COS 484: Natural Language Processing

Log-linear models

Fall 2019

Announcements

- Assignment 2 will be available soon
  - **due Monday, Oct 7, 11:59pm**
  - Start early!
- All assignments due **Mondays before lectures**

Last time

- Supervised classification:
  - Document to classify, d
  - Set of classes, $C = \{c_1, c_2, \ldots, c_k\}$
- Naive Bayes:

$$\hat{c} = \arg\max_{c} P(c)\, P(d \mid c)$$

Logistic Regression

- Powerful supervised model
- Baseline approach to most NLP tasks
- Connections with neural networks
- Binary (two classes) or multinomial (>2 classes)

Discriminative Model

- Logistic Regression is a *discriminative* model
- Naive Bayes is a *generative* model

Discriminative Model

- Logistic Regression: $\hat{c} = \underset{c}{\mathrm{argmax}}\ P(c \mid d)$
- Naive Bayes: $\hat{c} = \underset{c}{\mathrm{argmax}}\ P(c)\, P(d \mid c)$

Using Logistic Regression

Using Logistic Regression

- Inputs:

Using Logistic Regression

- Inputs:

  1. Classification instance in a **feature representation** $[x_1, x_2, \ldots, x_d]$

Using Logistic Regression

- Inputs:

1. Classification instance in a **feature representation** $[x_1, x_2, \ldots, x_d]$

2. **Classification function** to compute $\hat{y}$ using $P(\hat{y} \mid x)$

Using Logistic Regression

- Inputs:

1. Classification instance in a **feature representation** $[x_1, x_2, ..., x_d]$

2. **Classification function** to compute $\hat{y}$ using $P(\hat{y} | x)$

3. **Loss function** (for learning)

Using Logistic Regression

- Inputs:

1. Classification instance in a **feature representation** $[x_1, x_2, \ldots, x_d]$

2. **Classification function** to compute $\hat{y}$ using $P(\hat{y} \mid x)$

3. **Loss function** (for learning)

4. Optimization **algorithm**

Using Logistic Regression

- Inputs:

  1. Classification instance in a **feature representation** $[x_1, x_2, \ldots, x_d]$

  2. **Classification function** to compute $\hat{y}$ using $P(\hat{y} \mid x)$

  3. **Loss function** (for learning)

  4. Optimization **algorithm**

- Train phase: Learn the **parameters** of the model to minimize **loss function**

Using Logistic Regression

- Inputs:

  1. Classification instance in a **feature representation** $[x_1, x_2, \ldots, x_d]$

  2. **Classification function** to compute $\hat{y}$ using $P(\hat{y} \mid x)$

  3. **Loss function** (for learning)

  4. Optimization **algorithm**

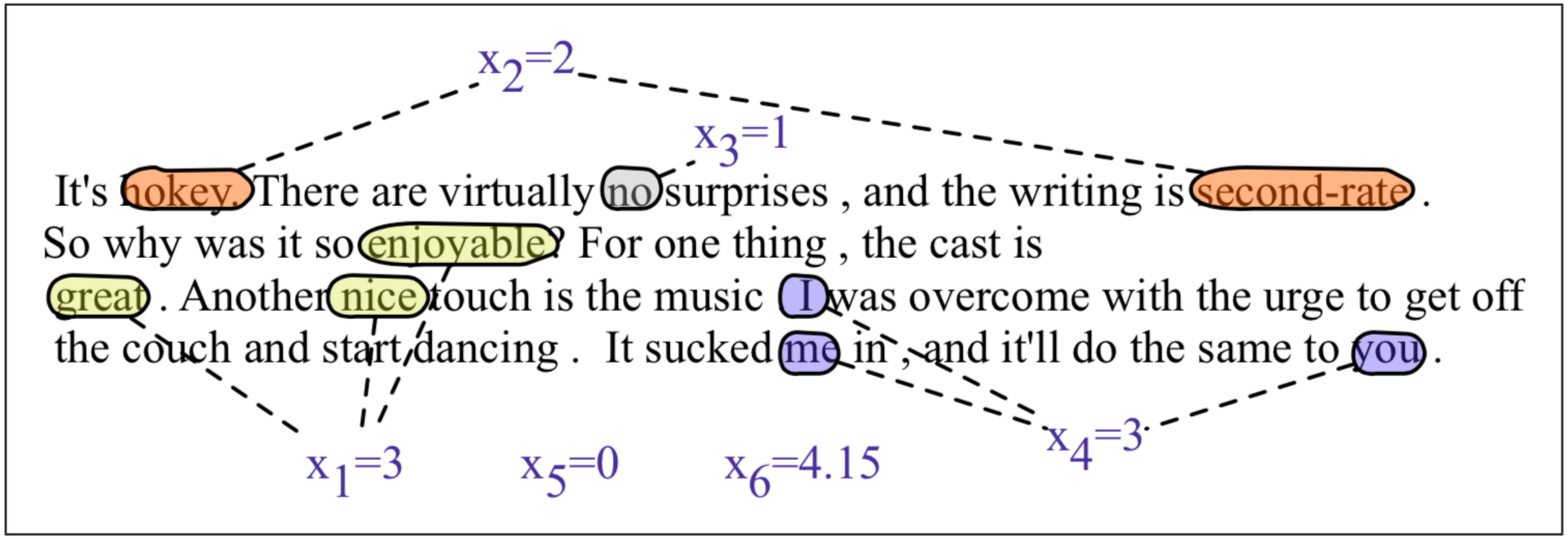
- Train phase: Learn the **parameters** of the model to minimize **loss function**

- Test phase: Apply **parameters** to predict class given a new input x

Feature representation

- Input observation: $x^{(i)}$
- Feature vector: $[x_1, x_2, \ldots, x_d]$
- Feature j of i[th] input : $x_j^{(i)}$

Sample feature vector



Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

Classification function

Classification function

- *Given*: Input feature vector $[x_1, x_2, \ldots, x_d]$

Classification function

- *Given*: Input feature vector $[x_1, x_2, \ldots, x_d]$

- *Output*: $P(y = 1 \,|\, x)$ and $P(y = 0 \,|\, x)$ *(binary classification)*

Classification function

- *Given*: Input feature vector $[x_1, x_2, \ldots, x_d]$
- *Output*: $P(y = 1 \mid x)$ and $P(y = 0 \mid x)$ *(binary classification)*
- Require a *function,* $F : \mathbb{R}^{\mathrm{d}} \to [0,1]$

Classification function

- *Given*: Input feature vector $[x_1, x_2, \ldots, x_d]$
- *Output*: $P(y = 1 \mid x)$ and $P(y = 0 \mid x)$ *(binary classification)*
- Require a *function,* $F : \mathbb{R}^{\mathrm{d}} \rightarrow [0,1]$
- Sigmoid:

Classification function

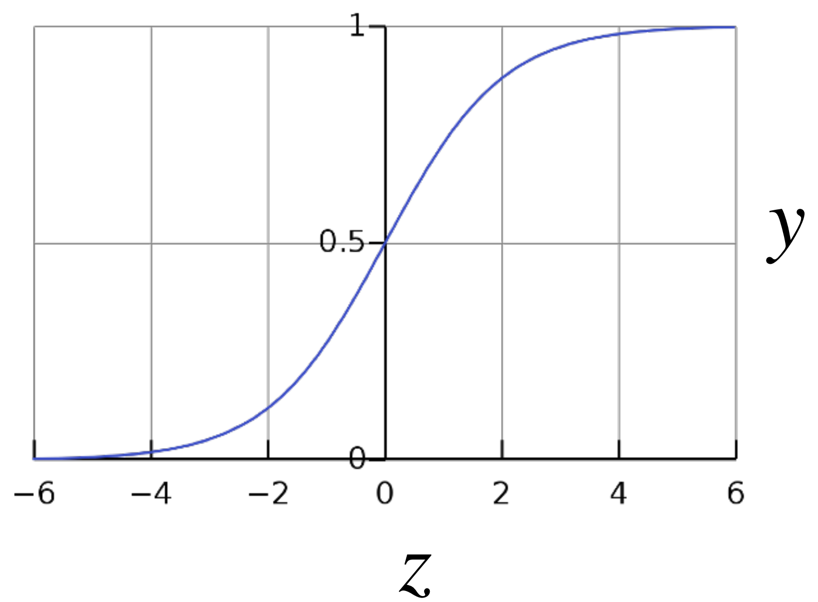
- *Given*: Input feature vector $[x_1, x_2, \ldots, x_d]$

- *Output*: $P(y = 1 \,|\, x)$ and $P(y = 0 \,|\, x)$ *(binary classification)*

- Require a *function,* $F : \mathbb{R}^{\mathrm{d}} \rightarrow [0,1]$

- Sigmoid:

$$y = \frac{1}{1 + e^{-z}}$$



Weights and Biases

Weights and Biases

- *Which features are important* and *how much*?

Weights and Biases

- *Which features are important* and *how much*?
- Learn a vector of **weights** and a **bias**

Weights and Biases

- *Which features are important* and *how much*?
- Learn a vector of **weights** and a **bias**
- **Weights:** Vector of real numbers, $w = [w_1, w_2, \ldots, w_d]$

Weights and Biases

- *Which features are important* and *how much*?
- Learn a vector of **weights** and a **bias**
- **Weights:** Vector of real numbers, $w = [w_1, w_2, \ldots, w_d]$
- **Bias:** Scalar intercept, b

Weights and Biases

- *Which features are important* and *how much*?
- Learn a vector of **weights** and a **bias**
- **Weights:** Vector of real numbers, $w = [w_1, w_2, \ldots, w_d]$
- **Bias:** Scalar intercept, b
- Given an instance, x: $z = \sum_{i=1}^{d} w_i x_i + b$ or $z = w \cdot x + b$

What is the bias?

- Let’s say we have a feature that is always set to 1 regardless of what the input text is.
- This is clearly not an informative feature. However, let’s say it was the only one I had…

first, how many weights do I need to learn for this feature?

(Credits: Richard Socher)

What is the bias?

- Let's say we have a feature that is always set to 1 regardless of what the input text is.
- This is clearly not an informative feature. However, let's say it was the only one I had…

$$w \cdot x + b$$

first, how many weights do I need to learn for this feature?

okay… what is the best set of weights for it?

(Credits: Richard Socher)

Putting it together

Putting it together

- Given x, compute $z = w \cdot x + b$

Putting it together

- Given x, compute $z = w \cdot x + b$
- Compute probabilities: $P(y = 1 \mid x) = \dfrac{1}{1 + e^{-z}}$

Putting it together

- Given x, compute $z = w \cdot x + b$
- Compute probabilities: $P(y = 1 \mid x) = \dfrac{1}{1 + e^{-z}}$

$$
\begin{aligned}
P(y = 1) &= \sigma(w \cdot x + b) \\
&= \frac{1}{1 + e^{-(w \cdot x + b)}}
\end{aligned}
$$

Putting it together

- Given x, compute $z = w \cdot x + b$
- Compute probabilities: $P(y = 1 \mid x) = \dfrac{1}{1 + e^{-z}}$

$$
\begin{aligned}
P(y = 1) &= \sigma(w \cdot x + b) \\
&= \frac{1}{1 + e^{-(w \cdot x + b)}}
\end{aligned}
$$

$$
\begin{aligned}
P(y = 0) &= 1 - \sigma(w \cdot x + b) \\
&= 1 - \frac{1}{1 + e^{-(w \cdot x + b)}} \\
&= \frac{e^{-(w \cdot x + b)}}{1 + e^{-(w \cdot x + b)}}
\end{aligned}
$$

Putting it together

- Given x, compute $z = w \cdot x + b$
- Compute probabilities: $P(y = 1 \mid x) = \dfrac{1}{1 + e^{-z}}$

$$\begin{aligned} P(y = 1) &= \sigma(w \cdot x + b) \\ &= \frac{1}{1 + e^{-(w \cdot x + b)}} \end{aligned}$$

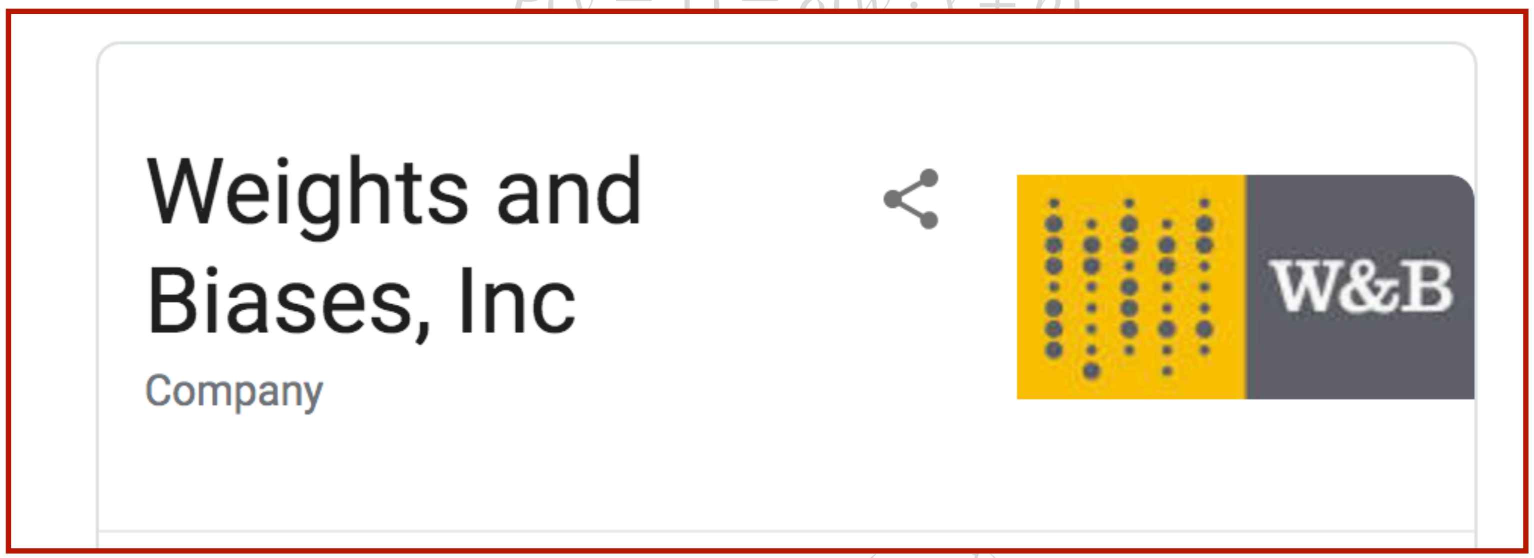
$$\begin{aligned} P(y = 0) &= 1 - \sigma(w \cdot x + b) \\ &= 1 - \frac{1}{1 + e^{-(w \cdot x + b)}} \\ &= \frac{e^{-(w \cdot x + b)}}{1 + e^{-(w \cdot x + b)}} \end{aligned}$$

- Decision boundary: $\hat{y} = \begin{cases} 1 & \text{if } P(y = 1 \mid x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$

Putting it together

- Given x, compute $z = w \cdot x + b$
- Compute probabilities: $P(y = 1 \mid x) = \dfrac{1}{1 + e^{-z}}$

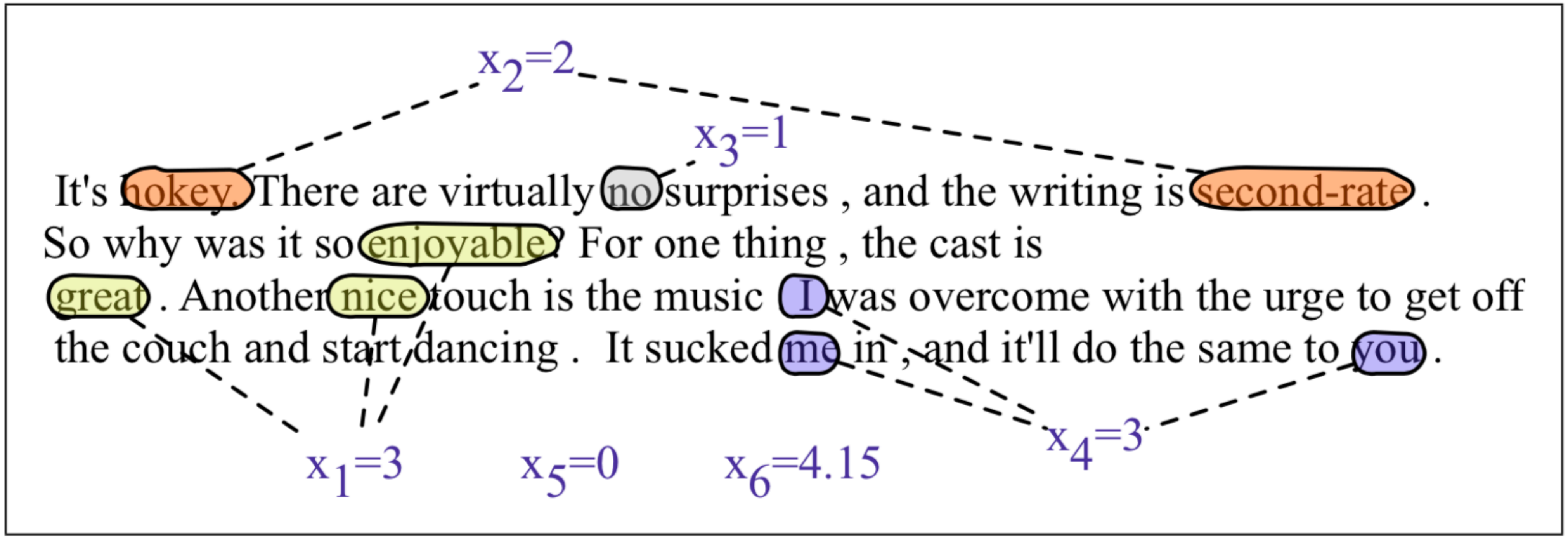
$$P(y = 1) = \sigma(w \cdot x + b)$$



$$= \frac{e^{-(w \cdot x + b)}}{1 + e^{-(w \cdot x + b)}}$$

- Decision boundary: $\hat{y} = \begin{cases} 1 & \text{if } P(y = 1 \mid x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$

Example: Sentiment classification



Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

Example: Sentiment classification

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

Example: Sentiment classification

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

- Assume weights $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and bias $b = 0.1$

Example: Sentiment classification

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

- Assume weights $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and bias $b = 0.1$

$$
\begin{aligned}
p(+|x) = P(Y=1|x) &= \sigma(w \cdot x + b) \\
&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.15] + 0.1) \\
&= \sigma(.805) \\
&= 0.69 \\
p(-|x) = P(Y=0|x) &= 1 - \sigma(w \cdot x + b) \\
&= 0.31
\end{aligned}
$$

Feature design

Feature design

- Most important rule: Data is *key*!

Feature design

- Most important rule: Data is *key*!
- Linguistic intuition (e.g. part of speech tags, parse trees)

Feature design

- Most important rule: Data is *key*!
- Linguistic intuition (e.g. part of speech tags, parse trees)
- Complex combinations

Feature design

- Most important rule: Data is *key*!
- Linguistic intuition (e.g. part of speech tags, parse trees)
- Complex combinations

$$
\begin{aligned}
x_1 &= \begin{cases} 1 & \text{if ``}Case(w_i) = \text{Lower''} \\ 0 & \text{otherwise} \end{cases} \\
x_2 &= \begin{cases} 1 & \text{if ``}w_i \in \text{AcronymDict''} \\ 0 & \text{otherwise} \end{cases} \\
x_3 &= \begin{cases} 1 & \text{if ``}w_i = \text{St. \& } Case(w_{i-1}) = \text{Cap''} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

Feature design

- Most important rule: Data is *key*!
- Linguistic intuition (e.g. part of speech tags, parse trees)
- Complex combinations

$$x_1 = \begin{cases} 1 & \text{if "}Case(w_i) = \text{Lower"} \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if "}w_i \in \text{AcronymDict"} \\ 0 & \text{otherwise} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{if "}w_i = \text{St. \& } Case(w_{i-1}) = \text{Cap"} \\ 0 & \text{otherwise} \end{cases}$$

- Feature templates
    - Sparse representations, hash only seen features into index
    - Ex. Trigram(“*logistic regression model”*) = Feature #78

Feature design

- Most important rule: Data is *key*!
- Linguistic intuition (e.g. part of speech tags, parse trees)
- Complex combinations

$$x_1 = \begin{cases} 1 & \text{if ``}Case(w_i) = \text{Lower''} \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if ``}w_i \in \text{AcronymDict''} \\ 0 & \text{otherwise} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{if ``}w_i = \text{St. \& } Case(w_{i-1}) = \text{Cap''} \\ 0 & \text{otherwise} \end{cases}$$

- Feature templates
  - Sparse representations, hash only seen features into index
  - Ex. Trigram("*logistic regression model"*) = Feature #78
- Advanced: Representation learning (we will see this later!)

Logistic Regression: what's good and what's not

Logistic Regression: what's good and what's not

- More freedom in designing features

Logistic Regression: what's good and what's not

- More freedom in designing features
  - No strong independence assumptions like Naive Bayes

Logistic Regression: what's good and what's not

- More freedom in designing features
  - No strong independence assumptions like Naive Bayes
  - More robust to correlated features ("San Francisco" vs "Boston") —LR is likely to work better than NB

Logistic Regression: what's good and what's not

- More freedom in designing features
  - No strong independence assumptions like Naive Bayes
  - More robust to correlated features ("San Francisco" vs "Boston") —LR is likely to work better than NB
  - Can even have the same feature twice! (*why?*)

Logistic Regression: what's good and what's not

- More freedom in designing features
  - No strong independence assumptions like Naive Bayes
  - More robust to correlated features ("San Francisco" vs "Boston") —LR is likely to work better than NB
  - Can even have the same feature twice! (*why?*)
- **However**: NB often better on very small datasets

Learning

Learning

- We have our **classification function -** how to assign weights and bias?

Learning

- We have our **classification function -** how to assign weights and bias?

- Goal: predicted label $\hat{y}$ as close as possible to actual label y

Learning

- We have our **classification function -** how to assign weights and bias?

- Goal: predicted label $\hat{y}$ as close as possible to actual label y

  - **Distance metric/Loss function** between $\hat{y}$ and y :
    $L(\hat{y}, y)$

Learning

- We have our **classification function -** how to assign weights and bias?

- Goal: predicted label $\hat{y}$ as close as possible to actual label y

  - **Distance metric/Loss function** between $\hat{y}$ and y : $L(\hat{y}, y)$

  - **Optimization algorithm** for updating weights

Loss function

Loss function

- Assume $\hat{y} = \sigma(w \cdot x + b)$

Loss function

- Assume $\hat{y} = \sigma(w \cdot x + b)$
- $L(\hat{y}, y) =$ Measure of difference between $\hat{y}$ and y. But what form?

Loss function

- Assume $\hat{y} = \sigma(w \cdot x + b)$
- $L(\hat{y}, y) =$ Measure of difference between $\hat{y}$ and y. But what form?
- Maximum likelihood estimation (conditional):

Loss function

- Assume $\hat{y} = \sigma(w \cdot x + b)$
- $L(\hat{y}, y) =$ Measure of difference between $\hat{y}$ and y. But what form?
- Maximum likelihood estimation (conditional):
  - Choose w and b such that $\log P(y \mid x)$ is maximized for true labels y paired with input x

Loss function

- Assume $\hat{y} = \sigma(w \cdot x + b)$
- $L(\hat{y}, y) =$ Measure of difference between $\hat{y}$ and y. But what form?
- Maximum likelihood estimation (conditional):
  - Choose w and b such that $\log P(y \mid x)$ is maximized for true labels y paired with input x
  - Similar to language models!

Loss function

- Assume $\hat{y} = \sigma(w \cdot x + b)$
- $L(\hat{y}, y) =$ Measure of difference between $\hat{y}$ and y. But what form?
- Maximum likelihood estimation (conditional):
  - Choose w and b such that $\log P(y \mid x)$ is maximized for true labels y paired with input x
  - Similar to language models!
    - $\max \log P(w_t \mid w_{t-n}, \ldots, w_{t-1})$ given a corpus

Cross Entropy loss

- Assume a single data point (x, y) and two classes

- Classifier probability: $P(y \mid x) = \hat{y}^{\,y}(1 - \hat{y})^{1-y}$

- Log probability:

- CE Loss:

Cross Entropy loss

- Assume a single data point (x, y) and two classes

- Classifier probability: $P(y \mid x) = \hat{y}^{\,y}(1 - \hat{y})^{1-y}$

- Log probability: $\log P(y|x) = \log\left[\hat{y}^{y}(1-\hat{y})^{1-y}\right]$

  $= y \log \hat{y} + (1-y)\log(1-\hat{y})$

- CE Loss:

Cross Entropy loss

- Assume a single data point (x, y) and two classes

- Classifier probability: $P(y \mid x) = \hat{y}^{\,y}(1 - \hat{y})^{1-y}$

- Log probability: $\log P(y|x) = \log\left[\hat{y}^{y}(1-\hat{y})^{1-y}\right]$

  $= y \log \hat{y} + (1-y)\log(1-\hat{y})$

- CE Loss:

$-\log P(y|x)$

$= -\left[y \log \hat{y} + (1-y)\log(1-\hat{y})\right]$

$y = 1 \Rightarrow -\log \hat{y}$, $y = 0 \Rightarrow -\log(1-\hat{y})$

Cross Entropy loss

- Assume n data points $(x^{(i)}, y^{(i)})$
- Classifier probability: $\Pi_{i=1}^{n} P(y \mid x) = \Pi_{i=1}^{n} \hat{y}^{y}(1 - \hat{y})^{1-y}$
- CE Loss: $-\log \prod_{i=1}^{n} P(y|x)$

$$= -\sum_{i=1}^{n} \log P(y|x)$$

$$L_{CE} = -\sum_{i=1}^{n} \left[y \log \hat{y} + (1-y) \log (1-\hat{y}) \right]$$

Example: Computing CE Loss

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

Example: Computing CE Loss

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

- Assume weights $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and bias $b = 0.1$

Example: Computing CE Loss

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

- Assume weights $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and bias $b = 0.1$
- If y = 1 (positive sentiment), $L_{CE} = -\log(0.69) = 0.37$

Example: Computing CE Loss

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

- Assume weights $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and bias $b = 0.1$
- If y = 1 (positive sentiment), $L_{CE} = -\log(0.69) = 0.37$
- If y = 0 (negative sentiment), $L_{CE} = -\log(0.31) = 1.17$

Properties of CE Loss

Properties of CE Loss

- $L_{CE} = -\sum_{i=1}^{n} [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$

Properties of CE Loss

- $$L_{CE} = -\sum_{i=1}^{n} [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

- Ranges from 0 (perfect predictions) to ∞

Properties of CE Loss

- $$L_{CE} = -\sum_{i=1}^{n} [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$
- Ranges from 0 (perfect predictions) to ∞
- Lower the value, better the classifier

Properties of CE Loss

- $L_{CE} = -\sum_{i=1}^{n} [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$

- Ranges from 0 (perfect predictions) to ∞

- Lower the value, better the classifier

- *Cross-entropy* between the true distribution $P(y \mid x)$ and predicted distribution $P(\hat{y} \mid x)$

Optimization

Optimization

- We have our **classification function** and **loss function** - how do we find the best w and b?

Optimization

- We have our **classification function** and **loss function** - how do we find the best w and b?

$$\theta = [w; b]$$

Optimization

- We have our **classification function** and **loss function** - how do we find the best w and b?

$$\theta = [w; b]$$

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L_{CE}(y^{(i)}, x^{(i)}; \theta)$$

Optimization

- We have our **classification function** and **loss function** - how do we find the best w and b?

$$\theta = [w; b]$$

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L_{CE}(y^{(i)}, x^{(i)}; \theta)$$

- Gradient descent:

Optimization

- We have our **classification function** and **loss function** - how do we find the best w and b?

$$\theta = [w; b]$$

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L_{CE}(y^{(i)}, x^{(i)}; \theta)$$

- Gradient descent:
  - Find direction of steepest slope

Optimization

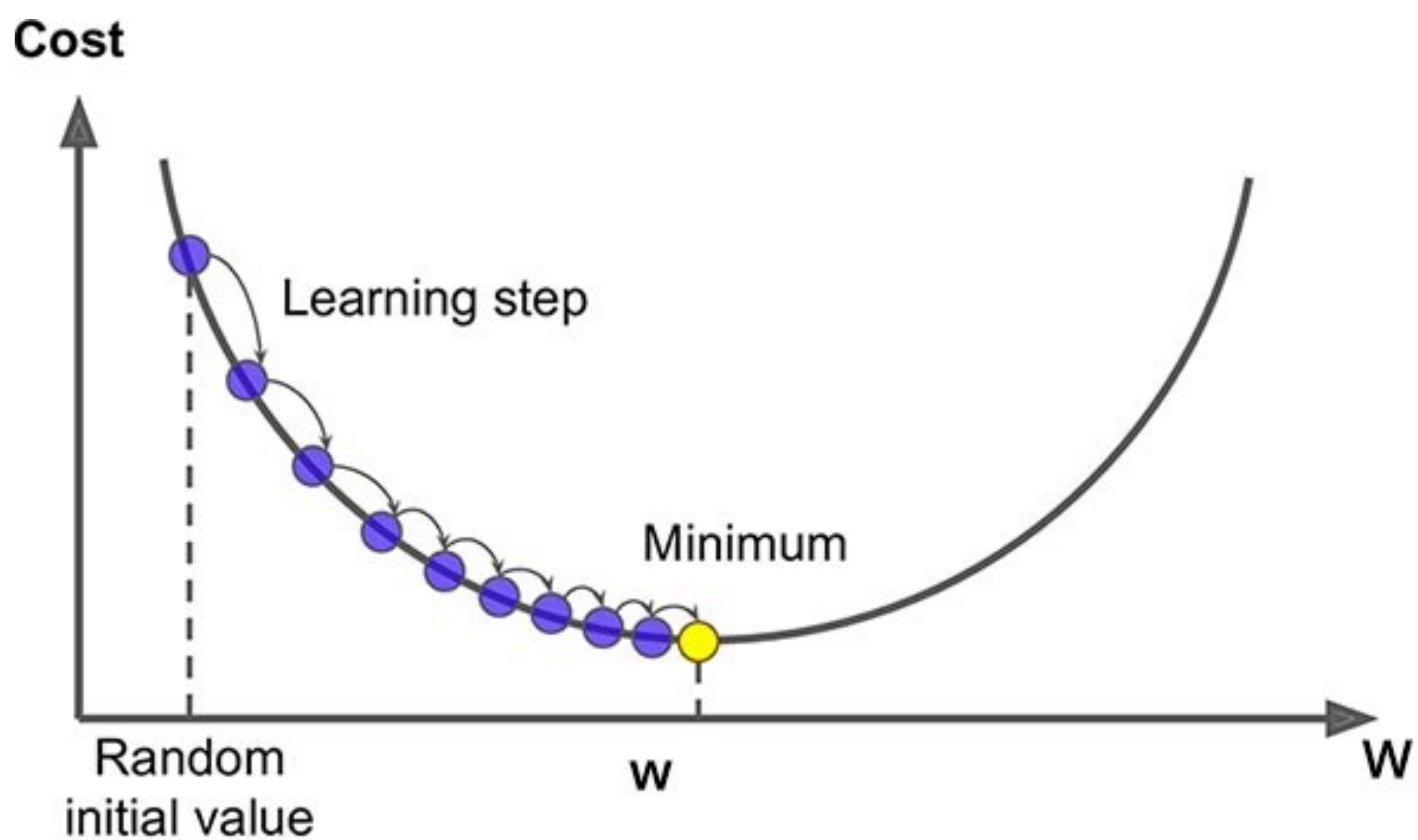
- We have our **classification function** and **loss function** - how do we find the best w and b?

$$\theta = [w; b]$$

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L_{CE}(y^{(i)}, x^{(i)}; \theta)$$

- Gradient descent:
  - Find direction of steepest slope
  - Move in the opposite direction

Gradient descent (1-D)



$$\theta^{t+1} = \theta^{t} - \eta \frac{d}{d\theta} f(x;\theta)$$

Gradient descent for LR

Gradient descent for LR

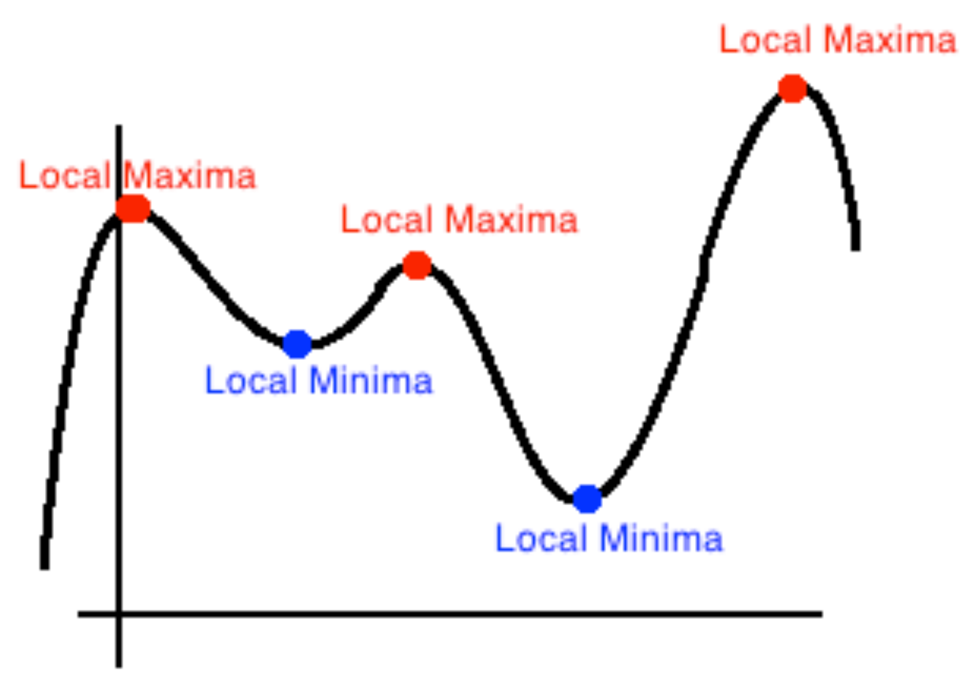
- Cross entropy loss for logistic regression is **convex** (i.e. has only one global minimum)
    - No local minima to get stuck in

Gradient descent for LR

- Cross entropy loss for logistic regression is **convex** (i.e. has only one global minimum)
  - No local minima to get stuck in
- Deep neural networks are not so easy
  - Non-convex

Gradient descent for LR

- Cross entropy loss for logistic regression is **convex** (i.e. has only one global minimum)
    - No local minima to get stuck in
- Deep neural networks are not so easy
    - Non-convex

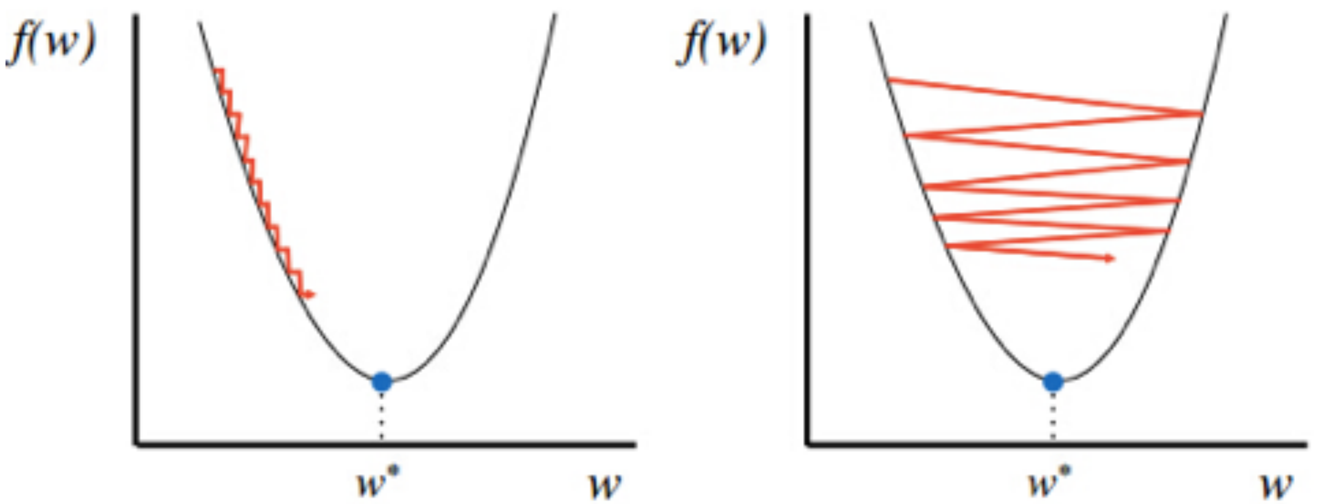


Learning Rate

- Updates: $\theta^{t+1} = \theta^{t} - \eta \frac{d}{d\theta} f(x;\theta)$

- Magnitude of movement along gradient

- Higher/faster learning rate = larger updates to parameters



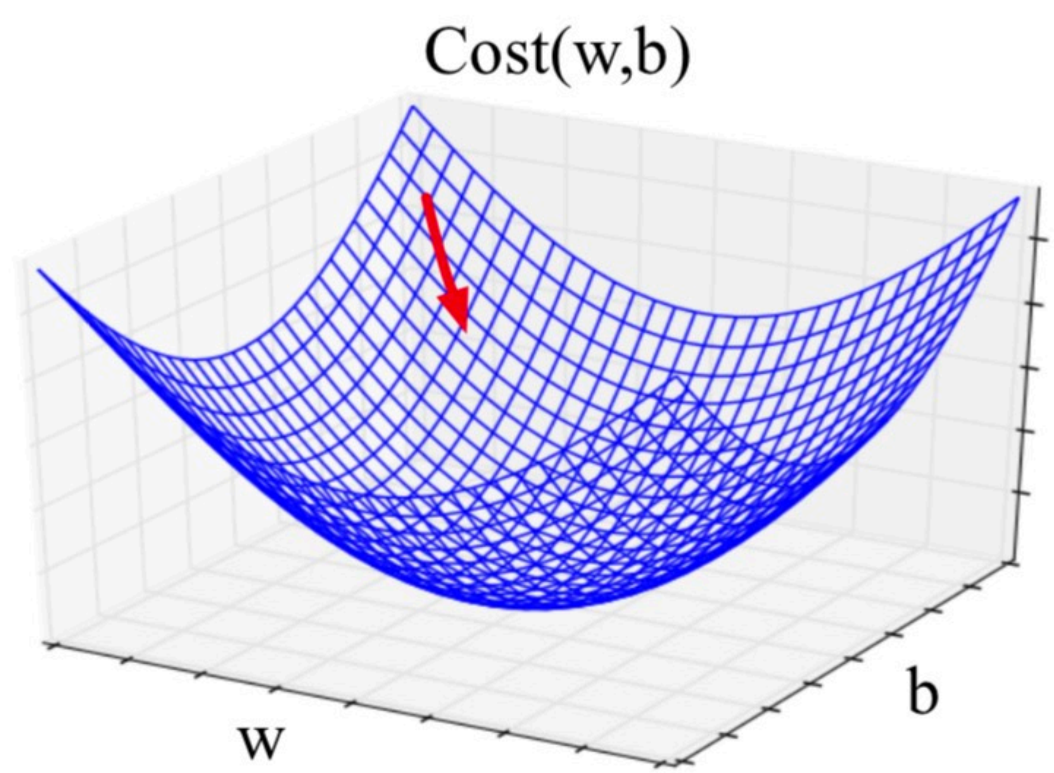
Too small: converge very slowly

Too big: overshoot and even diverge

Gradient descent with vector weights

- In LR: weight w is a vector
- Express slope as a partial derivative of loss w.r.t each weight:

$$\nabla_\theta L(f(x;\theta),y)) \;=\; \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \end{bmatrix}$$



Gradient descent with vector weights

- In LR: weight w is a vector

- Express slope as a partial derivative of loss w.r.t each weight:

$$\nabla_\theta L(f(x;\theta),y)) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \end{bmatrix}$$

- Updates: $\theta^{(t+1)} = \theta^t - \eta \nabla L(f(x;\theta), y)$

Gradient for logistic regression

Gradient for logistic regression

- $L_{CE} = -\sum_{i=1}^{n} [y^{(i)} \log \sigma(w \cdot x^{(i)} + b) + (1 - y^{(i)}) \log(1 - \sigma(w \cdot x^{(i)} + b))]$

Gradient for logistic regression

- $L_{CE} = -\sum_{i=1}^{n} [y^{(i)} \log \sigma(w \cdot x^{(i)} + b) + (1 - y^{(i)}) \log(1 - \sigma(w \cdot x^{(i)} + b))]$

- Gradient, $\dfrac{dL_{CE}(w, b)}{dw_j} = \sum_{i=1}^{n} [\sigma(w \cdot x^{(i)} + b) - y^{(i)}] x_j^{(i)}$

Gradient for logistic regression

- $L_{CE} = -\sum_{i=1}^{n} [y^{(i)} \log \sigma(w \cdot x^{(i)} + b) + (1 - y^{(i)}) \log(1 - \sigma(w \cdot x^{(i)} + b))]$
- Gradient, $\frac{dL_{CE}(w, b)}{dw_j} = \sum_{i=1}^{n} [\sigma(w \cdot x^{(i)} + b) - y^{(i)}] x_j^{(i)}$

Gradient for logistic regression

- $L_{CE} = -\sum_{i=1}^{n} [y^{(i)} \log \sigma(w \cdot x^{(i)} + b) + (1 - y^{(i)}) \log(1 - \sigma(w \cdot x^{(i)} + b))]$

- Gradient, $\dfrac{dL_{CE}(w, b)}{dw_j} = \sum_{i=1}^{n} [\sigma(w \cdot x^{(i)} + b) - y^{(i)}] x_j^{(i)}$

- $\dfrac{dL_{CE}(w, b)}{db} = \sum_{i=1}^{n} [\sigma(w \cdot x^{(i)} + b) - y^{(i)}]$

Gradient for logistic regression

- $L_{CE} = -\sum_{i=1}^{n} [y^{(i)} \log \sigma(w \cdot x^{(i)} + b) + (1 - y^{(i)}) \log(1 - \sigma(w \cdot x^{(i)} + b))]$

- Gradient, $\dfrac{dL_{CE}(w, b)}{dw_j} = \sum_{i=1}^{n} [\sigma(w \cdot x^{(i)} + b) - y^{(i)}] x_j^{(i)}$

Diff between true y and prediction

input feature value

- $\dfrac{dL_{CE}(w, b)}{db} = \sum_{i=1}^{n} [\sigma(w \cdot x^{(i)} + b) - y^{(i)}]$

Stochastic Gradient Descent

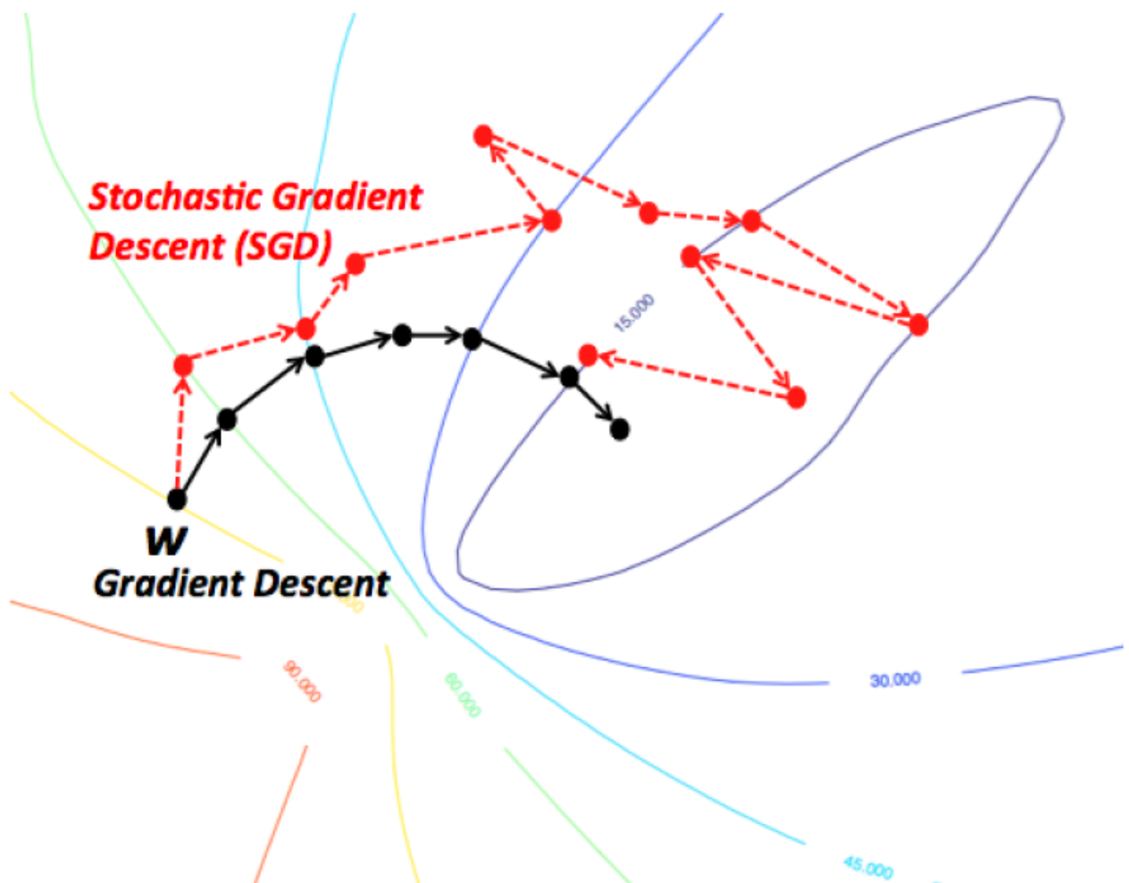
- Online optimization
- Compute loss and minimize after each training example
- **function** STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ

    \# where: L is the loss function
    \# f is a function parameterized by θ
    \# x is the set of training inputs $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$
    \# y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots, y^{(n)}$

    $\theta \leftarrow 0$
    repeat til done # see caption
      For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)
        1. Optional (for reporting): # How are we doing on this tuple?
          Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$ # What is our estimated output $\hat{y}$?
          Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$) from the true output $y^{(i)}$?
        2. $g \leftarrow \nabla_\theta L(f(x^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?
        3. $\theta \leftarrow \theta - \eta\, g$ # Go the other way instead
    return θ

Per Instance Loss →

Stochastic Gradient Descent

- Online optimization
- Compute loss and minimize after each training example
- 

Regularization

Regularization

- Training objective: $\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} | x^{(i)})$

Regularization

- Training objective: $\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} | x^{(i)})$

- This might fit the training set too well! (including noisy features)

Regularization

- Training objective: $\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} \mid x^{(i)})$

- This might fit the training set too well! (including noisy features)

- Poor generalization to the unseen test set — **Overfitting**

Regularization

- Training objective: $\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} | x^{(i)})$

- This might fit the training set too well! (including noisy features)

- Poor generalization to the unseen test set — **Overfitting**

- **Regularization** helps prevent overfitting

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} | x^{(i)}) - \alpha R(\theta)$$

Regularization

- Training objective: $\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} \mid x^{(i)})$

- This might fit the training set too well! (including noisy features)

- Poor generalization to the unseen test set — **Overfitting**

- **Regularization** helps prevent overfitting

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} \mid x^{(i)}) - \alpha R(\theta)$$

Penalize large weights

L2 regularization

- $R(\theta) = ||\theta||^2 = \sum_{j=1}^{d} \theta_j^2$

- Euclidean distance of weight vector θ from origin

- L2 regularized objective:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} | x^{(i)}) - \alpha \sum_{j=1}^{d} \theta_j^2$$

L1 Regularization

- $R(\theta) = ||\theta||_1 = \sum_{j=1}^{d} |\theta_j|$

- Manhattan distance of weight vector θ from origin

- L1 regularized objective:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y^{(i)} | x^{(i)}) - \alpha \sum_{j=1}^{d} |\theta_j|$$

L2 vs L1 regularization

L2 vs L1 regularization

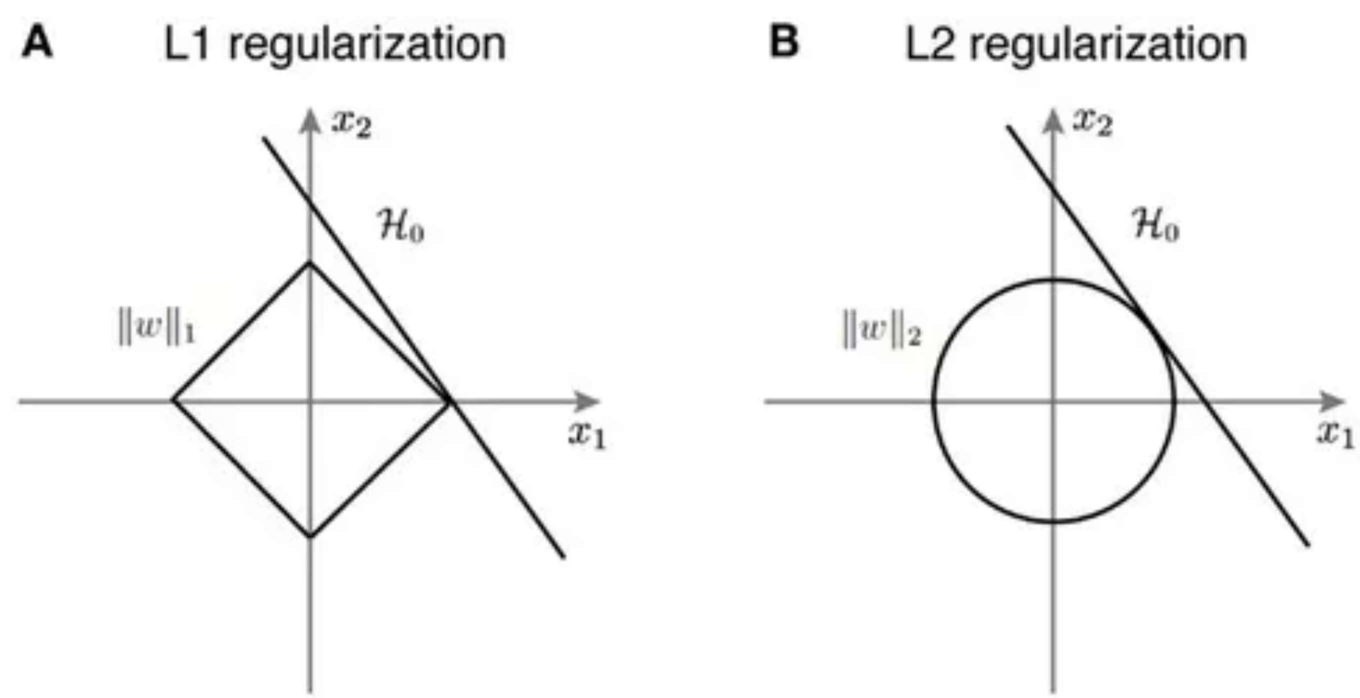
- L2 is easier to optimize - simpler derivation
  - L1 is complex since the derivative of $|\theta|$ is not continuous at 0

L2 vs L1 regularization

- L2 is easier to optimize - simpler derivation
  - L1 is complex since the derivative of $|\theta|$ is not continuous at 0
- L2 leads to many small weights (due to θ^2 term)
  - L1 prefers *sparse* weight vectors with many weights set to 0 (i.e. far fewer features used)

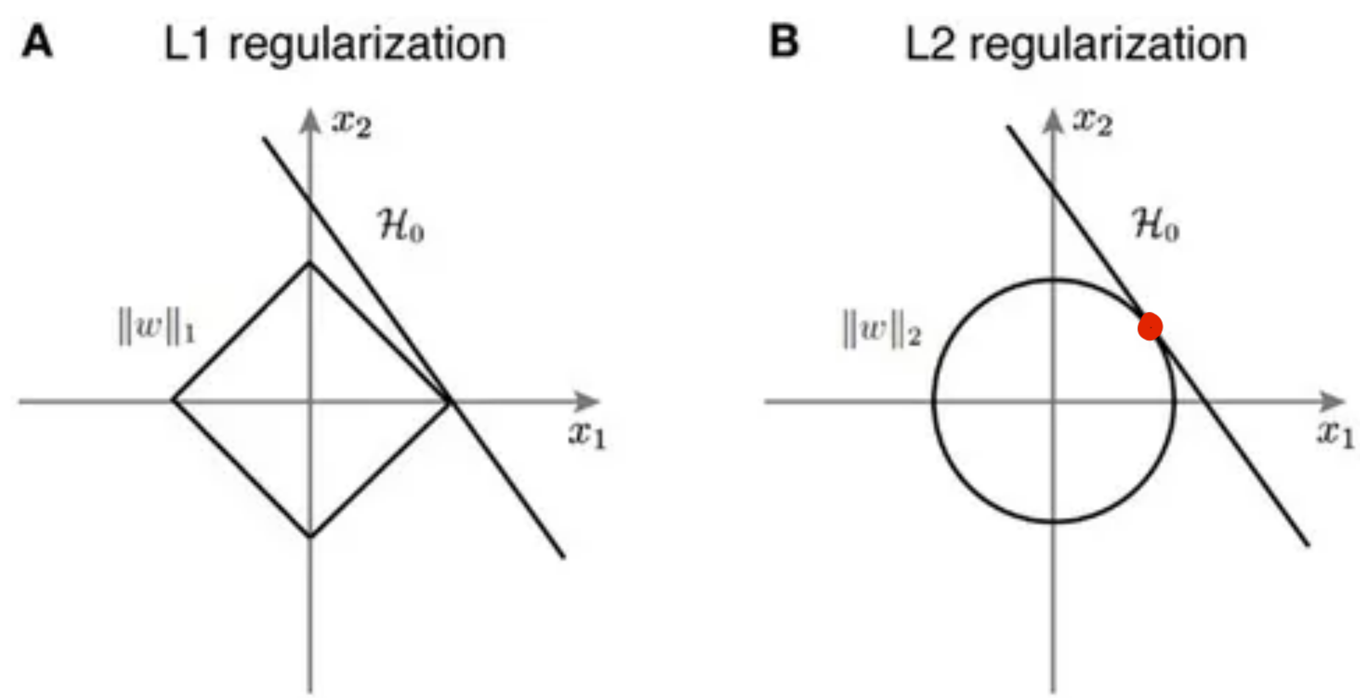
L2 vs L1 regularization

- L2 is easier to optimize - simpler derivation
  - L1 is complex since the derivative of $|\theta|$ is not continuous at 0
- L2 leads to many small weights (due to θ^2 term)
  - L1 prefers *sparse* weight vectors with many weights set to 0 (i.e. far fewer features used)



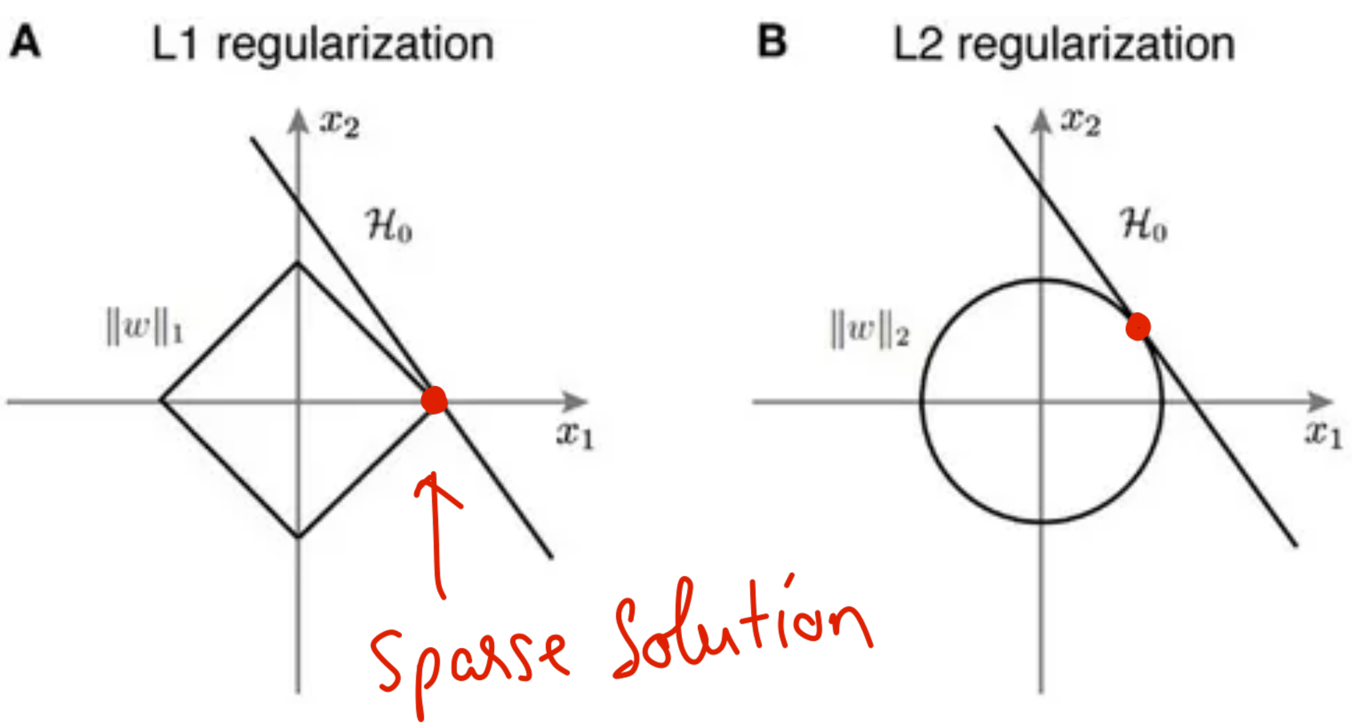
L2 vs L1 regularization

- L2 is easier to optimize - simpler derivation
  - L1 is complex since the derivative of $|\theta|$ is not continuous at 0
- L2 leads to many small weights (due to θ^2 term)
  - L1 prefers *sparse* weight vectors with many weights set to 0 (i.e. far fewer features used)



L2 vs L1 regularization

- L2 is easier to optimize - simpler derivation
  - L1 is complex since the derivative of $|\theta|$ is not continuous at 0
- L2 leads to many small weights (due to θ^2 term)
  - L1 prefers *sparse* weight vectors with many weights set to 0 (i.e. far fewer features used)



Multinomial Logistic Regression

Multinomial Logistic Regression

- What if we have more than 2 classes? (e.g. Part of speech tagging, Named Entity Recognition, language model!)

Multinomial Logistic Regression

- What if we have more than 2 classes? (e.g. Part of speech tagging, Named Entity Recognition, language model!)

- Need to model $P(y = c \mid x) \forall c \in C$

Multinomial Logistic Regression

- What if we have more than 2 classes? (e.g. Part of speech tagging, Named Entity Recognition, language model!)
- Need to model $P(y = c \mid x) \forall c \in C$
- Generalize **sigmoid** function to **softmax**

Multinomial Logistic Regression

- What if we have more than 2 classes? (e.g. Part of speech tagging, Named Entity Recognition, language model!)

- Need to model $P(y = c \mid x) \forall c \in C$

- Generalize **sigmoid** function to **softmax**

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad 1 \leq i \leq k$$

Multinomial Logistic Regression

- What if we have more than 2 classes? (e.g. Part of speech tagging, Named Entity Recognition, language model!)
- Need to model $P(y = c \mid x) \forall c \in C$
- Generalize **sigmoid** function to **softmax**

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad 1 \leq i \leq k$$

← Normalization

Softmax

Softmax

- Similar to sigmoid, softmax squashes values towards 0 or 1

Softmax

- Similar to sigmoid, softmax squashes values towards 0 or 1
- If $z = [0,1,2,3,4]$, then
  - softmax$(z) = ([0.0117,0.0317,0.0861,0.2341,0.6364])$

Softmax

- Similar to sigmoid, softmax squashes values towards 0 or 1
- If $z = [0,1,2,3,4]$, then
  - softmax$(z) = ([0.0117, 0.0317, 0.0861, 0.2341, 0.6364])$
- For multinomial LR,

Softmax

- Similar to sigmoid, softmax squashes values towards 0 or 1
- If $z = [0,1,2,3,4]$, then
  - softmax$(z) = ([0.0117, 0.0317, 0.0861, 0.2341, 0.6364])$
- For multinomial LR,

$$P(y = c \,|\, x) = \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_j}}$$

Softmax

- Similar to sigmoid, softmax squashes values towards 0 or 1
- If $z = [0,1,2,3,4]$, then
  - softmax$(z) = ([0.0117, 0.0317, 0.0861, 0.2341, 0.6364])$
- For multinomial LR,

$$P(y = c \mid x) = \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_j}}$$

$$\log P(y = c \mid x) \propto w_c \cdot x + b_c$$

(log-linear)

Features in multinomial LR

Features in multinomial LR

- Features need to include both input (x) and class (c)

Features in multinomial LR

- Features need to include both input (x) and class (c)
- Implicit in binary case

Features in multinomial LR

- Features need to include both input (x) and class (c)
- Implicit in binary case

Var	Definition	Wt
$f_1(0,x)$	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	-4.5
$f_1(+,x)$	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	2.6
$f_1(-,x)$	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1.3

Learning

Learning

- Generalize binary loss to multinomial CE loss:

$$L_{CE}(\hat{y}, y) = -\sum_{c=1}^{k} 1\{y = k\} \log P(y = k \mid x)$$

$$= -\sum_{c=1}^{k} 1\{y = k\} \log \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_c}}$$

Learning

- Generalize binary loss to multinomial CE loss:

$$L_{CE}(\hat{y}, y) = -\sum_{c=1}^{k} 1\{y = k\} \log P(y = k \mid x)$$

$$= -\sum_{c=1}^{k} 1\{y = k\} \log \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_c}}$$

- Gradient:

Learning

- Generalize binary loss to multinomial CE loss:

$$L_{CE}(\hat{y}, y) = -\sum_{c=1}^{k} 1\{y = k\} \log P(y = k \,|\, x)$$

$$= -\sum_{c=1}^{k} 1\{y = k\} \log \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_c}}$$

- Gradient:

$$\frac{dL_{CE}}{dw_c} = -(1\{y = c\} - P(y = c \,|\, x))x_c$$

$$= -\left(1\{y = c\} - \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_c}}\right) x_c$$