

Logistic Regression & Discriminative Classifier

Dr. Jianlin Cheng

**Department of Electrical Engineering and Computer
Science**

University of Missouri, Columbia

Fall, 2017

Slides Adapted from Book and CMU, Stanford Machine Learning Courses

Naïve Bayes Recap...

- Optimal Classifier: $f^*(x) = \arg \max_y P(y|x)$

- NB Assumption: $P(X_1 \dots X_d | Y) = \prod_{i=1}^d P(X_i | Y)$

- NB Classifier:

$$f_{NB}(x) = \arg \max_y \prod_{i=1}^d P(x_i | y) P(y)$$

- Assume parametric form for $P(X_i | Y)$ and $P(Y)$
 - Estimate parameters using MLE/MAP and plug in

Generative vs. Discriminative Classifiers

Generative classifiers (e.g. Naïve Bayes)

- Assume some functional form for $P(X,Y)$ (or $P(X|Y)$ and $P(Y)$)
- Estimate parameters of $P(X|Y)$, $P(Y)$ directly from training data
- Use Bayes rule to calculate $P(Y|X)$

Why not learn $P(Y|X)$ directly? Or better yet, why not learn the decision boundary directly?

Discriminative classifiers (e.g. Logistic Regression)

- Assume some functional form for $P(Y|X)$ or for the decision boundary
- Estimate parameters of $P(Y|X)$ directly from training data

Logistic Regression

Example:
Drug dose response experiments

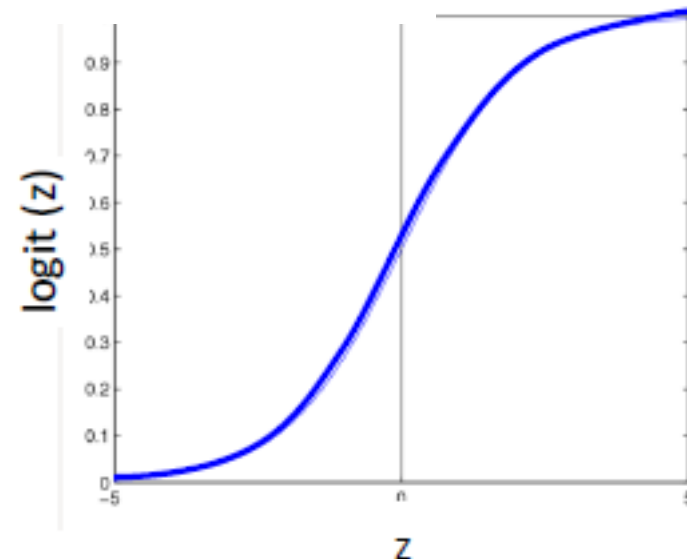
Assumes the following functional form for $P(Y|X)$:

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

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

Logistic function applied to a linear function of the data

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



Features can be discrete or continuous!

Logistic Regression is a Linear Classifier!

Assumes the following functional form for $P(Y|X)$:

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

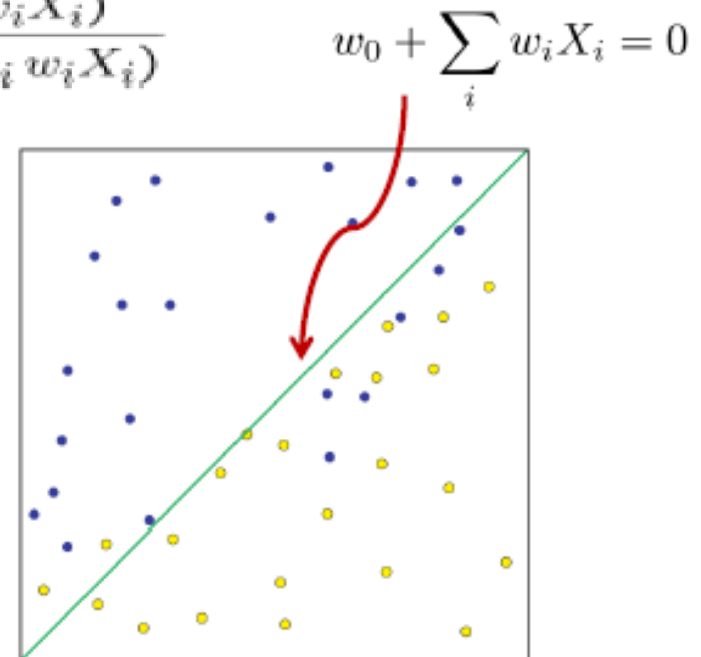
$$P(Y = 1|\mathbf{X}, \mathbf{w}) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

Decision boundary:

$$P(Y = 1|X) \stackrel{1}{\geq} P(Y = 0|X)$$

$$w_0 + \sum_i w_i X_i \stackrel{1}{\geq} 0$$

(Linear Decision Boundary)



Logistic Regression is a Linear Classifier!

Assumes the following functional form for $P(Y|X)$:

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

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

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

$$\Rightarrow w_0 + \sum_i w_i X_i \stackrel{\mathbf{1}}{\geq} \mathbf{0}$$

Logistic Regression for more than 2 classes

- Logistic regression in more general case, where $Y \in \{y_1, \dots, y_K\}$

for $k < K$

$$P(Y = y_k | X) = \frac{\exp(w_{k0} + \sum_{i=1}^d w_{ki} X_i)}{1 + \sum_{j=1}^{K-1} \exp(w_{j0} + \sum_{i=1}^d w_{ji} X_i)}$$

for $k=K$ (normalization, so no weights for this class)

$$P(Y = y_K | X) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(w_{j0} + \sum_{i=1}^d w_{ji} X_i)}$$

Is the decision boundary still linear?

Training Logistic Regression

We'll focus on binary classification:

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

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

How to learn the parameters w_0, w_1, \dots, w_d ?

Training Data $\{(X^{(j)}, Y^{(j)})\}_{j=1}^n$ $X^{(j)} = (X_1^{(j)}, \dots, X_d^{(j)})$

Maximum Likelihood Estimates

$$\hat{\mathbf{w}}_{MLE} = \arg \max_{\mathbf{w}} \prod_{j=1}^n P(X^{(j)}, Y^{(j)} | \mathbf{w})$$

But there is a problem ...

Don't have a model for $P(X)$ or $P(X|Y)$ - only for $P(Y|X)$

Training Logistic Regression

How to learn the parameters w_0, w_1, \dots, w_d ?

Training Data $\{(X^{(j)}, Y^{(j)})\}_{j=1}^n$ $X^{(j)} = (X_1^{(j)}, \dots, X_d^{(j)})$

Maximum (Conditional) Likelihood Estimates

$$\hat{\mathbf{w}}_{MCLE} = \arg \max_{\mathbf{w}} \prod_{j=1}^n P(Y^{(j)} | X^{(j)}, \mathbf{w})$$

Discriminative philosophy – Don't waste effort learning $P(X)$, focus on $P(Y|X)$ – that's all that matters for classification!

Expressing Conditional log Likelihood

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

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

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

MLE Estimate

$$P(O|w) = \prod_{j=1}^n P(y^j | x^j, w) = \prod_{j=1}^n P(y^j = 1 | x^j, w)^{y^j} P(y^j = 0 | x^j, w)^{1-y^j}$$
$$= \prod_{j=1}^n \left(\frac{\exp(w_0 + \sum_i w_i x_i^j)}{1 + \exp(w_0 + \sum_i w_i x_i^j)} \right)^{y^j} \left(\frac{1}{1 + \exp(w_0 + \sum_i w_i x_i^j)} \right)^{1-y^j}$$

$$l(w) = \ln P(O|w) = \sum_{j=1}^n \left[y^j (w_0 + \sum_i w_i x_i^j) - \ln(1 + \exp(w_0 + \sum_i w_i x_i^j)) - (1-y^j) (-\ln(1 + \exp(w_0 + \sum_i w_i x_i^j))) \right]$$

$$= \sum_{j=1}^n \left[y^j (w_0 + \sum_{i=1}^d w_i x_i^j) - \ln(1 + \exp(w_0 + \sum_{i=1}^d w_i x_i^j)) \right]$$

$$\frac{\partial l(w)}{\partial w_i} = \sum_{j=1}^n \left[x_i^j y^j - \frac{\exp(w_0 + \sum_{i=1}^d w_i x_i^j) \cdot x_i^j}{1 + \exp(w_0 + \sum_{i=1}^d w_i x_i^j)} \right]$$

$$= \sum_{j=1}^n x_i^j \left(y^j - \frac{\exp(w_0 + \sum_{i=1}^d w_i x_i^j)}{1 + \exp(w_0 + \sum_{i=1}^d w_i x_i^j)} \right)$$

$$= \sum_{j=1}^n x_i^j \left(y^j - P(y^j = 1 | x^j, w) \right)$$

Maximizing Conditional log Likelihood

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

Good news: $l(\mathbf{w})$ is concave function of \mathbf{w} \rightarrow no locally optimal solutions

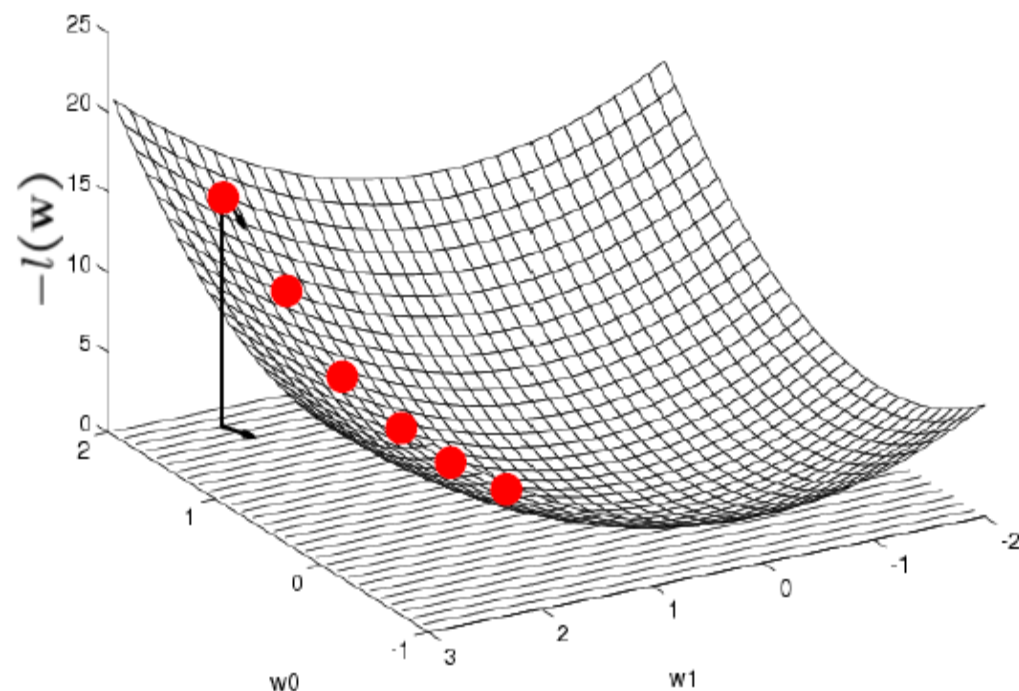
Bad news: no closed-form solution to maximize $l(\mathbf{w})$

Good news: concave functions easy to optimize (unique maximum)

Optimizing concave/convex function

- Conditional likelihood for Logistic Regression is concave
- Maximum of a concave function = minimum of a convex function

Gradient Ascent (concave)/ Gradient Descent (convex)



Gradient:

$$\nabla_{\mathbf{w}} l(\mathbf{w}) = \left[\frac{\partial l(\mathbf{w})}{\partial w_0}, \dots, \frac{\partial l(\mathbf{w})}{\partial w_n} \right]'$$

Update rule:

Learning rate, $\eta > 0$

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

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left. \frac{\partial l(\mathbf{w})}{\partial w_i} \right|_t$$

Calculate Partial Derivative – A Beautiful Result

$$\frac{\partial l(w)}{\partial w_i} = \sum_j y^j x_i^j - \frac{\exp(w_o + \sum_1^d w_i x_i^j) x_i^j}{1 + \exp(w_o + \sum_1^d w_i x_i^j)} = \sum_j x_i^j (y^j - \frac{\exp(w_o + \sum_1^d w_i x_i^j)}{1 + \exp(w_o + \sum_1^d w_i x_i^j)})$$

$$\frac{\partial l(w)}{\partial w_i} = \sum_j x_i^j (y^j - P(y^j = 1 | x^j, w))$$

Gradient Ascent for Logistic Regression

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

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

For $i=1, \dots, d$,

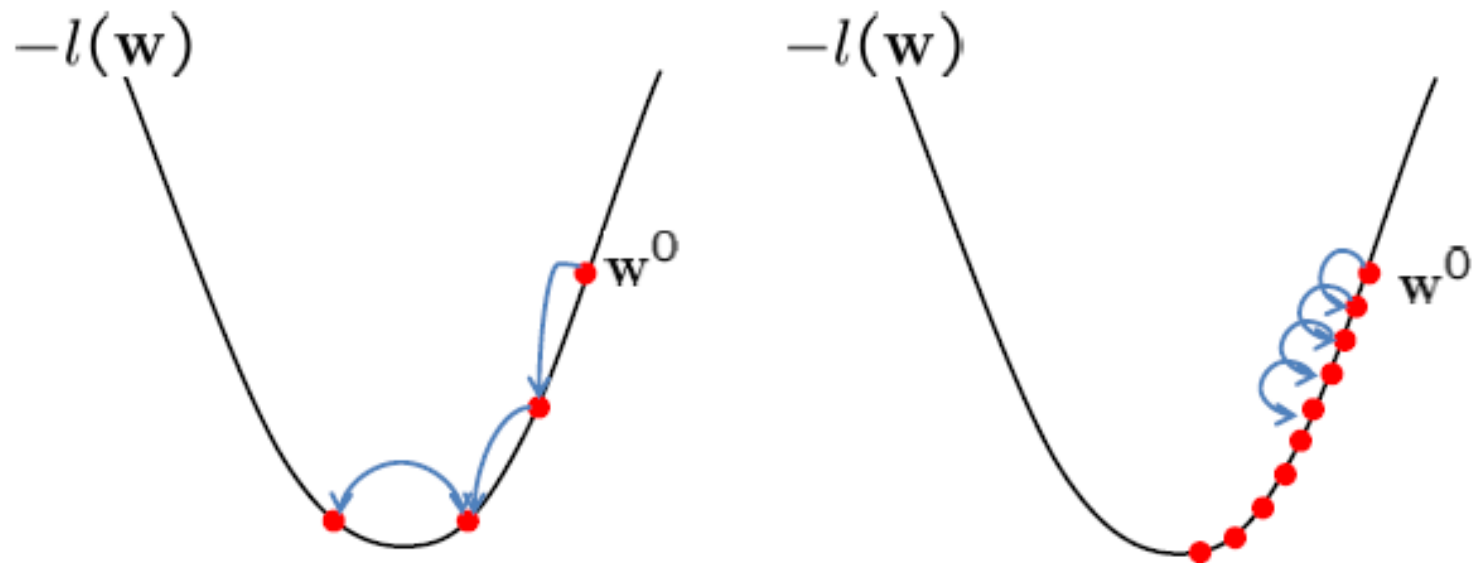
$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]$$

repeat

Predict what current weight thinks label Y should be

- Gradient ascent is simplest of optimization approaches
 - e.g., Newton method, Conjugate gradient ascent, IRLS (see Bishop 4.3.3)

Effect of step-size η



Large $\eta \Rightarrow$ Fast convergence but larger residual error
Also possible oscillations

Small $\eta \Rightarrow$ Slow convergence but small residual error

That's all M(C)LE. How about MAP?

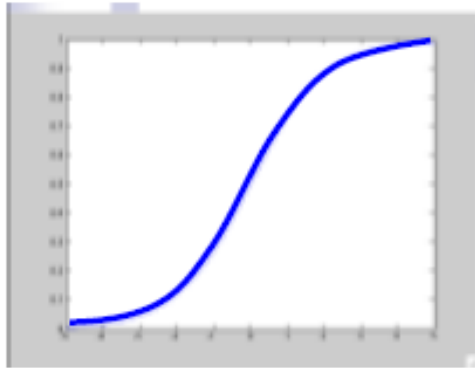
$$p(\mathbf{w} \mid Y, \mathbf{X}) \propto P(Y \mid \mathbf{X}, \mathbf{w})p(\mathbf{w})$$

- One common approach is to define priors on \mathbf{w}
 - Normal distribution, zero mean, identity covariance
 - “Pushes” parameters towards zero
- Corresponds to **Regularization**
 - Helps avoid very large weights and overfitting
 - More on this later in the semester

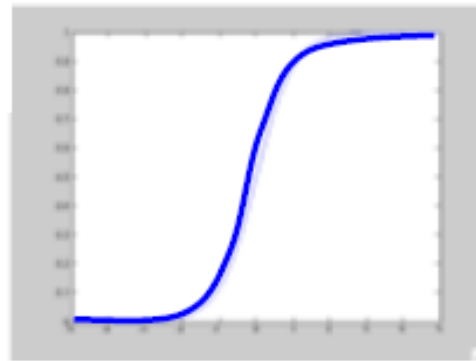
- M(C)AP estimate

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \ln \left[p(\mathbf{w}) \prod_{j=1}^n P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

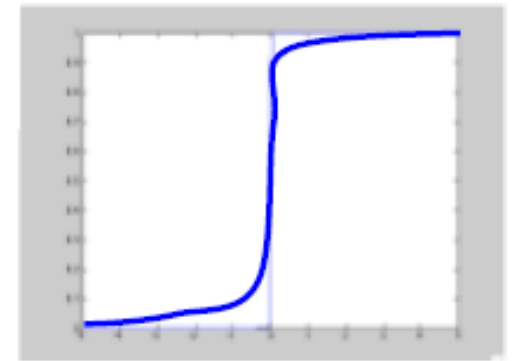
Large weights \rightarrow Overfitting



$$\frac{1}{1 + e^{-x}}$$



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



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

- Large weights lead to overfitting:

$$\begin{array}{ccc|ccc} & 1 & 1 & 1 & & & \\ & & 1 & & 0 & & \\ 1 & & & & 0 & 0 & \\ & 0 & & 0 & & & \end{array}$$

- Penalizing high weights can prevent overfitting...
 - again, more on this later in the semester

M(C)AP – Regularization

- Regularization

$$\arg \max_{\mathbf{w}} \ln \left[p(\mathbf{w}) \prod_{j=1}^n P(y^j | \mathbf{x}^j, \mathbf{w}) \right]$$

$$p(\mathbf{w}) = \prod_i \frac{1}{\kappa \sqrt{2\pi}} e^{\frac{-w_i^2}{2\kappa^2}}$$

Zero-mean Gaussian prior

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \sum_{j=1}^n \ln P(y^j | \mathbf{x}^j, \mathbf{w}) - \underbrace{\sum_{i=1}^d \frac{w_i^2}{2\kappa^2}}$$

Penalizes large weights

Calculate Partial Derivative – A Beautiful Result

$$l(\mathbf{w}) = \sum_{j=1}^n \ln P(y^j | \mathbf{x}^j, \mathbf{w}) - \sum_{i=1}^d \frac{w_i^2}{2\kappa^2}$$

$$\frac{\partial l(\mathbf{w})}{\partial w_i} = \sum_j x_i^j (y^j - P(y^j = 1 | \mathbf{x}^j, \mathbf{w})) - \frac{w_i}{\kappa^2}$$

Gradient Ascent for Logistic Regression

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

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

For $i=1, \dots, d$,

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

$-\eta \frac{w_i}{k^2}$

repeat

Predict what current weight thinks label Y should be

- Gradient ascent is simplest of optimization approaches
 - e.g., Newton method, Conjugate gradient ascent, IRLS (see Bishop 4.3.3)