

CSE 6331 Natural Language Processing

Log-linear Models

2026 Fall

Last time

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

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

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: Dog or Cat (2 pts)





- Given that we want to classify an animal into either a dog or a cat (no other choices), name 4 features you would use (can be numerical or categorical).

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 $P(\hat{y} \mid \boldsymbol{x})$
  3. **Loss function** (for learning)
  4. Optimization **algorithm**

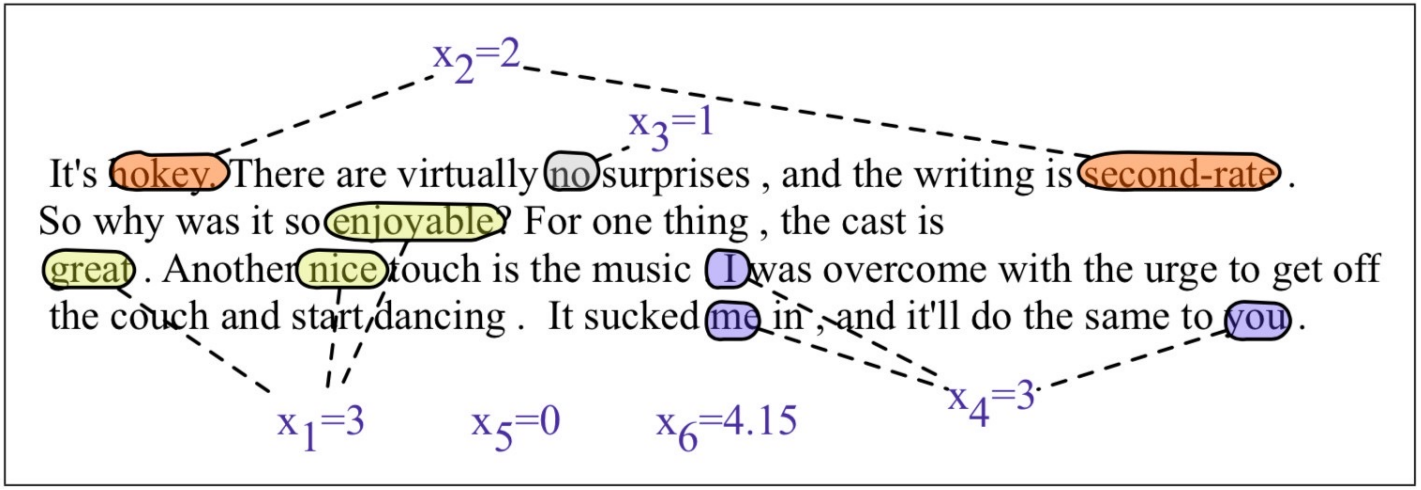
- Train phase:
  - Learn the **parameters** of the model to minimize **loss function**

- 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

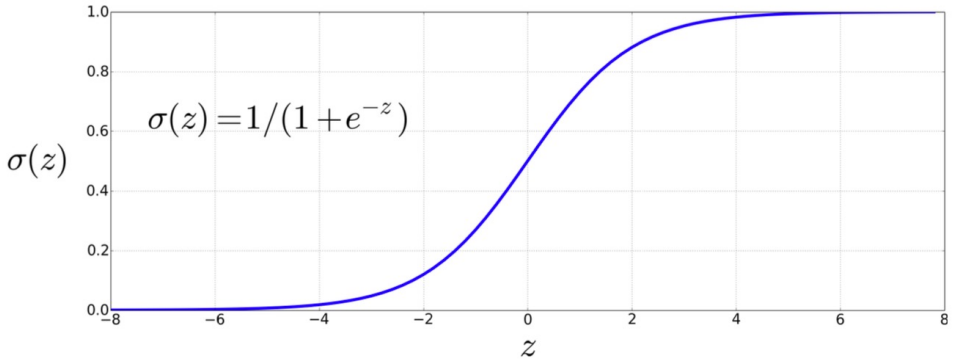


Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if ``no'' } \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if ``!'' } \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	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:

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



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:

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

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

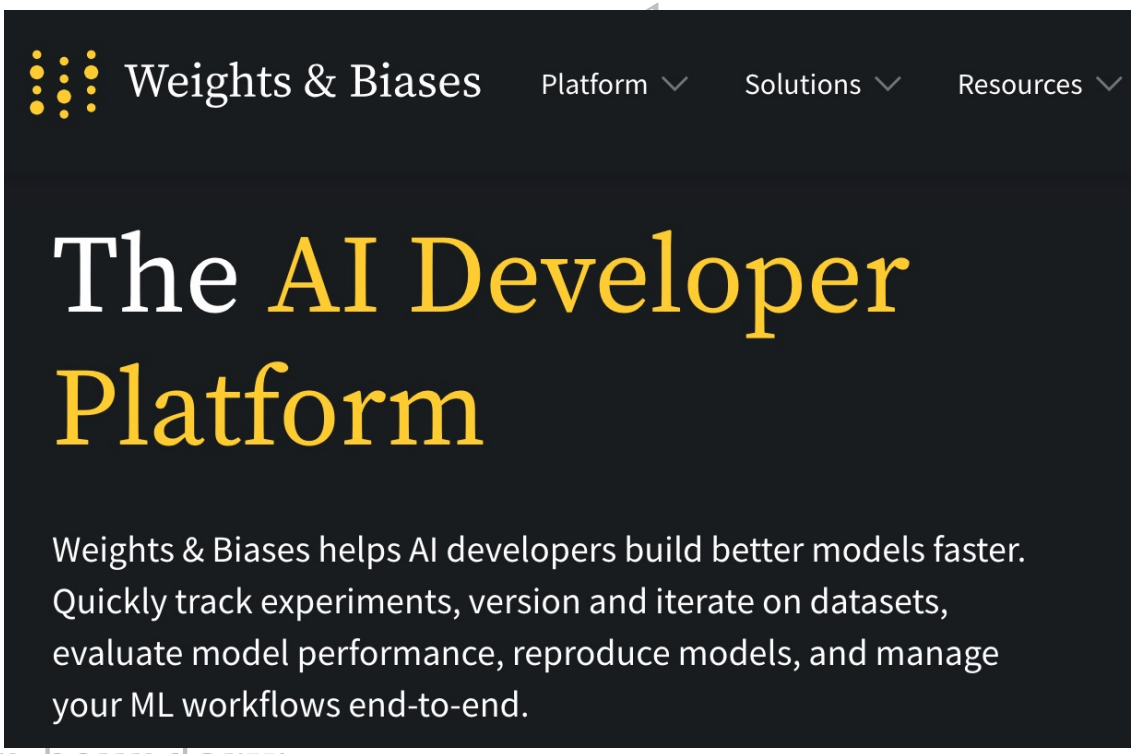
$$P(y = 0) = 1 - \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b) = 1 - \frac{1}{1 + e^{-(\boldsymbol{w} \cdot \boldsymbol{x} + b)}} = \frac{e^{-(\boldsymbol{w} \cdot \boldsymbol{x} + b)}}{1 + e^{-(\boldsymbol{w} \cdot \boldsymbol{x} + b)}}$$

- 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:



- Decision boundary:

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

Quiz: Sigmoid function (1 pt)

- 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)

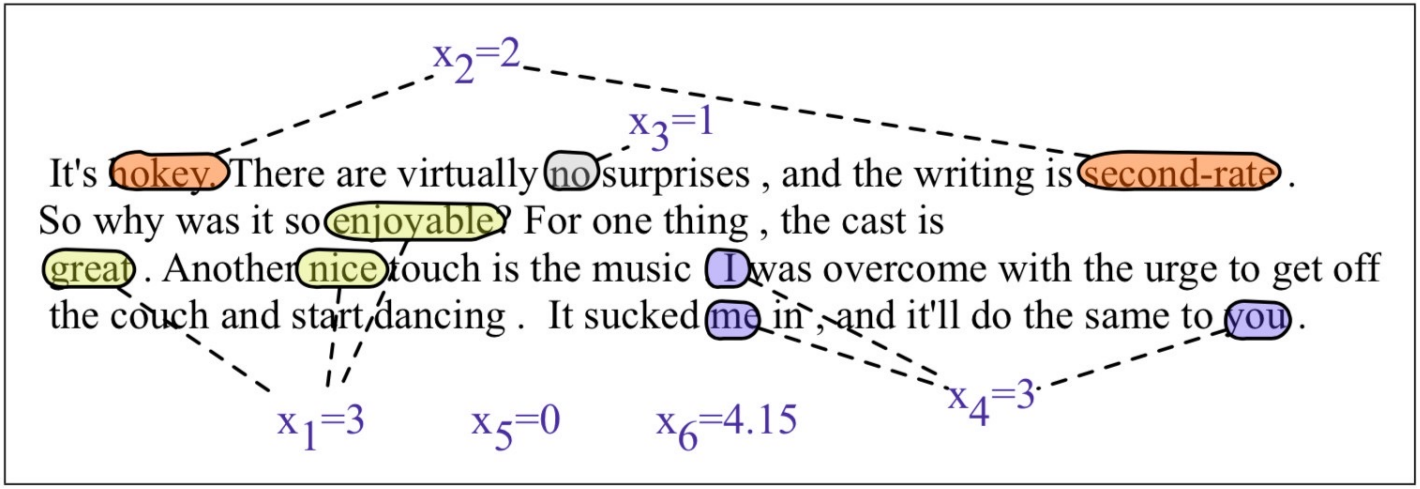
c) Produces a value between -1 to 1

d) A total function with domain (-inf, inf)

e) Integrates to 1 from –inf to inf

f) Differentiable

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

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

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

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

- 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!)

Quiz: Bias (3 pt)

- Suppose we are using a logistic regression classifier to classify an animal into either a cat (0) or a dog (1), and we know a priori that there are 9 times more cats than dogs (meaning the ratio between cats and dogs is 9:1). Can you compute the bias b using the following equations (show steps):

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

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!

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

Quiz: Logistic Regression (1 pt)

- Why can you have the same feature twice in logistic regression? Because:

a) the features are multiplied

b) the bias can offset the excessive feature values

c) the features are independent from each other

d) the features are weighted and summed

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 (note that $\hat{y}$ is real while y is binary)
  - **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):

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

  - Remember $0 \leq \hat{y} \leq 1$.

- Log probability:

$$\log P(y|x) = \log[\hat{y}^{y}\ (1 - \hat{y})^{1-y}$$
$$= \mathrm{y} \log \hat{y} + (1 - y) \log(1 - \hat{y})$$

- CE Loss (we want to minimize):

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

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

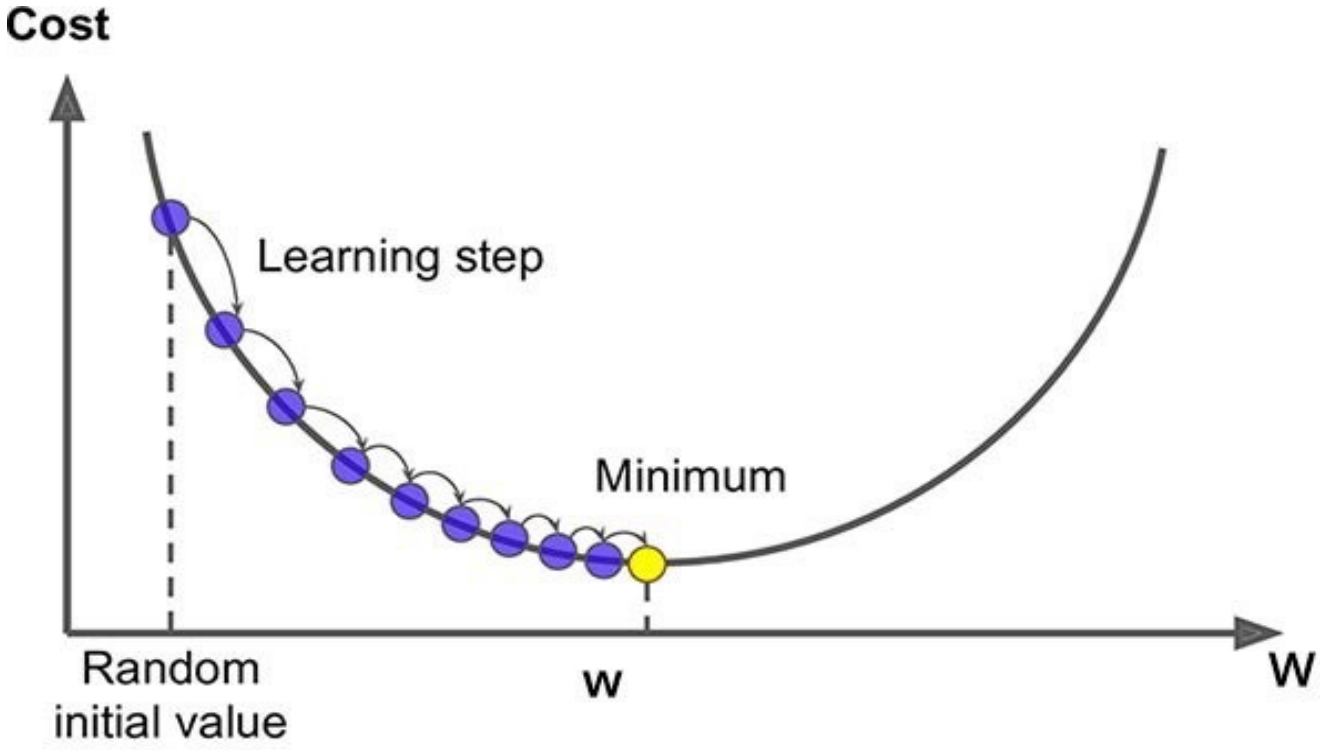
- We have our **classification function** and **loss function** - how do we find the best w and b?

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

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

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

GRADIENT DESCENT (1-D)



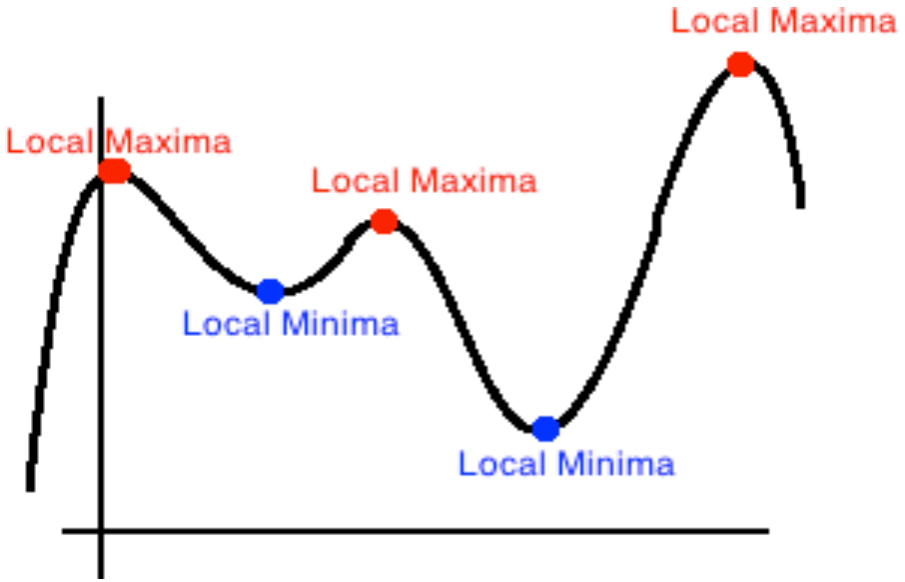
$$\theta^{t+1} = \theta^t - \eta \frac{d}{d\theta} f(x;\theta)$$

(f is differentiable and θ is the w in the figure above)

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

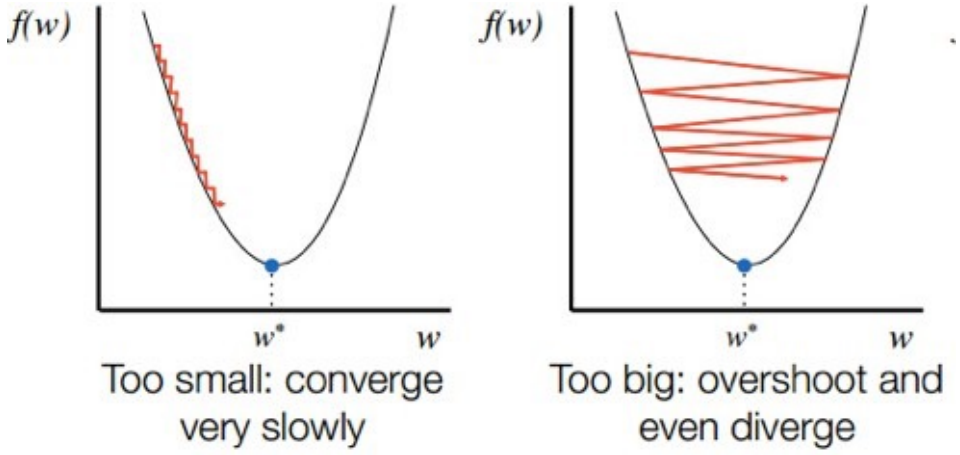


Learning Rate

- 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



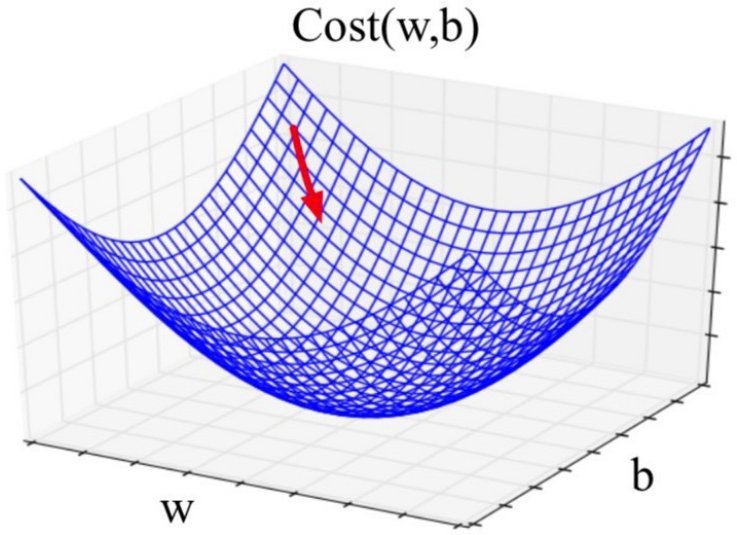
Too small: converge very slowly

Too big: overshoot and even diverge

GRADIENT DESCENT WITH VECTOR WEIGHTS

- In LR: weight $\boldsymbol{w}$ 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 $\boldsymbol{w}$ 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

- Cross entropy loss:

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

- Gradient:

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

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

- Recall:

$$\frac{d}{dx} \ln(x) = \frac{1}{x} \qquad \frac{d\sigma(z)}{dz} = \sigma(z)(1 - \sigma(z))$$

Quiz: Derivative of CE Loss (5 pts)

- Given that: $\frac{d}{dx}\ln(x) = \frac{1}{x}$ $\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$

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

Derive (showing steps) that:

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

Gradient for logistic regression

- Cross entropy loss:

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

- Gradient:

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

input feature value

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

$$\frac{dL_{CE}(w,b)}{dw_j} = \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)}) x_j^{(i)}$$

Stochastic Gradient Descent

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

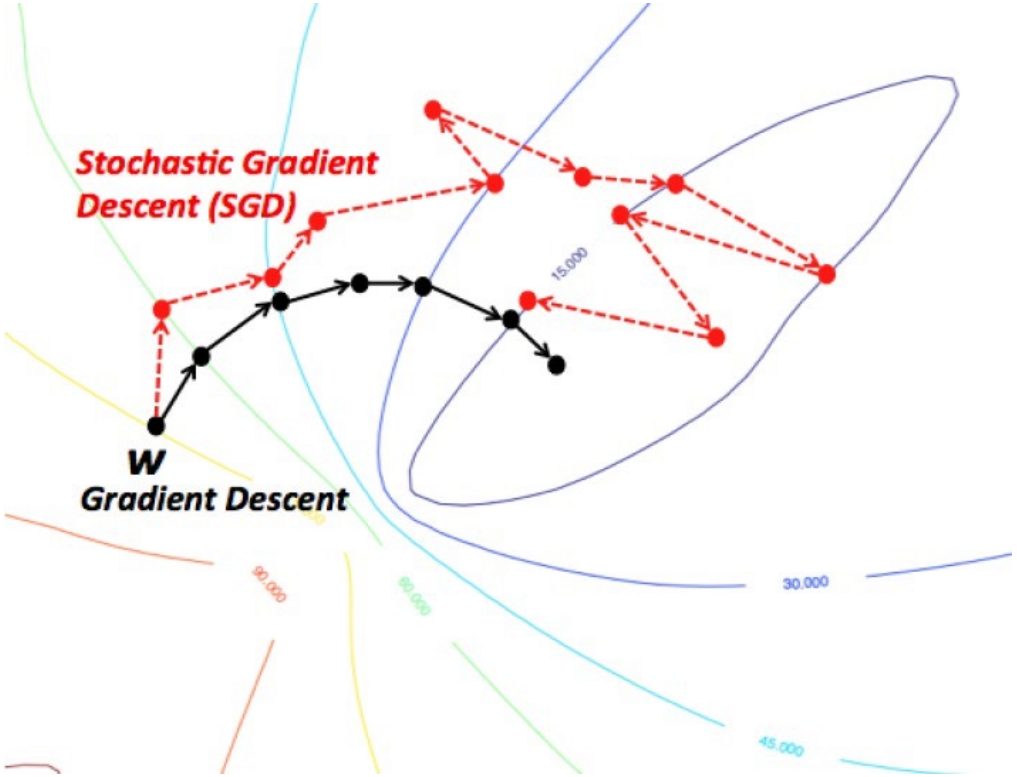
function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ

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

θ ← 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 ŷ^(i) = f(x^(i); θ)        # What is our estimated output ŷ?
       Compute the loss L(ŷ^(i), y^(i))   # How far off is ŷ^(i)) from the true output y^(i)?
    2. g ← ∇_θ L(f(x^(i); θ), y^(i))      # How should we move θ to maximize loss?
    3. θ ← θ − η g                        # Go the other way instead
return θ
```

Stochastic Gradient Descent

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



REGULARIZAATION

- 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

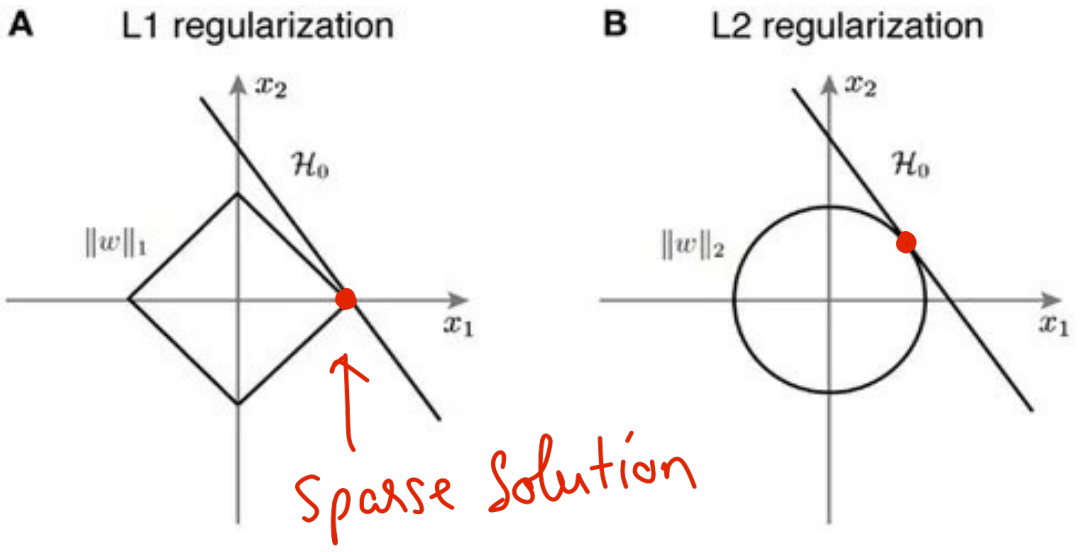
$$R(\theta) = ||\theta||_1 = \sum_{j=1}^{d} |\theta_j|$$

- 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 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

- 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

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

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

$$\log P(y = c \mid \boldsymbol{x}) \propto \boldsymbol{w_c} \cdot \boldsymbol{x} + b_c$$ (log-linear!)

Features in multinomial LR

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

Var	Definition	Wt
$f_1(0,x)$	1 if "!" $\in$ doc 0 otherwise	−4.5
$f_1(+,x)$	1 if "!" $\in$ doc 0 otherwise	2.6
$f_1(-,x)$	1 if "!" $\in$ doc 0 otherwise	1.3

Learning

- Generalize binary CE loss to multinomial CE loss:

$$
\begin{aligned}
L_{\text{CE}}(\hat{\mathbf{y}}, \mathbf{y}) &= -\sum_{k=1}^{K} \mathbf{y}_k \log \hat{\mathbf{y}}_k \\
&= -\log \hat{\mathbf{y}}_c, \quad \text{(where } c \text{ is the correct class)} \\
&= -\log \hat{p}(\mathbf{y}_c = 1 | \mathbf{x}) \quad \text{(where } c \text{ is the correct class)} \\
&= -\log \frac{\exp(\mathbf{w_c} \cdot \mathbf{x} + b_c)}{\sum_{j=1}^{K} \exp(\mathbf{w_j} \cdot \mathbf{x} + b_j)} \quad (c \text{ is the correct class})
\end{aligned}
$$

- Gradient:

$$
\begin{aligned}
\frac{\partial L_{\text{CE}}}{\partial \mathbf{w}_{k,i}} &= -(\mathbf{y}_k - \hat{\mathbf{y}}_k)\mathbf{x}_i \\
&= -(\mathbf{y}_k - p(\mathbf{y}_k = 1 | \mathbf{x}))\mathbf{x}_i \\
&= -\left(\mathbf{y}_k - \frac{\exp(\mathbf{w_k} \cdot \mathbf{x} + b_k)}{\sum_{j=1}^{K} \exp(\mathbf{w_j} \cdot \mathbf{x} + b_j)}\right)\mathbf{x}_i
\end{aligned}
$$