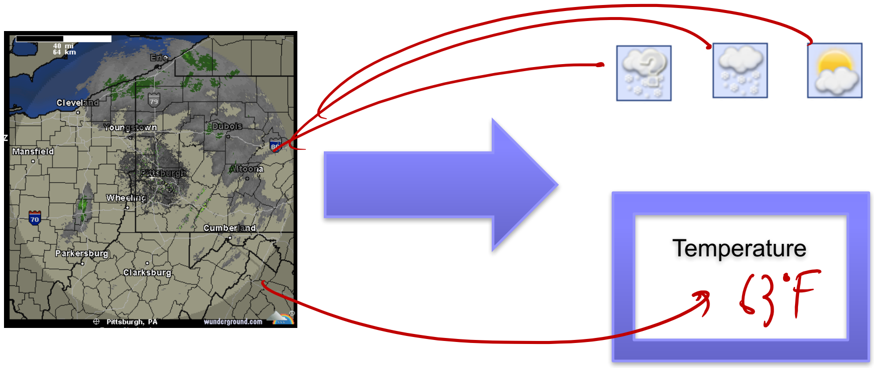
Classification
Logistic Regression

Machine Learning – CSE546
Sham Kakade
University of Washington

October 13, 2016

THUS FAR, REGRESSION: PREDICT A CONTINUOUS VALUE GIVEN SOME INPUTS

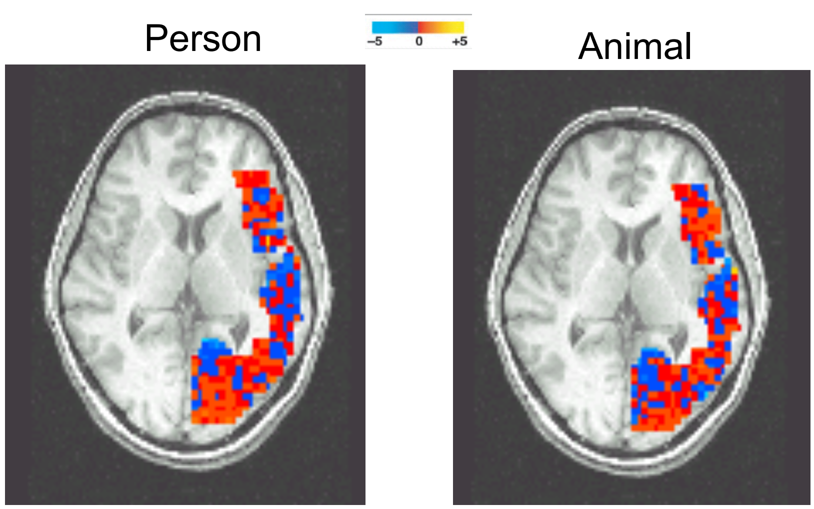
Weather prediction revisted



Reading Your Brain, Simple Example

[Mitchell et al.]

Pairwise classification accuracy: 85%



Classification

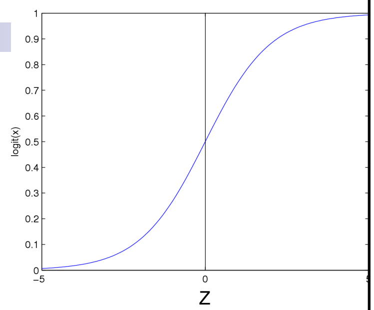
- **Learn**: h:**X** $\mapsto$ Y
  - **X** – features
  - Y – target classes
- Conditional probability: P(Y|**X**)
- Suppose you know P(Y|**X**) exactly, how should you classify?
  - Bayes optimal classifier:
- **How do we estimate P(Y|X)?**

Link Functions

- Estimating P(Y|**X**): Why not use standard linear regression?
- Combing regression and probability?
  - Need a mapping from real values to [0,1]
  - A link function!

Logistic Regression

Logistic function (or Sigmoid): $\frac{1}{1+exp(-z)}$



- Learn P(Y|**X**) directly
  - Assume a particular functional form for link function
  - Sigmoid applied to a linear function of the input features:

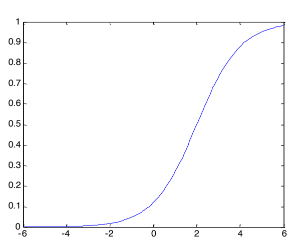
$$P(Y=0|X,W) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

Features can be discrete or continuous!

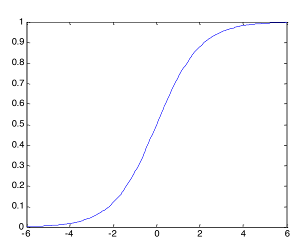
Understanding the sigmoid

$$g(w_0+\sum_i w_i x_i) = \frac{1}{1+e^{w_0+\sum_i w_i x_i}}$$

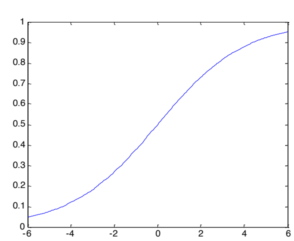
w_0=-2, w_1=-1



w_0=0, w_1=-1

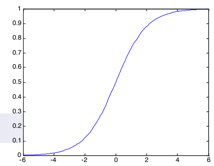


w_0=0, w_1=-0.5



Logistic Regression – a Linear classifier

$$\frac{1}{1 + exp(-z)}$$



$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{w_0 + \sum_i w_i x_i}}$$

Very convenient!

$$P(Y = 0 | X = < X_1, ... X_n >) = \frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

implies

$$P(Y = 1 | X = < X_1, ... X_n >) = \frac{exp(w_0 + \sum_i w_i X_i)}{1 + exp(w_0 + \sum_i w_i X_i)}$$

implies

$$\frac{P(Y = 1 | X)}{P(Y = 0 | X)} = exp(w_0 + \sum_i w_i X_i)$$

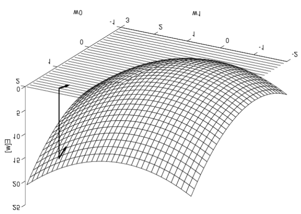
linear classification rule!

implies

$$\ln \frac{P(Y = 1 | X)}{P(Y = 0 | X)} = w_0 + \sum_i w_i X_i$$

Optimizing concave function – Gradient ascent

- Conditional likelihood for Logistic Regression is concave. Find optimum with gradient ascent



Gradient: $$\nabla_{\mathbf{w}} l(\mathbf{w}) = [\frac{\partial l(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial l(\mathbf{w})}{\partial w_n}]'$$

Step size, $\eta > 0$

Update rule: $$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$

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

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form:

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

$$\ln P(\mathcal{D}_Y \mid \mathcal{D}_{\mathbf{X}}, \mathbf{w}) = \sum_{j=1}^{N} \ln P(y^j \mid \mathbf{x}^j, \mathbf{w})$$

Expressing Conditional Log Likelihood

$$P(Y=0|\mathbf{X},\mathbf{w}) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

$$P(Y=1|\mathbf{X},\mathbf{w}) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

$$l(\mathbf{w}) \equiv \sum_j \ln P(y^j|\mathbf{x}^j,\mathbf{w})$$

$$\ell(\mathbf{w}) = \sum_j y^j \ln P(Y=1|\mathbf{x}^j,\mathbf{w}) + (1-y^j)\ln P(Y=0|\mathbf{x}^j,\mathbf{w})$$

Maximizing Conditional Log Likelihood

$$P(Y=0|X,W) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

$$P(Y=1|X,W) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

$$\begin{aligned} l(\mathbf{w}) &\equiv \ln \prod_j P(y^j|\mathbf{x}^j,\mathbf{w}) \\ &= \sum_j y^j(w_0 + \sum_i^n w_i x_i^j) - \ln(1+exp(w_0+\sum_i^n w_i x_i^j)) \end{aligned}$$

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

Bad news: no closed-form solution to maximize $l(\mathbf{w})$

Good news: concave functions easy to optimize

Maximize Conditional Log Likelihood: Gradient ascent

$$l(\mathbf{w}) = \sum_j y^j(w_0 + \sum_i^n w_i x_i^j) - \ln(1 + exp(w_0 + \sum_i^n w_i x_i^j))$$

Gradient Ascent for LR

Gradient ascent algorithm: iterate until change < ε

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

For i=1,…,k,

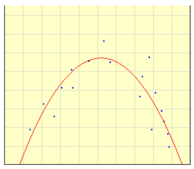
$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]$$

repeat

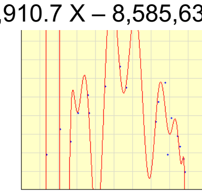
Regularization in linear regression

- Overfitting usually leads to very large parameter choices, e.g.:

-2.2 + 3.1 X – 0.30 X^2



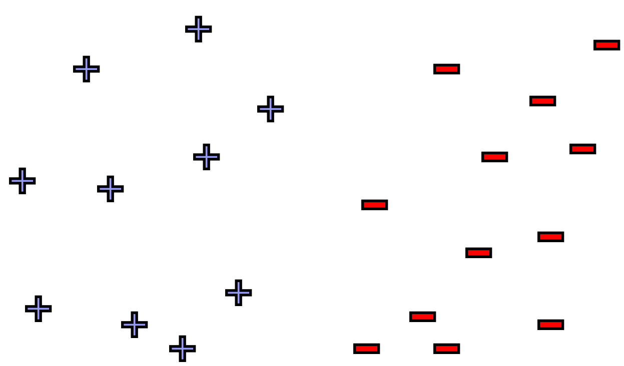
-1.1 + 4,700,910.7 X – 8,585,638.4 X^2 + …



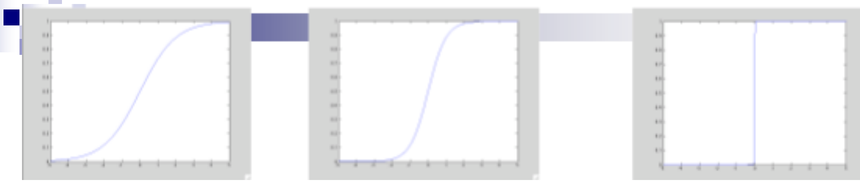
- Regularized least-squares (a.k.a. ridge regression), for $\lambda>0$:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

Linear Separability



Large parameters → Overfitting



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

- If data is linearly separable, weights go to infinity
  - In general, leads to overfitting:
- Penalizing high weights can prevent overfitting…

Regularized Conditional Log Likelihood

- Add regularization penalty, e.g., L_2:

$$\ell(\mathbf{w}) = \ln \prod_{j=1}^{N} P(y^j|\mathbf{x}^j, \mathbf{w}) - \frac{\lambda}{2}||\mathbf{w}||_2^2$$

- Practical note about w_0:
- Gradient of regularized likelihood:

Standard v. Regularized Updates

- Maximum conditional likelihood estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ \ln \prod_{j=1}^{N} P(y^j | \mathbf{x}^j, \mathbf{w})$$

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

- Regularized maximum conditional likelihood estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ \ln \prod_{j=1}^{N} P(y^j | \mathbf{x}^j, \mathbf{w}) - \frac{\lambda}{2} \sum_{i=1}^{k} w_i^2$$

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

Please Stop!! Stopping criterion

$$\ell(\mathbf{w}) = \ln \prod_j P(y^j | \mathbf{x}^j, \mathbf{w})) - \lambda ||\mathbf{w}||_2^2$$

- When do we stop doing gradient descent?

- Because $l(\mathbf{w})$ is strongly concave:
  - i.e., because of some technical condition

$$\ell(\mathbf{w}^*) - \ell(\mathbf{w}) \leq \frac{1}{2\lambda} ||\nabla \ell(\mathbf{w})||_2^2$$

- Thus, stop when:

Digression: Logistic regression for more than 2 classes

- Logistic regression in more general case (C classes), where *Y in* {0,…,C-1}

Digression: Logistic regression more generally

- Logistic regression in more general case, where *Y in* {0,…,C-1}

for *c>0*

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

for *c=0* (normalization, so no weights for this class)

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

Learning procedure is basically the same as what we derived!

Stochastic Gradient Descent

Machine Learning – CSE546
Sham Kakade
University of Washington

October 13, 2016

The Cost, The Cost!!! Think about the cost…

- What's the cost of a gradient update step for LR???

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

Learning Problems as Expectations

- Minimizing loss in training data:
  - Given dataset:
    - Sampled iid from some distribution p(**x**) on features:
  - Loss function, e.g., hinge loss, logistic loss,…
  - We often minimize loss in training data:

$$\ell_{\mathcal{D}}(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} \ell(\mathbf{w}, \mathbf{x}^j)$$

- However, we should really minimize expected loss on all data:

$$\ell(\mathbf{w}) = E_{\mathbf{x}}\left[\ell(\mathbf{w}, \mathbf{x})\right] = \int p(\mathbf{x})\ell(\mathbf{w}, \mathbf{x})d\mathbf{x}$$

- So, we are approximating the integral by the average on the training data

Gradient ascent in Terms of Expectations

- "True" objective function:

$$\ell(\mathbf{w}) = E_{\mathbf{x}}\left[\ell(\mathbf{w}, \mathbf{x})\right] = \int p(\mathbf{x})\ell(\mathbf{w}, \mathbf{x})d\mathbf{x}$$

- Taking the gradient:

- "True" gradient ascent rule:

- How do we estimate expected gradient?

SGD: Stochastic Gradient Ascent (or Descent)

- "True" gradient: $\nabla \ell(\mathbf{w}) = E_{\mathbf{x}}\left[\nabla \ell(\mathbf{w}, \mathbf{x})\right]$
- Sample based approximation:
- What if we estimate gradient with just one sample???
  - Unbiased estimate of gradient
  - Very noisy!
  - Called stochastic gradient ascent (or descent)
    - Among many other names
  - VERY useful in practice!!!

Stochastic Gradient Ascent for Logistic Regression

- Logistic loss as a stochastic function:

$$E_{\mathbf{x}}\left[\ell(\mathbf{w}, \mathbf{x})\right] = E_{\mathbf{x}}\left[\ln P(y|\mathbf{x}, \mathbf{w}) - \lambda ||\mathbf{w}||_2^2\right]$$

- Batch gradient ascent updates:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \frac{1}{N} \sum_{j=1}^{N} x_i^{(j)} [y^{(j)} - P(Y = 1|\mathbf{x}^{(j)}, \mathbf{w}^{(t)})] \right\}$$

- Stochastic gradient ascent updates:
  - Online setting:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \left\{ -\lambda w_i^{(t)} + x_i^{(t)} [y^{(t)} - P(Y = 1|\mathbf{x}^{(t)}, \mathbf{w}^{(t)})] \right\}$$

Stochastic Gradient Ascent: general case

- Given a stochastic function of parameters:
  - Want to find maximum

- Start from $\mathbf{w}^{(0)}$
- Repeat until convergence:
  - Get a sample data point $\mathbf{x}^t$
  - Update parameters:

- Works on the online learning setting!
- Complexity of each gradient step is constant in number of examples!
- In general, step size changes with iterations

What you should know…

- Classification: predict discrete classes rather than real values
- Logistic regression model: Linear model
  - Logistic function maps real values to [0,1]
- Optimize conditional likelihood
- Gradient computation
- Overfitting
- Regularization
- Regularized optimization
- Cost of gradient step is high, use stochastic gradient descent

Stopping criterion

$$\ell(\mathbf{w}) = \ln \prod_j P(y^j|\mathbf{x}^j, \mathbf{w})) - \lambda||\mathbf{w}||_2^2$$

- Regularized logistic regression is strongly concave
  - Negative second derivative bounded away from zero:
- Strong concavity (convexity) is super helpful!!
- For example, for strongly concave *l*(**w**):

$$\ell(\mathbf{w}^*) - \ell(\mathbf{w}) \leq \frac{1}{2\lambda}||\nabla\ell(\mathbf{w})||_2^2$$

Convergence rates for gradient descent/ascent

- Number of Iterations to get to accuracy

$$\ell(\mathbf{w}^*) - \ell(\mathbf{w}) \leq \epsilon$$

- If func Lipschitz: $O(1/\epsilon^2)$
- If gradient of func Lipschitz: $O(1/\epsilon)$
- If func is strongly convex: $O(\ln(1/\epsilon))$