Classification

- **Learn**: $h : X \mapsto Y$
  - $X$ – features
  - $Y$ – target classes

- Conditional probability: $P(Y|X)$
  
- Suppose you know $P(Y|X)$ exactly, how should you classify?
  - Bayes optimal classifier:
    
- How do we estimate $P(Y|X)$?
Logistic Regression

- Learn $P(Y|X)$ directly
  - Assume a particular functional form for link function
  - Sigmoid applied to a linear function of the input features:
    $$P(Y = 0|X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i x_i)}$$

Features can be discrete or continuous!

Logistic Regression – a Linear classifier

$$P(Y=0|x, w) = g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + \exp(-z)}$$
Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: 
  \[(x^i, y^i)_{i=1}^n = D = (D_x, D_y)\]

- Discriminative (logistic regression) loss function: 
  Conditional Data Likelihood

\[
\arg\max_w \prod_{j=1}^n P(y^j | x^j, w) = \arg\max_x \sum_{j=1}^n \ln P(y^j | x^j, w)
\]

\[
\ln P(D_y | D_x, w) = \sum_{j=1}^N \ln P(y^j | x^j, w)
\]

Maximizing Conditional Log Likelihood

\[
l(w) \equiv \ln \prod_j P(y^j | x^j, w)
\]

\[
= \sum_j y^j w_0 + \sum_i w_i x_i^j - \ln (1 + \exp(w_0 + \sum_i w_i x_i^j))
\]

**Good news:** \(l(w)\) is concave function of \(w\), no local optima problems

**Bad news:** no closed-form solution to maximize \(l(w)\)

**Good news:** concave functions easy to optimize
Optimizing concave function – Gradient ascent

- Conditional likelihood for Logistic Regression is concave. Find optimum with gradient ascent
  
  **Gradient:** \( \nabla_w l(w) = \left[ \frac{\partial l(w)}{\partial w_0}, \ldots, \frac{\partial l(w)}{\partial w_n} \right]' \)

  **Update rule:** 
  \[
  \Delta w = \eta \nabla_w l(w) \\
  w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(w)}{\partial w_i}
  \]

- Gradient ascent is simplest of optimization approaches
  - e.g., Conjugate gradient ascent can be much better

  Often, especially in proof, \( \eta \) gets smaller with iterations.
  - E.g., \( \eta_t = \frac{\alpha}{t} \) is a constant.

Maximize Conditional Log Likelihood: Gradient ascent

\[
\frac{\partial l(w)}{\partial w} = f'(s) e^{f(s)}
\]

\[
l(w) = \sum_{j=1}^{k} y_{ij} (w_0 + \sum_{i=1}^{n} w_i x_{ij}) - \ln(1 + \exp(w_0 + \sum_{i=1}^{n} w_i x_{ij}))
\]

\[
\frac{\partial l}{\partial w_i} = \sum_{j=1}^{k} \left[ y_{ij} x_{ij} - \frac{x_{ij} \exp(w_0 + \sum_{i=1}^{n} w_i x_{ij})}{\left(1 + \exp(w_0 + \sum_{i=1}^{n} w_i x_{ij})\right)} \right] \frac{\partial}{\partial w_i}
\]

\[
\frac{\partial l}{\partial w_i} = \sum_{j=1}^{k} \left( y_{ij} - \hat{p}(y=1|x_i,w) \right)
\]

Weighed by contribution of \( i \) th feature to predict \( j \).
Gradient Ascent for LR

Gradient ascent algorithm: iterate until change $< \varepsilon$

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \sum_j [y^j - \hat{P}(Y^j = 1 \mid x^j, w^{(0)})]$$

For $i=1,\ldots,k$,

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid x^j, w^{(i)})]$$

repeat

Regularization in linear regression

- Overfitting usually leads to very large parameter choices, e.g.:
  - $-2.2 + 3.1 X - 0.30 X^2$
  - $-1.1 + 4,700,910.7 X - 8,585,638.4 X^2 + \ldots$

- Regularized least-squares (a.k.a. ridge regression), for $\lambda > 0$:

$$w^* = \arg \min_w \sum_j \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 + \lambda \sum_{i=1}^k w_i^2$$
Linear Separability

- If data is linearly separable, weights go to infinity

\[ \frac{1}{1 + e^{-x}} \]

- In general, leads to overfitting:
  - Penalizing high weights can prevent overfitting...

Large parameters → Overfitting

```
\frac{1}{1 + e^{-x}}
\frac{1}{1 + e^{-2x}}
\frac{1}{1 + e^{-100x}}
```

©Carlos Guestrin 2005-2013
Regularized Conditional Log Likelihood

- Add regularization penalty, e.g., $L_2$:
  \[
  \ell(w) = \ln \prod_{j=1}^{N} P(y_j | x^j, w) - \frac{\lambda}{2} \|w\|_2^2
  \]

- Practical note about $w_0$:
  \[
  \text{don’t regularize}
  \]

- Gradient of regularized likelihood:
  \[
  \frac{\partial \ell}{\partial w_i} = \frac{\partial}{\partial w_i} \left( \ln \prod_{j} P(y_j | x^j, w) \right) - \frac{\lambda}{2} \frac{\partial \|w\|_2^2}{\partial w_i}
  \]

Standard v. Regularized Updates

- Maximum conditional likelihood estimate
  \[
  w^* = \arg \max_w \ln \prod_{j=1}^{N} P(y_j | x^j, w)
  \]
  \[
  w_i^{(t+1)} = w_i^{(t)} + \eta \sum_j x_i^j [y_j - \hat{P}(Y^j = 1 | x^j, w)]
  \]

- Regularized maximum conditional likelihood estimate
  \[
  w^* = \arg \max_w \ln \prod_{j=1}^{N} P(y_j | x^j, w) - \frac{\lambda}{2} \sum_{i=1}^{k} w_i^2
  \]
  \[
  w_i^{(t+1)} = w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_j x_i^j [y_j - \hat{P}(Y^j = 1 | x^j, w)] \right\}
  \]
Please Stop!! Stopping criterion

\[ \ell(w) = \ln \prod_j P(y_j^i | x_j^i, w) - \lambda ||w||_2^2 \]

- When do we stop doing gradient descent?
  \[ \ell(w^t) - \ell(w) \leq \varepsilon \]
- Because \( \ell(w) \) is strongly concave:
  - i.e., because of some technical condition
  \[ \ell(w^*) - \ell(w) \leq \frac{1}{2\lambda} ||\nabla \ell(w)||_2^2 < \varepsilon \]
- Thus, stop when:
  \[ \frac{1}{2\lambda} ||\nabla \ell(w||_2^2 < \varepsilon \]

Digression: Logistic regression for more than 2 classes

- Logistic regression in more general case (C classes), where \( Y \) in \{1,...,C\}
  - for \( C \) classes \( (C-1)(k+1) \) params
  - A class \( c \in \{1,...,C-1\} \)
    \[ P(Y = c | x, w) \propto e^{w_0 + \sum_{k=1}^k w_k x_k} \]
    \[ P(Y = C | x, w) = 1 - \sum_{c=1}^{C-1} P(Y = c | x, w) \]
- \( C = 2 \)
  - Aims to learn: \( K+1 \) params to learn
    \[ P(Y = 1 | x, w) = e^{w_0 + \sum_{k=1}^k w_k x_k} \]
    \[ P(Y = 0 | x, w) = 1 - P(Y = 1 | x, w) = \frac{1}{1 + e^{w_0 + \sum_{k=1}^k w_k x_k}} \]
Digression: Logistic regression more generally

- Logistic regression in more general case, where $Y \in \{1, \ldots, C\}$

  for $c < C$
  $$P(Y = c | x, w) = \frac{\exp(w_{c0} + \sum_{i=1}^{k} w_{ci}x_i)}{1 + \sum_{c'=1}^{C-1} \exp(w_{c'0} + \sum_{i=1}^{k} w_{c'i}x_i)}$$

  for $c = C$ (normalization, so no weights for this class)
  $$P(Y = C | x, w) = \frac{1}{1 + \sum_{c'=1}^{C-1} \exp(w_{c'0} + \sum_{i=1}^{k} w_{c'i}x_i)}$$

Learning procedure is basically the same as what we derived!