FNLP
Classification – Log-linear Models

Yansong Feng
fengyansong@pku.edu.cn

Wangxuan Institute of Computer Technology
Peking University

February 26, 2025

Recall in WSD

We have considered the following clues as useful:

- neighbouring words around the target
- Part-of-Speech tags of the target, and neighbouring words
- syntax clues
- ...

Recall in WSD

We have considered the following clues as useful:

- neighbouring words around the target
- Part-of-Speech tags of the target, and neighbouring words
- syntax clues
- ...

These hints work like:

when you see **blablabla**, the target word should have the **XXX** sense...

Recall in WSD

We have considered the following clues as useful:

- neighbouring words around the target
- Part-of-Speech tags of the target, and neighbouring words
- syntax clues
- ...

These hints work like:

when you see **blablabla**, the target word should have the **XXX** sense...

- there may be many applicable hints
- are those hints equally important?
- how should I make the final decision accordingly?

Recall in WSD

We have considered the following clues as useful:

- neighbouring words around the target
- Part-of-Speech tags of the target, and neighbouring words
- syntax clues
- ...

These hints work like:
when you see **blablabla**, the target word should have the **XXX** sense...

- there may be many applicable hints
- are those hints equally important?
- how should I make the final decision accordingly?

(1) Design those hints
(2) Weight those hints
(3) Do the scoring
(4) Rank the candidates!

New View: Features

Features: pieces of evidences describing some aspects of observed data x, usually with respect to the predicted label y

- computer vision
  - the shape, color, texture, size.....of an object
  - other objects nearby, relative postions
  - number of objects available
  - ...
- natural language processing
  - the target word itself, POS, prefix, suffix, capital or not, ...
  - context: words before/after the target, their morphology, POS, ...
  - number of those indications
  - ...

New View: Features

Features: pieces of evidences describing some aspects of observed data x, usually with respect to the predicted label y

- computer vision
  - the shape, color, texture, size.....of an object
  - other objects nearby, relative postions
  - number of objects available
  - ...
- natural language processing
  - the target word itself, POS, prefix, suffix, capital or not, ...
  - context: words before/after the target, their morphology, POS, ...
  - number of those indications
  - ...
- dense vector representations
  - word embedding
  - tree embedding, graph embedding, ...
  - ...

New View: Features

Features: pieces of evidences describing some aspects of observed data x, usually with respect to the predicted label y

- computer vision
  - the shape, color, texture, size.....of an object
  - other objects nearby, relative postions
  - number of objects available
  - ...
- natural language processing
  - the target word itself, POS, prefix, suffix, capital or not, ...
  - context: words before/after the target, their morphology, POS, ...
  - number of those indications
  - ...
- dense vector representations
  - word embedding
  - tree embedding, graph embedding, ...
  - embedding anything ...
  - ...

Features in NLP

Features in NLP: pieces of evidences describing some aspects of observed data x with respect to the predicted label y, usually with the purpose of providing a conditional probability $p(y|x)$

Features in NLP

Features in NLP: pieces of evidences describing some aspects of observed data x with respect to the predicted label y, usually with the purpose of providing a conditional probability $p(y|x)$
$\rightarrow$ **discriminative models**

Features in NLP

- A feature is **a function of x and y,** $f_i(x, y) \in \mathcal{R}$
- more often, it is a binary or indicator function:

Example

- when we do WSD for the target word <u>*bank*</u>,

$$f_i = \begin{cases} 1 & \text{if } w_{-1} = \text{transfer and } y = \textbf{FINANCIAL}, \\ 0 & \text{otherwise} \end{cases}$$

if the previous word is *transfer*, the current target should have the sense of **FINANCIAL**.

Features in NLP

- A feature is **a function of** x **and** y**,** $f_i(x, y) \in \mathcal{R}$
- more often, it is a binary or indicator function:

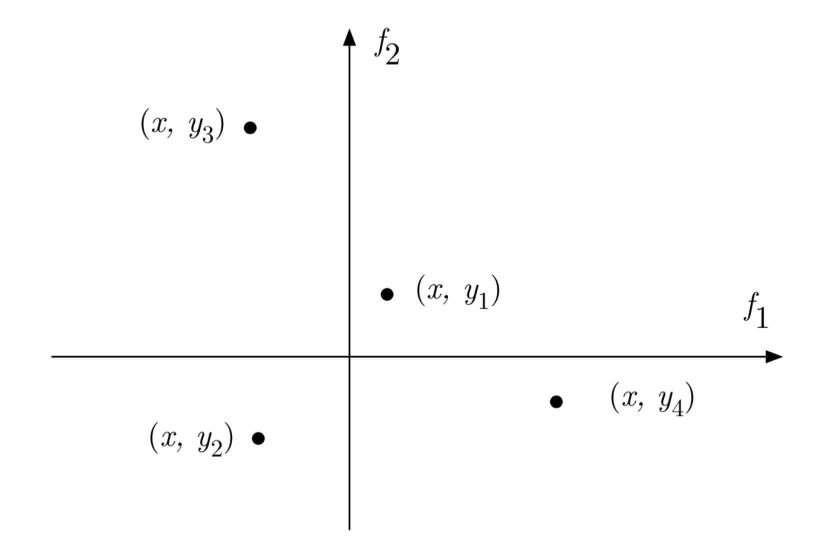
Example

- when we do WSD for the target word *bank*,

$$f_i = \begin{cases} 1 & \text{if } w_{-1} = \text{transfer and } y = \textbf{FINANCIAL}, \\ 0 & \text{otherwise} \end{cases}$$

- if we have m aspects to describe an instance, i.e., m features, then:
  - **a feature vector** for each instance, (x, y)
  - $[f_1(x, y), f_2(x, y), f_3(x, y), ..., f_m(x, y)]$
  - $[1, 0, 0, ..., 1, 0]$

Features in NLP



[from Noah Smith]

Examples of Features in NLP

WSD for **bank**:

- if the previous word is transfer, it should be of **FINANCIAL**.
- if the next word is note, it should be of **FINANCIAL**.
- if the sentence is more than 10 words long, it should be of **FINANCIAL**.
- if **bank** is the first word of the sentence, it should be of **FINANCIAL**.
- ...

Examples of Features in NLP

WSD for **bank**:

- if the previous word is <u>transfer</u>, it should be of **FINANCIAL**.
- if the next word is <u>note</u>, it should be of **FINANCIAL**.
- if the sentence is more than <u>10</u> words long, it should be of **FINANCIAL**.
- if **bank** is the <u>first</u> word of the sentence, it should be of **FINANCIAL**.
- ...
- Many features are relevant, thus useful
- Features can be local or beyond

Examples of Features in NLP

WSD for **bank**:

- if the previous word is transfer, it should be of **FINANCIAL**.
- if the next word is note, it should be of **FINANCIAL**.
- if the sentence is more than 10 words long, it should be of **FINANCIAL**.
- if **bank** is the first word of the sentence, it should be of **FINANCIAL**.
- ...
- Many features are relevant, thus useful
- Features can be local or beyond

Any differences compared to what we have done last week?

Naïve Bayes?

Examples of Features in NLP

Text Classification for news reports:

- if the document contains Kobe, its category should be of **Sports**
- if the title contains NBA, its category should be of **Sports**
- if football appear in the upper half, its category should be of **Sports**
- ...

An Example in WSD

I cash a check in that **bank** *and transfer 100 dollars to my mom.*

simply written as: $f(x, y)$

i.e., $f(x, Fi)$, $f(x, Sp)$, $f(x, Po)$, $f(x, En)$,...

An Example in WSD

I cash a check in that **bank** *and transfer 100 dollars to my mom.*

Feature templates

- words before target words: $w_{-1} = $ *, $w_{-2} = $ *, $w_{-3} = $ *, ...
- words after target words: $w_1 = $ *, $w_2 = $ *, $w_3 = $ *, ...
- whether at the beginning of sent.: $@1 = \{$YES, NO$\}$,
- whether capitalized or not: $Cap. = \{$YES, NO$\}$,
- ...

An Example in WSD

I cash a check in that **bank** *and transfer 100 dollars to my mom.*

Feature templates

- words before target words: $w_{-1} =$ *, $w_{-2} =$ *, $w_{-3} =$ *, ...
- words after target words: $w_1 =$ *, $w_2 =$ *, $w_3 =$ *, ...
- whether at the beginning of sent.: $@1 = \{$YES, NO$\}$,
- whether capitalized or not: $Cap. = \{$YES, NO$\}$,
- ...

Apply those feature templates to the input sentence

- $w_{-1} =$ that, $w_{-2} =$ in, $w_{-3} =$ check, ...
- $w_1 =$ and, $w_2 =$ transfer, $w_3 =$ 100, ...
- $@1 =$ NO, $Cap. =$ NO, ...
- ...

An Example in WSD

I cash a check in that **bank** *and transfer 100 dollars to my mom.*

Apply those feature templates to the input sentence

- $w_{-1} =$ that, $w_{-2} =$ in, $w_{-3} =$ check, ...
- $w_1 =$ and, $w_2 =$ transfer, $w_3 =$ 100, ...
- $@1 =$ NO, $Cap. =$ NO, ...
- ...

when we consider a candidate label Fi for this case:

- $f_{w-1=that,Fi} = 1$, $f_{w-2=in,Fi} = 1$, $f_{w-3=check,Fi} = 1$, ...
- $f_{w1=and,Fi} = 1$, $f_{w2=transfer,Fi} = 1$, $f_{w3=100,Fi} = 1$, ...
- $f_{at1,Fi} = 0$, $f_{Cap.,Fi} = 0$, ...

simply written as: $f(x, y)$

i.e., $f(x, Fi)$, $f(x, Sp)$, $f(x, Po)$, $f(x, En)$,...

An Example in WSD

I cash a check in that **bank** *and transfer 100 dollars to my mom.*

Apply those feature templates to the input sentence

- $w_{-1} =$ that, $w_{-2} =$ in, $w_{-3} =$ check, ...
- $w_1 =$ and, $w_2 =$ transfer, $w_3 = 100$, ...
- $@1 =$ NO, $Cap. =$ NO, ...
- ...

when we consider a candidate label Fi for this case:

- $f_{w-1=that,Fi} = 1$, $f_{w-2=in,Fi} = 1$, $f_{w-3=check,Fi} = 1$, ...
- $f_{w1=and,Fi} = 1$, $f_{w2=transfer,Fi} = 1$, $f_{w3=100,Fi} = 1$, ...
- $f_{at1,Fi} = 0$, $f_{Cap.,Fi} = 0$, ...
- ... $\Longrightarrow [1, 1, 1, 1, 0, 0, ..., 1, 0, 0]$

simply written as: $f(x, y)$

i.e., $f(x, Fi)$, $f(x, Sp)$, $f(x, Po)$, $f(x, En)$,...

An Example in WSD

I cash a check in that **bank** *and transfer 100 dollars to my mom.*

Apply those feature templates to the input sentence

- $w_{-1} =$ that, $w_{-2} =$ in, $w_{-3} =$ check, ...
- $w_1 =$ and, $w_2 =$ transfer, $w_3 = 100$, ...
- $@1 =$ NO, $Cap. =$ NO, ...
- ...

when we consider a candidate label Fi for this case:

- $f_{w-1=that,Fi} = 1$, $f_{w-2=in,Fi} = 1$, $f_{w-3=check,Fi} = 1$, ...
- $f_{w1=and,Fi} = 1$, $f_{w2=transfer,Fi} = 1$, $f_{w3=100,Fi} = 1$, ...
- $f_{at1,Fi} = 0$, $f_{Cap.,Fi} = 0$, ...
- ... $\Longrightarrow [1, 1, 1, 1, 0, 0, ..., 1, 0, 0]$

try more candidate labels ... Sp, Po, En, ...

simply written as: $f(x, y)$

i.e., $f(x, Fi)$, $f(x, Sp)$, $f(x, Po)$, $f(x, En)$,...

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

- Using words themselfves as features ...
- [..., around, ..., fans, ..., game, ..., many, ..., soccer, ..., world, ...]

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

- Using words themselfves as features ...
- [..., around, ..., fans, ..., game, ..., many, ..., soccer, ..., world, ...]

when we consider a candidate label Sp for this case:

- $f_{around,Sp} = 1$, $f_{fans,Sp} = 1$, $f_{game,Sp} = 1$, $f_{Messi,Sp} = 1$, ...
- $f_{soccer,Sp} = 1$, $f_{w2=world,Sp} = 1$, $f_{cup,Sp} = 1$, ...
- $f_{Portugal,Sp} = 0$, $f_{Cristiano,Sp} = 0$, ...

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

- Using words themselfves as features ...
- [..., around, ..., fans, ..., game, ..., many, ..., soccer, ..., world, ...]

when we consider a candidate label Sp for this case:

- $f_{around,Sp} = 1$, $f_{fans,Sp} = 1$, $f_{game,Sp} = 1$, $f_{Messi,Sp} = 1$, ...
- $f_{soccer,Sp} = 1$, $f_{w2=world,Sp} = 1$, $f_{cup,Sp} = 1$, ...
- $f_{Portugal,Sp} = 0$, $f_{Cristiano,Sp} = 0$, ...
- ... $\Longrightarrow [1, 1, 1, 1, 0, 0, ..., 1, 0, 0]$

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

- Using words themselfves as features ...
- [..., around, ..., fans, ..., game, ..., many, ..., soccer, ..., world, ...]

when we consider a candidate label Sp for this case:

- $f_{around,Sp} = 1$, $f_{fans,Sp} = 1$, $f_{game,Sp} = 1$, $f_{Messi,Sp} = 1$, ...
- $f_{soccer,Sp} = 1$, $f_{w2=world,Sp} = 1$, $f_{cup,Sp} = 1$, ...
- $f_{Portugal,Sp} = 0$, $f_{Cristiano,Sp} = 0$, ...
- ... $\Longrightarrow [1, 1, 1, 1, 0, 0, ..., 1, 0, 0]$
- Or, using frequencies: $\Longrightarrow [1, 2, 1, 4, 0, 0, ..., 90, 0, 0]$

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

- Using words themselfves as features ...
- [..., around, ..., fans, ..., game, ..., many, ..., soccer, ..., world, ...]

when we consider a candidate label Sp for this case:

- $f_{around,Sp} = 1$, $f_{fans,Sp} = 1$, $f_{game,Sp} = 1$, $f_{Messi,Sp} = 1$, ...
- $f_{soccer,Sp} = 1$, $f_{w2=world,Sp} = 1$, $f_{cup,Sp} = 1$, ...
- $f_{Portugal,Sp} = 0$, $f_{Cristiano,Sp} = 0$, ...
- ... $\Longrightarrow [1, 1, 1, 1, 0, 0, ..., 1, 0, 0]$
- Or, using frequencies: $\Longrightarrow [1, 2, 1, 4, 0, 0, ..., 90, 0, 0]$
- Or, using TF-IDF weighting: $\Longrightarrow [0.4, 4.5, 1.0, 0.9, 0, 0, ..., 8.1, 0, 0]$

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

- Using words themselfves as features ...
- [..., around, ..., fans, ..., game, ..., many, ..., soccer, ..., world, ...]

when we consider a candidate label Sp for this case:

- $f_{around,Sp} = 1$, $f_{fans,Sp} = 1$, $f_{game,Sp} = 1$, $f_{Messi,Sp} = 1$, ...
- $f_{soccer,Sp} = 1$, $f_{w2=world,Sp} = 1$, $f_{cup,Sp} = 1$, ...
- $f_{Portugal,Sp} = 0$, $f_{Cristiano,Sp} = 0$, ...
- ... $\Longrightarrow [1, 1, 1, 1, 0, 0, ..., 1, 0, 0]$
- Or, using frequencies: $\Longrightarrow [1, 2, 1, 4, 0, 0, ..., 90, 0, 0]$
- Or, using TF-IDF weighting: $\Longrightarrow [0.4, 4.5, 1.0, 0.9, 0, 0, ..., 8.1, 0, 0]$

try more candidate labels ... Fi, Po, En, ...

Weighting a Term

The weighted value or importance of a word t in a document d can be considered by taking both t's term frequency and inverse document frequency:

$$TF_{t,d} = count(t, d)$$

Weighting a Term

The weighted value or importance of a word t in a document d can be considered by taking both t's term frequency and inverse document frequency:

$$TF_{t,d} = \begin{cases} 1 + \log_{10} count(t,d) & \text{if } count(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Weighting a Term

The weighted value or importance of a word t in a document d can be considered by taking both t's term frequency and inverse document frequency:

$$TF_{t,d} = \begin{cases} 1 + \log_{10} count(t,d) & \text{if } count(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$IDF_t = \log_{10} \frac{N}{DF_t}$$

Weighting a Term

The weighted value or importance of a word t in a document d can be considered by taking both t's term frequency and inverse document frequency:

$$TF_{t,d} = \begin{cases} 1 + \log_{10} count(t,d) & \text{if } count(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$IDF_t = \log_{10} \frac{N}{DF_t}$$

$$TF - IDF = TF_{t,d} \times IDF_t$$

Weighting a Term

Take an hour to research the TF-IDF weighting scheme

The weighted value or importance of a word t in a document d can be considered by taking both t's term frequency and inverse document frequency:

$$TF_{t,d} = \begin{cases} 1 + \log_{10} count(t,d) & \text{if } count(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$IDF_t = \log_{10} \frac{N}{DF_t}$$

$$TF - IDF = TF_{t,d} \times IDF_t$$

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

$\Longrightarrow$:

- $[1, 1, 1, 1, 0, 0, ..., 1, 0, 0]$, also called One-Hot vector
- Or, using frequencies: $[1, 2, 1, 4, 0, 0, ..., 90, 0, 0]$
- Or, using TF-IDF weighting: $[0.4, 4.5, 1.0, 0.9, 0, 0, ..., 8.1, 0, 0]$

Bag-of-words

The so-called `Bag-of-words` format

[by students@CMU LTI]

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

$\Longrightarrow$:

- $[1, 1, 1, 1, 0, 0, ..., 1, 0, 0]$, also called One-Hot vector
- Or, using frequencies: $[1, 2, 1, 4, 0, 0, ..., 90, 0, 0]$
- Or, using TF-IDF weighting: $[0.4, 4.5, 1.0, 0.9, 0, 0, ..., 8.1, 0, 0]$

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

$\Longrightarrow$:

- $[1, 1, 1, 1, 0, 0, ..., 1, 0, 0]$, also called One-Hot vector
- Or, using frequencies: $[1, 2, 1, 4, 0, 0, ..., 90, 0, 0]$
- Or, using TF-IDF weighting: $[0.4, 4.5, 1.0, 0.9, 0, 0, ..., 8.1, 0, 0]$

What are those with very small values or even zeros?

An Example in News Classification

For many soccer fans around the world, Lionel Messi is the most magical player they have seen play the game in their lifetime. ...

$\Longrightarrow$:

- $[1, 1, 1, 1, 0, 0, ..., 1, 0, 0]$, also called One-Hot vector
- Or, using frequencies: $[1, 2, 1, 4, 0, 0, ..., 90, 0, 0]$
- Or, using TF-IDF weighting: $[0.4, 4.5, 1.0, 0.9, 0, 0, ..., 8.1, 0, 0]$

What are those with very small values or even zeros?

- Take an hour to research the the stop-word list.

A Simple Features based Discriminative Model

Imagine a linear classifiers with the form like, $\lambda_{f(x,y)} f(x,y)$, where λs are weights,

- build a linear function to map $f(x,y)$ to label y
- possibly need a weight $\lambda_{f_i(x,y)}$ for each feature $f_i(x,y)$
- then, for each possible label y of instance x, we can compute a score:

$$score(x,y) = \sum_i \lambda_{f_i(x,y)} f_i(x,y)$$

- the classifier should choose y^*:

$$y^* = \arg\max_y \sum_i \lambda_{f_i(x,y)} f_i(x,y)$$

A Simple Features based Discriminative Model

Imagine a linear classifiers with the form like, $\lambda_{f(x,y)} f(x, y)$, where λs are weights,

- build a linear function to map $f(x, y)$ to label y
- possibly need a weight $\lambda_{f_i(x,y)}$ for each feature $f_i(x, y)$
- then, for each possible label y of instance x, we can compute a score:

$$score(x, y) = \sum_i \lambda_{f_i(x,y)} f_i(x, y)$$

- the classifier should choose y^*:

$$y^* = \arg\max_y \sum_i \lambda_{f_i(x,y)} f_i(x, y)$$

That is, for each y, compute its score, and select the y^* that gives the largest score.

A Simple Features based Discriminative Model

Imagine a linear classifiers with the form like, $\lambda_{f(x,y)} f(x,y)$, where λs are weights,

- build a linear function to map $f(x,y)$ to label y
- possibly need a weight $\lambda_{f_i(x,y)}$ for each feature $f_i(x,y)$
- then, for each possible label y of instance x, we can compute a score:

$$score(x,y) = \sum_i \lambda_{f_i(x,y)} f_i(x,y)$$

- the classifier should choose y^*:

$$y^* = \arg\max_y \sum_i \lambda_{f_i(x,y)} f_i(x,y)$$

Note that it may not be a probabilistic model.

A Simple Features based Discriminative Model

Imagine a linear classifiers with the form like, $\lambda_{f(x,y)} f(x,y)$, where λs are weights,

- build a linear function to map $f(x,y)$ to label y
- possibly need a weight $\lambda_{f_i(x,y)}$ for each feature $f_i(x,y)$
- then, for each possible label y of instance x, we can compute a score:

$$score(x,y) = \sum_i \lambda_{f_i(x,y)} f_i(x,y)$$

- the classifier should choose y^*:

$$y^* = \arg\max_y \sum_i \lambda_{f_i(x,y)} f_i(x,y)$$

How to figure out those λs?

Our Expectations

We want:

- nicely fit to our linear story
- (at least, looks like) simple
- (may be) easy to drive
- (hopefully) probabilistic
- ...

Features based Linear Models: Algorithms

The key is to choose/learn proper weights λs for different features

- the Perceptron algorithm
- Margin-based models (the Support Vector Machines, SVM)
- Exponential Models:
  - log-linear models, maximum entropy models, logistic models, ...

Features based Linear Models: Algorithms

The key is to choose/learn proper weights λs for different features

- the Perceptron algorithm (will be covered soon)
- Margin-based models (the Support Vector Machines, SVM)
- Exponential Models:
  - log-linear models, maximum entropy models, logistic models, ...

Features based Linear Models: Algorithms

The key is to choose/learn proper weights λs for different features

- the Perceptron algorithm
- Margin-based models (the Support Vector Machines, SVM)
- Exponential Models:
    - log-linear models, maximum entropy models, logistic models, ...

Features based Linear Models: Algorithms

The key is to choose/learn proper weights λs for different features

- the Perceptron algorithm
- Margin-based models (the Support Vector Machines, SVM)
- Exponential Models:
  - log-linear models, maximum entropy models, logistic models, ...
  - basically, produce a probabilistic model according to $score(x, y)$

$$p(y|x) = \frac{\exp score(x,y)}{\sum_{y'} \exp score(x,y')} = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x,y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x,y')}$$

  - numerator: positive score for label y
  - denominator: normalization over all labels

Features based Linear Models: Algorithms

The key is to choose/learn proper weights λs for different features

- the Perceptron algorithm
- Margin-based models (the Support Vector Machines, SVM)
- Exponential Models:
  - log-linear models, maximum entropy models, logistic models, ...
  - basically, produce a probabilistic model according to $score(x, y)$

$$p(y|x) = \frac{\exp score(x, y)}{\sum_{y'} \exp score(x, y')} = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x, y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x, y')}$$

  - numerator: positive score for label y
  - denominator: normalization over all labels
  - everything is positive; looks like a probabilistic interpretation!

Features based Linear Models: Algorithms

The key is to choose/learn proper weights λs for different features

- the Perceptron algorithm
- Margin-based models (the Support Vector Machines, SVM)
- Exponential Models:
  - log-linear models, maximum entropy models, logistic models, ...
  - basically, produce a probabilistic model according to $score(x, y)$

$$p(y|x) = \frac{\exp score(x, y)}{\sum_{y'} \exp score(x, y')} = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x, y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x, y')}$$

  - numerator: positive score for label y
  - denominator: normalization over all labels
  - everything is positive; looks like a probabilistic interpretation!
  - a very powerful tool!

Log-Linear Models

For a data sample (x, y);

- We care:

$$p(y|x) = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x, y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x, y')}$$

Log-Linear Models

For a data sample (x, y);

- We care:

$$p(y|x) = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x,y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x,y')}$$

- write it as:

$$p(y|x) = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x,y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x,y')} = \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x,y))}{\sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x,y'))}$$

Log-Linear Models

For a data sample (x, y);

- We care:

$$p(y|x) = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x, y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x, y')}$$

- write it as:

$$p(y|x) = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x, y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x, y')} = \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x, y))}{\sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x, y'))}$$

- and then:

$$\log p(y|x; \boldsymbol{\lambda}) = \boldsymbol{\lambda} \cdot \boldsymbol{f}(x, y) - \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x, y'))$$

Log-Linear Models

For a data sample (x, y);

- We care:

$$p(y|x) = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x, y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x, y')}$$

- write it as:

$$p(y|x) = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x, y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x, y')} = \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x, y))}{\sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x, y'))}$$

- and then:

$$\log p(y|x; \boldsymbol{\lambda}) = \boldsymbol{\lambda} \cdot \boldsymbol{f}(x, y) - \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x, y'))$$

  - linear: $\boldsymbol{\lambda} \cdot \boldsymbol{f}(x, y)$
  - Normalization: $\log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x, y'))$

Maximum-Likelihood Estimations

How likely do we observe the data given the current parameters?

- Given the training data $\{(x_1, y_1), (x_2, y_2), ..., (x_k, y_k)\}$:
  the Likelihood of $\boldsymbol{\lambda}$ is:

$$L(\boldsymbol{\lambda}) = \prod_k p(y_k|x_k; \boldsymbol{\lambda}) = \prod_k \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k))}{\sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}$$

Maximum-Likelihood Estimations

How likely do we observe the data given the current parameters?

- Given the training data $\{(x_1, y_1), (x_2, y_2), ..., (x_k, y_k)\}$:
  the Likelihood of $\boldsymbol{\lambda}$ is:

$$L(\boldsymbol{\lambda}) = \prod_k p(y_k|x_k; \boldsymbol{\lambda}) = \prod_k \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k))}{\sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}$$

- $L(\boldsymbol{\lambda})$ gets bigger, the model works better **on this dataset**

Maximum-Likelihood Estimations

How likely do we observe the data given the current parameters?

- Given the training data $\{(x_1, y_1), (x_2, y_2), ..., (x_k, y_k)\}$:
  the Likelihood of $\boldsymbol{\lambda}$ is:

$$L(\boldsymbol{\lambda}) = \prod_k p(y_k|x_k;\boldsymbol{\lambda}) = \prod_k \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k))}{\sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}$$

  After $\log$:

$$\begin{aligned} LL(\boldsymbol{\lambda}) &= \sum_k \log p(y_k|x_k;\boldsymbol{\lambda}) \\ &= \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y')) \end{aligned}$$

- $L(\boldsymbol{\lambda})$ gets bigger, the model works better **on this dataset**

"Maximum-Likelihood Estimations"

- Need to maximize:

$$LL(\boldsymbol{\lambda}) = \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

"Maximum-Likelihood Estimations"

- Need to maximize:

$$LL(\boldsymbol{\lambda}) = \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

- So, we need to calculate gradients w.r.t. each λ:

$$\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(,)}} = \sum_k f_i(x_k, y_k) - \sum_k \frac{\sum_{y'} f_i(x_k, y') \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))}$$

"Maximum-Likelihood Estimations"

- Need to maximize:

$$LL(\boldsymbol{\lambda}) = \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

- So, we need to calculate gradients w.r.t. each λ:

$$\begin{aligned}\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(,)}} &= \sum_k f_i(x_k, y_k) - \sum_k \frac{\sum_{y'} f_i(x_k, y') \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))} \\ &= \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))}\end{aligned}$$

"Maximum-Likelihood Estimations"

- Need to maximize:

$$LL(\boldsymbol{\lambda}) = \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

- So, we need to calculate gradients w.r.t. each λ:

$$\begin{aligned}\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(,)}} &= \sum_k f_i(x_k, y_k) - \sum_k \frac{\sum_{y'} f_i(x_k, y') \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))} \\ &= \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))}\end{aligned}$$

"Maximum-Likelihood Estimations"

- Need to maximize:

$$LL(\boldsymbol{\lambda}) = \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

- So, we need to calculate gradients w.r.t. each λ:

$$\begin{aligned}
\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(,)}} &= \sum_k f_i(x_k, y_k) - \sum_k \frac{\sum_{y'} f_i(x_k, y') \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))} \\
&= \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))} \\
&= \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') p(y' | x_k; \boldsymbol{\lambda})
\end{aligned}$$

"Maximum-Likelihood Estimations"

- Need to maximize:

$$LL(\boldsymbol{\lambda}) = \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

- So, we need to calculate gradients w.r.t. each λ:

$$\begin{aligned}
\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(,)}} &= \sum_k f_i(x_k, y_k) - \sum_k \frac{\sum_{y'} f_i(x_k, y') \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))} \\
&= \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))} \\
&= \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') p(y'|x_k; \boldsymbol{\lambda})
\end{aligned}$$

"Maximum-Likelihood Estimations"

- Need to maximize:

$$LL(\boldsymbol{\lambda}) = \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

- So, we need to calculate gradients w.r.t. each λ:

$$\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(,)}} = \sum_k f_i(x_k, y_k) - \sum_k \frac{\sum_{y'} f_i(x_k, y') \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))}$$

$$= \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') \frac{\exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))}{\sum_{z'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, z'))}$$

$$= \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') p(y' | x_k; \boldsymbol{\lambda})$$

  - The first component: **Empirical Counts**
  - The second component: **Expected Counts**

Gradient Ascend Methods

- Need to maximize $LL(\boldsymbol{\lambda})$ where

$$\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(x,y)}} = \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') p(y'|x_k; \boldsymbol{\lambda})$$

- Initialize all λs to be 0
- Iterate until convergence
  - Calculate $\Delta = \frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda}$
  - Calculate $\beta_* = \arg\max_\beta LL(\boldsymbol{\lambda} + \beta \cdot \Delta)$
  - Set $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \beta_* \cdot \Delta$

Gradient Ascend Methods

- Need to maximize $LL(\boldsymbol{\lambda})$ where

$$\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(x,y)}} = \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') p(y'|x_k; \boldsymbol{\lambda})$$

- Initialize all λs to be 0
- Iterate until convergence
  - Calculate $\Delta = \frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda}$
  - Calculate $\beta_* = \arg\max_\beta LL(\boldsymbol{\lambda} + \beta \cdot \Delta)$ (search along Δ)
  - Set $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \beta_* \cdot \Delta$

Gradient Ascend Methods

- Need to maximize $LL(\boldsymbol{\lambda})$ where

$$\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(x,y)}} = \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') p(y'|x_k; \boldsymbol{\lambda})$$

- Initialize all λs to be 0
- Iterate until convergence
  - Calculate $\Delta = \frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda}$
  - Calculate $\beta_* = \arg\max_\beta LL(\boldsymbol{\lambda} + \beta \cdot \Delta)$ (search along Δ)
  - Set $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \beta_* \cdot \Delta$
- Slow!
  - Optimizations available : Conjugate Gradient Methods
  - or, Stochastic Gradient Methods

Stochastic Gradient Ascend

- Need to maximize $LL(\boldsymbol{\lambda})$ where

$$\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(x,y)}} = \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') p(y'|x_k; \boldsymbol{\lambda})$$

- Initialize all λs, number of epochs T, learning rate α
- For $t \in \{1, ..., T\}$:
  - choose a random permutation π of $\{1, 2, ..., k, ...\}$ (the whole dataset)
  - for $i \in \{1, 2, ..., k, ...\}$:
    - calculate $\Delta = \frac{\partial LL(\boldsymbol{\lambda})_{\pi(i)}}{\partial \lambda}$
    - set $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \alpha \cdot \Delta$
- output $\boldsymbol{\lambda}$

More to Mention

Just anther way of talking this:

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} LL(\boldsymbol{\lambda})$$

More to Mention

Just anther way of talking this:

$$\begin{aligned}\boldsymbol{\lambda}^* &= \arg\max_{\boldsymbol{\lambda}} LL(\boldsymbol{\lambda}) \\ &= \arg\max_{\boldsymbol{\lambda}} \sum_k \log p(y_k|x_k;\boldsymbol{\lambda})\end{aligned}$$

More to Mention

Just anther way of talking this:

$$\begin{aligned}\boldsymbol{\lambda}^* &= \arg\max_{\boldsymbol{\lambda}} LL(\boldsymbol{\lambda}) \\ &= \arg\max_{\boldsymbol{\lambda}} \sum_k \log p(y_k|x_k;\boldsymbol{\lambda}) \\ &= \arg\min_{\boldsymbol{\lambda}} \sum_k -\log p(y_k|x_k;\boldsymbol{\lambda})\end{aligned}$$

More to Mention

Just anther way of talking this:

$$\begin{aligned}\boldsymbol{\lambda}^* &= \arg\max_{\boldsymbol{\lambda}} LL(\boldsymbol{\lambda}) \\ &= \arg\max_{\boldsymbol{\lambda}} \sum_k \log p(y_k|x_k;\boldsymbol{\lambda}) \\ &= \arg\min_{\boldsymbol{\lambda}} \sum_k -\log p(y_k|x_k;\boldsymbol{\lambda})\end{aligned}$$

- the *log loss*
- the *cross entropy loss*

More to Mention

Just anther way of talking this:

$$\begin{aligned}\boldsymbol{\lambda}^* &= \arg\max_{\boldsymbol{\lambda}} LL(\boldsymbol{\lambda}) \\ &= \arg\max_{\boldsymbol{\lambda}} \sum_k \log p(y_k|x_k;\boldsymbol{\lambda}) \\ &= \arg\min_{\boldsymbol{\lambda}} \sum_k -\log p(y_k|x_k;\boldsymbol{\lambda})\end{aligned}$$

- the *log loss*
- the *cross entropy loss*
- then, use Stochastic Gradient **Descend** instead of *Ascend* in the last slide

More to Mention

Just anther way of talking this:

$$\begin{aligned}\boldsymbol{\lambda}^* &= \arg\max_{\boldsymbol{\lambda}} LL(\boldsymbol{\lambda}) \\ &= \arg\max_{\boldsymbol{\lambda}} \sum_k \log p(y_k|x_k;\boldsymbol{\lambda}) \\ &= \arg\min_{\boldsymbol{\lambda}} \sum_k -\log p(y_k|x_k;\boldsymbol{\lambda})\end{aligned}$$

- the *log loss*
- the *cross entropy loss*
- then, use Stochastic Gradient **Descend** instead of *Ascend* in the last slide
- all seem exciting terms :-)

Anything More?

- How do we get the probablistic thing?
- Do we have other choices?
- Can we do something else?

Anything More?

- How do we get the probablistic thing?
- Do we have other choices?
- Can we do something else?

$$LL(\boldsymbol{\lambda}) = \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

Anything More?

- How do we get the probablistic thing?
- Do we have other choices?
- Can we do something else?

$$LL(\boldsymbol{\lambda}) = \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

Anything More?

- How do we get the probablistic thing?
- Do we have other choices?
- Can we do something else?

$$LL(\boldsymbol{\lambda}) = \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

or

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}} - \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) + \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y'))$$

Is It Done?

It looks we have everything ready to build a model!

- Say, we have a feature f_1, defined as:

$$f_1(x, y) = \begin{cases} 1 & \text{if } x \text{ contains } \textbf{NFL} \text{ and } y = \textbf{ Sports}, \\ 0 & \text{otherwise} \end{cases}$$

- In our training data, **NFL** is seen 100 times, with **Sports** every time
- Our MLE should satisfy:

$$\sum_k f_1(x_k, y_k) = \sum_k \sum_y p(y|x_k; \boldsymbol{\lambda}) f_1(x_k, y)$$

Is It Done?

It looks we have everything ready to build a model!

- Say, we have a feature f_1, defined as:

$$f_1(x,y) = \begin{cases} 1 & \text{if } x \text{ contains } \textbf{NFL} \text{ and } y = \textbf{ Sports}, \\ 0 & \text{otherwise} \end{cases}$$

- In our training data, **NFL** is seen 100 times, with **Sports** every time
- Our MLE should satisfy:

$$\underbrace{\sum_k f_1(x_k, y_k)}_{\text{Empirical Counts}} = \underbrace{\sum_k \sum_y p(y|x_k; \boldsymbol{\lambda}) f_1(x_k, y)}_{\text{Expected Counts}}$$

Is It Done?

It looks we have everything ready to build a model!

- Say, we have a feature f_1, defined as:

$$f_1(x, y) = \begin{cases} 1 & \text{if } x \text{ contains } \textbf{NFL} \text{ and } y = \textbf{ Sports}, \\ 0 & \text{otherwise} \end{cases}$$

- In our training data, **NFL** is seen 100 times, with **Sports** every time
- Our MLE should satisfy:

$$\sum_k f_1(x_k, y_k) = \sum_k \sum_y p(y|x_k; \boldsymbol{\lambda}) f_1(x_k, y)$$

  - $p(\textbf{Sports}|x_k; \boldsymbol{\lambda}) = 1$ for <u>any training sample</u> x_k where **NFL**$\in x_k$

Is It Done?

It looks we have everything ready to build a model!

- Say, we have a feature f_1, defined as:

$$f_1(x, y) = \begin{cases} 1 & \text{if } x \text{ contains } \textbf{NFL} \text{ and } y = \textbf{ Sports}, \\ 0 & \text{otherwise} \end{cases}$$

- In our training data, **NFL** is seen 100 times, with **Sports** every time
- Our MLE should satisfy:

$$\sum_k f_1(x_k, y_k) = \sum_k \sum_y p(y|x_k; \boldsymbol{\lambda}) f_1(x_k, y)$$

  - $p(\textbf{Sports}|x_k; \boldsymbol{\lambda}) = 1$ for <u>any training sample</u> x_k where **NFL**$\in x_k$
  - $\lambda_1 \to \infty$ for maximum-likelihood solutions (most likely)

Is It Done?

It looks we have everything ready to build a model!

- Say, we have a feature f_1, defined as:

$$f_1(x,y) = \begin{cases} 1 & \text{if } x \text{ contains } \textbf{NFL} \text{ and } y = \textbf{ Sports}, \\ 0 & \text{otherwise} \end{cases}$$

- In our training data, **NFL** is seen 100 times, with **Sports** every time
- Our MLE should satisfy:

$$\sum_k f_1(x_k, y_k) = \sum_k \sum_y p(y|x_k; \boldsymbol{\lambda}) f_1(x_k, y)$$

  - $p(\textbf{Sports}|x_k; \boldsymbol{\lambda}) = 1$ for any training sample x_k where **NFL**$\in x_k$
  - $\lambda_1 \to \infty$ for maximum-likelihood solutions (most likely)
  - $p(\textbf{Sports}|x_k; \boldsymbol{\lambda}) = 1$ for any test sample x where **NFL**$\in x$

Is It Done?

It looks we have everything ready to build a model!

- Say, we have a feature f_1, defined as:

$$f_1(x, y) = \begin{cases} 1 & \text{if } x \text{ contains } \textbf{NFL} \text{ and } y = \textbf{ Sports}, \\ 0 & \text{otherwise} \end{cases}$$

- In our training data, **NFL** is seen 100 times, with **Sports** every time
- Our MLE should satisfy:

$$\sum_k f_1(x_k, y_k) = \sum_k \sum_y p(y|x_k; \boldsymbol{\lambda}) f_1(x_k, y)$$

  - $p(\textbf{Sports}|x_k; \boldsymbol{\lambda}) = 1$ for any training sample x_k where **NFL**$\in x_k$
  - $\lambda_1 \to \infty$ for maximum-likelihood solutions (most likely)
  - $p(\textbf{Sports}|x_k; \boldsymbol{\lambda}) = 1$ for ANY test sample x where **NFL**$\in x$

Regularization

What can we do to prevent such cases?

Regularization

What can we do to prevent such cases?

$\rightarrow$ Regularize the learning process

Regularization

What can we do to prevent such cases?

$\rightarrow$ Regularize the learning process

- Modified the loss function

$$
\begin{aligned}
LL(\boldsymbol{\lambda}) &= \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y')) - \frac{\alpha}{2} ||\boldsymbol{\lambda}||^2 \\
&= \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y')) - \frac{\alpha}{2} \sum_i \lambda_i^2
\end{aligned}
$$

- When calculating the gradients

$$
\frac{\partial LL(\boldsymbol{\lambda})}{\partial \lambda_{f_i(x,y)}} = \sum_k f_i(x_k, y_k) - \sum_k \sum_{y'} f_i(x_k, y') p(y'|x_k; \boldsymbol{\lambda}) - \alpha \lambda_{f_i(x,y)}
$$

- Adds a penalty for large weights

Regularization

Prevent every parameter from becoming too large in magnitude.

$$\arg\min_{\boldsymbol{\lambda}} loss(\boldsymbol{\lambda}) + \alpha||\boldsymbol{\lambda}||_p$$

where $\alpha > 0$, p could choose from 1, 2 or others.

Regularization

Prevent every parameter from becoming too large in magnitude.

$$\arg\min_{\boldsymbol{\lambda}} loss(\boldsymbol{\lambda}) + \alpha||\boldsymbol{\lambda}||_p$$

where $\alpha > 0$, p could choose from 1, 2 or others.
Gives us:

- L-2 Regularization

Regularization

Prevent every parameter from becoming too large in magnitude.

$$\arg\min_{\boldsymbol{\lambda}} loss(\boldsymbol{\lambda}) + \alpha||\boldsymbol{\lambda}||_p$$

where $\alpha > 0$, p could choose from 1, 2 or others.
Gives us:

- L-2 Regularization
- L-1 Regularization
  - Sparsity! (parameters)
  - but ......

Reasons for Log-linear Models

You will find it useful in the (near) future...

- You love feature engineering;
- You hate feature engineering

In Short

- A strong classifier you would be more familiar in the future
  - connection to many dominant classifiers
- Take care of features for your tasks
  - feature selection
  - biased feature selection
  - purposely feature selection
- Take care of the learning process and also evaluation protocal.
  - regularization
  - randomness

Reading

- M. Collins, Notes on Log-Linear Models (http://www.cs.columbia.edu/ mcollins/loglinear.pdf)
- Lecture Note, Linear Regression and Gradient Ascent, CS109@Stanford, (https://web.stanford.edu/class/archive/cs/cs109/cs109.1208/lectures/2
- Griffiths, T. L., and Steyvers, M. (2004). Finding scientific topics. Proceedings of the National Academy of Sciences, 101, 5228-5235

Book Chapter 5, Dan's book

Further Reading

- David M. Blei, Andrew Y. Ng and Michael I. Jordan. Latent Dirichlet Allocation, Journal of Machine Learning Research 3 (2003) 993-1022
- Zhang and Oles (2010), Text Categorization Based on Regularized Linear Classification Methods (http://www.stat.yale.edu/lc436/papers/temp/Zhang Oles 2001.pdf)
- Lecture Notes 1, Discriminative Algorithms, CS229-Machine Learning@Stanford, Andrew Ng (http://cs229.stanford.edu/materials.html)
- Adam Berger, Stephen Della Pietra, and Vincent Della Pietra, A maximum entropy approach to natural language processing. Computational Linguistics, 22(1):39-71, 1996.
- Galen Andrew and Jianfeng Gao, Scalable training of L1-regularized log-linear models. In Proc. of ICML, 2007.