Linear Threshold Units

- The basic model: \( \text{Output} = f(\sum w_k x_k) \)
- Where \( f \) can be one of:
  - \( f = \text{sign}() \) Value in \( \{-1, 1\} \)
  - \( f = \text{step}() \) Value in \( \{0, 1\} \)

\[
\sigma(a) = \frac{1}{1 + e^{-a}} \\
p(f = 1) = \sigma(\sum w_k x_k)
\]

- Most of the lecture uses the sign function with values in \( \{-1, 1\} \)
- Standard trick: instead of saying \( 5x_1 + 6x_2 > 9 \)
\[
\text{add zero'th index and set } x_0 = 1 \\
-9x_0 + 5x_1 + 6x_2 > 0
\]
- In this way our hyperplanes always go through the origin.

- \( f = \text{sign}(\sum w_k x_k) \)
- Is a powerful representation
  - Express \( x_1 \) And \( x_2 \) And \( x_3 \):
  - Express \( x_1 \) or \( x_2 \) or \( x_3 \):
  - Express \( m \) of \( k \) functions:

- But it cannot represent XOR

- LTU have gone up/down in popularity many times
Some Geometric Intuition

- \( w \) imposes constraints on the label of \( x \)
  \( x, \pm / - \) imposes constraints on possible \( w \)'s

- \( x=(1,1) + \) implies \( w_0 + w_1 > 0 \)
- \( x=(2,3) + \) implies \( 2w_0 + 3w_1 > 0 \)
- \( x=(5,1) - \) implies \( 5w_0 + w_1 < 0 \)

- What does this look like?

On line updates

- Current weight vector is: \( (1,1,1) \)
- \( X=(1,2,4) \) is -

  - Our prediction: \( 1+2+4=7 > 0 \) \( \rightarrow + \)
  - How can we fix this?

Perceptron Learning Algorithm

\[
\text{Repeat}
\]
\[
\hat{y} = \text{sign}(\sum_k w_k x_k)
\]
\[
\text{if} \ (y \neq \hat{y}) \text{ then } w \leftarrow w + \eta y x
\]

- If given training set then can go over it multiple times

  - Perceptron Convergence Theorem:
    if data is separable with margin \( \delta \) then at most \( R^2/\delta^2 \) mistakes

Margin Error

- We want to get prediction correct
- Want points to be far from \( h \)-plane
- Otherwise penalize \( w \)

\[
\text{margin-err} = \max\{0, \gamma - y (\sum_k w_k x_k)\}
\]

- Intuition: \( y (\sum_k w_k x_k) \) measures distance from hyperplane in the right direction
- Taking derivative we get \( PwM \) algorithm as gradient descent

Perceptron with Margin

- Aim to maintain a margin around the separating hyperplane

\[
\text{Repeat}
\]
\[
\hat{y} = \text{sign}(\sum_k w_k x_k)
\]
\[
\text{if} \ (y (\sum_k w_k x_k) < \gamma) \text{ then } w \leftarrow w + \eta y x
\]

- Updating on mistake (as in perceptron) and in addition on "margin deficiency"
**Voted Perceptron**

- In sequential updates the last vector may not be the best
- Idea: use all weight vector in sequence
- $x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ \ldots \ \ x_N$
- $w_0 \ w_1 \ w_2 \ w_3 \ w_4 \ \ldots \ \ w_N$

Using $i$ for example/iteration index, $k$ for feature index

- Prediction on $x$: $\text{sign}\left[\sum_i \text{sign}(\sum_k w_{i,k} x_k)\right]$

- Has theoretical justification. Works well

---

**Average Perceptron**

- In sequential updates the last vector may not be the best
- Idea: use all weight vector in sequence
- $x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ \ldots \ \ x_N$
- $w_0 \ w_1 \ w_2 \ w_3 \ w_4 \ \ldots \ \ w_N$

Using $i$ for example/iteration index, $k$ for feature index

- Prediction on $x$: $\text{sign}\left[\sum_i (\sum_k w_{i,k} x_k)\right]$

Less expensive and works well in practice.

---

**Online Logistic Regression**

- The sigmoid based probabilistic model

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

$$p(f = 1) = \sigma(\sum_k w_k x_k)$$

- With independent examples likelihood is

$$L = \prod_i \sigma(\sum_k w_k x_{i,k})^{y_i} (1 - \sigma(\sum_k w_k x_{i,k}))^{(1-y_i)}$$

$$\log L = \sum_i y_i \log \sigma(\sum_k w_k x_{i,k}) + (1 - y_i) \log(1 - \sigma(\sum_k w_k x_{i,k}))$$

---

**Linear Threshold Units**

- Useful fact about sigmoids

$$h(z) = \frac{1}{1+e^{-z}} \Rightarrow h'(z) = \frac{-e^{-z}}{1+e^{-z}}$$

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

$$\sigma'(a) = \frac{-e^{-a}}{(1+e^{-a})^2} = \sigma(a)(1 - \sigma(a))$$

---

**Online Logistic Regression**

- For maximum likelihood, apply gradient ascent algorithm

- To get online algorithm we focus on contribution of one example at a time to the likelihood (cf. error in perceptron)

$$\frac{\partial \log L}{\partial w_k} = \ldots = [y_i - \sigma(\sum_k w_k x_{i,k})] x_{i,k}$$

$$w \leftarrow w + \eta[y_i - \sigma(w^T x_i)] x_i$$

Allows for Probabilistic predictions.

Compare to Perceptron.