Binary Classification

This Lecture

- Two discriminative models:
  - logistic regression
  - perceptron

Classification

Classification

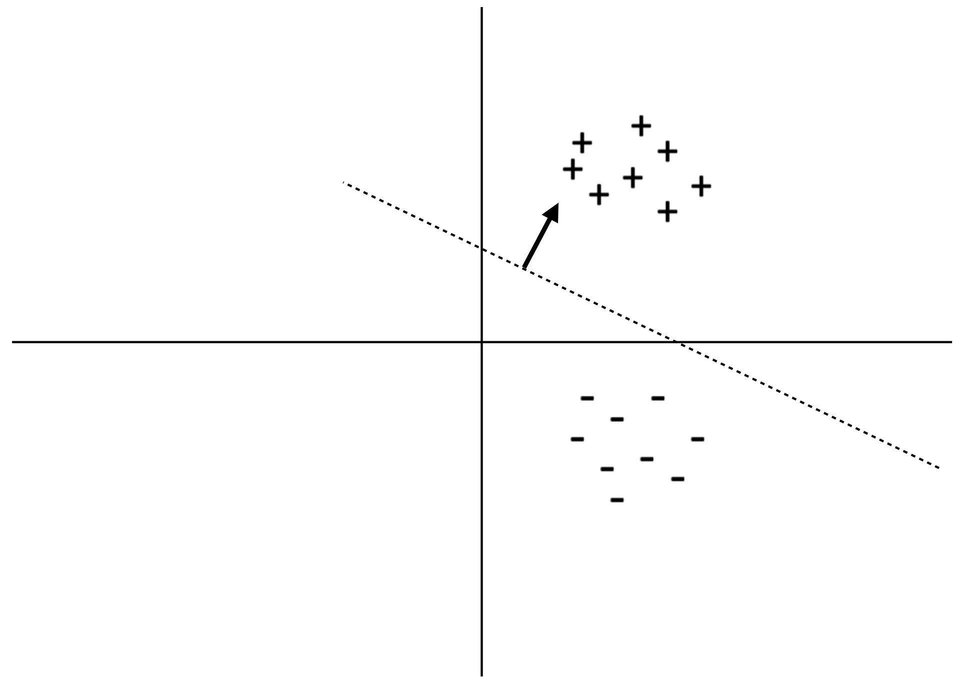
- Datapoint x with label $y \in \{0, 1\}$

Classification

- Datapoint x with label $y \in \{0, 1\}$
- Embed datapoint in a feature space $f(x) \in \mathbb{R}^n$

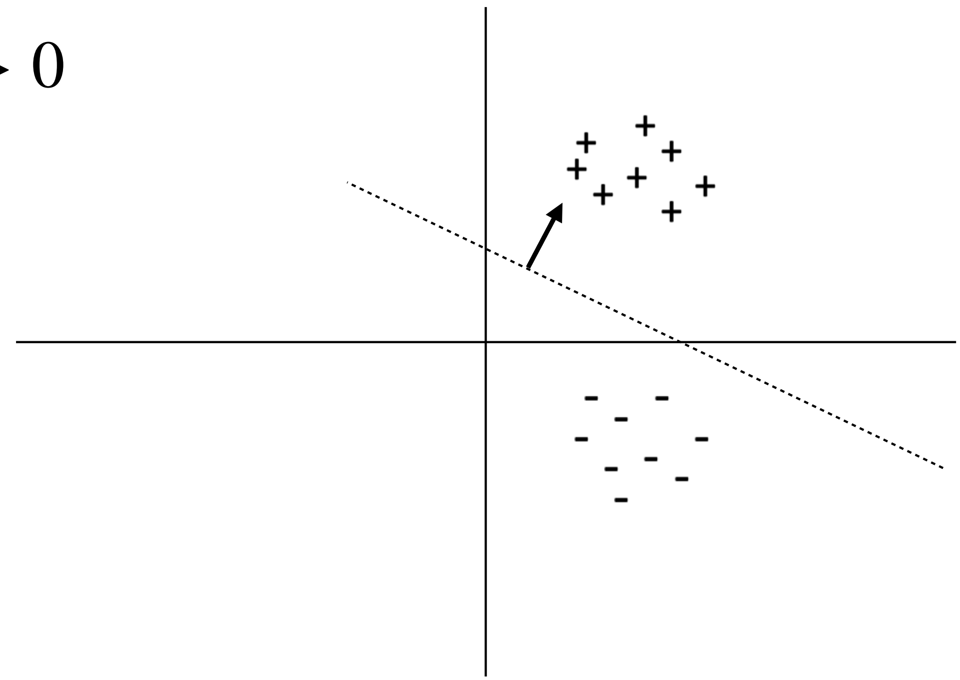
Classification

- Datapoint x with label $y \in \{0, 1\}$
- Embed datapoint in a feature space $f(x) \in \mathbb{R}^n$



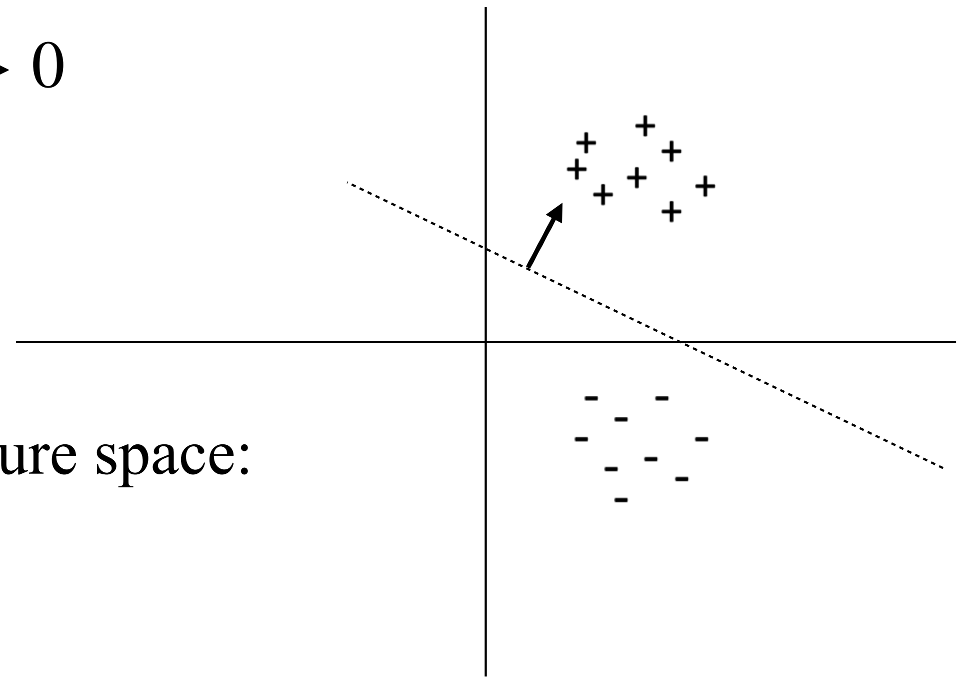
Classification

- Datapoint x with label y $\in \{0, 1\}$
- Embed datapoint in a feature space $f(x) \in \mathbb{R}^n$
- Linear decision rule: $w^T f(x) + b > 0$



Classification

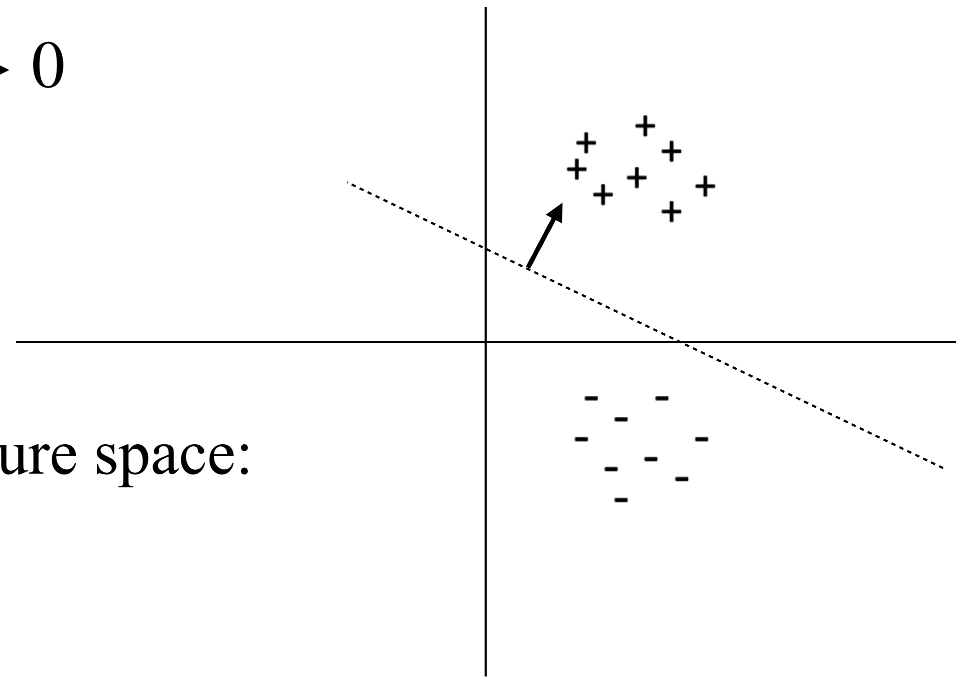
- Datapoint x with label $y \in \{0, 1\}$
- Embed datapoint in a feature space $f(x) \in \mathbb{R}^n$
- Linear decision rule: $w^T f(x) + b > 0$



- Can delete bias if we augment feature space:

Classification

- Datapoint x with label y $\in$ {0 , 1 }
- Embed datapoint in a feature space $f(x) \in \mathbb{R}^n$
- Linear decision rule: $w^T f(x) + b > 0$



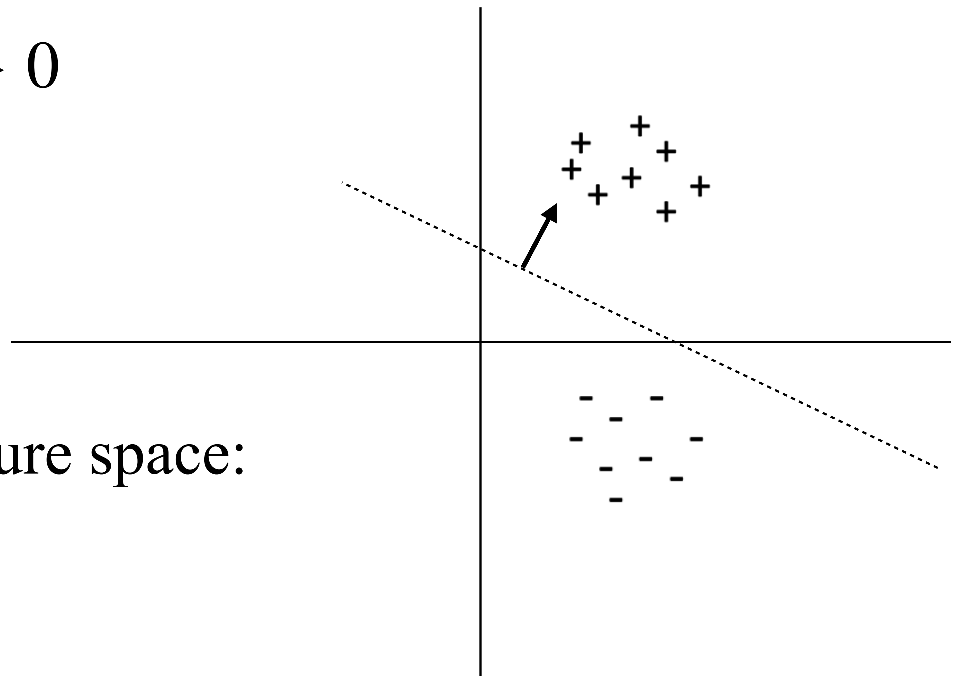
- Can delete bias if we augment feature space:
  f (x) = [0.5, 1.6, 0.3]

Classification

- Datapoint x with label $y \in \{0, 1\}$

- Embed datapoint in a feature space $f(x) \in \mathbb{R}^n$
- Linear decision rule: $w^T f(x) + b > 0$

$$w^T f(x) > 0$$



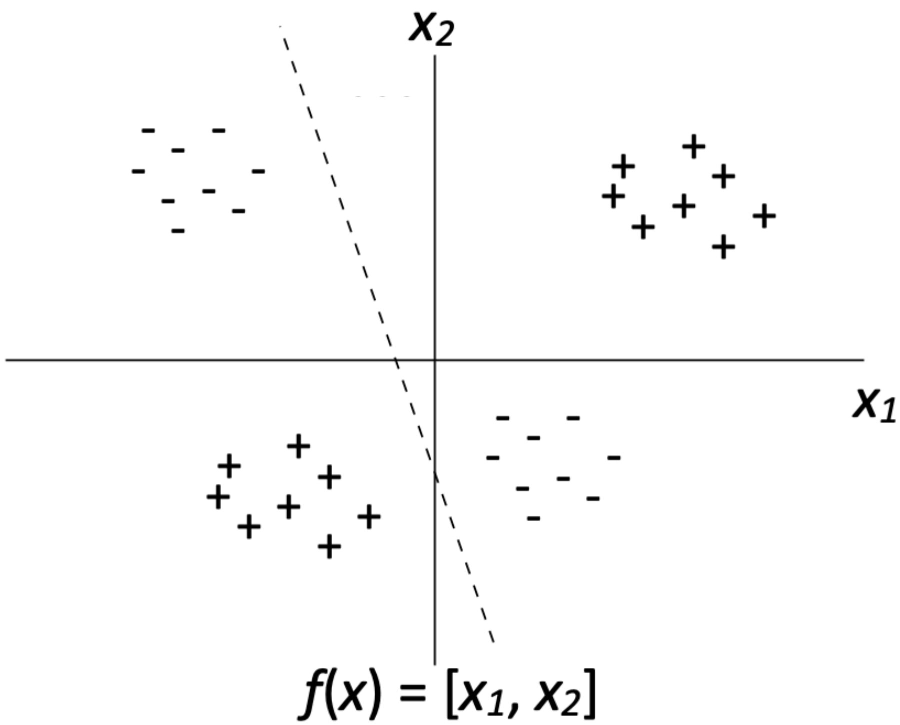
- Can delete bias if we augment feature space:

f (x) = [0.5, 1.6, 0.3]

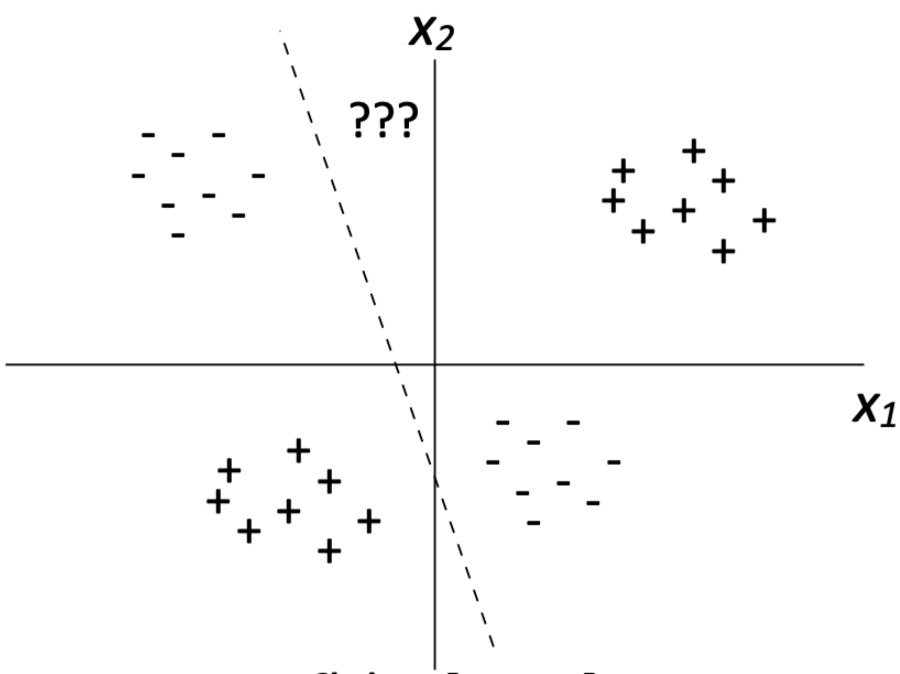
↓

[0.5, 1.6, 0.3, **1**]

Linear functions are powerful!

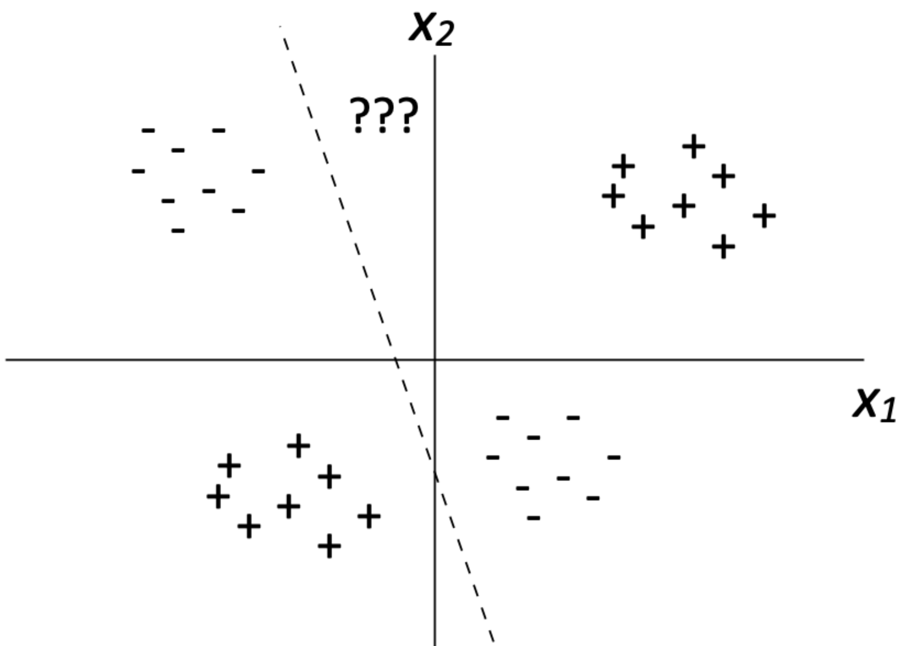


Linear functions are powerful!



$f(x) = [x_1, x_2]$

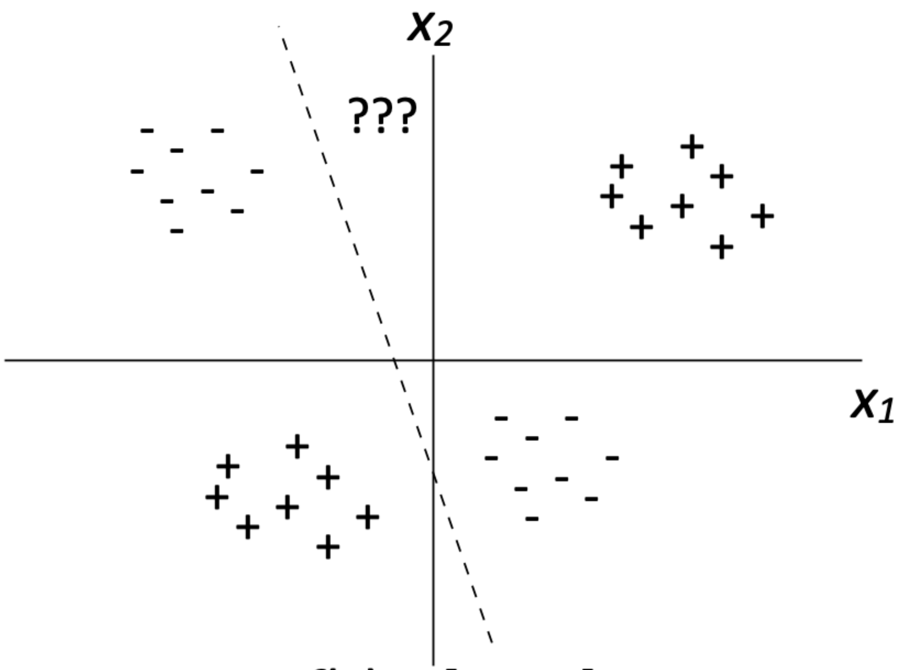
Linear functions are powerful!



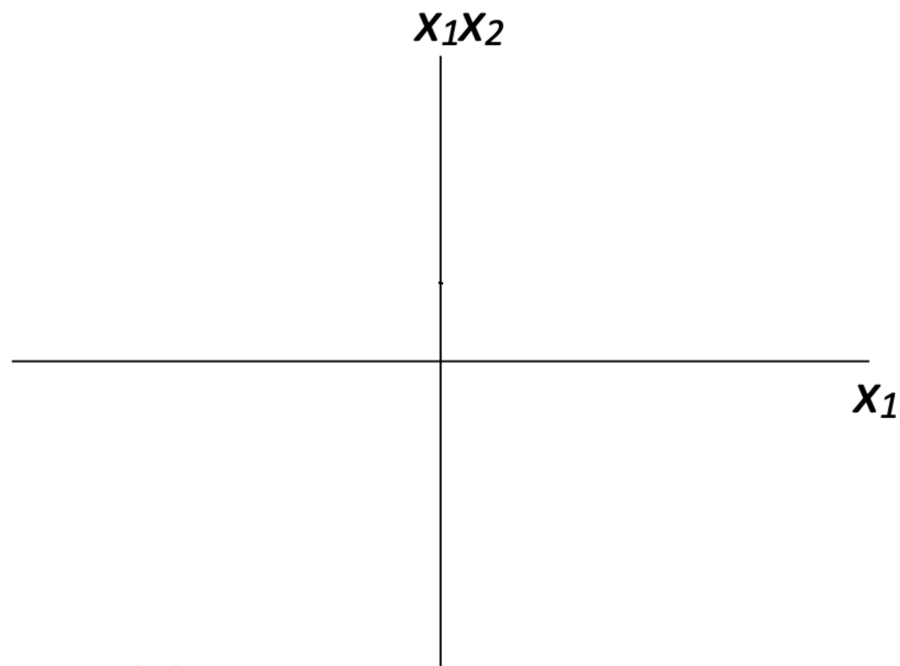
$f(x) = [x_1, x_2]$

$f(x) = [x_1, x_2, x_1^2, x_2^2, x_1x_2]$

Linear functions are powerful!

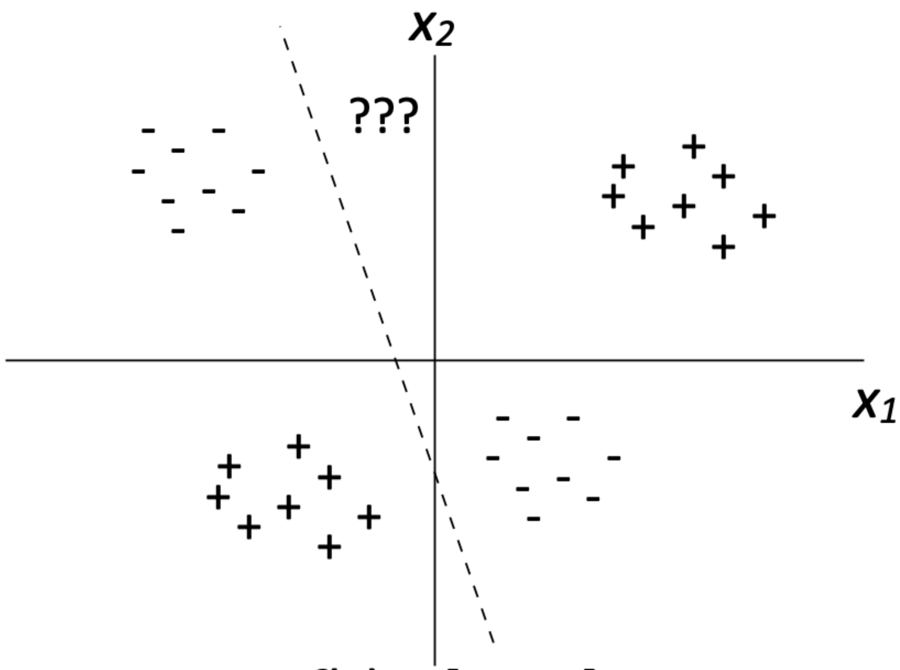


$f(x) = [x_1, x_2]$

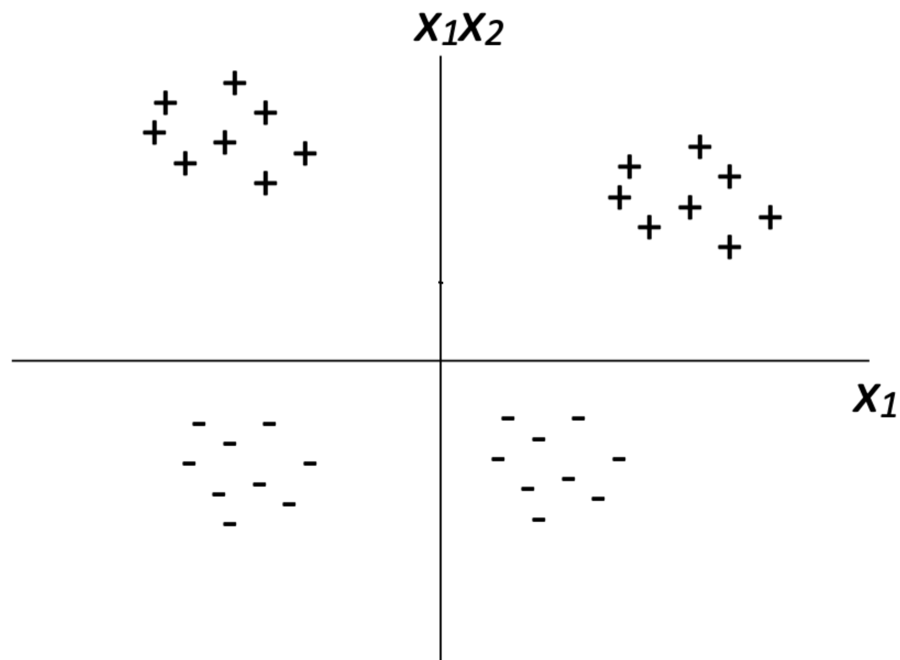


$f(x) = [x_1, x_2, x_1^2, x_2^2, x_1x_2]$

Linear functions are powerful!

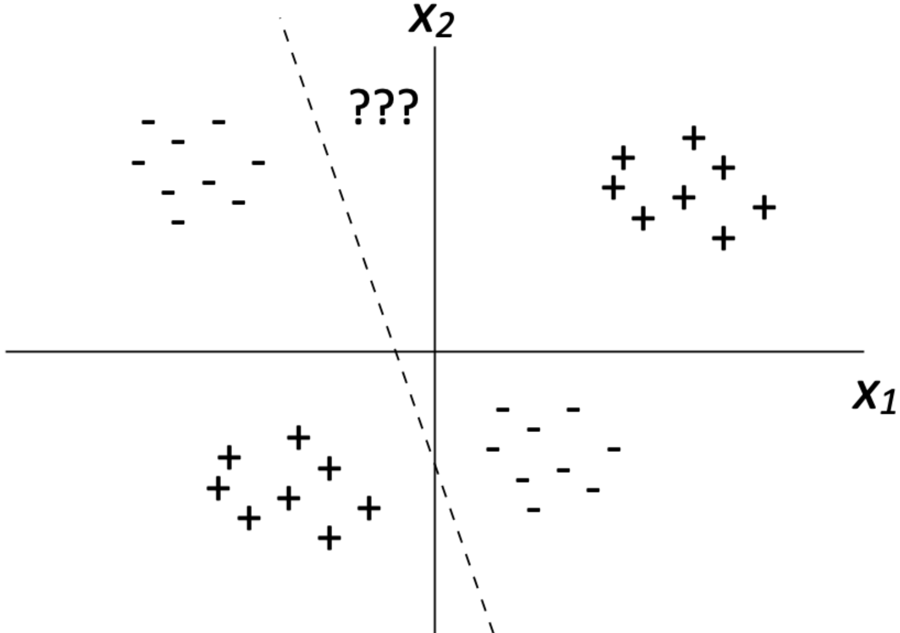


$f(x) = [x_1, x_2]$

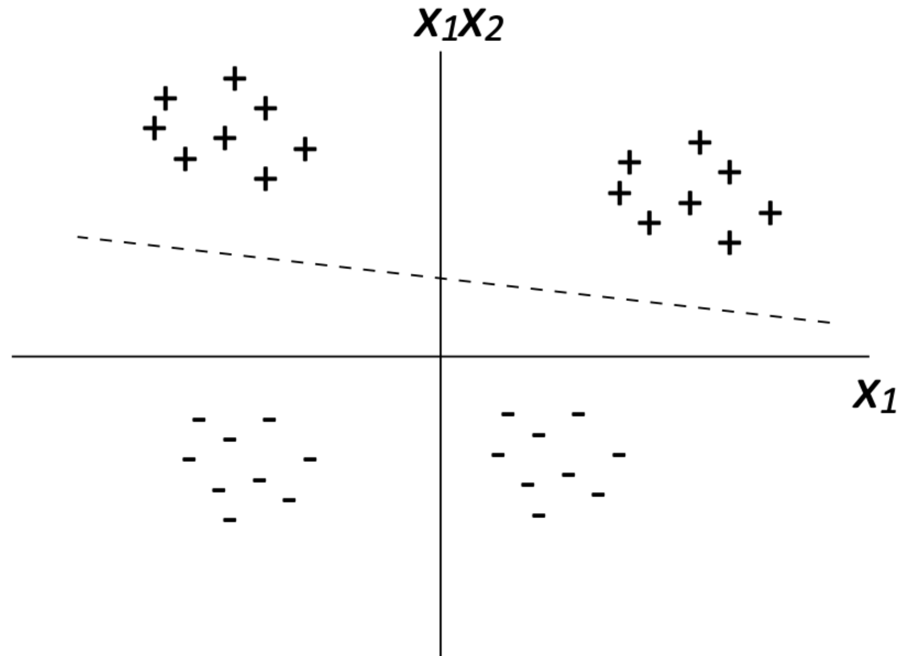


$f(x) = [x_1, x_2, x_1^2, x_2^2, x_1x_2]$

Linear functions are powerful!



$f(x) = [x_1, x_2]$



$f(x) = [x_1, x_2, x_1^2, x_2^2, x_1x_2]$

Classification: Sentiment Analysis

this movie was great! would watch again Positive

that film was awful, I'll never watch again Negative

Classification: Sentiment Analysis

this movie was great! would watch again Positive

that film was awful, I'll never watch again Negative

- Surface cues can basically tell you what's going on here:
    - presence or absence of certain words (*great*, *awful*)

Classification: Sentiment Analysis

this movie was great! would watch again Positive

that film was awful, I'll never watch again Negative

- Surface cues can basically tell you what's going on here:
  - presence or absence of certain words (*great*, *awful*)

- Steps to classification:
  - Turn examples like this into feature vectors
  - Pick a model / learning algorithm
  - Train weights on data to get our classifier

Feature Representation

this movie was great! would watch again Positive

Feature Representation

this movie was great! would watch again   Positive

- Convert this example to a vector using *bag-of-words features*

Feature Representation

this movie was great! would watch again   Positive

- Convert this example to a vector using *bag-of-words features*

[contains *the*]   [contains *a*]   [contains *was*]   [contains *movie*]   [contains *film*] …

feature 0   feature 1   feature 2   feature 3   feature 4

Feature Representation

this movie was great! would watch again   Positive

- Convert this example to a vector using *bag-of-words features*

[contains *the*]	[contains *a*]	[contains *was*]	[contains *movie*]	[contains *film*]	…
feature 0	feature 1	feature 2	feature 3	feature 4	
$f(x) = [$ 0	0	1	1	0	…

Feature Representation

this movie was great! would watch again Positive

- Convert this example to a vector using *bag-of-words features*

[contains *the*]	[contains *a*]	[contains *was*]	[contains *movie*]	[contains *film*]	…
feature 0	feature 1	feature 2	feature 3	feature 4	
$f(x) = [$ 0	0	1	1	0	…

- Very large vector space (size of vocabulary), sparse features

Feature Representation

this movie was great! would watch again Positive

- Convert this example to a vector using *bag-of-words features*

[contains *the*]	[contains *a*]	[contains *was*]	[contains *movie*]	[contains *film*]	…
feature 0	feature 1	feature 2	feature 3	feature 4	
$f(x) = [\ 0$	0	1	1	0	…

- Very large vector space (size of vocabulary), sparse features
- Requires *indexing* the features (mapping them to axes)

Feature Representation

this movie was great! would watch again Positive

- Convert this example to a vector using *bag-of-words features*

[contains *the*]	[contains *a*]	[contains *was*]	[contains *movie*]	[contains *film*]	…
feature 0	feature 1	feature 2	feature 3	feature 4	
$f(x) = [\ 0$	0	1	1	0	…

- Very large vector space (size of vocabulary), sparse features
- Requires *indexing* the features (mapping them to axes)
- More sophisticated feature mappings possible:
    - character n-grams, parts of speech, lemmas, …

Maximum Likelihood Estimation

Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (j indexes over examples)

Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (j indexes over examples)
- Find values of $P(y)$, $P(x_j \mid y)$ that maximized data likelihood:

Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (j indexes over examples)
- Find values of $P(y)$, $P(x_j \mid y)$ that maximized data likelihood:

$$\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[\prod_{i=1}^{n} P(x_{ji} | y_j) \right]$$

Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (j indexes over examples)
- Find values of $P(y)$, $P(x_j \mid y)$ that maximized data likelihood:

$$\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[\prod_{i=1}^{n} P(x_{ji} | y_j) \right]$$

data point (*j*)

Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (j indexes over examples)
- Find values of $P(y)$, $P(x_j \mid y)$ that maximized data likelihood:

$$\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[\prod_{i=1}^{n} P(x_{ji} | y_j) \right]$$

data point (*j*)

features (i)

Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (j indexes over examples)
- Find values of $P(y)$, $P(x_j \mid y)$ that maximized data likelihood:

$$\underbrace{\prod_{j=1}^{m}}_{\text{data point } (j)} P(y_j, x_j) = \underbrace{\prod_{j=1}^{m}}_{\text{data point } (j)} P(y_j) \left[\underbrace{\prod_{i=1}^{n}}_{\text{features (i)}} P(\underbrace{x_{ji}}_{\text{i-th feature of j-th example}} | y_j) \right]$$

Maximum Likelihood Estimation

- Imagine a coin flip which is heads with probability p

Maximum Likelihood Estimation

- Imagine a coin flip which is heads with probability p
- Observe (H, H, H, T)

Maximum Likelihood Estimation

- Imagine a coin flip which is heads with probability *p*
- Observe (H, H, H, T) and maximize likelihood: $\prod_{j=1}^{m} P(y_j) = p^3(1-p)$

Maximum Likelihood Estimation

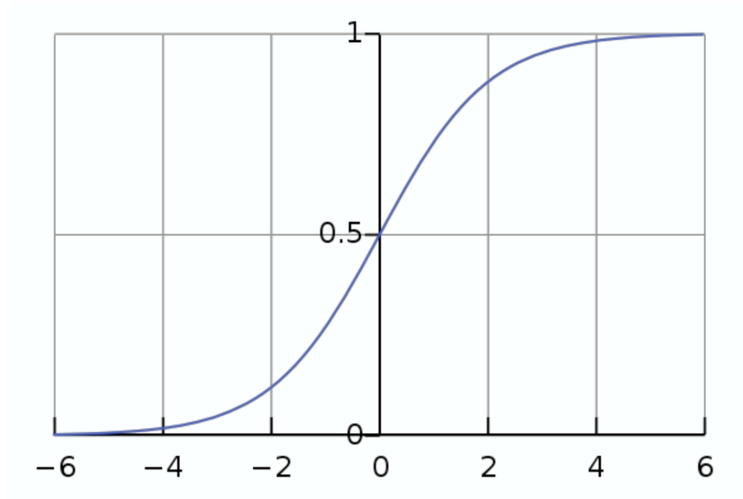
- Imagine a coin flip which is heads with probability *p*
- Observe (H, H, H, T) and maximize likelihood: $\prod_{j=1}^{m} P(y_j) = p^3(1-p)$
- Easier: maximize *log* likelihood

$$\sum_{j=1}^{m} \log P(y_j) = 3\log p + \log(1-p)$$

Logistic Regression

Logistic Regression

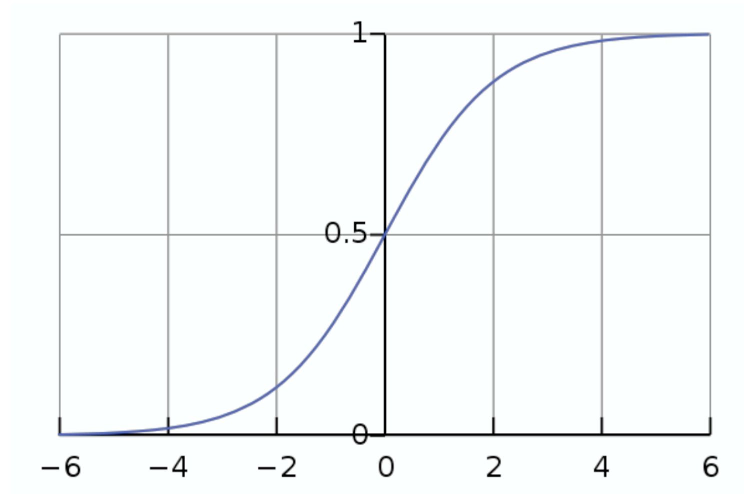
$$P(y = +|x) = \text{logistic}(w^\top x)$$



Logistic Regression

$$P(y=+|x) = \text{logistic}(w^\top x)$$

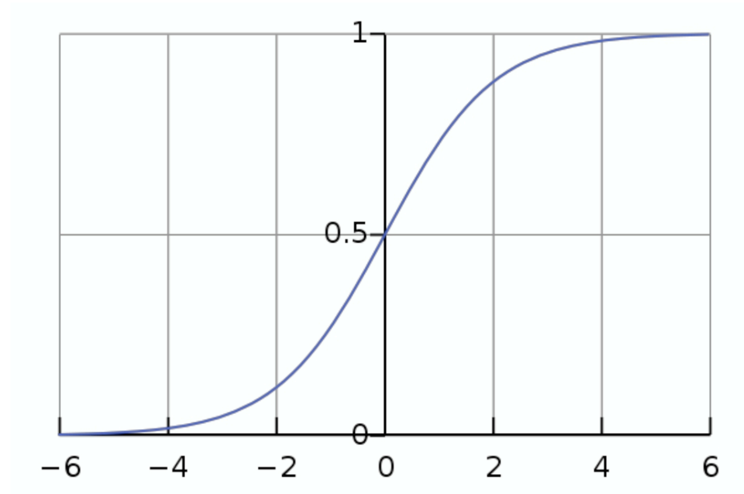
$$P(y=+|x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1+\exp(\sum_{i=1}^{n} w_i x_i)}$$



Logistic Regression

$$P(y = +|x) = \text{logistic}(w^\top x)$$

$$P(y = +|x) = \frac{\exp(\sum_{i=1}^n w_i x_i)}{1 + \exp(\sum_{i=1}^n w_i x_i)}$$

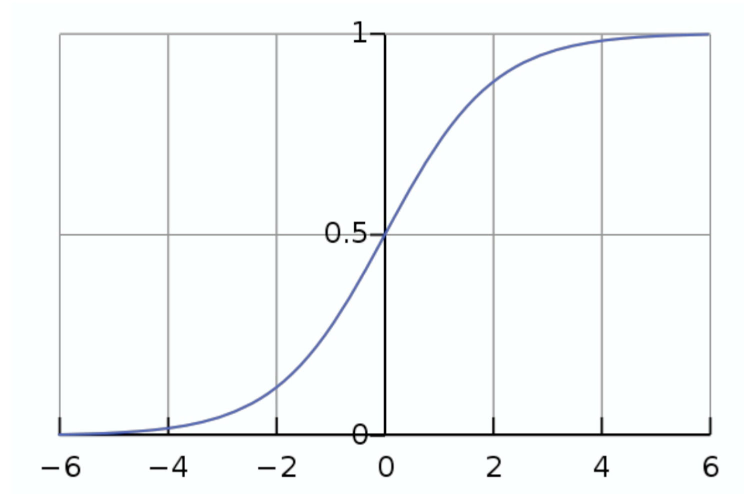


- To learn weights: maximize discriminative log likelihood of data *P(y|x)*

Logistic Regression

$$P(y=+|x) = \text{logistic}(w^\top x)$$

$$P(y=+|x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1+\exp(\sum_{i=1}^{n} w_i x_i)}$$



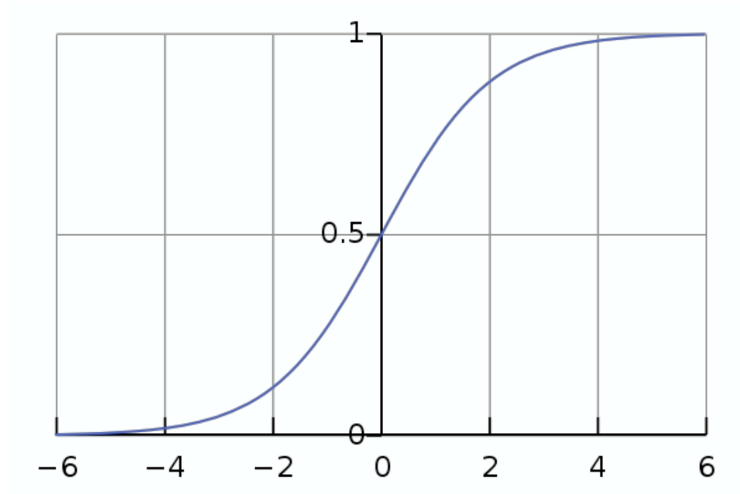
- To learn weights: maximize discriminative log likelihood of data *P(y|x)*

$$\mathcal{L}(x_j, y_j=+) = \log P(y_j=+|x_j)$$

Logistic Regression

$$P(y=+|x) = \text{logistic}(w^\top x)$$

$$P(y=+|x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1+\exp(\sum_{i=1}^{n} w_i x_i)}$$



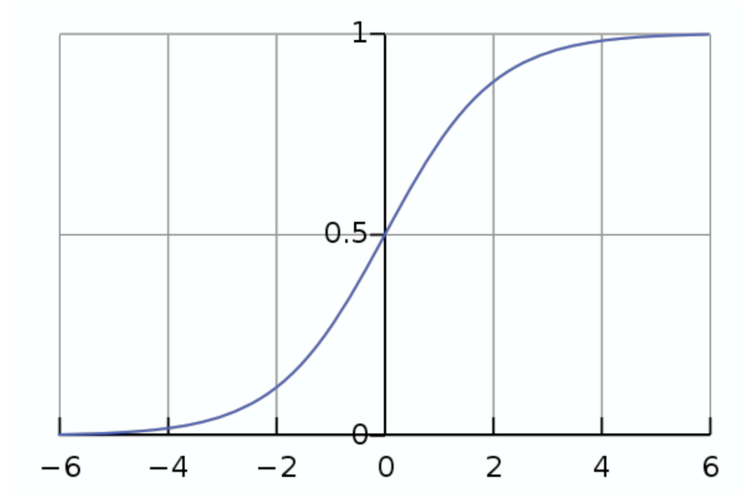
- To learn weights: maximize discriminative log likelihood of data *P(y|x)*

$$\begin{aligned}\mathcal{L}(x_j, y_j=+) &= \log P(y_j=+|x_j) \\ &= \sum_{i=1}^{n} w_i x_{ji} - \log\left(1+\exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)\end{aligned}$$

Logistic Regression

$$P(y = +|x) = \text{logistic}(w^\top x)$$

$$P(y = +|x) = \frac{\exp(\sum_{i=1}^n w_i x_i)}{1 + \exp(\sum_{i=1}^n w_i x_i)}$$



- To learn weights: maximize discriminative log likelihood of data *P(y|x)*

$$\mathcal{L}(x_j, y_j = +) = \log P(y_j = +|x_j)$$
$$= \sum_{i=1}^n w_i x_{ji} - \log\left(1 + \exp\left(\sum_{i=1}^n w_i x_{ji}\right)\right)$$

sum over features

Logistic Regression

$$\mathcal{L}(x_j, y_j = +) = \log P(y_j = +|x_j) = \sum_{i=1}^{n} w_i x_{ji} - \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

$$\frac{\partial \mathcal{L}(x_j, y_j)}{\partial w_i} =$$

Logistic Regression

$$\mathcal{L}(x_j, y_j = +) = \log P(y_j = +|x_j) = \sum_{i=1}^{n} w_i x_{ji} - \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

$$\frac{\partial \mathcal{L}(x_j, y_j)}{\partial w_i} = \; x_{ji} - \frac{\partial}{\partial w_i} \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

Logistic Regression

$$\mathcal{L}(x_j, y_j = +) = \log P(y_j = + | x_j) = \boxed{\sum_{i=1}^{n} w_i x_{ji}} - \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

$$\frac{\partial \mathcal{L}(x_j, y_j)}{\partial w_i} = \boxed{x_{ji}} - \frac{\partial}{\partial w_i} \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

Logistic Regression

$$\mathcal{L}(x_j, y_j = +) = \log P(y_j = +|x_j) = \boxed{\sum_{i=1}^{n} w_i x_{ji}} - \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

$$\frac{\partial \mathcal{L}(x_j, y_j)}{\partial w_i} = \boxed{x_{ji}} - \frac{\partial}{\partial w_i} \log\left(\boxed{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}\right)$$

$$= x_{ji} - \frac{1}{\boxed{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}} \frac{\partial}{\partial w_i}\left(\boxed{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}\right)$$

deriv
of log

Logistic Regression

$$\mathcal{L}(x_j, y_j = +) = \log P(y_j = +|x_j) = \boxed{\sum_{i=1}^{n} w_i x_{ji}} - \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

$$\frac{\partial \mathcal{L}(x_j, y_j)}{\partial w_i} = \boxed{x_{ji}} - \frac{\partial}{\partial w_i} \log\left(\boxed{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}\right)$$

$$= x_{ji} - \frac{1}{\boxed{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}} \frac{\partial}{\partial w_i}\left(\boxed{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}\right)$$

deriv of log

$$= x_{ji} - \frac{1}{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)} x_{ji} \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)$$

deriv of exp

Logistic Regression

$$\mathcal{L}(x_j, y_j = +) = \log P(y_j = +|x_j) = \sum_{i=1}^{n} w_i x_{ji} - \log\left(1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)\right)$$

$$\frac{\partial \mathcal{L}(x_j, y_j)}{\partial w_i} = \boxed{x_{ji}} - \frac{\partial}{\partial w_i} \log\left(\boxed{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}\right)$$

$$= x_{ji} - \frac{1}{\boxed{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}} \frac{\partial}{\partial w_i}\left(\boxed{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}\right)$$

deriv of log

$$= x_{ji} - \frac{1}{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)} x_{ji} \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)$$

deriv of exp

$$= x_{ji} - x_{ji} \frac{\exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}{1 + \exp\left(\sum_{i=1}^{n} w_i x_{ji}\right)}$$

Logistic Regression

- Recall that , for positive instances, $y_j = 1$ and for negative instances, $y_j = 0$
- Gradient of w_i on positive example = $x_{ji}(y_j - P(y_j = + \mid x_j))$
    - If P(+) is close to 1, make very little update
    - Otherwise make w_i look more like x_{ji} , which will increase P(+)

Logistic Regression

- Recall that , for positive instances, $y_j = 1$ and for negative instances, $y_j = 0$
- Gradient of w_i on positive example = $x_{ji}(y_j - P(y_j = + \mid x_j))$
    - If P(+) is close to 1, make very little update
    - Otherwise make w_i look more like x_{ji} , which will increase P(+)
- Gradient of w_i on positive example = $x_{ji}(-P(y_j = + \mid x_j))$
    - If P(+) is close to 0, make very little update
    - Otherwise make w_i look more like x_{ji} , which will decrease P(+)

Logistic Regression

- Recall that , for positive instances, $y_j = 1$ and for negative instances, $y_j = 0$
- Gradient of w_i on positive example = $x_{ji}(y_j - P(y_j = + \mid x_j))$
    - If P(+) is close to 1, make very little update
    - Otherwise make w_i look more like x_{ji} , which will increase P(+)
- Gradient of w_i on positive example = $x_{ji}(-P(y_j = + \mid x_j))$
    - If P(+) is close to 0, make very little update
    - Otherwise make w_i look more like x_{ji} , which will decrease P(+)
- Can combine these gradient as : $x_j(y_j - P(y_j = 1 \mid x_j))$

Logistic Regression: Summary

- Model

$$P(y=+|x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1+\exp(\sum_{i=1}^{n} w_i x_i)}$$

Logistic Regression: Summary

- Model

$$P(y=+|x)=\frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1+\exp(\sum_{i=1}^{n} w_i x_i)}$$

- Inference

$$\text{argmax}_y P(y|x)$$

$$P(y=1|x)\geq 0.5 \Leftrightarrow w^\top x \geq 0$$

Logistic Regression: Summary

- Model

$$P(y=+|x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1+\exp(\sum_{i=1}^{n} w_i x_i)}$$

- Inference

$$\text{argmax}_y P(y|x)$$

$$P(y=1|x) \geq 0.5 \Leftrightarrow w^\top x \geq 0$$

- Learning: gradient ascent on the (regularized) discriminative log-likelihood

Perceptron

Perceptron

- Simple error-driven learning approach similar to logistic regression

Perceptron

- Simple error-driven learning approach similar to logistic regression
- Decision rule: $w^T x > 0$

Perceptron

- Simple error-driven learning approach similar to logistic regression
- Decision rule: $w^T x > 0$
  - If incorrect: if positive, $w \leftarrow w + x$

    if negative, $w \leftarrow w - x$

Perceptron

- Simple error-driven learning approach similar to logistic regression
- Decision rule: $w^T x > 0$
  - If incorrect: if positive, $w \leftarrow w + x$

    if negative, $w \leftarrow w - x$

Logistic Regression

$$w \leftarrow w + x(1 = P(y = 1 \mid x))$$

$$w \leftarrow w - x(1 = P(y = 1 \mid x))$$

Perceptron

- Simple error-driven learning approach similar to logistic regression
- Decision rule: $w^T x > 0$
  - If incorrect: if positive, $w \leftarrow w + x$

    if negative, $w \leftarrow w - x$

Logistic Regression

$$w \leftarrow w + x(1 = P(y = 1 \mid x))$$

$$w \leftarrow w - x(1 = P(y = 1 \mid x))$$

- Guaranteed to eventually separate the data if the data are separable

Comparing Gradient Updates (Reference)

Logistic regression

$x(y - P(y = 1|x))$ = $x(y - \text{logistic}(w^\top x))$

Perceptron

$(2y - 1)x$ if classified incorrectly

0 else

y = 1 for pos,
0 for neg

Classification: Sentiment Analysis

this movie was great! would watch again Positive

that film was awful, I'll never watch again Negative

- Surface cues can basically tell you what's going on here:
  - presence or absence of certain words (*great*, *awful*)

- Steps to classification:
  - Turn examples like this into feature vectors
  - Pick a model / learning algorithm
  - Train weights on data to get our classifier

Sentiment Analysis

this movie was great! would watch again +

the movie was gross and overwrought, but I liked it +

this movie was not really very enjoyable —

- Bag-of-words doesn’t seem sufficient (discourse structure, negation)

- There are some ways around this:
    - extract bigram feature for “not X” for all X following the not
    - character n-grams, parts of speech, lemmas, …

Sentiment Analysis

	Movie Reviews	Product Reviews
Unigram only	64.1	42.91
Bigram only	76.15	69.62
Trigram only	76.1	71.37
(Uni + Bi) gram	77.15	72.94
(Uni + Bi + Tri) gram	**80.15**	**78.67**

Table 1: Results of Simple NGram

	Movie Reviews	Product Reviews
POS-(U + B + T)-JJ	75.00	50.425
POS-(U + B + T)-RB	65.50	36.76
POS-(U + B + T)-(JJ + RB)	**76.50**	**62.06**

Table 2: Results of POS-Tagged NGram. U = Unigram, B = Bigram, T = Trigram

Bakliwal et al. (2011)

Recap

- Logistic regression, and perceptron are closely related

- Perceptron inference require taking maxes, logistic regression has a similar update but is “softer” due to its probabilistic nature

- All gradient updates: “make it look more like the right thing and less like the wrong thing”