Lecture 2: Text Classification

Cornell Bowers C·IS
Computer Science

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

Announcements

- HW1 will be released on Wednesday.
    - Due on 20 February, 11.59 p.m.
- Conflict sheet for the midterm released on Ed.
    - Deadline to fill this is Feb 15 (barring emergencies).

Today

- **N-grams revisited.**
- Text Classification
- Feature Engineering
- Binary Logistic Regression

What is a Language Model?

- A model that computes the probability of **any** sequence of words:

$$P(w_1 w_2 w_3 \dots w_n)$$

$$e.g. \quad P(\textit{Mayenne ate my shoes today.}) = 10^{-12}$$

$$P(\textit{Mayenne my ate no}) = 10^{-30}$$

- A model that computes a probability distribution over possible next words:

$$P(w_n \mid w_1 w_2 w_3 \dots w_{n-1})$$

$$e.g. \quad P(\textit{today} \mid \textit{Mayenne ate my shoes}) = 10^{-3}$$

Language Modeling Problem

- Let $\mathscr{V}$ be a finite vocabulary of words.

$$\mathscr{V} = \{\text{ the, a, man, telescope, Madrid, two, } \ldots\}$$

- We can construct (infinite) word sequences $\mathbf{w}$

$$\mathscr{V}^{\dagger} = \{\text{ the, a, the a, the fan, the man, the man with a telescope}\}$$

- **Given**: a dataset of **M** sentences $\mathscr{D} = \{\mathbf{w}\}_{i=1}^{M}$
- **Goal/ Output**: estimate a probability distribution $P(\mathbf{w}) \geq 0$ over **all** word sequences $\mathbf{w} \in \mathscr{V}^{+}$.

Language Modeling Problem

$$P(\mathbf{w}_1^n) = P(w_1 w_2 w_3 \dots w_n) = \prod_{i=1}^{n} P(w_i \mid w_1 \dots w_{i-1})$$

Key idea: Markov Assumption: Probability of each word in a sequence only depends on a fixed number of previous words

Unigram Model $\rightarrow P(w_i \mid w_1 \dots w_{i-1}) := P(w_i)$

Bigram Model $\rightarrow P(w_i \mid w_1 \dots w_{i-1}) := P(w_i \mid w_{i-1})$

Trigram Model $\rightarrow P(w_i \mid w_1 \dots w_{i-1}) := P(w_i \mid w_{i-2} w_{i-1})$

N-gram language models: Probability of each word depends on N-1 previous words.

$$:= \prod_{i=1}^{n} P(w_i \mid w_{i-k+1} \dots w_{i-1})$$

Training a 2-gram Language Model

Given: a training dataset of M sentences $\mathscr{D} = \{\mathbf{w}\}_{i=1}^{M}$

Goal: Be able to estimate the probability of any sequence $\mathbf{w}$.

$$P(\mathbf{w}_1^n) = \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1})$$

$$= \prod_{i=1}^{n} P(w_i \mid w_{i-1}) \quad \textbf{(Bigram LM)}$$

We will estimate $P(w_i \mid w_{i-1})$ from the training data by:

$$P(w_i \mid w_{i-1}) = \frac{C(w_{i-1} w_i)}{C(w_{i-1})} \begin{array}{l} \longleftarrow \text{Bigram Counts} \\ \longleftarrow \text{Unigram counts} \end{array}$$

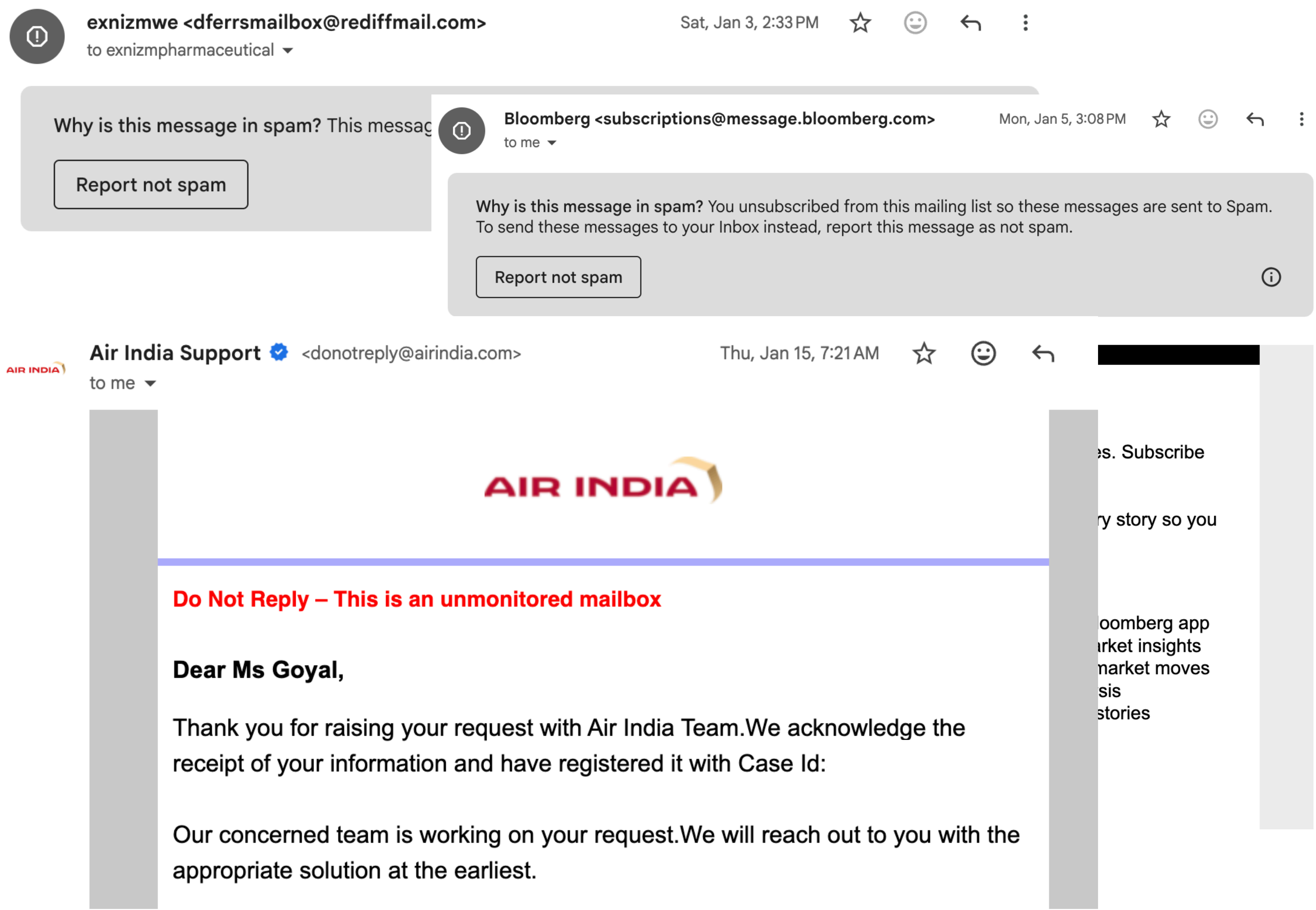
Look out for N-gram related questions in HW1

- Last lecture, we walked through an example of “training” an N-gram language model in class.
- Written component of HW1 will have other such questions + more conceptual questions about N-grams.

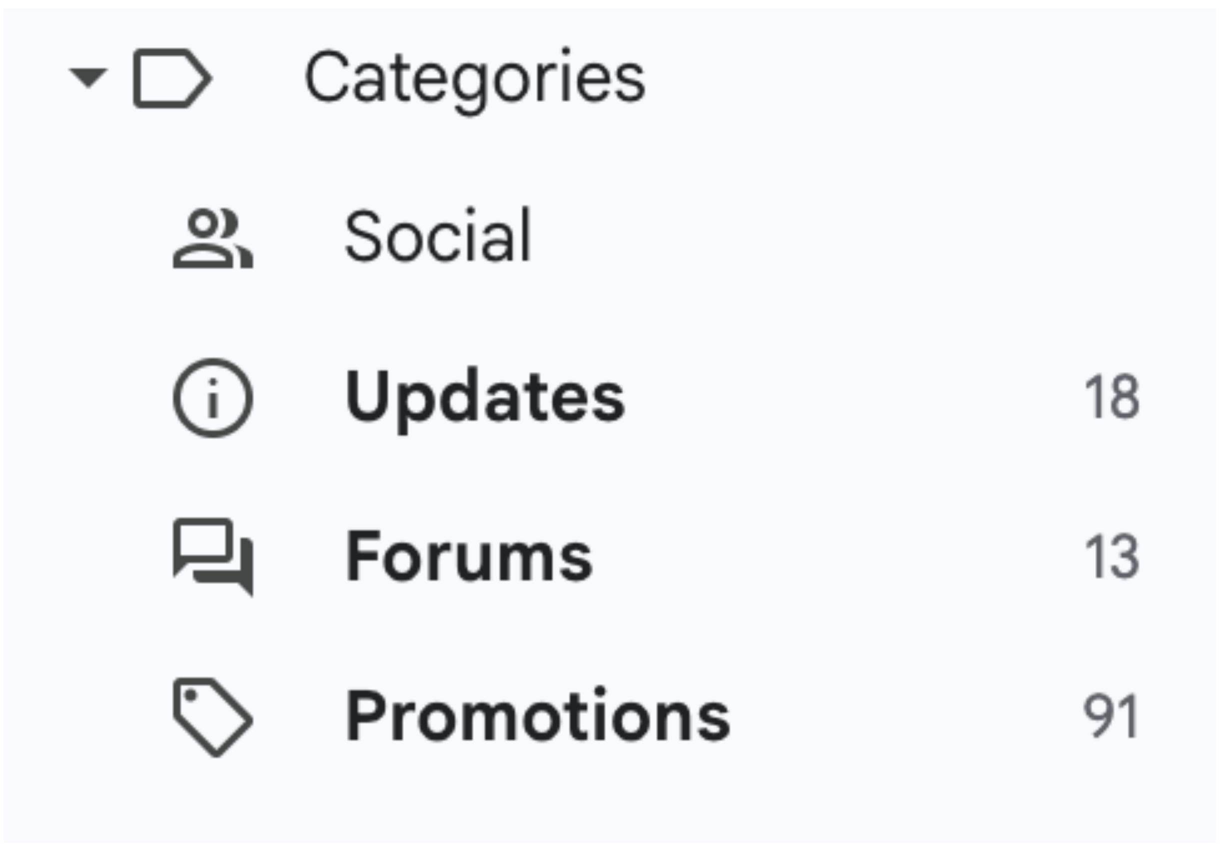
Today

- N-grams revisited.
- **Text Classification**
- Feature Engineering
- Binary Logistic Regression

Text Classification



- Gmail automatically detects which emails are spam vs "ham".
- Automatically classifies into pre-determined categories.



- All these are instances of text classification.

Text Classification

User queries to ChatGPT (safe vs unsafe)

"Help me design my personal website …." →

"Help me build a bomb …." → **Do not generate**

"How do I build a transformer library from scratch?" →

"How do I apply the binomial theorem to this problem…" →

"Generate a report justifying unequal pay for men and women…" → **Do not generate**

Binary Text Classification

Text Classification

Named Entity Recognition

In a given text input, identify all:

- Named locations, named persons, named organizations, dates, monetary amounts…
- Fixed set of NE types

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	The **Mt. Sanitas** loop is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states, provinces	**Palo Alto** is raising the fees for parking.
Facility	FAC	bridges, buildings, airports	Consider the **Golden Gate Bridge**.
Vehicles	VEH	planes, trains, automobiles	It was a classic **Ford Falcon**.

Figure 17.1 A list of generic named entity types with the kinds of entities they refer to.

Text Classification

Named Entity Recognition

In fact, the Chinese NORP market has the three CARDINAL most influential names of the retail and tech space – Alibaba GPE , Baidu ORG , and Tencent PERSON (collectively touted as BAT ORG), and is betting big in the global AI GPE in retail industry space . The three CARDINAL giants which are claimed to have a cut-throat competition with the U.S. GPE (in terms of resources and capital) are positioning themselves to become the 'future AI PERSON platforms'. The trio is also expanding in other Asian NORP countries and investing heavily in the U.S. GPE based AI GPE startups to leverage the power of AI GPE . Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing one CARDINAL , with an anticipated CAGR PERSON of 45% PERCENT over 2018 - 2024 DATE .

To further elaborate on the geographical trends, North America LOC has procured more than 50% PERCENT of the global share in 2017 DATE and has been leading the regional landscape of AI GPE in the retail market. The U.S. GPE has a significant credit in the regional trends with over 65% PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as Google ORG , IBM ORG , and Microsoft ORG .

- Each word is classified as one of {**NORP**, **PERSON**, **DATE**, **LOC**, **GPE**, **ORG**, ….. **NULL**}
- **NULL** used for words that don't correspond to Named Entities.
- How do we deal with multi-word named entities like "North America"?

Multi-Class Text Classification

Text Classification

- **Formally,**
  - Given a dataset of (x, y) pairs,
    - input: text x
    - output: a label y (from a finite set)
  - goal: learn a mapping function $P(y \mid x)$

In our NER example,
y = {PERSON, LOC, ORG, …, NULL}

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, ham}

Today

- N-grams revisited.
- Text Classification
- **Feature Engineering**
- **Binary Logistic Regression**

Classification

- **Formally,**
    - Given a dataset of (x, y) pairs,
    - **Goal**: learn a mapping function $P(y \mid x)$

Extract Features from x.

$f(x) =$ [#positive words, # negative words]

x= "The movie was great"

$f(x) =$ [1, 0]

x = "The movie was great"	**y** = 1
x = "The movie was terrible"	**y** = 0

- What are some "rules" we can use to make this labeling decision?
- Define "features" that are informative of the output label.

Classification

- **Formally,**
  - Given a dataset of (x, y) pairs,
  - **Goal**: learn a mapping function $P(y \mid x)$

x = "The movie was great"	**y** = 1
x = "The movie was terrible "	**y** = 0

Extract Features from x.

$f(x) =$ [#positive words, # negative words]

Goal: learn a mapping function $P(y \mid f(x))$

x = "The movie was great"

$f(x) =$ [1, 0]

Classification

- **Formally,**
  - Given a dataset of (x, y) pairs,
  - **Goal**: learn a mapping function $P(y \mid x)$

x = "The movie was great" **y** = 1

x = "The movie was terrible " **y** = 0

Extract Features from x.

$f(x) =$ [#positive words, # negative words]

x = "The movie was great"

$f(x) =$ [1, 0]

Feature Extraction

Goal: learn a mapping function $P(y \mid f(x))$

Learning Algorithm

In class, we will only learn the **binary logistic regression algorithm.**

Binary Logistic Regression Model

- **Formally,**
  - Given a dataset of (x, y) pairs,
  - **Goal**: learn a mapping function $P(y \mid f(x))$

y = {0, 1}

Let w be a vector of the same size as $f(x)$.

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

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

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

Binary Logistic Regression Model

- **Formally,**
  - Given a dataset of (x, y) pairs,

y = {0, 1}

  - **Goal**: learn a mapping function $P(y \mid f(x))$ → *learn weights w_i*

Let w be a vector of the same size as $f(x)$.

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

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

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

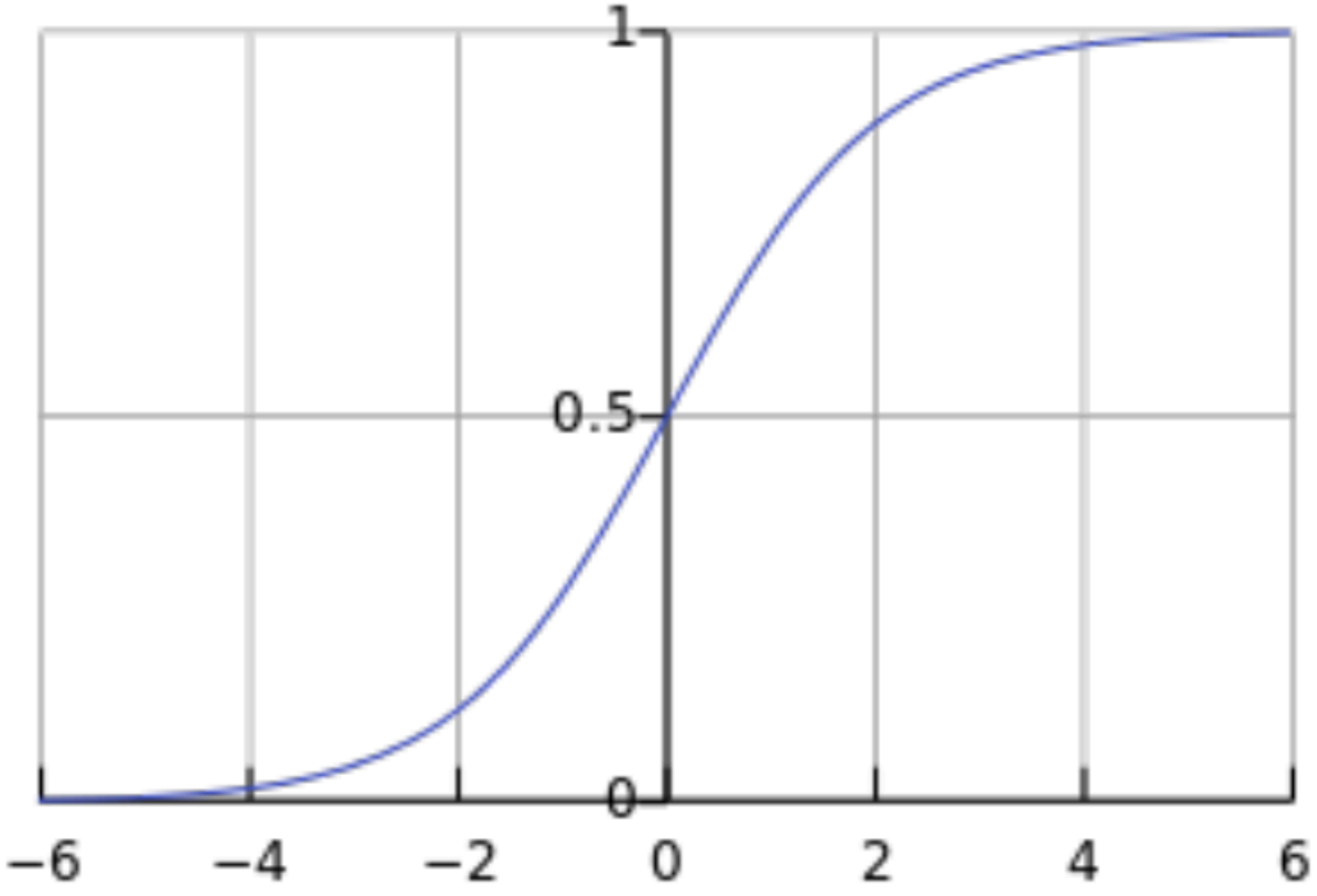
Properties of Logistic Function

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

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

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

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



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

Binary Logistic Regression Model

Sentiment Analysis

x = “The movie was great”

y = 1

Step1: Extract Features

$f =$

$f_0 = 1$ f_1 = #words f_2 = #“great”

f_3 = # positive words (from a pre-defined lexicon of positive words)

f_4 = # negative words (from a pre-defined lexicon of negative words)

~~f_5 = # adjectives~~ ~~f_6 = # “not”~~

~~f_7 = # “not” before a +ve word~~

....

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

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

$f_0 = 1$

$f_1 = \#\text{words}$

$f_2 = \#\text{"great"}$

$f_3 = \#\text{ positive words}$

$f_4 = \#\text{ negative words}$

Sentiment Analysis

$\mathbf{x}$ = "The movie was okay"

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

Step2: Dot product w. weights

$$f = ??$$

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

$$z = \sum_i f_i w_i = ??$$

Step3: Compute Probabilities

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

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

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,**

$(x^1 = \langle 1, 2, 1, -1, 3 \rangle, y^1 = 1)$

$(x^2 = \langle 1, -3, -2, -1, 4 \rangle, y^2 = 0)$

$(x^3 = \langle 1, -2, 0, -1, 3 \rangle, y^3 = 1)$

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

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

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

$$=$$

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

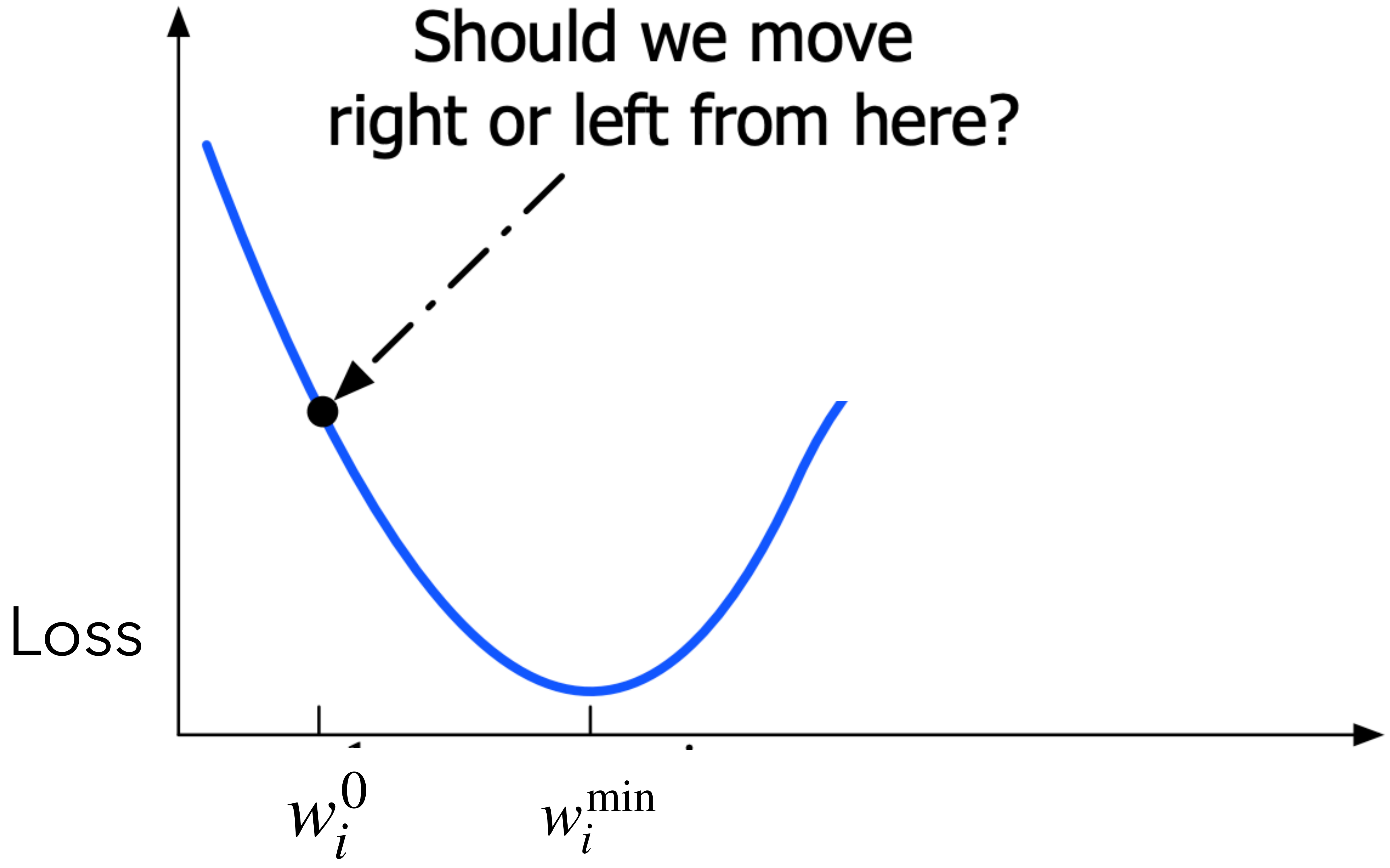
Log Loss L^j

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

Learning Weights

- Logistic regression loss function is convex → one minimum.

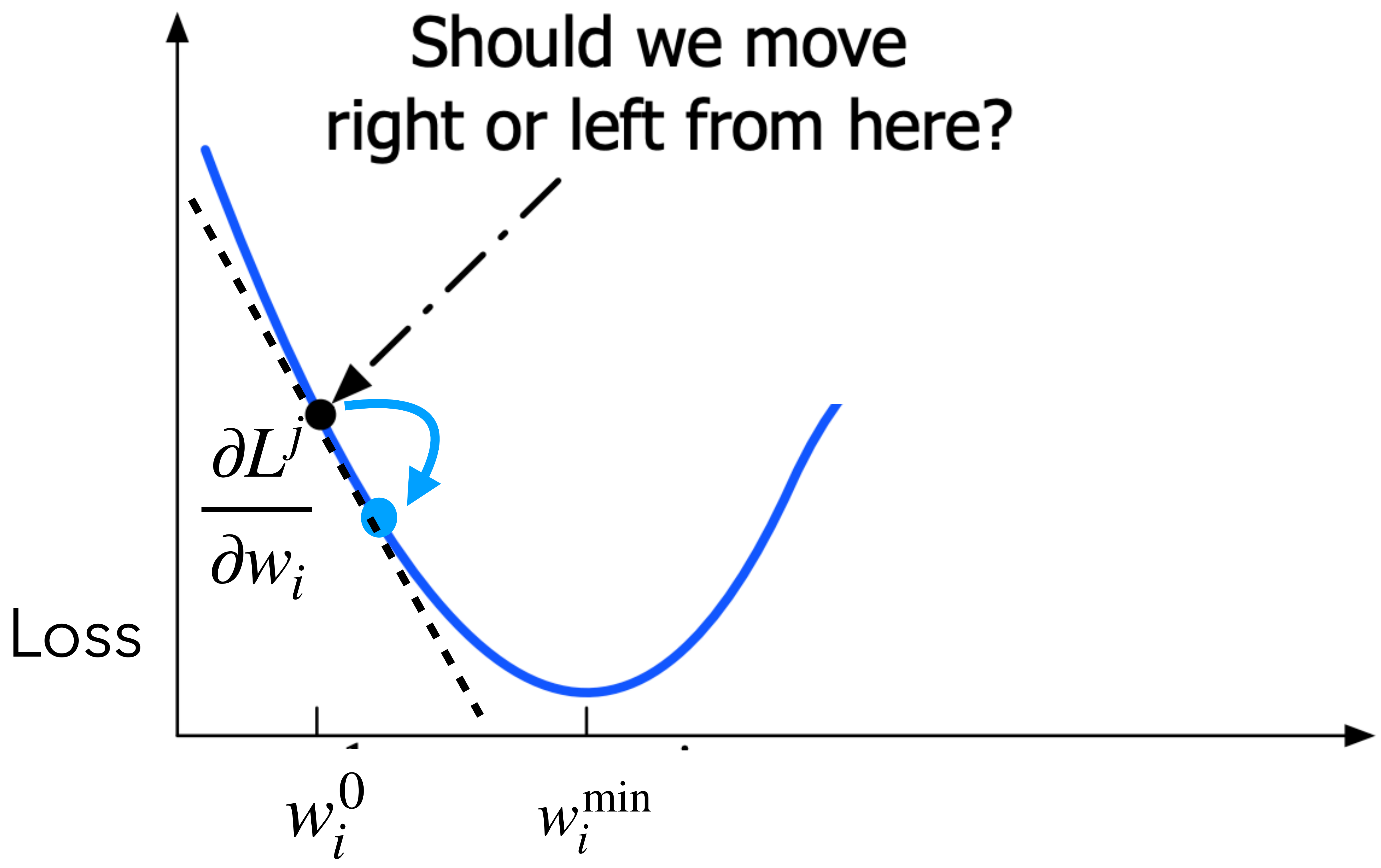
Visualizing one dim $\mathbf{w_i}$



Learning Weights

- Logistic regression loss function is convex → one minimum.

Visualizing one dim $\mathbf{w_i}$



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

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

Stochastic Gradient Descent

Initialize w^0

For e in range(0, #epochs)

For j in range(0, #num_datapoints):

Compute Loss L^j

Compute $\dfrac{\partial L^j}{\partial w_i} = \dfrac{\partial(-\log P(y = y^j \,|\, x^j))}{\partial w_i}$ for each weight w_i

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

$t = t + 1$

Learning Weights

Negative Log Likelihood

$$w^{\text{MLE}} = \arg\min_{w} \sum_{i=0}^{N} -\log P(y^i \,|\, x^i; w)$$

- $$\frac{\partial L^j}{\partial w_i} = \frac{\partial(-\log P(y = y^j \,|\, x^j))}{\partial w_i}$$

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 $y^j = 0$

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

Learning Weights

Negative Log Likelihood

$$w^{\text{MLE}} = \arg\min_w \sum_{i=0}^{N} -\log P(y^i \mid x^i; w)$$

- $$\frac{\partial L^j}{\partial w_i} = \frac{\partial(-\log P(y = y^j \mid x^j))}{\partial w_i}$$

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 $y^j = 0$

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

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

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

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

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)
- In hw1, we will train a logistic regression model + perform feature engineering for a binary classification task.

Slide Acknowledgements

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