criminal novels Mannen med ljåen (1942), Ridderne av øksen (1945) and Den hvite streken (1946), and translated the book S.S. Murder by Quentin Patrick as Mord ombord in 1945. He released several historical books: Rørleggernes Fagforenings historie gjennem 50 år (1934), Telefonmontørenes Forening Oslo, gjennem 50 år (1939), Norsk nærings- og nydelsesmiddelarbeiderforbund: 25-års beretning (1948), De tause vitner: av rettskjemiker Ch. Bruffs memoarer (1949, with Fridtjof Knudsen) and Elektriske montørers fagforening gjennom 50 år (1949).

Sentence-boundary-detection revisited

- P(boundary | subeditor . In)
- P(boundary | O . Solbakken)
- P(boundary | Solbakken . In)
- P(boundary | Arbeiderbladet . Under)
- P(boundary | Mr . George)
- P(boundary | 1945 . He)

Sentence-boundary-detection revisited

- P(boundary | L=subeditor . R=In)
- P(boundary | L=O . R=Solbakken)
- P(boundary | L=Solbakken . R=In)
- P(boundary | L=Arbeiderbladet . R=Under)
- P(boundary | L=Mr . R=George)
- P(boundary | L=1945 . R=He)

Sentence-boundary-detection revisited

- P(boundary | L=subeditor . R=In)
- P(boundary | L=O . R=Solbakken)
- P(boundary | L=Solbakken . R=In)
- P(boundary | L=Arbeiderbladet . R=Under)
- P(boundary | L=Mr . R=George)
- P(boundary | L=1945 . R=He)

Useful indicators

- Identity of L
- Identity of R
- Length of L
- Is R capitalized
- ...

Sentence-boundary-detection revisited

- P(boundary | L=subeditor . R=In)
- P(boundary | L=O . R=Solbakken)
- P(boundary | L=Solbakken . R=In)
- P(boundary | L=Arbeiderbladet . R=Under)
- P(boundary | L=Mr . R=George)
- P(boundary | L=1945 . R=He)

Useful indicators

- **Identity of L**
- Identity of R
- **Length of L**
- Is R capitalized
- ...

Highly correlated.
Do we need both?
How do we integrate information?

Sentence-boundary-detection revisited

$$P(\text{boundary}|\text{L=subeditor, R=In, len(L)=9, isCap(R)=True, isCap(L)=False, }\dots)$$

- How do we compute the probability model?
- Do we really need a probability model here?

Probabilities $\rightarrow$ Scores

- Instead of $P(y|x)$, compute $score(x, y)$
- Return $\arg\max_y score(x, y)$
  - Or, if we just have two classes, return class 1 iff $score(x) > 0$
- But how do we compute the score?

Feature Representation

Each indicator will be a feature.

Feature

- A **feature** is a function $\phi_i(x)$ looking at a particular property of x and returning a value.
  - Generally, the value can be any number, $\phi_i(x) \in \mathbb{R}$
  - Often, the value is binary, $\phi_i(x) \in \{0, 1\}$.

Feature Extractor / Feature Vector

- A **feature extractor** is a collection of features $\phi = \phi_1, \ldots, \phi_m$
- A **feature vector** is the result of ϕ applied on a particular x
  - $\phi(x) = \phi_1(x), \ldots, \phi_m(x)$
  - For binary features: $\phi(x) \in \{0, 1\}^m$

Feature Representation

Feature Examples

$$\phi_1(L.R) = \begin{cases} 1 & \text{if L='subeditor'} \\ 0 & otherwise \end{cases}$$

$$\phi_2(L.R) = \begin{cases} 1 & \text{if L='O'} \\ 0 & otherwise \end{cases}$$

$$\phi_{57}(L.R) = \begin{cases} 1 & \text{if R='George'} \\ 0 & otherwise \end{cases}$$

$$\phi_{1039}(L.R) = \begin{cases} 1 & \text{if len(L)=1} \\ 0 & otherwise \end{cases}$$

$$\phi_{1040}(L.R) = \begin{cases} 1 & \text{if len(L)=2} \\ 0 & otherwise \end{cases}$$

$$\phi_{1045}(L.R) = \begin{cases} 1 & \text{if len(L)>4} \\ 0 & otherwise \end{cases}$$

$$\phi_{1092}(L.R) = \begin{cases} 1 & \text{if isCap(R)} \\ 0 & otherwise \end{cases}$$

$$\phi_{1093}(L.R) = \begin{cases} 1 & \text{if isNumber(L)} \\ 0 & otherwise \end{cases}$$

$$\phi_{2045}(L.R) = \begin{cases} 1 & \text{if L='US' and isCap(R)} \\ 0 & otherwise \end{cases}$$

$$\phi_{3066}(L.R) = \begin{cases} 1 & \text{if len(L)=1 and isCap(R)} \\ 0 & otherwise \end{cases}$$

$$\phi_{5032}(L.R) = \begin{cases} 1 & \text{if L='Ca' and R='The'} \\ 0 & otherwise \end{cases}$$

$$\phi_{5033}(L.R) = \begin{cases} 1 & \text{if L='Ca' and R='snow'} \\ 0 & otherwise \end{cases}$$

Feature Representation

Feature Vector

$\phi(\text{L=subeditor . R=In}) = 1, 0, 0, 0, 0, \ldots, 1, 0, 0, 0, 1 \ldots$
$\phi(\text{L=Mr . R=George}) = 0, 0, 0, 1, 0, \ldots, 0, 0, 1, 0, 1 \ldots$

- Each example is mapped to a very long vector of 0 and 1.
- Most values are 0.

Sparse Representation

We don't need to write the zeros:

$\phi(\text{L=subeditor . R=In}) = \text{1:1 1045:1 1092:1 } \ldots$

Machine Learning

We had a set of example:

- subeditor . In $\rightarrow$ boundary
- O . Solbakken $\rightarrow$ no-boundary
- Mr . George $\rightarrow$ no-boundary
- ...

We can map them to a list of vectors

- ϕ(L=subeditor . R=In) = 1:1 1045:1 1092:1 ...
- ϕ(L=O . R=Solbakken) = 2:1 1039:1 1092:1 ...
- ϕ(L=Mr . R=George) = 57:1 1040:1 1092:1 ...
- ...

And a list of answers:

- $+1, -1, -1, \ldots$

Feature Representation

Why did we do this?

- We mapped each example x to a vector of numbers $\phi(x)$.
- We mapped each answer y to a number.
- Instead of $P(y|x)$ we now have $P(y|\phi(x)) \;\; y \in \{-1, +1\}$.
  - x was a sequence of words.
  - $\phi(x)$ is a vector of numbers.
- The field of **Machine Learning** is very good at learning to classify vectors.

Machine Learning

Dataset

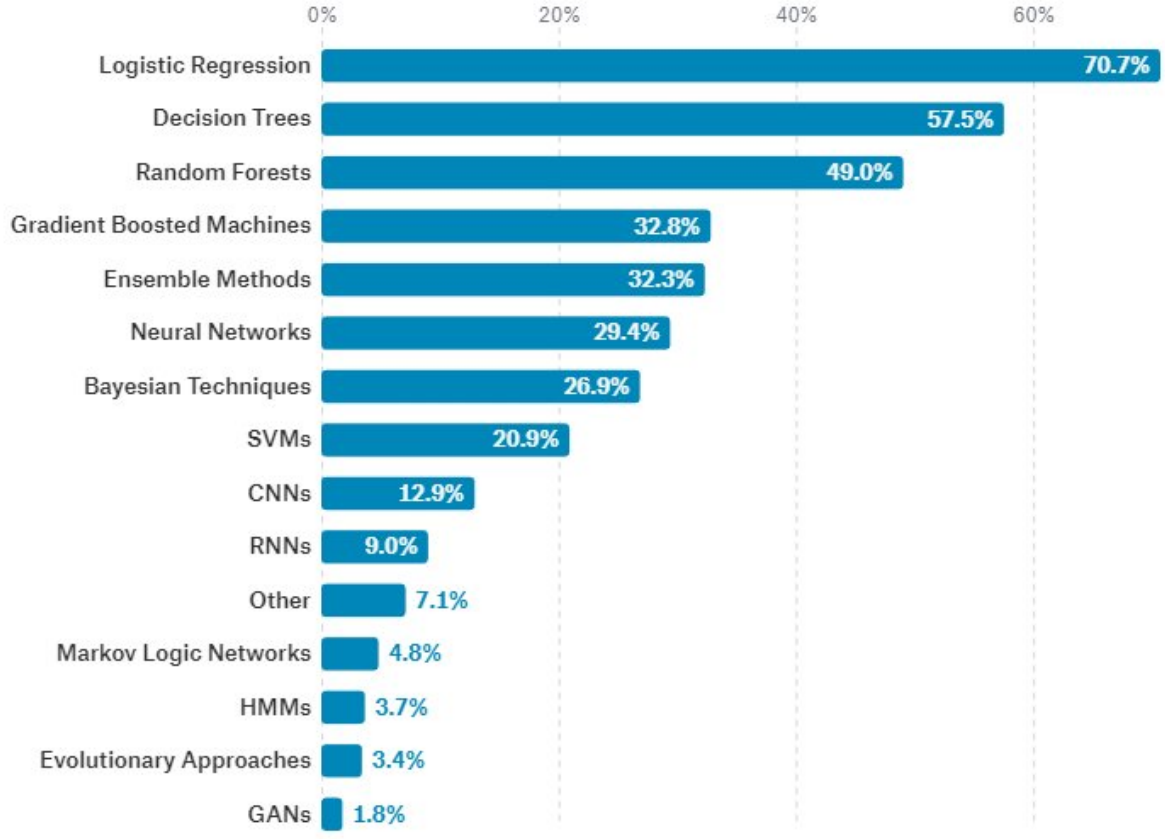
$+1$ 1:1 1045:1 1092:1 ...
-1 2:1 1039:1 1092:1 ...
-1 57:1 1040:1 ...
...

- We can feed this dataset to a machine-learning algorithm.
- Many machine learning algorithms exist:
  - SVM, boosting, decision trees, knn, logistic regression, random forests, perceptron, neural networks, ...
- The important distinctions:
  - output type:
    - Binary, Multiclass, Numeric, Structured
  - scoring type:
    - Linear vs. Non Linear

What data science methods are used at work?

Logistic regression is the most commonly reported data science method used at work for all industries *except* Military and Security where Neural Networks are used slightly more frequently.

Company Size ▾ Financial ▾ Job Title ▾



789 responses

View code in Kaggle Kernels

https://www.kaggle.com/surveys/2017

Types of learning problems

Binary

- Answer is binary: $y \in \{-1, +1\}$
  - boundary/no-boundary, verb-attach/noun-attach, yes/no

Multiclass

- Answer is one of k options: $y \in \{1, \ldots, k\}$
  - choose the part-of-speech of a word.
  - identify the language of a document.

Numeric / Regression

- Answer is a number: $y \in \mathbb{R}$
  - Predict the height of a person.

Structured

- Answer is a complex object:
  - Predict a sequence of tags for a sentence.

Generic NLP Solution

- Find an annotated corpus
- Split it into train and test parts
- Convert it to a vector representation
  - Decide on output type
  - Decide on features
  - Convert each training example to feature vector
- Train a machine-learning model on training set
- Apply machine-learning model to test set
- Measure accuracy

Linear Models

Machine Learning

- Many learning algorithms exist.
- Some of them are based on a linear model:
  - SVM, boosting, logistic-regression, perceptron
- Others are non-linear:
  - Kernel-SVM, decision-trees, knn, random-forests, neural-networks.
- We will work mostly with linear classifiers in this course.
- Neural networks are useful and popular. But there's a separate course for them.

Linear Models

- Some features are indicators for “boundary” and others are good indicators for “no boundary”.
- We will assign a score (weight) to each feature.
  - Features in favor of boundary will receive a positive score.
  - Features in favor of no boundary will receive a negative score.
- **Use the sum of the scores to classify.**

Linear Models

- Some features are indicators for “boundary” and others are good indicators for “no boundary”.
- We will assign a score (weight) to each feature.
  - Features in favor of boundary will receive a positive score.
  - Features in favor of no boundary will receive a negative score.
- **Use the sum of the scores to classify.**

Binary Linear Classifier

$$score(x) = \sum_{i \in 1,\ldots,m} w_i \times \phi_i(x) = \mathbf{w} \cdot \phi(x)$$

$$\hat{y} = predict(x) = \begin{cases} 1 & score(x) \geq 0 \\ -1 & score(x) < 0 \end{cases}$$

Setting the Weights

Feature Examples

$$\phi_1(L.R) = \begin{cases} 1 & \text{if L='subeditor'} \\ 0 & otherwise \end{cases}$$

$$\phi_2(L.R) = \begin{cases} 1 & \text{if L='O'} \\ 0 & otherwise \end{cases}$$

$$\phi_{57}(L.R) = \begin{cases} 1 & \text{if R='George'} \\ 0 & otherwise \end{cases}$$

$$\phi_{1039}(L.R) = \begin{cases} 1 & \text{if len(L)=1} \\ 0 & otherwise \end{cases}$$

$$\phi_{1040}(L.R) = \begin{cases} 1 & \text{if len(L)=2} \\ 0 & otherwise \end{cases}$$

$$\phi_{1045}(L.R) = \begin{cases} 1 & \text{if len(L)>4} \\ 0 & otherwise \end{cases}$$

$$\phi_{1092}(L.R) = \begin{cases} 1 & \text{if isCap(R)} \\ 0 & otherwise \end{cases}$$

$$\phi_{1093}(L.R) = \begin{cases} 1 & \text{if isNumber(L)} \\ 0 & otherwise \end{cases}$$

$$\phi_{2045}(L.R) = \begin{cases} 1 & \text{if L='US' and isCap(R)} \\ 0 & otherwise \end{cases}$$

$$\phi_{3066}(L.R) = \begin{cases} 1 & \text{if len(L)=1 and isCap(R)} \\ 0 & otherwise \end{cases}$$

$$\phi_{5032}(L.R) = \begin{cases} 1 & \text{if L='Ca' and R='The'} \\ 0 & otherwise \end{cases}$$

$$\phi_{5033}(L.R) = \begin{cases} 1 & \text{if L='Ca' and R='snow'} \\ 0 & otherwise \end{cases}$$

Setting the Weights

Binary Linear Classifier

$$score(x) = \sum_{i \in 1,\ldots,m} w_i \times \phi_i(x) = \mathbf{w} \cdot \phi(x)$$

$$\hat{y} = predict(x) = \begin{cases} 1 & score(x) \geq 0 \\ -1 & score(x) < 0 \end{cases}$$

Machine Learning

Provide algorithms for **learning** the values of w from examples.

- A supervised binary learning problem:
  - **Input**: a set of (x, y) examples, features extractor ϕ.
  - **Output:** weights $\mathbf{w}$ such that $\mathbf{w} \cdot \phi(x)$ classifies many of the y's correctly.

Setting weights: the perceptron algorithm

$\mathbf{w} \leftarrow 0$
for x, y in training examples **do**
    $\hat{y} \leftarrow sign(\mathbf{w} \cdot \phi(x))$
    if $\hat{y} < 0$ and $y \geq 0$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \phi(x)$
    if $\hat{y} \geq 0$ and $y < 0$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} - \phi(x)$
return $\mathbf{w}$

Setting the Weights

Multiclass Linear Classifier

$$score(x, y) = \sum_{i \in 1,\ldots,m} w_i^{(y)} \times \phi_i(x, y) = \mathbf{w} \cdot \phi(x, y)$$

$$\hat{y} = predict(x) = \arg\max_y score(x, y)$$

Setting the Weights

Multiclass Linear Classifier

$$score(x, y) = \sum_{i \in 1,\ldots,m} w_i^{(y)} \times \phi_i(x, y) = \mathbf{w} \cdot \phi(x, y)$$

$$\hat{y} = predict(x) = \arg\max_y score(x, y)$$

Alternative view

$$score(x, y) = \sum_{i \in 1,\ldots,m} w_i \times \phi_i(x) = \mathbf{w}^{(y)} \cdot \phi(x)$$

$$\hat{y} = predict(x) = \arg\max_y score(x, y)$$

Here, each class receives its own weight vector.

Setting weights: the multiclass perceptron algorithm

1: $\mathbf{w} \leftarrow 0$
2: **for** x, y in training examples **do**
3: $\quad \hat{y} \leftarrow \arg\max_{y'}(\mathbf{w} \cdot \phi(x, y'))$
4: $\quad$ **if** $\hat{y} \neq y$ **then**
5: $\quad\quad \mathbf{w} \leftarrow \mathbf{w} + \phi(x, y)$
6: $\quad\quad \mathbf{w} \leftarrow \mathbf{w} - \phi(x, \hat{y})$
7: **return** $\mathbf{w}$

Setting weights: the multiclass perceptron algorithm

$\mathbf{w} \leftarrow 0$
for x, y in training examples **do**
    $\hat{y} \leftarrow \arg\max_{y'}(\mathbf{w} \cdot \phi(x, y'))$
    if $\hat{y} \neq y$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \phi(x, y) - \phi(x, \hat{y})$
return $\mathbf{w}$

Perceptron → Averaged Perceptron

$\mathbf{w} \leftarrow 0$
for x, y in training examples **do**
    $\hat{y} \leftarrow \arg\max_{y'}(\mathbf{w} \cdot \phi(x, y'))$
    if $\hat{y} \neq y$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \phi(x, y) - \phi(x, \hat{y})$
return $\mathbf{w}$

Averaged Perceptron

- The perceptron algorithm is not a very good learner
- But can be easily improved

Perceptron → Averaged Perceptron

1: $\mathbf{w} \leftarrow 0$
2: **for** x, y in training examples **do**
3: $\quad \hat{y} \leftarrow \arg\max_{y'}(\mathbf{w} \cdot \phi(x, y'))$
4: $\quad$ **if** $\hat{y} \neq y$ **then**
5: $\quad\quad \mathbf{w} \leftarrow \mathbf{w} + \phi(x, y) - \phi(x, \hat{y})$
6: **return** $\mathbf{w}$

Averaged Perceptron

- The perceptron algorithm is not a very good learner
- But can be easily improved
- Instead of returning $\mathbf{w}$, return $avg(\mathbf{w})$
  - $avg(\mathbf{w})$ is the average of all versions of $\mathbf{w}$ seen in training

Perceptron → Averaged Perceptron

1: $\mathbf{w} \leftarrow 0$
2: **for** x, y in training examples **do**
3: $\quad \hat{y} \leftarrow \arg\max_{y'}(\mathbf{w} \cdot \phi(x, y'))$
4: $\quad$ **if** $\hat{y} \neq y$ **then**
5: $\quad\quad \mathbf{w} \leftarrow \mathbf{w} + \phi(x, y) - \phi(x, \hat{y})$
6: **return** $\mathbf{w}$

Averaged Perceptron

- The perceptron algorithm is not a very good learner
- But can be easily improved
- Instead of returning $\mathbf{w}$, return $avg(\mathbf{w})$
  - $avg(\mathbf{w})$ is the average of all versions of $\mathbf{w}$ seen in training
- Require a bit more book-keeping for calculating the average at the end
- **The result is a very competitive algorithm**

PP-attachment revisited

We calculated

$P(V|v = \text{saw}, n1 = \text{mouse}, p = \text{with}, n2 = \text{telescope})$

Problems

- Was not trivial to come up with a formula.
- Hard to add more sources of information.

New solution

- Encode as a binary or multiclass classification.
- Decide on features.
- Apply learning algorithm.

PP-attachment

Multiclass classification problem

- Previously, we had a binary classification problem:
  - $\Rightarrow$ x = (v,n1,p,n2)
  - $\Rightarrow$ y $\in$ {V,N}
- Let's do a multiclass problem:
  - $\Rightarrow$ y $\in$ {V,N,Other}

PP-attachment

Types of features

- Single items
  - Identity of v
  - Identity of p
  - Identity of n1
  - Identity of n2
- Pairs
  - Identity of (v,p)
  - Identity of (n1,p)
  - Identity of (p,n1)
- Triplets
  - Identity of (v,n1,p)
  - Identity of (v,p,n2)
  - Identity of (n1,p,n2)
- Quadruple
  - Identity of (v,n1,p,n2)

PP-attachment

Types of features

- Single items
  - **Identity of v**
  - Identity of p
  - Identity of n1
  - Identity of n2
- Pairs
  - Identity of (v,p)
  - Identity of (n1,p)
  - Identity of (p,n1)
- Triplets
  - Identity of (v,n1,p)
  - Identity of (v,p,n2)
  - Identity of (n1,p,n2)
- Quadruple
  - Identity of (v,n1,p,n2)

$$\phi_{1039} = \begin{cases} 1 & \text{if v='ate' and y='N'} \\ 0 & otherwise \end{cases}$$

$$\phi_{1040} = \begin{cases} 1 & \text{if v='ate' and y='V'} \\ 0 & otherwise \end{cases}$$

$$\phi_{1041} = \begin{cases} 1 & \text{if v='ate' and y='O'} \\ 0 & otherwise \end{cases}$$

$$\phi_{1042} = \begin{cases} 1 & \text{if v='saw' and y='N'} \\ 0 & otherwise \end{cases}$$

$\ldots$

PP-attachment

Types of features

- Single items
  - Identity of v
  - Identity of p
  - Identity of n1
  - Identity of n2
- Pairs
  - **Identity of (v,p)**
  - Identity of (n1,p)
  - Identity of (p,n1)
- Triplets
  - Identity of (v,n1,p)
  - Identity of (v,p,n2)
  - Identity of (n1,p,n2)
- Quadruple
  - Identity of (v,n1,p,n2)

$$\phi_{2349} = \begin{cases} 1 & \text{if v='ate' p='with' and y='N'} \\ 0 & otherwise \end{cases}$$

$$\phi_{2350} = \begin{cases} 1 & \text{if v='ate' p='with' and y='V'} \\ 0 & otherwise \end{cases}$$

$$\phi_{2351} = \begin{cases} 1 & \text{if v='ate' p='with' and y='O'} \\ 0 & otherwise \end{cases}$$

...

PP-attachment

Types of features

- Single items
  - Identity of v
  - Identity of p
  - Identity of n1
  - Identity of n2
- Pairs
  - Identity of (v,p)
  - Identity of (n1,p)
  - Identity of (p,n1)
- Triplets
  - Identity of (v,n1,p)
  - Identity of (v,p,n2)
  - Identity of (n1,p,n2)
- Quadruple
  - Identity of (v,n1,p,n2)

$$\phi_{2349} = \begin{cases} 1 & \text{if v='ate' p='with' and y='N'} \\ 0 & otherwise \end{cases}$$

$$\phi_{2350} = \begin{cases} 1 & \text{if v='ate' p='with' and y='V'} \\ 0 & otherwise \end{cases}$$

$$\phi_{2351} = \begin{cases} 1 & \text{if v='ate' p='with' and y='O'} \\ 0 & otherwise \end{cases}$$

...

PP-attachment

More features?

- Corpus Level
  - Did we see the (v,p) pair in a 5-word window in a big corpus?
  - Did we see the (n1,p) pair in a 5-word window in a big corpus?
  - Did we see the (n1,p,n2) triplet in a 5-word window in a big corpus?
    - We can also use counts, or binned counts.
- Distance
  - Distance (in words) between v and p
  - Distance (in words) between n1 and p
  - ...

PP-attachment

- Compute features for each example.
- Feed into a machine learning algorithm (for example SVM or perceptron).
- Get a weight vector.
- Classify new examples.

POS-tagging revisited

POS-tagging

$$x = x_1, \ldots, x_n = \text{Holly came from Miami , FLA}$$

$$y = y_1, \ldots, y_n = \text{NNP VBD IN NNP , NNP}$$

POS-tagging

$$x = x_1, \ldots, x_n = \text{Holly came from Miami , FLA}$$

$$y = y_1, \ldots, y_n = \text{NNP VBD IN NNP , NNP}$$

- Our job is to predict y. We will score each y_i in turn.

POS-tagging

$$x = x_1, \ldots, x_n = \text{Holly came from Miami , FLA}$$

$$y = y_1, \ldots, y_n = \text{NNP VBD IN NNP , NNP}$$

- Our job is to predict y. We will score each y_i in turn.
  - Reminder: in the HMM case we had:

$$score(y_i|x, y_{i-1}, y_{i-2}) = q(y_i|y_{i-1}, y_{i-2})e(x_i|y_i)$$

where e, q are MLE estimates.

POS-tagging

$$x = x_1, \ldots, x_n = \text{Holly came from Miami , FLA}$$

$$y = y_1, \ldots, y_n = \text{NNP VBD IN NNP , NNP}$$

- Our job is to predict y. We will score each y_i in turn.
  - Reminder: in the HMM case we had:

$$score(y_i|x, y_{i-1}, y_{i-2}) = q(y_i|y_{i-1}, y_{i-2})e(x_i|y_i)$$

    where e, q are MLE estimates.
- In the feature based approach:

$$predict(y_i|x, y_{[0:i-1]}) = \arg\max_{y_i \in tagset} w \cdot \phi(x, y_{[0:i-1]}, i, y_i)$$

- We define ϕ to take position into account: $\phi(x, y_{[0:i-1]}, i, y_i)$

POS-tagging

Feature Examples

Sequence:

$$\phi_{349} = \begin{cases} 1 & \text{if } y_{i-1}\text{='JJ'}, \wedge\ y_{i-2}\text{='DT'} \wedge\ y_i\text{='NN'} \\ 0 & otherwise \end{cases}$$

$$\phi_{355} = \begin{cases} 1 & \text{if } \wedge\ y_{i-1}\text{='JJ'} \wedge\ y_i\text{='NN'} \\ 0 & otherwise \end{cases}$$

$$\phi_{392} = \begin{cases} 1 & \text{if } y_i\text{='NN'} \\ 0 & otherwise \end{cases}$$

Local:

$$\phi_{231} = \begin{cases} 1 & \text{if } x_i\text{='saw'} \wedge\ y_i\text{='VBD'} \\ 0 & otherwise \end{cases}$$

POS-tagging

Feature Examples

Sequence:

$$\phi_{349} = \begin{cases} 1 & \text{if } y_{i-1}\text{='JJ'}, \wedge\ y_{i-2}\text{='DT'} \wedge\ y_i\text{='NN'} \\ 0 & otherwise \end{cases}$$

$$\phi_{355} = \begin{cases} 1 & \text{if } \wedge\ y_{i-1}\text{='JJ'} \wedge\ y_i\text{='NN'} \\ 0 & otherwise \end{cases}$$

$$\phi_{392} = \begin{cases} 1 & \text{if } y_i\text{='NN'} \\ 0 & otherwise \end{cases}$$

Local:

$$\phi_{231} = \begin{cases} 1 & \text{if } x_i\text{='saw'} \wedge\ y_i\text{='VBD'} \\ 0 & otherwise \end{cases}$$

- These correspond to q and e from the HMM.
- Did we gain anything yet?

POS-tagging

More Feature Examples

3 Suffix:

$$\phi_{121} = \begin{cases} 1 & \text{if } x_i \text{ endsWith 'ing'} \wedge y_i\text{='VBD'} \\ 0 & otherwise \end{cases}$$

$$\phi_{122} = \begin{cases} 1 & \text{if } x_i \text{ endsWith 'int'} \wedge y_i\text{='VBD'} \\ 0 & otherwise \end{cases}$$

$\ldots$

2 Suffix:

$$\phi_{222} = \begin{cases} 1 & \text{if } x_i \text{ endsWith 'ed'} \wedge y_i\text{='VBD'} \\ 0 & otherwise \end{cases}$$

$$\phi_{225} = \begin{cases} 1 & \text{if } x_i \text{ endsWith 'ng'} \wedge y_i\text{='VBD'} \\ 0 & otherwise \end{cases}$$

$$\phi_{228} = \begin{cases} 1 & \text{if } x_i \text{ endsWith 'he'} \wedge y_i\text{='VBD'} \\ 0 & otherwise \end{cases}$$

$\ldots$

POS-tagging

More Feature Examples

Word-features:

$$\phi_{552} = \begin{cases} 1 & \text{if hasHyphen}(x_i) \wedge y_i\text{='JJ'} \\ 0 & otherwise \end{cases}$$

$$\phi_{553} = \begin{cases} 1 & \text{if hasDigit}(x_i) \wedge y_i\text{='JJ'} \\ 0 & otherwise \end{cases}$$

$$\phi_{554} = \begin{cases} 1 & \text{if allDigits}(x_i) \wedge y_i\text{='JJ'} \\ 0 & otherwise \end{cases}$$

$$\phi_{555} = \begin{cases} 1 & \text{if isUpper}(x_i) \wedge y_i\text{='JJ'} \\ 0 & otherwise \end{cases}$$

$$\phi_{556} = \begin{cases} 1 & \text{if allUpper}(x_i) \wedge y_i\text{='JJ'} \\ 0 & otherwise \end{cases}$$

...

POS-tagging

More Feature Examples

Next word:

$$\phi_{621} = \begin{cases} 1 & \text{if } x_{i+1} \text{ endsWith 'ing'} \wedge y_i\text{='JJ'} \\ 0 & otherwise \end{cases}$$

$$\phi_{649} = \begin{cases} 1 & \text{if } x_{i+1}\text{='yesterday'} \wedge y_i\text{='JJ'} \\ 0 & otherwise \end{cases} \quad \ldots$$

Prev word:

$$\phi_{721} = \begin{cases} 1 & \text{if } x_{i-1} \text{ endsWith 'ing'} \wedge y_i\text{='JJ'} \\ 0 & otherwise \end{cases}$$

$$\phi_{749} = \begin{cases} 1 & \text{if } x_{i-1}\text{='yesterday'} \wedge y_i\text{='JJ'} \\ 0 & otherwise \end{cases}$$

$\ldots$

POS-tagging

- We gained much more flexibility in what information sources we can use in the scoring.
- But how do we tag?
  - Can we use viterbi?

POS-tagging

- We cannot use viterbi because the scores are "meaningless".
- But greedy tagging works quite well!

1: **Input:** sentence x, parameter-vector $\mathbf{w}$, feature extractor ϕ
2: $y \leftarrow None$
3: **for** i in $1, \ldots, |x|$ **do**
4: $\quad y_i \leftarrow \arg\max_{y_i \in tagset} \mathbf{w} \cdot \phi(x, y_{[0:i-1]}, i, y_i)$
5: **return** y

POS-tagging

- We cannot use viterbi because the scores are "meaningless".
- But greedy tagging works quite well!

Input: sentence x, parameter-vector $\mathbf{w}$, feature extractor ϕ
$y \leftarrow None$
for i in $1, \ldots, |x|$ **do**
    $y_i \leftarrow \arg\max_{y_i \in tagset} \mathbf{w} \cdot \phi(x, y_{[0:i-1]}, i, y_i)$
return y

- This was a bad idea before. What changed?

POS-tagging

Improving Greedy Tagging

Training Algorithm

1: $\mathbf{w} \leftarrow 0$
2: **for** x, y in examples **do**
3: **for** i in $1, \ldots, |x|$ **do**
4: $\hat{y}_i \leftarrow \arg\max_{y_i \in tagset} \mathbf{w} \cdot \phi(x, y_{[0:i-1]}, i, y_i)$
5: **if** $\hat{y}_i \neq y_i$ **then**
6: $\mathbf{w} \leftarrow \mathbf{w} + \phi(x, y_{[0:i-1]}, i, y_i) - \phi(x, y_{[0:i-1]}, i, \hat{y}_i)$
7: **return** $\mathbf{w}$

POS-tagging

Improving Greedy Tagging

Better Training Algorithm

1: $\mathbf{w} \leftarrow 0$
2: **for** x, y in examples **do**
3: $\quad y' \leftarrow y$
4: $\quad$ **for** i in $1, \ldots, |x|$ **do**
5: $\quad\quad \hat{y}_i \leftarrow \arg\max_{y_i \in tagset} \mathbf{w} \cdot \phi(x, y'_{[0:i-1]}, i, y_i)$
6: $\quad\quad$ **if** $\hat{y}_i \neq y_i$ **then**
7: $\quad\quad\quad \mathbf{w} \leftarrow \mathbf{w} + \phi(x, y'_{[0:i-1]}, i, y_i) - \phi(x, y'_{[0:i-1]}, i, \hat{y}_i)$
8: $\quad\quad y'_i \leftarrow \hat{y}_i$
9: **return** $\mathbf{w}$

POS-tagging

Improving Greedy Tagging

Better Training Algorithm

1: $\mathbf{w} \leftarrow 0$
2: **for** x, y in examples **do**
3: $\quad y' \leftarrow y$
4: $\quad$ **for** i in $1, \ldots, |x|$ **do**
5: $\quad\quad \hat{y}_i \leftarrow \arg\max_{y_i \in tagset} \mathbf{w} \cdot \phi(x, y'_{[0:i-1]}, i, y_i)$
6: $\quad\quad$ **if** $\hat{y}_i \neq y_i$ **then**
7: $\quad\quad\quad \mathbf{w} \leftarrow \mathbf{w} + \phi(x, y'_{[0:i-1]}, i, y_i) - \phi(x, y'_{[0:i-1]}, i, \hat{y}_i)$
8: $\quad\quad y'_i \leftarrow \hat{y}_i$
9: **return** $\mathbf{w}$

- Predict y_i based on previous predictions, can recover from errors.

POS-tagging

- Greedy tagging can be quite accurate (and very fast).

POS-tagging

- Greedy tagging can be quite accurate (and very fast).
- But a global search is still preferable.
- What if we do want to have global search?

POS-tagging

- Greedy tagging can be quite accurate (and very fast).
- But a global search is still preferable.
- What if we do want to have global search?

$\Rightarrow$ We need meaningful scores.

Log-linear Modeling

Log-linear Modeling

- Our linear classifiers give us a score:

$$score(x, y_{[0:i-1]}, i, y_i) = \mathbf{w} \cdot \phi(x, y_{[0:i-1]}, i, y_i)$$

- This is great, if we are looking for the best y.
- But sometimes we do want a probability:

$$p(y_i | x, y_{[0:i-1]}, i)$$

Log-linear Modeling

- Our linear classifiers give us a score:

$$score(x, y_{[0:i-1]}, i, y_i) = \mathbf{w} \cdot \phi(x, y_{[0:i-1]}, i, y_i)$$

- This is great, if we are looking for the best y.
- But sometimes we do want a probability:

$$p(y_i | x, y_{[0:i-1]}, i)$$

- In a log linear model, we define:

$$p(y_i | x, y_{[0:i-1]}, i) = \frac{e^{\mathbf{w} \cdot \phi(x, y_{[0:i-1]}, i, y_i)}}{\sum_{y'_i \in tagset} e^{\mathbf{w} \cdot \phi(x, y_{[0:i-1]}, i, y'_i)}}$$

Log-linear Modeling

Name

- In a log linear model, we define:

$$p(y_i|x, y_{[0:i-1]}, i) = \frac{e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y_i)}}{\sum_{y'_i \in tagset} e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y'_i)}}$$

- Why is it called log-linear?

$$\log p(y_i|x, y_{[0:i-1]}, i) = \mathbf{w}\cdot\phi(x, y_{[0:i-1]}, i, y_i) - \log \sum_{y'_i \in tagset} e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y'_i)}$$

Log-linear Modeling

Training Objective

- In a log linear model, we define:

$$p(y_i|x, y_{[0:i-1]}, i) = \frac{e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y_i)}}{\sum_{y'_i \in tagset} e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y'_i)}}$$

- The exponentiation makes everything positive.
- The denominator is normalizing everything to sum to 1.
  - $\Rightarrow$ The result is a formal probability.
  - $\Rightarrow$ But is it the "real" probability?

Log-linear Modeling

Training Objective

- In a log linear model, we define:

$$p(y_i|x, y_{[0:i-1]}, i) = \frac{e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y_i)}}{\sum_{y'_i \in tagset} e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y'_i)}}$$

- The exponentiation makes everything positive.
- The denominator is normalizing everything to sum to 1.
  - $\Rightarrow$ The result is a formal probability.
  - $\Rightarrow$ But is it the "real" probability?
- The job of learning is to find $\mathbf{w}$ such to maximize:

$$\sum_{x,y \in data} \sum_{i \in 1,\ldots,|x|} \log p(y_i|x, y_{[0:i-1]}, i)$$

(maximize the log likelihood of the data)

Log-linear Modeling

Smoothing

- In a log linear model, we define:

$$p(y_i|x, y_{[0:i-1]}, i) = \frac{e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y_i)}}{\sum_{y'_i \in tagset} e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y'_i)}}$$

- The job of learning is to find $\mathbf{w}$ such to maximize:

$$\sum_{x,y \in data} \sum_{i \in 1,\ldots,|x|} \log p(y_i|x, y_{[0:i-1]}, i)$$

Log-linear Modeling

Smoothing

- In a log linear model, we define:

$$p(y_i|x, y_{[0:i-1]}, i) = \frac{e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y_i)}}{\sum_{y'_i \in tagset} e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y'_i)}}$$

- The job of learning is to find $\mathbf{w}$ such to maximize:

$$\sum_{x,y \in data} \sum_{i \in 1,\ldots,|x|} \log p(y_i|x, y_{[0:i-1]}, i)$$

- We can make it better by adding a regularization term:

$$\sum_{x,y \in data} \sum_{i \in 1,\ldots,|x|} \log p(y_i|x, y_{[0:i-1]}, i) - \frac{\lambda}{2} \sum w_i^2$$

- This prefers that most weights are small $\rightarrow$ avoid overfitting

Log-linear Modeling

Note

- This form of the model and objective is specialized to our tagging setup:

$$p(y_i|x, y, i) = \frac{e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y_i)}}{\sum_{y'_i \in tagset} e^{\mathbf{w}\cdot\phi(x,y_{[0:i-1]},i,y'_i)}}$$

$$\arg\max_{\mathbf{w}\in\mathbb{R}^m} \sum_{x,y\in data} \sum_{i\in 1,\ldots,|x|} \log p(y_i|x, y_{[0:i-1]}, i) - \frac{\lambda}{2}\sum w_i^2$$

- The general form is:

$$p(y|x) = \frac{e^{\mathbf{w}\cdot\phi(x,y)}}{\sum_{y'} e^{\mathbf{w}\cdot\phi(x,y')}}$$

$$\arg\max_{\mathbf{w}\in\mathbb{R}^m} \sum_{x,y\in data} \log p(y|x) - \frac{\lambda}{2}\sum w_i^2$$

Log-linear Modeling

Summary

- We defined $p(y|x) = \frac{e^{\mathbf{w}\cdot\phi(x,y)}}{\sum_{y'_i \in tagset} e^{\mathbf{w}\cdot\phi(x,y)}}$
- And train the model to optimize the data log-likelihood.
  - We also added a regularization term.

Log-linear Modeling

Summary

- We defined $p(y|x) = \frac{e^{\mathbf{w}\cdot\phi(x,y)}}{\sum_{y'_i \in tagset} e^{\mathbf{w}\cdot\phi(x,y)}}$
- And train the model to optimize the data log-likelihood.
  - We also added a regularization term.
- This is a very common approach, and goes by two names:
  - Multinomail Logistic Regression
  - Maximum Entropy Model (Maxent)

Log-linear Modeling

Summary

- We defined $p(y|x) = \frac{e^{\mathbf{w}\cdot\phi(x,y)}}{\sum_{y'_i \in tagset} e^{\mathbf{w}\cdot\phi(x,y)}}$
- And train the model to optimize the data log-likelihood.
  - We also added a regularization term.
- This is a very common approach, and goes by two names:
  - Multinomail Logistic Regression
  - Maximum Entropy Model (Maxent)
- There are many tools for training such models:
  - MegaM
  - Weka
  - Liblinear (when you pass the correct options)
  - VW (when you pass in the correct options)
  - scikit-learn (when you pass in the correct options)
  - Many others
  - (some of you implemented it in the deep-learning course...)

Back to tagging

MEMM – Maximum Entropy Markov Model

MEMM – MaxEnt Markov Models

- In a trigram HMM, we had:

$$p(x, y) = \prod_{i \in 1,\ldots,|x|} q(y_i | y_{i-1}, y_{i-2}) e(x_i | y_i)$$

$$\hat{y} = \arg\max_t p(x, y)$$

MEMM – MaxEnt Markov Models

- In a trigram HMM, we had:

$$p(x, y) = \prod_{i \in 1,\ldots,|x|} q(y_i | y_{i-1}, y_{i-2}) e(x_i | y_i)$$

$$\hat{y} = \arg\max_t p(x, y)$$

- In a trigram MEMM we have:

$$p(y|x) = \prod_{i \in 1,\ldots,|x|} p(y_i | x, y_{i-1}, y_{i-2})$$

$$\hat{y} = \arg\max_y p(y|x)$$

MEMM – MaxEnt Markov Models

- In a trigram HMM, we had:

$$p(x, y) = \prod_{i \in 1,\ldots,|x|} q(y_i | y_{i-1}, y_{i-2}) e(x_i | y_i)$$

$$\hat{y} = \arg\max_t p(x, y)$$

- In a trigram MEMM we have:

$$p(y|x) = \prod_{i \in 1,\ldots,|x|} p(y_i | x, y_{i-1}, y_{i-2}) = \prod_{i \in 1,\ldots,|x|} \frac{e^{\mathbf{w} \cdot \phi(x, y_{[0:i-1]}, i, y_i)}}{\sum_{y'_i} e^{\mathbf{w} \cdot \phi(x, y_{[0:i-1]}, i, y'_i)}}$$

$$\hat{y} = \arg\max_y p(y|x)$$

MEMM – MaxEnt Markov Models

- In a trigram MEMM we have:

$$p(y|x) = \prod_{i \in 1,\ldots,|x|} p(y_i|x, y_{i-1}, y_{i-2}) = \prod_{i \in 1,\ldots,|x|} \frac{e^{\mathbf{w}\cdot\phi(x, y_{[0:i-1]}, i, y_i)}}{\sum_{y'_i} e^{\mathbf{w}\cdot\phi(x, y_{[0:i-1]}, i, y'_i)}}$$

$$\hat{y} = \arg\max_y p(y|x)$$

$\Rightarrow$ Train $\mathbf{w}$ using a Maxent learner (new)

$\Rightarrow$ Decode (solve the argmax) using viterbi (like before)

POS-tagging

Summary

- MEMM combine the flexibility of the feature-based approach with the global search and probability score of HMM
- But greedy decoding is very fast and very competitive.
- MEMM still a “locally-trained” model.

Structured Perceptron

POS-tagging

- MEMM combine the flexibility of the feature-based approach with the global search and probability score of HMM
- But greedy decoding is very fast and very competitive.
- MEMM still a “locally-trained” model.

$\Rightarrow$ Can we have a globally trained model?

POS-tagging

- MEMM combine the flexibility of the feature-based approach with the global search and probability score of HMM
- But greedy decoding is very fast and very competitive.
- MEMM still a “locally-trained” model.

$\Rightarrow$ Can we have a globally trained model?

- Assume for now that we do not care about probability.

POS-tagging

$\Rightarrow$ Can we have a globally trained model?

- Assume for now that we do not care about probability.

- Define $score(x, y, i, y_i) = \mathbf{w} \cdot \phi(x, y, i, y_i)$
- Then the sequence score is $score(x, y) = \sum_i \mathbf{w} \cdot \phi(x, y, i, y_i)$
- We can find $\arg\max_y score(x, y)$ using viterbi.
  - (assuming ϕ is "well behaved" – not looking at y_{i-3} or y_{i+1})
- Let's write:

$$\Phi(x, y) = \sum_i \phi(x, y, i, y_i)$$

- and now: $score(x, y) = \mathbf{w} \cdot \Phi(x, y)$
- we need $\mathbf{w} \cdot \Phi(x, y) > \mathbf{w} \cdot \Phi(x, y')$ for all incorrect y'.

Structured Perceptron

$$\Phi(x, y) = \sum_i \phi(x, y, i, y_i)$$

$$score(x, y) = \mathbf{w} \cdot \Phi(x, y)$$

we need $\mathbf{w} \cdot \Phi(x, y) > \mathbf{w} \cdot \Phi(x, y')$ for all incorrect y'

Structured Perceptron

$$\Phi(x, y) = \sum_i \phi(x, y, i, y_i)$$

$$score(x, y) = \mathbf{w} \cdot \Phi(x, y)$$

we need $\mathbf{w} \cdot \Phi(x, y) > \mathbf{w} \cdot \Phi(x, y')$ for all incorrect y'

Training – structured perceptron

1: $\mathbf{w} \leftarrow 0$
2: **for** x, y in examples **do**
3: $\quad \hat{y} \leftarrow \arg\max_{y'} \mathbf{w} \cdot \Phi(x, y')$ ▷ Using viterbi
4: $\quad$ **if** $\hat{y} \neq y$ **then**
5: $\quad\quad \mathbf{w} \leftarrow \mathbf{w} + \Phi(x, y) - \Phi(x, \hat{y})$
6: **return** $\mathbf{w}$

Structured Perceptron

$$\Phi(x, y) = \sum_i \phi(x, y, i, y_i)$$

$$score(x, y) = \mathbf{w} \cdot \Phi(x, y)$$

we need $\mathbf{w} \cdot \Phi(x, y) > \mathbf{w} \cdot \Phi(x, y')$ for all incorrect y'

Training – structured perceptron

1: $\mathbf{w} \leftarrow 0$
2: **for** x, y in examples **do**
3: $\quad \hat{y} \leftarrow \arg\max_{y'} \mathbf{w} \cdot \Phi(x, y')$ ▷ Using viterbi
4: $\quad$ **if** $\hat{y} \neq y$ **then**
5: $\quad\quad \mathbf{w} \leftarrow \mathbf{w} + \Phi(x, y) - \Phi(x, \hat{y})$
6: **return** $\mathbf{w}$

Structured Perceptron Tagger

- Globally optimized – optimizing directly what we care about
- Does not provide probabilities

Structured Perceptron Tagger

- Globally optimized – optimizing directly what we care about
- Does not provide probabilities
  - There are global model that do provide probabilities – CRFs
  - CRFs are the log-linear version of the structured-percepton.

Summary

Discriminative Learning

- Feature representation, feature vectors
- Linear models (binary, multiclass)
  - The perceptron algorithm
- Log-linear models
  - Regularization

Summary

Discriminative Learning

- Feature representation, feature vectors
- Linear models (binary, multiclass)
  - The perceptron algorithm
- Log-linear models
  - Regularization

Sequence Tagging

- HMM (last time)
- Greedy, classifier based
- Greedy, classifier based – improved
- MEMM
- Structured Perceptron