CPSC 4420/6420

Artificial Intelligence

25 – Logistic and Softmax Regression

November 17, 2020

Announcements

- Projects 4&5 are due on 11/25
- Quiz 8 is due on 11/19

Quick Recap

Binary classification

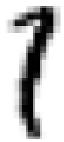
x (input) y (label)

```
Hello,

Do you want free printr
cartriges?  Why pay more
when you can get them
ABSOLUTELY FREE!  Just
```

```
# free       : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```

SPAM
or
+



```
PIXEL-7,12   : 1
PIXEL-7,13   : 0
...
NUM_LOOPS    : 1
...
```

“1”
or
+

Linear classifier & stochastic gradient descent

- Linear regression
  - $$h_w(x) = \sum_{i}^{n} w_i \, x_i = w \cdot x$$
  - Update rule: $w_j \leftarrow w_j + \alpha \, (y^{(k)} - h_w(x^{(k)}))x_j^{(k)}$
- Linear classifier
  - $$h_w(x) = \begin{cases} 1 & \text{if } w \cdot x \geq 0 \\ 0 & \text{if } w \cdot x < 0 \end{cases}$$
  - Update rule: $w_j \leftarrow w_j + \alpha \, (y^{(k)} - h_w(x^{(k)}))x_j^{(k)}$

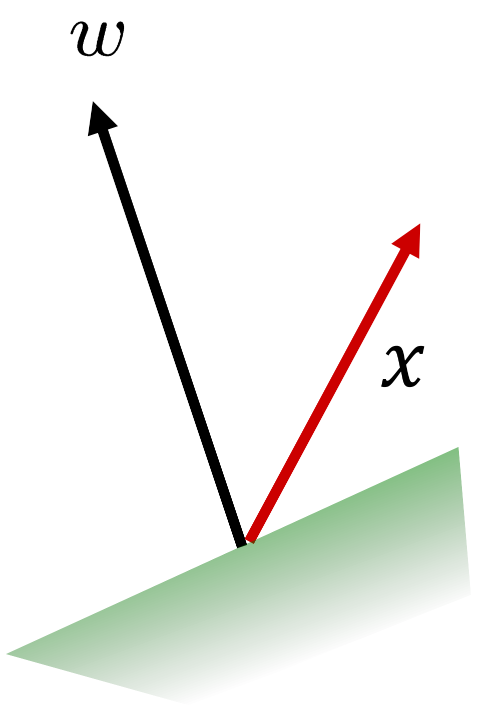
Binary perceptron algorithm

- Start with w = 0
- For each training instance:
  - Classify with current weights

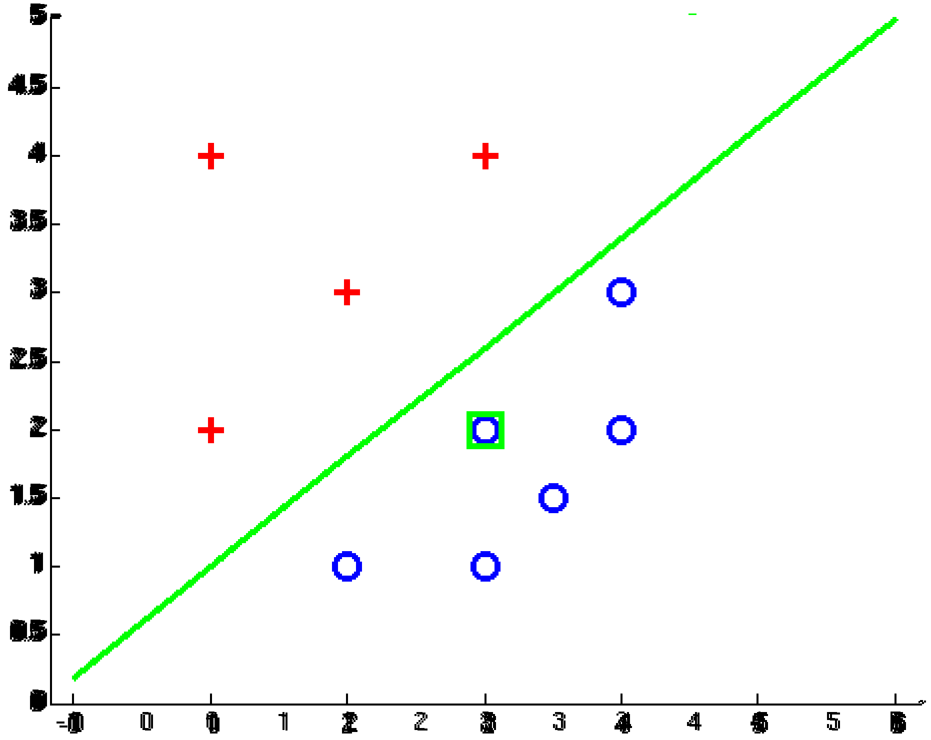
$$\hat{y} = \begin{cases} 1 & \text{if } \; w \cdot x \geq 0 \\ 0 & \text{if } \; w \cdot x < 0 \end{cases}$$

  - If correct (i.e., $\hat{y}$=y), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y is 0

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

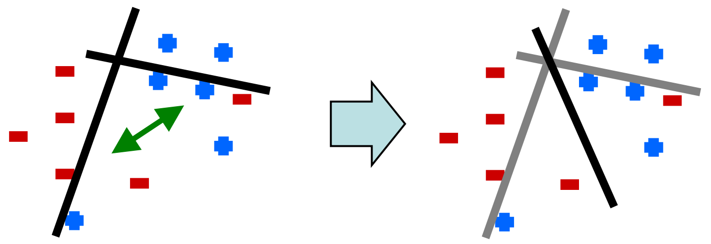


Separable case example

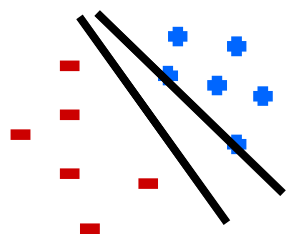


Issues

- Noise: if the data isn't separable, weights might thrash

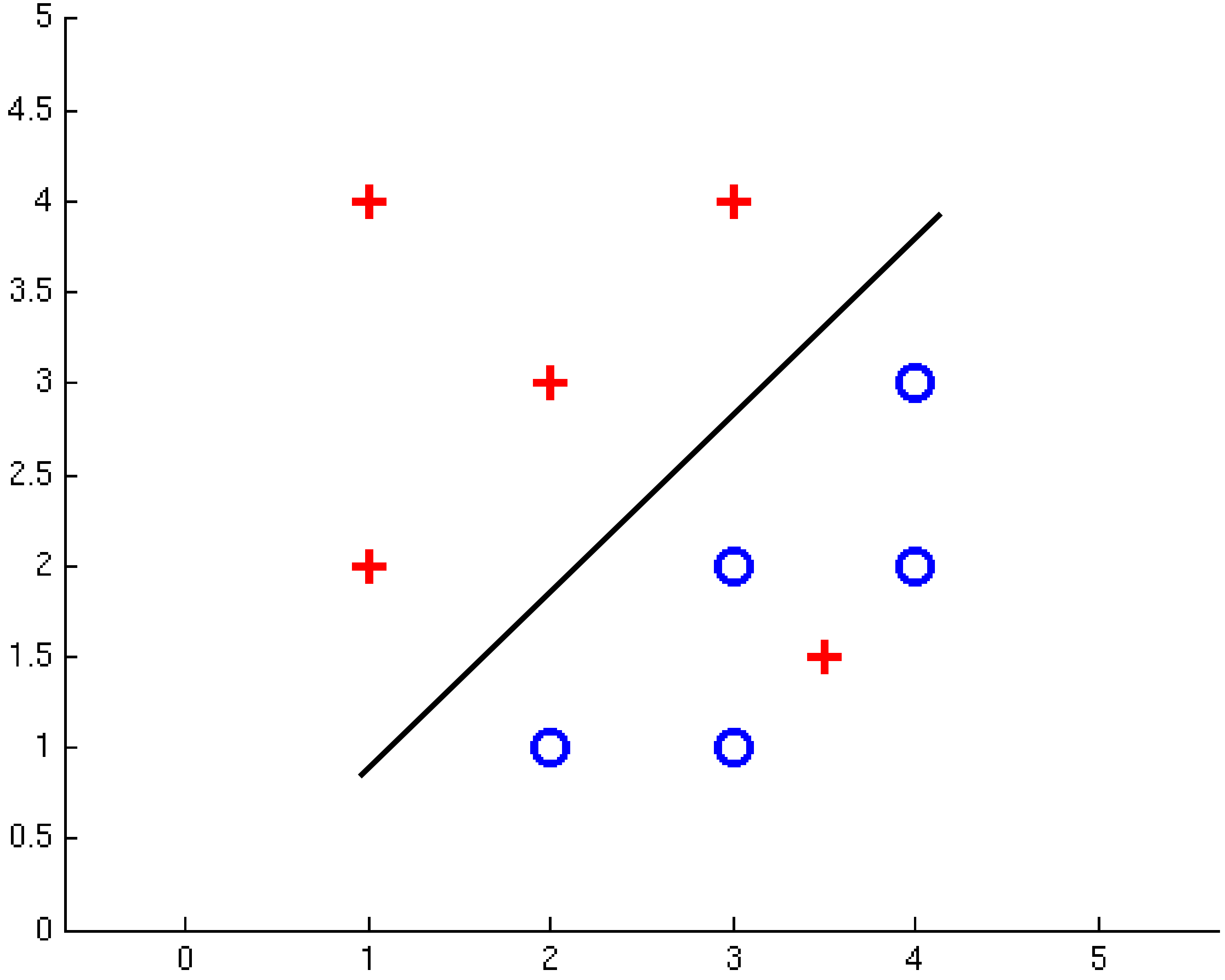


- Mediocre generalization: finds a "barely" separating solution

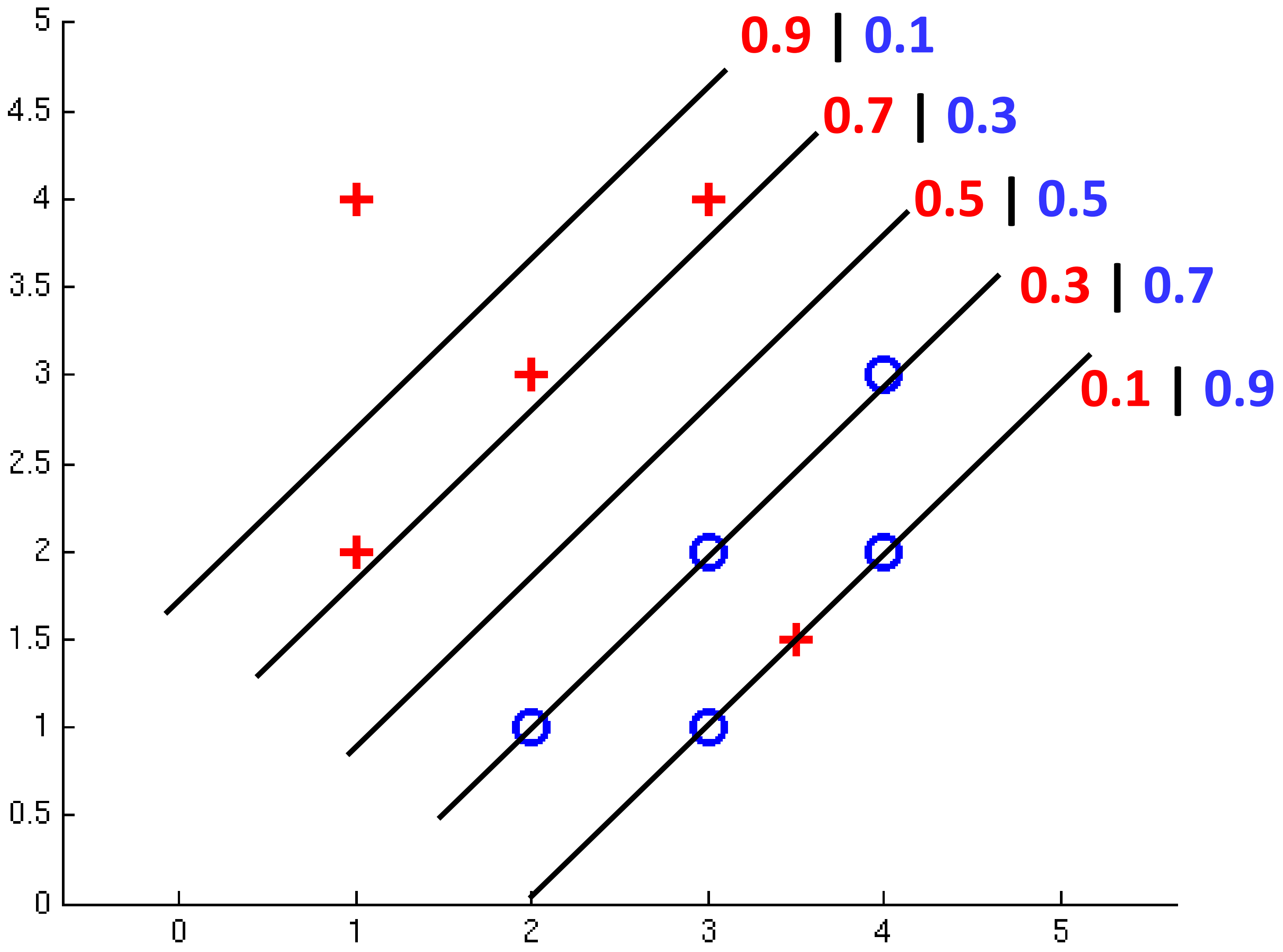


Logistic Regression

Non-separable case: probabilistic decision



Non-separable case: probabilistic decision

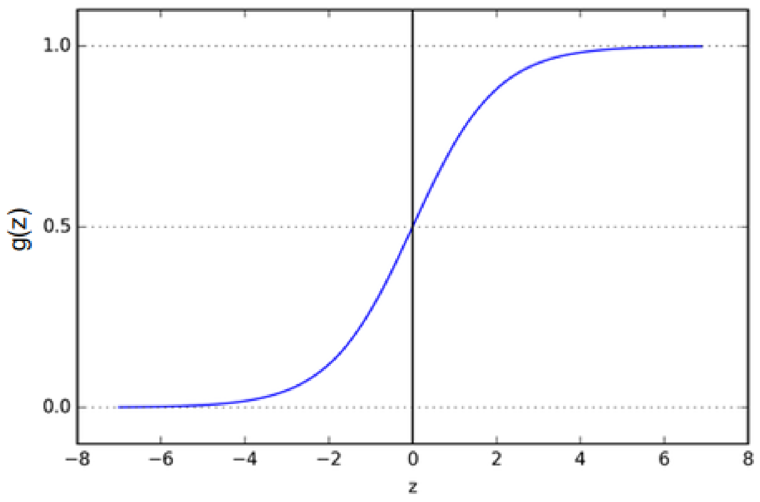


Probabilistic decisions

- Activation: $z = w \cdot x$
- If $z = w \cdot x$ very positive → want probability going to 1
- If $z = w \cdot x$ very negative → want probability going to 0

- Sigmoid function

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



Logistic regression

- Given a training set consisting of m examples, learn a function h_w
  - $h_w(x) = g(w \cdot x) = \dfrac{1}{1+e^{-w \cdot x}} = P(y=1 \mid x; w)$
  - $1 - h_w(x) = 1 - g(w \cdot x) = 1 - \dfrac{1}{1+e^{-w \cdot x}} = P(y=0 \mid x; w)$

Sidebar: Maximum Likelihood Estimation

- Data: Observed set D of α_H Heads and α_T Tails
- Assuming P(Heads)=θ and P(Tails)=1-θ, the likelihood of *D* is

$$P(\mathcal{D};\theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$$

- **MLE:** Choose θ to maximize probability of *D*

$$\begin{aligned}\widehat{\theta} &= \arg\max_{\theta} \; P(\mathcal{D};\theta) \\ &= \arg\max_{\theta} \; \ln P(\mathcal{D};\theta) \\ &= \arg\max_{\theta} \; \ln \theta^{\alpha_H}(1-\theta)^{\alpha_T}\end{aligned}$$

- Set derivative to zero, and solve!

$$\frac{d}{d\theta}\ln P(\mathcal{D};\theta) = \frac{\alpha_H}{\theta} - \frac{\alpha_T}{1-\theta} = 0 \quad \Rightarrow \quad \widehat{\theta} = \frac{\alpha_H}{\alpha_H+\alpha_T}$$

Maximum likelihood

- Let $P(y=1 \mid x; w) = h_w(x)$

  $P(y=0 \mid x; w) = 1 - h_w(x)$

- More compactly: $p(y \mid x; w) = (h_w(x))^y \, (1 - h_w(x))^{1-y}$
- Assuming that the m training example are independent

$$\begin{aligned} L(w) &= \prod^{m} p(y^{(i)} \mid x^{(i)}; w) \\ &= \prod_{i=1}^{m} \left(h_w(x^{(i)})\right)^{y^{(i)}} \left(1 - h_w(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

- Maximum likelihood estimation

$$\max_w \; \log L(w) = \max_w \; \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$$

Logistic regression

- Search for w that minimizes the following cost function

$$J(w) = -\sum_{i=1}^{m} \left(y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_w(x^{(i)}))\right)$$

- We need to compute ∇_w. For one coordinate of the weight vector

$$\frac{\partial J(w)}{\partial w_j} = x_j^{(i)} \left(h_w(x^{(i)}) - y^{(i)}\right)$$

Follows from the fact that $g'(z) = g(z)(1 - g(z))$

Logistic regression update rules

- Batch gradient descent (for every *j*)

$$w_j \leftarrow w_j - \alpha \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

- Stochastic gradient descent (for every *j*)

$$w_j \leftarrow w_j - \alpha \left(h_w(x^{(k)}) - y^{(k)}\right) x_j^{(k)}$$

Where $h_w(x)$ is the sigmoid function, i.e. $h_w = \frac{1}{1+e^{-w \cdot x}}$

Logistic regression

- Given a training set consisting of m examples, learn a function h_w
  - $h_w(x) = g(w \cdot x) = \dfrac{1}{1 + e^{-w \cdot x}} = P(y = 1 \mid x; w)$
  - $1 - h_w(x) = 1 - g(w \cdot x) = 1 - \dfrac{1}{1 + e^{-w \cdot x}} = P(y = 0 \mid x; w)$

Multiclass Classification

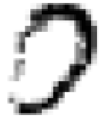
Classification

- Binary classification
  - Given inputs $x^{(i)} \in \mathbb{R}^n$ predict binary labels/classes $y^{(i)} \in \{0, 1\}$
- Multiclass classification
  - Given inputs $x^{(i)} \in \mathbb{R}^n$ predict labels/classes $y^{(i)} \in \{1, \dots, \mathrm{K}\}$
  - K is the number of classes

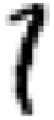
Example: Digit classification

- Input: images / pixel grids
  - Each input maps to a feature vector
  - E.g. one feature (variable) for each grid position based on pixel intensity

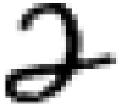
     $\rightarrow \langle 0, 0, 1, 1, 0 \ldots 0 \rangle$
- Output: a digit 0-9
- Setup
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images



0



1



2

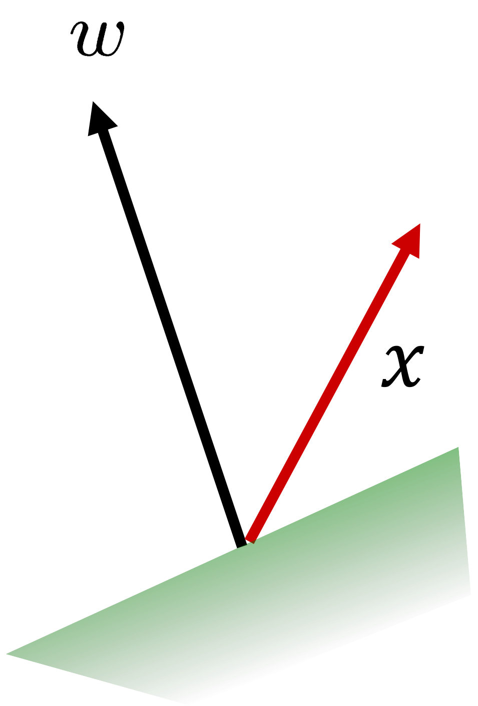
Recall: Binary perceptron

- Start with w = 0
- For each training instance:
  - Classify with current weights

$$\hat{y} = \begin{cases} 1 & \text{if} \quad w \cdot x \geq 0 \\ 0 & \text{if} \quad w \cdot x < 0 \end{cases}$$

  - If correct (i.e., $\hat{y}$=y), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y is 0

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



Multiclass decision rules

- Extending to K-classes:
  - A weight vector for each class $k = 1, \ldots, K$:

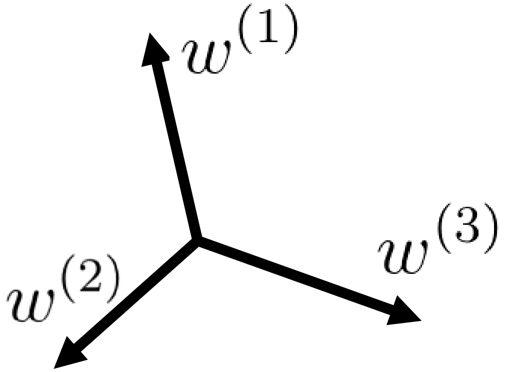
$$w^{(k)} \in \mathbb{R}^n$$

  - Score (activation) of a class y:

$$w^{(k)} \cdot x$$

  - Prediction highest score wins

$$\hat{y} = \arg\max_k \; w^{(k)} \cdot x$$



Multiclass decision rules

- Extending to K-classes:
  - A weight vector for each class $k = 1, \ldots, K$:

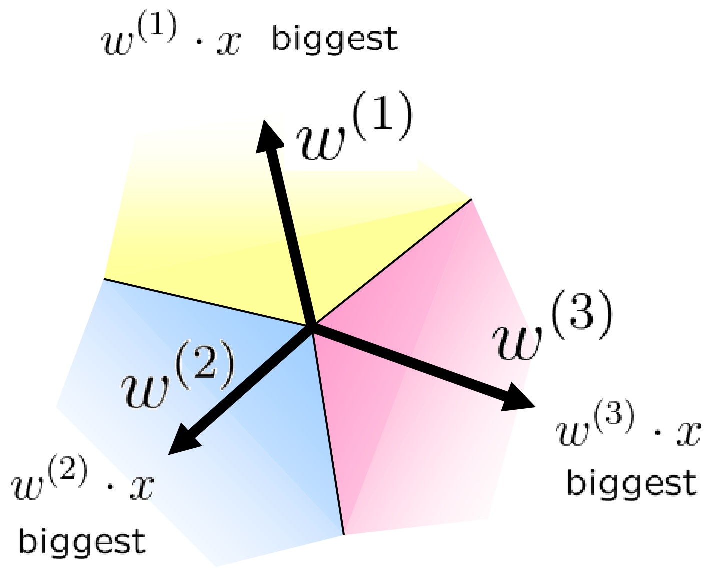
    $$w^{(k)} \in \mathbb{R}^n$$

  - Score (activation) of a class y:

    $$w^{(k)} \cdot x$$

  - Prediction highest score wins

    $$\hat{y} = \arg\max_k \; w^{(k)} \cdot x$$



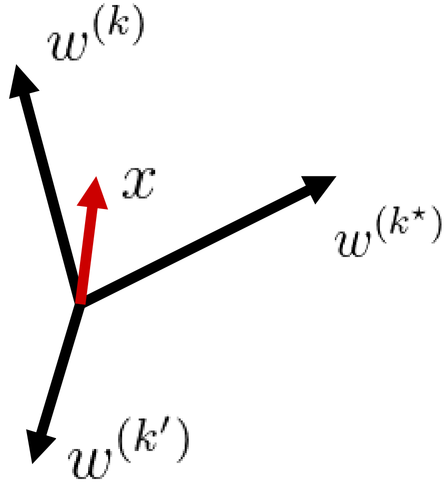
Binary = multiclass where the negative class has weight zero

Multiclass perceptron algorithm

- Start with all weights = 0
- For each training instance
  - Predict with current weights

$$\hat{y} = \arg\max_k \; w^{(k)} \cdot x$$

  - If correct, no change!
  - If wrong: lower score of wrong answer, raise score of right answer



Multiclass perceptron algorithm

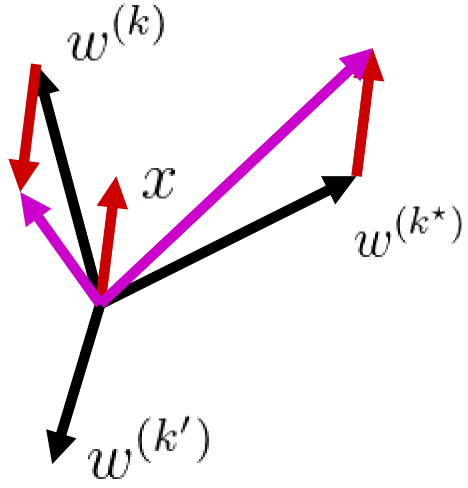
- Start with all weights = 0
- For each training instance
  - Predict with current weights

    $$\widehat{y} = \arg\max_k \; w^{(k)} \cdot x$$

  - If correct, no change!
  - If wrong: lower score of wrong answer, raise score of right answer

$$w^{(k)} = w^{(k)} - x$$

$$w^{(k^\star)} = w^{(k^\star)} + x$$



Example

“win the vote”

“win the election”

“win the game”

$$w_{SPORTS}$$

```
BIAS  : 1
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

$$w_{POLITICS}$$

```
BIAS  : 0
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

$$w_{TECH}$$

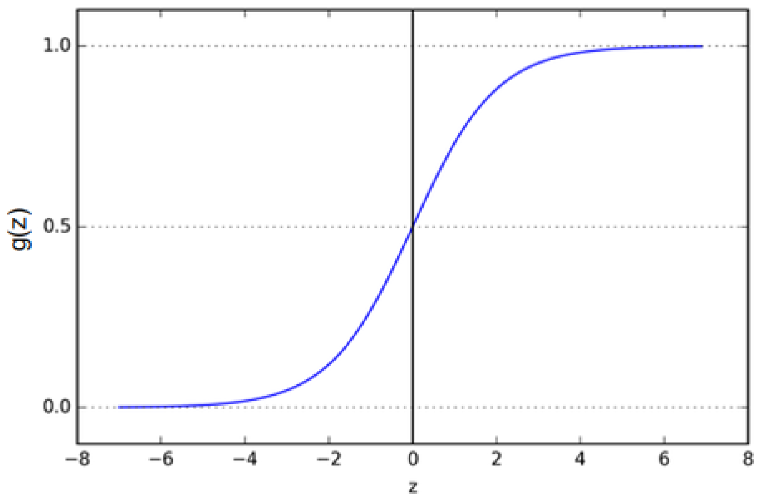
```
BIAS  : 0
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

Recall: logistic regression

- Activation: $z = w \cdot x$
- If $z = w \cdot x$ very positive → want probability going to 1
- If $z = w \cdot x$ very negative → want probability going to 0

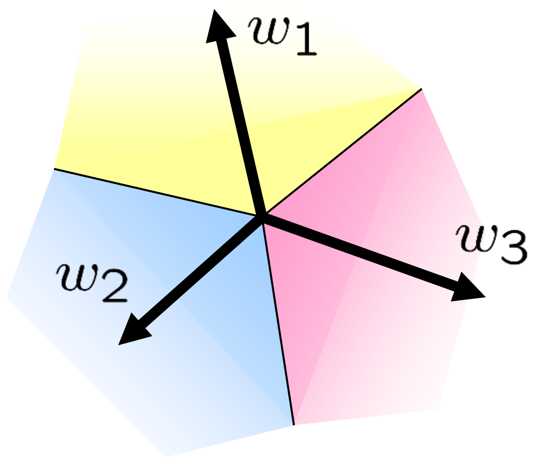
- Sigmoid function

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



Multiclass logistic regression

- Recall Perceptron:
  - A weight vector for each class: $w^{(k)}$
  - Score (activation) of a class k: $w^{(k)} \cdot x$
  - Prediction highest score wins $\hat{y} = \arg\max_k \; w^{(k)} \cdot x$



- How to make the scores into probabilities?

$$\underbrace{z_1, z_2, z_3}_{\text{original activations}} \rightarrow \underbrace{\frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}}_{\text{softmax activations}}$$

Softmax regression

- Given an input $x^{(i)}$, we want h_w to estimate $P(y^{(i)}{=}k|x)$ for each $k = 1, \dots, K$

$$h_w(x^{(i)}) = \begin{bmatrix} P(y^{(i)} = 1|x^{(i)}; w) \\ P(y^{(i)} = 2|x^{(i)}; w) \\ \vdots \\ P(y^{(i)} = K|x^{(i)}; w) \end{bmatrix}$$

where $\forall k : P(y^{(i)} = k|x^{(i)}; w) = \dfrac{\mathrm{e}^{w^{(k)} \cdot x^{(i)}}}{\sum_{j=1}^{K} \mathrm{e}^{w^{(j)} \cdot x^{(i)}}}$

- Here, $w^{(1)}, w^{(2)}, \ldots, w^{(K)} \in \mathbb{R}^n$ are the parameters of our model (stored in a n-by-K matrix)

Softmax regression

- For each training example, we want to learn

$$P(y^{(i)}|x^{(i)};w) = \sum_{k=1}^{K} 1\left\{y^{(i)} = k\right\} P(y^{(i)} = k|x^{(i)};w)$$, where $1\{\cdot\}$ is the indicator function

- Assuming that the m training examples are independent

$$\begin{aligned} L(w) &= \prod^{m} P(y^{(i)}|x^{(i)};w) \\ &= \prod_{i=1}^{m} \sum_{k=1}^{K} 1\left\{y^{(i)} = k\right\} P(y^{(i)} = k|x^{(i)};w) \end{aligned}$$

- Maximum likelihood estimation

$$\max_{w} \log L(w) = \max_{w} \sum_{i=1}^{m} \sum_{k=1}^{K} 1\left\{y^{(i)} = k\right\} \log P(y^{(i)} = k|x^{(i)};w)$$

Best w?

- Maximum likelihood estimation: search for w that minimizes the cost function

$$J(w) = -\left[\sum_{i=1}^{m}\sum_{k=1}^{K} 1\left\{y^{(i)} = k\right\} \log P(y^{(i)} = k | x^{(i)}; w)\right]$$

where $$P(y^{(i)} = k | x^{(i)}; w) = \frac{\mathrm{e}^{w^{(k)} \cdot x^{(i)}}}{\sum_{j=1}^{K} \mathrm{e}^{w^{(j)} \cdot x^{(i)}}}$$

- We need to compute the gradient ∇_w. It holds

$$\nabla_{w^{(k)}} J = -x^{(i)} \left(1\{y^{(i)} = k\} - P(y^{(i)} = k | x^{(i)}; w)\right)$$

Softmax regression update rules

- Batch gradient descent (for every class k)

$$w^{(k)} \leftarrow w^{(k)} + \alpha \sum_{i=1}^{m} \left[x^{(i)} \left(1\{y^{(i)} = k\} - P(y^{(i)} = k | x^{(i)}; w)\right)\right]$$

- Stochastic gradient descent (for every class k)

$$w^{(k)} \leftarrow w^{(k)} + \alpha \left[x^{(i)} \left(1\{y^{(i)} = k\} - P(y^{(i)} = k | x^{(i)}; w)\right)\right]$$

Where $P(\cdot)$ is the softmax function, i.e. $P(y^{(i)} = k | x^{(i)}; w) = \frac{\mathrm{e}^{w^{(k)} \cdot x^{(i)}}}{\sum_{j=1}^{K} \mathrm{e}^{w^{(j)} \cdot x^{(i)}}}$

Relationship to logistic regression

- When $K = 2$, softmax regression reduces to logistic regression
  - The function h is given by

$$h_w(x^{(i)}) = \begin{bmatrix} P(y^{(i)} = 1 | x^{(i)}; w) \\ P(y^{(i)} = 2 | x^{(i)}; w) \end{bmatrix} = \frac{1}{e^{w^{(1)} \cdot x^{(i)}} + e^{w^{(2)} \cdot x^{(i)}}} \begin{bmatrix} e^{w^{(1)} \cdot x^{(i)}} \\ e^{w^{(2)} \cdot x^{(i)}} \end{bmatrix}$$

  - By setting $\psi = w^{(2)}$, and subtracting from ψ each weight vector

$$h_w(x^{(i)}) = \frac{1}{e^{(w^{(1)} - w^{(2)}) \cdot x^{(i)}} + e^{\mathbf{0} \cdot x^{(i)}}} \begin{bmatrix} e^{(w^{(1)} - w^{(2)}) \cdot x^{(i)}} \\ e^{\mathbf{0} \cdot x^{(i)}} \end{bmatrix}$$

$$\overset{w' = w^{(1)} - w^{(2)}}{=\!=\!=} \begin{bmatrix} 1 - \frac{1}{1 + e^{w' \cdot x^{(i)}}} \\ \frac{1}{1 + e^{w' \cdot x^{(i)}}} \end{bmatrix}$$