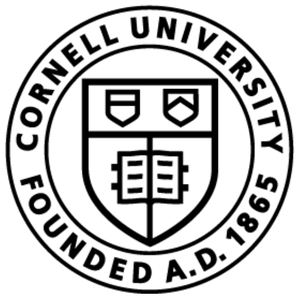
Lecture 6: Text Classification



Cornell Bowers CIS
Computer Science

Claire Cardie, Tanya Goyal

CS 4740 (and crosslists): Introduction to Natural Language Processing

Reminders

- **HW1 milestone due on 12 February, 11.59 p.m.**
- HW1 due on 21 February, 11.59 p.m.

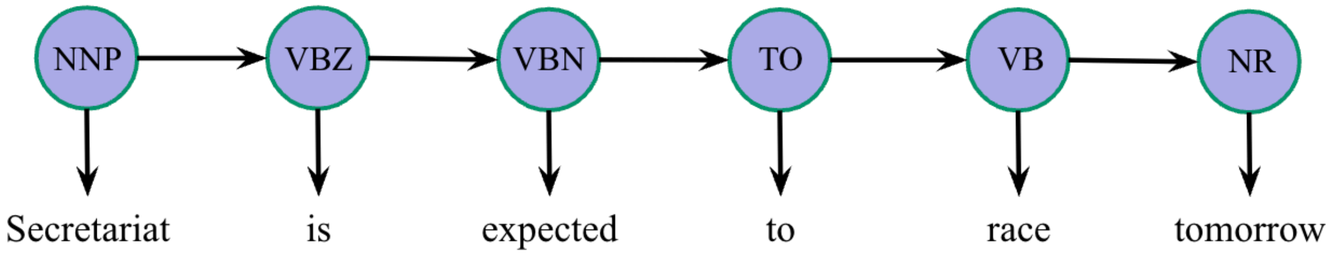
Today

- HMMs vs MEMMs
- Text Classification
  - Logistic Regression

MEMMs recap

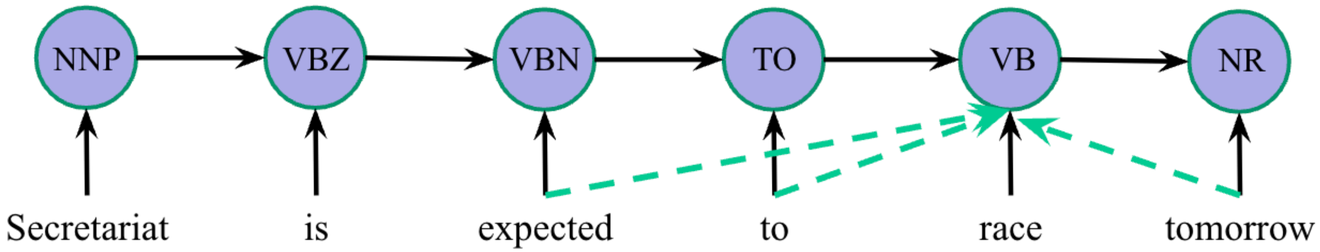
- HMMs:

$$\arg\max_{t_1 \ldots t_N} P(t_1 \ldots t_N \,|\, o_{1:N}) = \arg\max_{t_1 \ldots t_N} \prod_i P(o_i \,|\, t_i) \times P(t_i \,|\, t_{i-1})$$



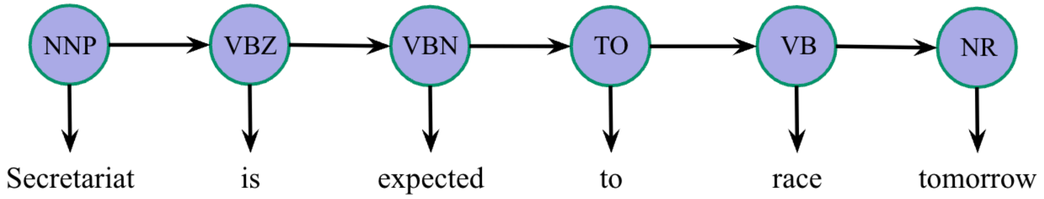
- MEMMs (Max Entropy Markov Models) assumptions:
  - Tag is independent of all other tags *except* the previous one.
  - But it can depend on the entire observation!

$$\arg\max_{t_1 \ldots t_N} P(t_1 \ldots t_N \,|\, o_{1:N}) = \arg\max_{t_1 \ldots t_N} \prod_i P_{\text{MEMM}}(t_i \,|\, t_{i-1}, o_{1:N})$$

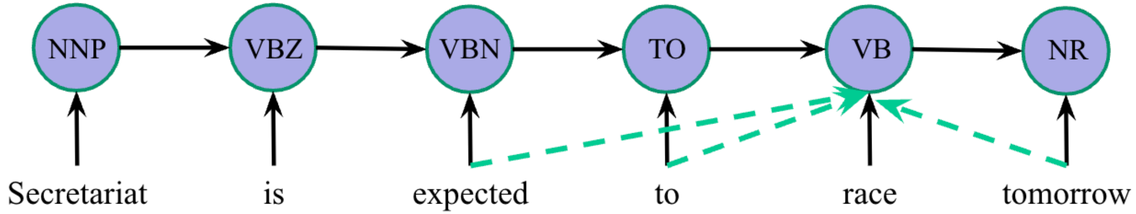


MEMMs advantages over HMMs

HMMs



MEMMs



- **Can condition on the whole input.**

I visited Jordan to see the Red Sea.

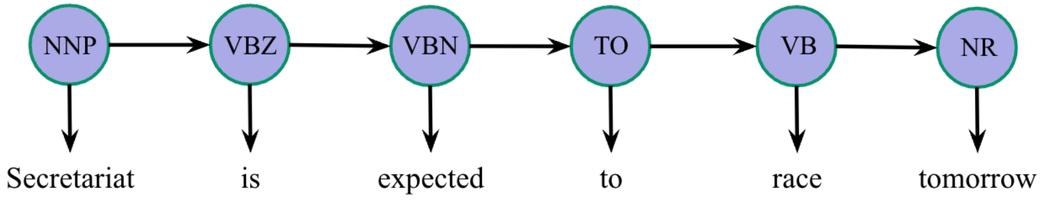
LOC

I met Jordan for an autograph.

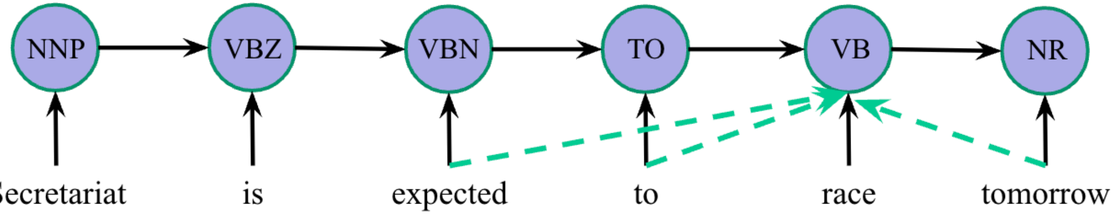
PER

MEMMs advantages over HMMs

HMMs



MEMMs



- **Can use features!**

The plot of the movie was discombobulating.

[word unseen in the training data]

The plot of the movie was [UNK].

Gives no clues about POS tag.

$$P_{\text{MEMM}}(t_i \mid t_{i-1}, o_{1:N})$$

Extract features from these.

$$P_{\text{MEMM}}(t_i \mid t_{i-1}, f_1 = \text{current word ends in "ing"}, f_2 = \text{current word is 13 chars}, \ldots)$$

Features

- f_1 = Does the token end in "ing" –> Likely a verb."
- f_2 = Length of the token –> Probably not a preposition."

Q: What features would be helpful for:

1. **Proper nouns?**
2. **Determiner?**

- For a possible tag, some "*features*" of a token raise the chance of that tag and some lower it.
- How do we compute $P_{\text{MEMM}}(t_i \mid t_{i-1}, o_{1:N})$?

Value of the "feature"

... multiplied by …

a weight indicating how much positive/negative evidence the presence of that feature gives to the tag.

Formalizing features and evidence weights

- f_1 = Does the token end in "ing" –> Likely a verb."
- f_2 = Length of the token –> Probably not a preposition."

The weight of this feature is **3** for tag VERB.	The weight of this feature is **0** for VERB
The weight of this feature is **-1** for PREP.	The weight of this feature is **-2** for PREP

- How do we compute $P_{\text{MEMM}}(t_i \mid t_{i-1}, o_{1:N})$?
  - For given tag t_i and your fixed collection of $\{f_k\}$ and $\{w_k^{t_i}\}$ of feature functions and weights for **that tag**.
  - $$P_{\text{MEMM}}(t_i \mid t_{i-1}, o_{1:N}) = \frac{\exp(\sum_k w_k^{t_i} . f_k(t_i, t_{i-1}, o_{1:N}, i))}{Z}$$

Formalizing features and evidence weights

- f_1 = Does the token end in "ing"
- f_2 = Length of the token

Weights for VERB $= [w_1^{VERB}, w_2^{VERB}] = [3,0]$,

Weights for PREP $= [w_1^{PREP}, w_2^{PREP}] = [-1,-2]$

$o_{1:N}$ = I am sitting in

$$P_{\text{MEMM}}(t_i \mid t_{i-1}, o_{1:N}) = \frac{\exp(\sum_k w_k^{t_i} . f_k(t_i, t_{i-1}, o_{1:N}, i))}{Z}$$

Step1: Extract features

$f_1 = 1$

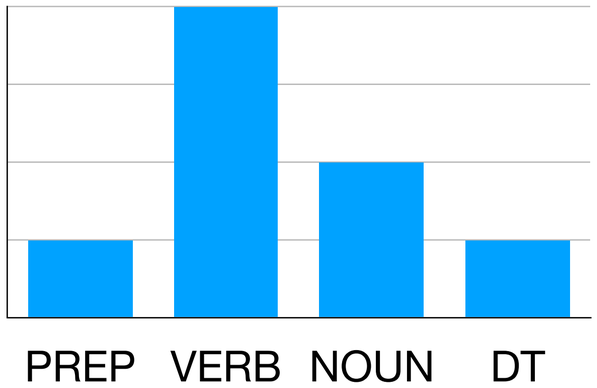
$f_2 = 7$

Step2: Compute Exponentials

$P_{\text{MEMM}}(t_3 = VERB \mid .) = \exp(3 \times 1 + 0 \times 7)/Z$

$P_{\text{MEMM}}(t_3 = PREP \mid .) = \exp((-1) \times 1 + (-2) \times 7)/Z$

Step3: Get Probability Distribution over tags



HMMs vs MEMMs as taggers via Viterbi

- HMMs: $v_i(t) = \max_{\text{possible prev}_t} P_{\text{HMM}}(t \mid \text{prev}_t)\, P_{\text{HMM}}(o_i \mid t)\, v_{i-1}(\text{prev}_t)$

No emission/transition matrices

- MEMMs: $v_i(t) = \max_{\text{possible prev}_t} P_{\text{MEMMs}}(t \mid \text{prev}_t, o_{1:N})\, v_{i-1}(\text{prev}_t)$

Multinomial Logistic Regression Model

- $P(t_i \mid t_{i-1}, o_{1:N}) = \dfrac{\exp(\sum_k w_k^{t_i} \cdot f_k(t_i, t_{i-1}, o_{1:N}, i))}{Z}$

How do we learn these weights?

- Let's work with an easier classifier in class: ***Binary*** *Logistic Regression Model*

Text Classification

- **Formally,**
  - input: text **x**
  - output: a label **y** (from a finite set)
  - goal: learn a mapping function $P(\mathbf{y} \mid \mathbf{x})$

In our tagging example,
Y = {VERB, PREP, NOUN, …}

Task	Input **x**	Output **y**
Sentiment Analysis	"The movie was great" "The actor is great, movie is dull"	{positive, negative}
Spam / Not spam	"Win $10Million" "CS4740 announcement"	{spam, not-spam}

Binary Logistic Regression Model

- **Formally,**
  - input: text **x**
  - output: a label **y** (from a finite set)
  - goal: learn a mapping function $P(\mathbf{y} \mid \mathbf{x})$

Define $z = \sum_{i=1}^{|f|} w_i f_i$

$$P(\mathbf{y} = 1 \mid \mathbf{x}) = \frac{e^z}{1 + e^z}$$

$$P(\mathbf{y} = 0 \mid \mathbf{x}) = \frac{1}{1 + e^z}$$

Self-study: why is this equivalent to the multinomial case?

Binary Logistic Regression Model

- **Formally,**
  - input: text **x**
  - output: a label **y** (from a finite set)
  - goal: ~~learn a mapping function $P(\mathbf{y} \mid \mathbf{x})$~~ → *learn weights w_i*

Define $z = \sum_{i=1}^{|f|} w_i f_i$

$$P(\mathbf{y} = 1 \mid \mathbf{x}) = \frac{e^z}{1 + e^z}$$

$$P(\mathbf{y} = 0 \mid \mathbf{x}) = \frac{1}{1 + e^z}$$

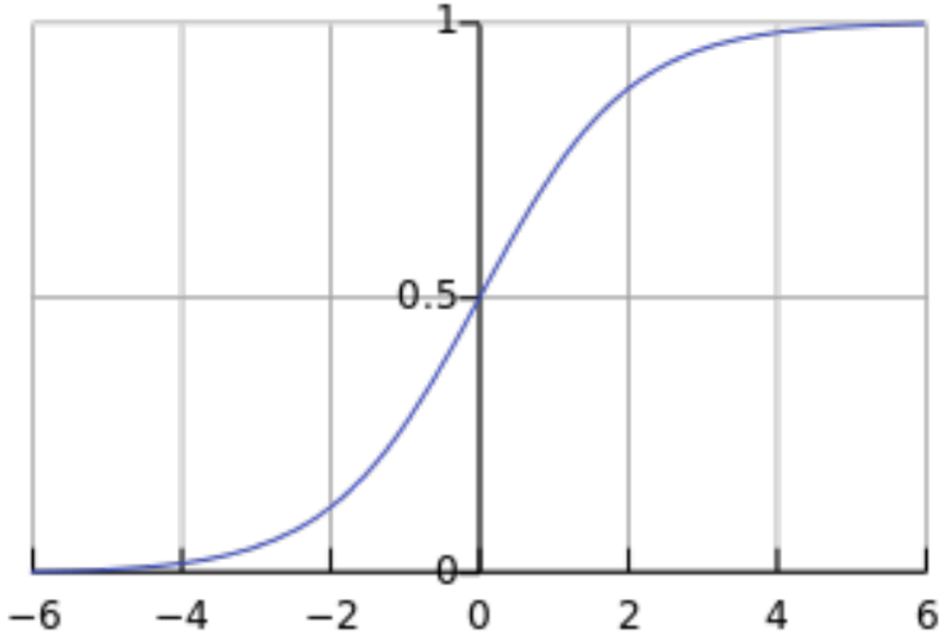
Properties of Logistic Function

$$z = \sum_{i=1}^{|f|} w_i f_i$$

$$P(\mathbf{y} = 1 \,|\, \mathbf{x}) = \frac{e^z}{1 + e^z}$$

$$P(\mathbf{y} = 0 \,|\, \mathbf{x}) = \frac{1}{1 + e^z}$$

- Logistic function: $\sigma(z) = \dfrac{e^z}{1 + e^z} = \dfrac{1}{1 + e^{-z}}$
- $\sigma(z) : \mathbb{R} \rightarrow [0,1]$



- $P(\mathbf{y} = 1 \,|\, \mathbf{x}) = \sigma(z) = \dfrac{1}{2}$ when $z = 0$.

Binary Logistic Regression Model

Sentiment Analysis

x = "The movie was great"

y = 1

Step1: Extract Features

$$f = \begin{pmatrix} f_0 = 1 \quad f_1 = \#\text{words} \quad f_2 = \#\text{"great"} \\ f_3 = \#\text{ positive words (from a pre-defined lexicon of positive words)} \\ f_4 = \#\text{ negative words (from a pre-defined lexicon of negative words)} \\ f_5 = \#\text{ adjectives} \quad f_6 = \#\text{ "not"} \\ f_7 = \#\text{ "not" before a +ve word} \\ \ldots. \end{pmatrix}$$

$$f = < 1,\ 4,\ 1,\ 1,\ 0 >$$

Binary Logistic Regression Model

Sentiment Analysis

x = "The movie was great"

y = 1

Assume we have learnt the weights of the logistic regression model.

Step2: Dot product w. weights

$$f = < 1,\ 4,\ 1,\ 1,\ 0 >$$

$$w = < 2, -0.5,\ 2,\ 1, -2 >$$

$$z = \sum_i f_i w_i = 3$$

Step3: Compute Probabilities

$$P(\mathbf{y} = 1 \mid \mathbf{x}) = \sigma(3) = 0.95$$

$$P(\mathbf{y} = 0 \mid \mathbf{x}) = 1 - \sigma(3) = 0.05$$

Binary Logistic Regression M

Sentiment Analysis

x = "The movie was okay"

$f_0 = 1$

$f_1 = $ #words

$f_2 = $ #"great"

$f_3 = $ # positive words

$f_4 = $ # negative words

Assume we have learnt the weights of the logistic regression model.

Step2: Dot product w. weights

Step3: Compute Probabilities

$f = $??

$w = < 2, -0.5, \ 2, \ 1, -2 >$

$z = \sum_i f_i w_i = $??

$P(\mathbf{y} = 1 \mid \mathbf{x}) = $??

$P(\mathbf{y} = 0 \mid \mathbf{x}) = $??

Binary Logistic Regression M

Sentiment Analysis

x = "The movie was okay"

$f_0 = 1$

$f_1 = $ #words

$f_2 = $ #"great"

$f_3 = $ # positive words

$f_4 = $ # negative words

Assume we have learnt the weights of the logistic regression model.

Step2: Dot product w. weights

Step3: Compute Probabilities

$$f = < 1,\ 4,\ 0,\ 0,\ 0 >$$

$$w = < 2, -0.5,\ 2,\ 1, -2 >$$

$$z = \sum_i f_i w_i = \text{??}$$

$$P(\mathbf{y} = 1 \mid \mathbf{x}) = \text{??}$$

$$P(\mathbf{y} = 0 \mid \mathbf{x}) = \text{??}$$

Binary Logistic Regression M

$f_0 = 1$

$f_1 =$ #words

$f_2 =$ #"great"

$f_3 =$ # positive words

$f_4 =$ # negative words

Sentiment Analysis

x = "The movie was okay"

Assume we have learnt the weights of the logistic regression model.

Step2: Dot product w. weights

Step3: Compute Probabilities

$$f = < 1,\ 4,\ 0,\ 0,\ 0 >$$

$$w = < 2, -0.5,\ 2,\ 1, -2 >$$

$$z = \sum_i f_i w_i = 0$$

$$P(\mathbf{y} = 1 \mid \mathbf{x}) = 0.5$$

$$P(\mathbf{y} = 0 \mid \mathbf{x}) = 0.5$$

Learning Weights

But how do we learn the weights!!

- **Given,**
  - dataset with (**x**, **y**) pairs. → dataset with ($< f_1, f_2, \ldots f_N >$, **y**) pairs.

Learning Weights

But how do we learn the weights!!

- **Given,**

$(x^1 = <1, 2, 1, -1, 3>, y^1 = 1)$

$(x^2 = <1, -3, -2, -1, 4>, y^2 = 0)$

$(x^3 = <1, -2, 0, -1, 3>, y^3 = 1)$

$$w^{\text{MLE}} = \arg\max_{w} \prod_{i=1}^{N} P(y = y^i \mid x^i ; w)$$

Let's try to learn a w that maximizes the probability of the entire dataset – maximum likelihood estimation

Learning Weights

But how do we learn the weights!!

- **Given,**

  $(x^1 = <1, 2, 1, -1, 3>, y^1 = 1)$

  $(x^2 = <1, -3, -2, -1, 4>, y^2 = 0)$

  $(x^3 = <1, -2, 0, -1, 3>, y^3 = 1)$

$$w^{\text{MLE}} = \arg\max_{w} \prod_{j=1}^{N} P(y = y^j \mid x^j ; w)$$

Log space.

$$w^{\text{MLE}} = \arg\max_{w} \sum_{j=1}^{N} \log P(y^j \mid x^j; w)$$

Learning Weights

But how do we learn the weights!!

Negative Log Likelihood

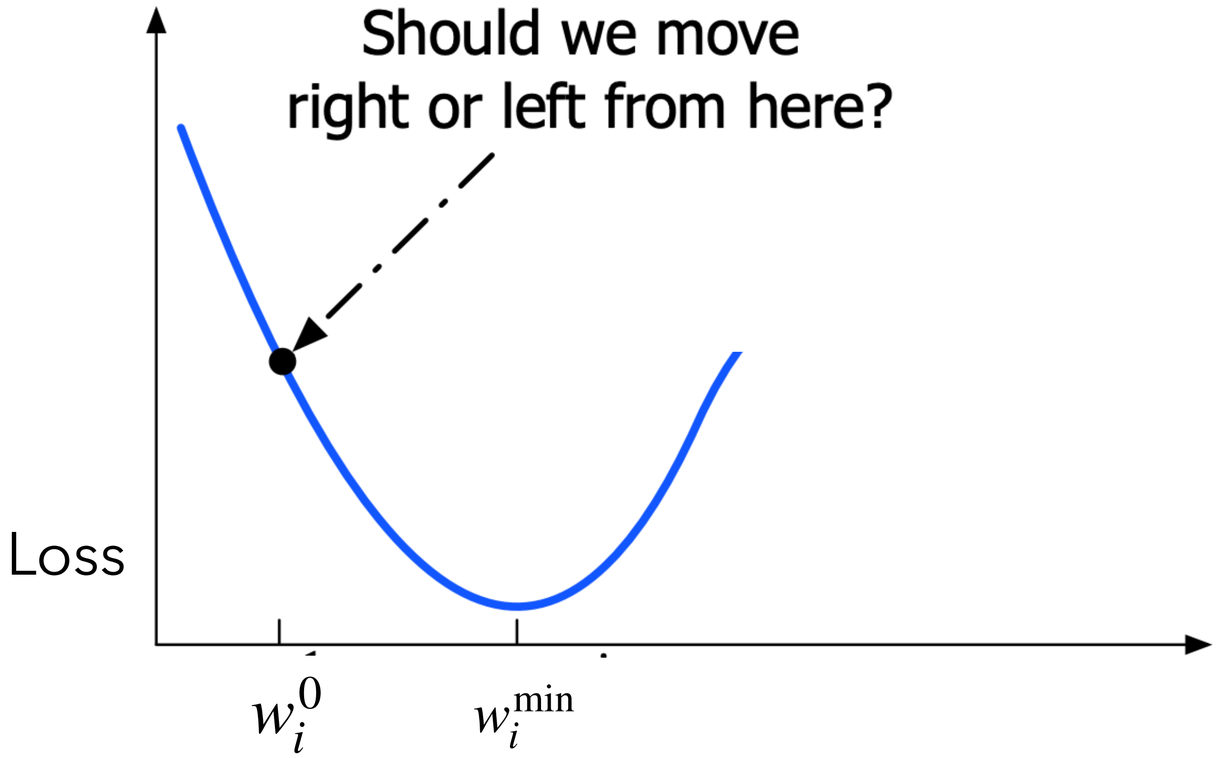
$$w^{\text{MLE}} = \arg\max_{w} \sum_{j=1}^{N} \log P(y^j \mid x^j; w) = w^{\text{MLE}} = \arg\min_{w} \sum_{j=1}^{N} \underbrace{-\log P(y^j \mid x^j; w)}_{\text{Log Loss } L^j}$$

- We can learn **w** using stochastic gradient descent (SGD).

Learning Weights

- Logistic regression loss function is convex → one minimum.

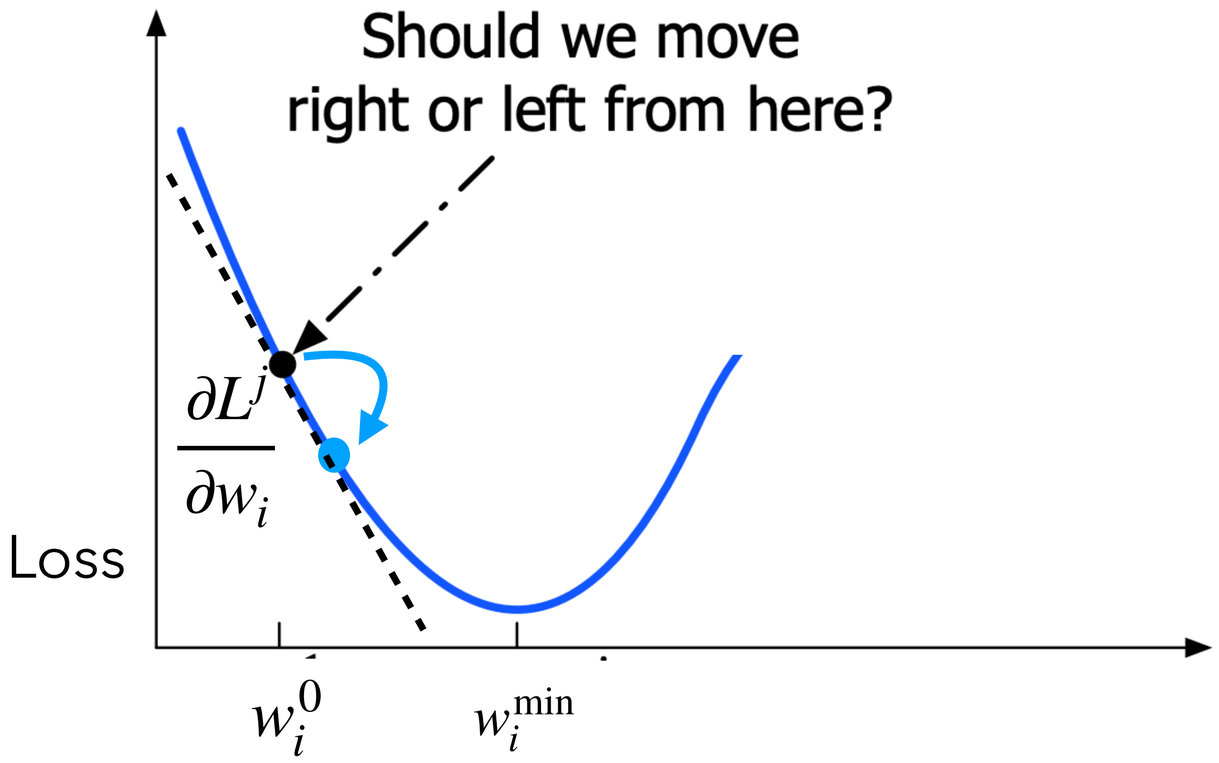
Visualizing one dim $\mathbf{w_i}$



Learning Weights

- Logistic regression loss function is convex → one minimum.

Visualizing one dim w_i



Slope is negative. → Update should move w_i^0 in the positive direction.

$$w_i^{t+1} = w_i^t - \alpha \frac{\partial L(y^j, x^j, w_i^t)}{\partial w_i}$$

Learning Weights

Negative Log Likelihood

$$w^{\text{MLE}} = \arg\min_{w} \sum_{i=0}^{N} -\log P(y_i \mid x_i; w)$$

- Initialize w^0
- $\dfrac{\partial L^j}{\partial w_i} = \dfrac{\partial}{\partial w_i} -\log P(y = y^j \mid x^j)$

Assume $y^j = 1$

$$= \frac{\partial}{\partial w_i} -\log \left[\frac{e^{\sum w_i f_i^j}}{1 + e^{\sum w_i f_i^j}} \right]$$

Assume $y^j = 0$

$$= \frac{\partial}{\partial w_i} -\log \left[\frac{1}{1 + e^{\sum w_i f_i}} \right]$$

Self-study: derive this!
hw0 had hints!

Predicted $P(y^j = 1 \mid x^j)$

True y^j

$$\frac{\partial L^j}{\partial w_i} = f_i^j \left[\sigma\left(\sum_i w_i f_i^j \right) - y^j \right]$$

Learning Weights

Negative Log Likelihood

$$w^{\text{MLE}} = \arg\min_{w} \sum_{i=0}^{N} -\log P(y_i \mid x_i; w)$$

- Initialize w^0
- $\dfrac{\partial L^j}{\partial w_i} = \dfrac{\partial}{\partial w_i} -\log P(y = y^j \mid x^j)$

Predicted $P(y^j = 1 \mid x^j)$ — True y^j

$$\frac{\partial L^j}{\partial w_i} = f_i^j \left[\sigma\left(\sum_i w_i f_i^j \right) - y^j \right]$$

If predicted probability is close to 1, and true label is $y^j = 1$, we make a smaller update!

- Update $w_i = w_i - \alpha \,.\, \dfrac{\partial L^j}{\partial w_i}$

Logistic Regression: Takeaways

- Feature engineering is important!
- Learn feature weights **w** by maximizing the log likelihood / minimizing the negative log likelihood of the training dataset.
- Lots of python libraries to train a logistic model (e.g. scikit-learn)

Slide Acknowledgements

- Earlier versions of this course offerings including materials from Claire Cardie, Marten van Schijndel, Lillian Lee.