CSE 4392 Special Topics
Natural Language Processing

Log-linear Models

2024 Spring

Last time

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

$$\hat{c} = \underset{c}{\arg\max}\ P(c)P(d|c)$$

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:

$$\hat{c} = \arg\max_{c \in C} P(c|d)$$

- Naive Bayes:

$$\hat{c} = \arg\max_{c \in C} P(c)\, P(d|c)$$

Quiz

- Given that we want to classify an image into either a dog or a cat (no other choices), name the features you would use (can be numerical or categorical).

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 $P(\hat{y} \mid \boldsymbol{x})$
  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 $\boldsymbol{x}$

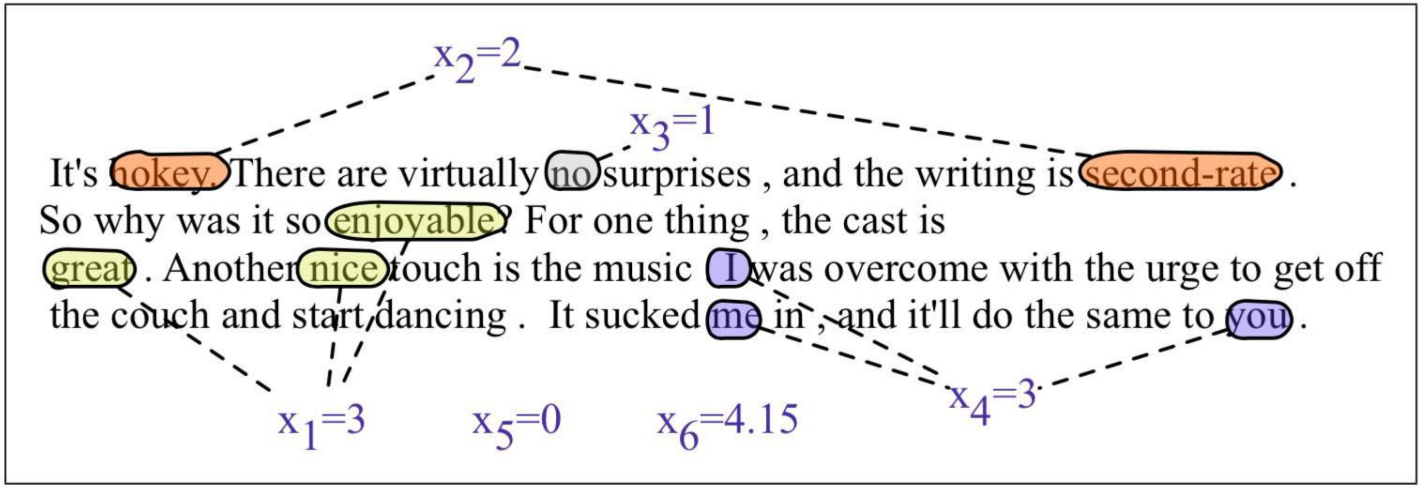
Feature Representation

- 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

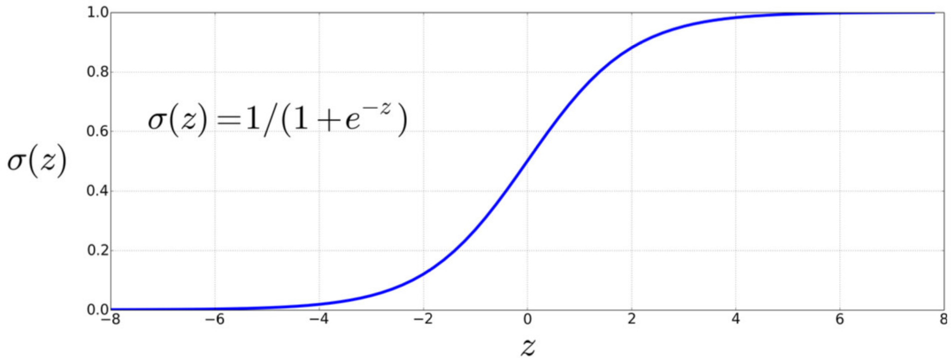
- *Given*: Input feature vector $[x_1, x_2, \ldots, x_d]$

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

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

- Sigmoid (or logistic) Function:

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-z)}$$



Quiz

- Why do we use Sigmoid/Logistic function as our classification function? (Select all that apply)

a) Produces a value between 0 and 1
b) A partial function with domain [0, +inf)
c) Produces a value between -1 to 1
d) A total function with domain (-inf, inf)
e) Integrates to 1 from –inf to inf
f) Differentiable

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 $\boldsymbol{x}$:

$$z = \sum_{i=1}^{d} w_i x_i + b$$

$$\text{or } z = \boldsymbol{w} \cdot \boldsymbol{x} + b$$

What is the Bias?

$$z = \boldsymbol{w} \cdot \boldsymbol{x} + b$$

- Bias, or intercept, gives the default behavior of the classifier when no useful information about x is known.
- Try setting x_i to be all 0:

$$z = b$$

- Gives the prior probability distribution of the classes without looking at the input features:

$$prediction\_bias = avg\_predictions - avg\ of\ labels\ in\ data\ set$$

PUTTING IT TOGETHER

- Given $\boldsymbol{x}$, compute $z = \boldsymbol{w} \cdot \boldsymbol{x} + b$
- Compute probabilities:

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

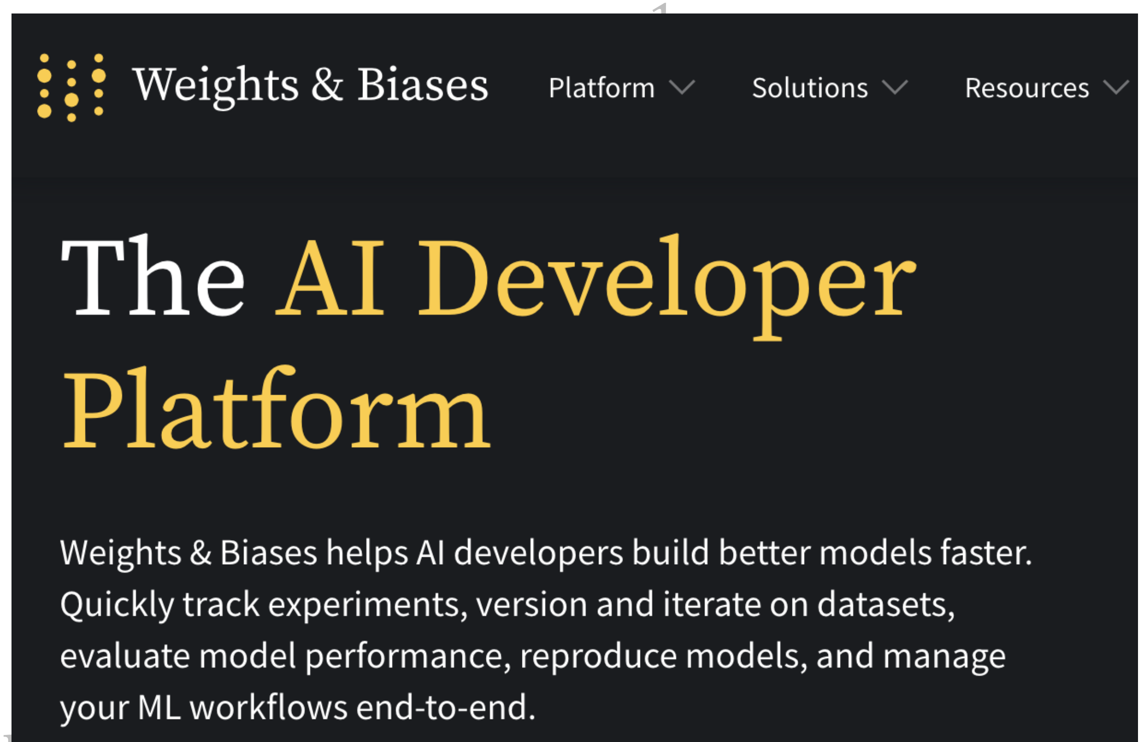
$$\begin{aligned} P(y = 1) &= \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b) \\ &= \frac{1}{1 + e^{-(\boldsymbol{w} \cdot \boldsymbol{x} + b)}} \\ P(y = 0) &= 1 - \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b) \\ &= 1 - \frac{1}{1 + e^{-(\boldsymbol{w} \cdot \boldsymbol{x} + b)}} \\ &= \frac{e^{-(\boldsymbol{w}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

- Decision boundary:

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

Putting it together

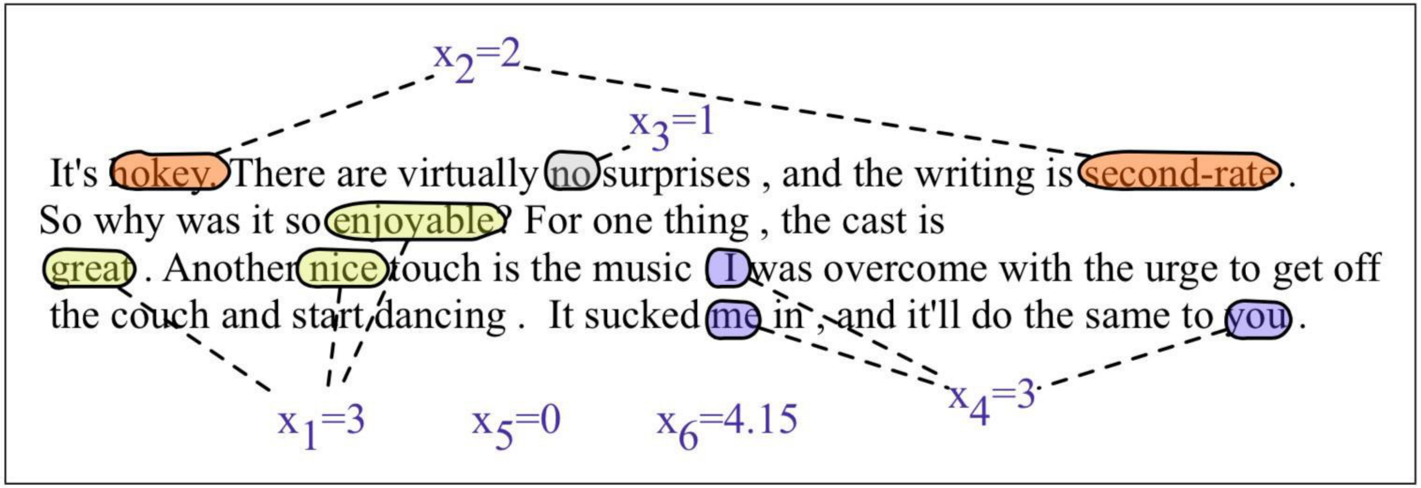
- Given $\boldsymbol{x}$, compute $z = \boldsymbol{w} \cdot \boldsymbol{x} + b$
- Compute probabilities:



- Decisio

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

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)	$\ln(64) = 4.15$

- Assume weights $\boldsymbol{w}$ = [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

- 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!)

Pros and Cons of Logistic Regression

- 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**: Naïve Bayes (NB) often better on very small datasets

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

- Assume $\hat{y} = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$
- $L(\hat{y}, y)$ = Measure of difference between $\hat{y}$ and y. But what form?
- Maximum likelihood estimation (conditional):
  - Choose w and b such that $\log P(y \mid \boldsymbol{x})$ 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
- Binary classifier probability (Bernoulli distribution):

$$P(y \mid x) = \hat{y}^{y}(1 - \hat{y})^{1-y}$$

- Log probability:

$$\begin{aligned} \log P(y|x) &= \log[\hat{y}^{y}(1 - \hat{y})^{1-y} \\ &= \text{y} \log \hat{y} + (1 - y) \log(1 - \hat{y}) \end{aligned}$$

- CE Loss (we want to minimize):

$$\begin{aligned} -\log P(y|x) &= -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) \\ &= \begin{cases} -\log \hat{y} & if\ y = 1 \\ -\log(1 - \hat{y}) & if\ y = 0 \end{cases} \end{aligned}$$

Cross Entropy Loss

- Assume n data points $(x^{(i)}, y^{(i)})$

- Classifier probability:

$$\Pi_{i=1}^{n} P(y \mid x) = \Pi_{i=1}^{n} \hat{y}^{y}(1-\hat{y})^{1-y}$$

(I omitted the (i) here for brevity)

- CE Loss:

$$L_{CE} = -\log \Pi_{i=1}^{n} P\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}\right)$$

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

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)	$\ln(64) = 4.15$

- 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), LCE = − log(0.69) = 0.37
- If y = 0 (negative sentiment), LCE = − log(0.31) = 1.17

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 $P(y \mid \boldsymbol{x})$ and predicted distribution $P(\hat{y} \mid \boldsymbol{x})$

OPTIMIZATION

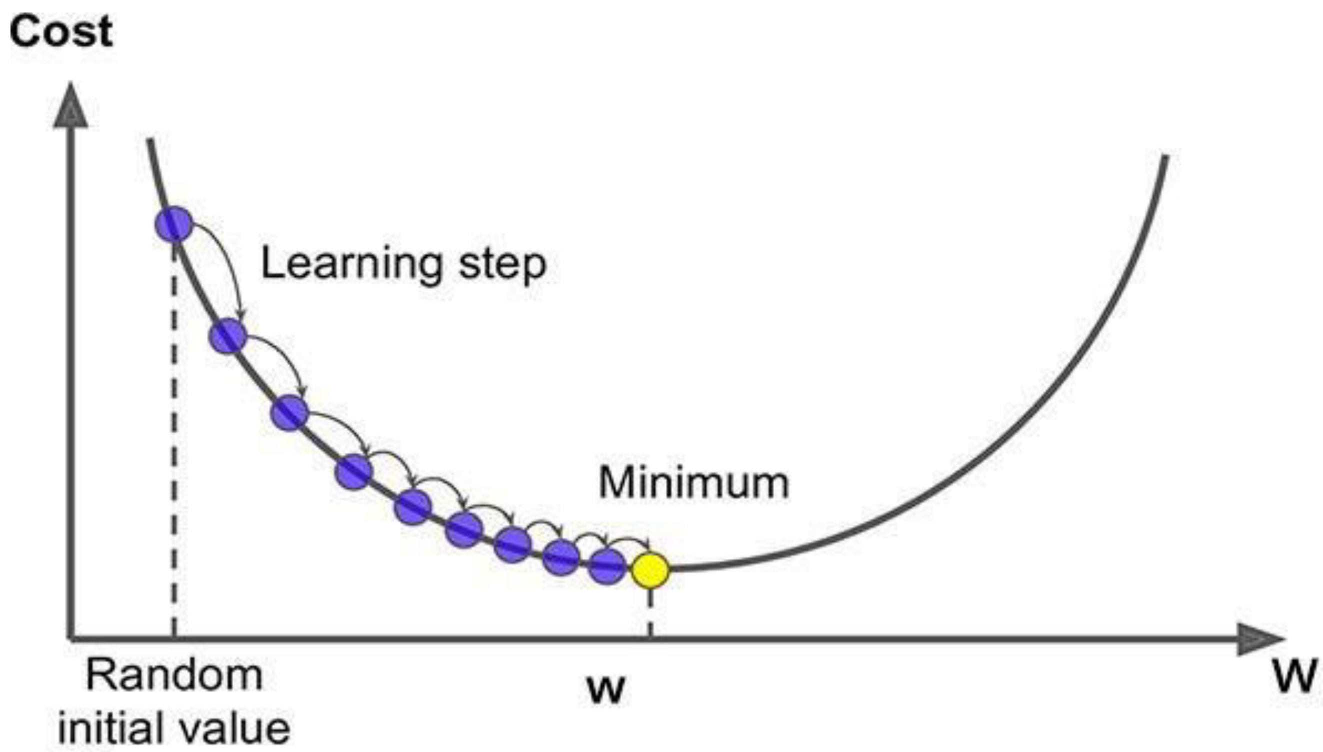
- We have our **classification function** and **loss function** - how do we find the best *w* and *b*?

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

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

- Gradient descent:
  - For a differentiable function *f*:
  - Find direction of steepest slope
  - Move in the opposite direction

GRADIENT DESCENT (1-D)



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

(f is differentiable)

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