Linear Threshold Units

- **Motivation:** neurons

- **The basic model:**
  \[ \text{Output} = f(\sum_k w_k x_k) \]

- Where \( f \) can be one of:
  \[ f = \text{sign}() \text{ Value in } \{-1,1\} \]
  \[ f = \text{step}() \text{ Value in } \{0,1\} \]

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

- **Most of the lecture uses the sign function with values in \{-1,1\}**
- **Standard trick:** instead of saying
  \[ 5x_1 + 6x_2 > 9 \]
  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.

- **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 $x$
  \[ x, +/ - \text{ imposes constraints on possible } w's \]

• $x=(1,1)$ + 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

• Cannot see the entire dataset

• See one example at a time
  - First: Predict its label
  - Then: see true label
  - Then: update hypothesis

• Want overall “good performance”
  How to define this?

Perceptron Learning Algorithm

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

• 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

Gradient Descent Algorithms

• Reminder: GD can be used to find minima of functions.

• With multiple variables: we calculate gradient on all variables, and take a gradient step on all variables together.

• Quick review of gradient descent for optimization.
### Margin Error
- We want to get prediction correct
- Want points to be far from h-plane
- Otherwise penalize w
  \[ \text{margin-err} = \max\{0, \gamma - y \langle \sum_k w_k x_k \rangle \} \]
- Intuition: \( y \langle \sum_k w_k x_k \rangle \) measures distance from hyperplane in the right direction
- Taking derivative we get \( \text{PwM} \) algorithm as gradient descent

### Perceptron with Margin
- Aim to maintain a margin around the separating hyperplane
  \[
  \begin{align*}
  \hat{y} &= \text{sign}(\sum_k w_k x_k) \\
  \text{if } (y \langle \sum_k w_k x_k \rangle < \gamma) \text{ then } w &\leftarrow w + \eta y x
  \end{align*}
  \]
- Updating on mistake (as in perceptron) and in addition on "margin deficiency"
  \( \Rightarrow \text{PwM and Perceptron seen as GD alg} \)

### From on line algorithm to batch hyp
- In sequential updates the last vector may not be the best
  - \( x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ \ldots \ \ldots \ x_N \)
  - \( w_0 \ w_1 \ w_2 \ w_3 \ w_4 \ w_4 \ w_N \)
- What predictor should we use if learning stops at step N and we just want a classifier for future examples?

### Voted Perceptron
- In sequential updates the last vector may not be the best
- Idea: use all weight vector in sequence
  - \( x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ \ldots \ \ldots \ x_N \)
  - \( w_0 \ w_1 \ w_2 \ w_3 \ w_4 \ w_4 \ w_N \)
  - Using \( i \) for example/iteration index, \( k \) for feature index
  - Prediction on \( x \):
    \[ \text{sign} \left[ \sum_i \text{sign}(\sum_k w_{i,k} x_k) \right] \]
  - Works well but expensive (b/c need to store all \( w_i \) and calc many inner products)

### Average Perceptron
- In sequential updates the last vector may not be the best
- Idea: use all weight vector in sequence
  - \( x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ \ldots \ \ldots \ x_N \)
  - \( w_0 \ w_1 \ w_2 \ w_3 \ w_4 \ w_4 \ w_N \)
  - 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
  
  \[ h(z) = \frac{1}{y(a)} \Rightarrow h'(z) = \frac{y'(a)}{y(a)} \]

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

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

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)

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

  \[ w \leftarrow w + \eta[y_i - \sigma(\sum_k w_k x_{i,k})]x_i \]

  Allows for probabilistic predictions.

  Compare to Perceptron.

Linear Threshold Units - Recap

- Single unit expressive but has limitations

- Geometry motivated algorithms possible

- We focused on on-line algorithms which are "memory lean" and efficient to run
  - Perceptron algorithm and variants
  - On-line Logistic regression

- Derived as gradient descent algorithms on corresponding error functions

- Online to batch produces better final predictor