

# Logistic Regression

Maria-Florina Balcan

02/07/2018

# Naïve Bayes Recap

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

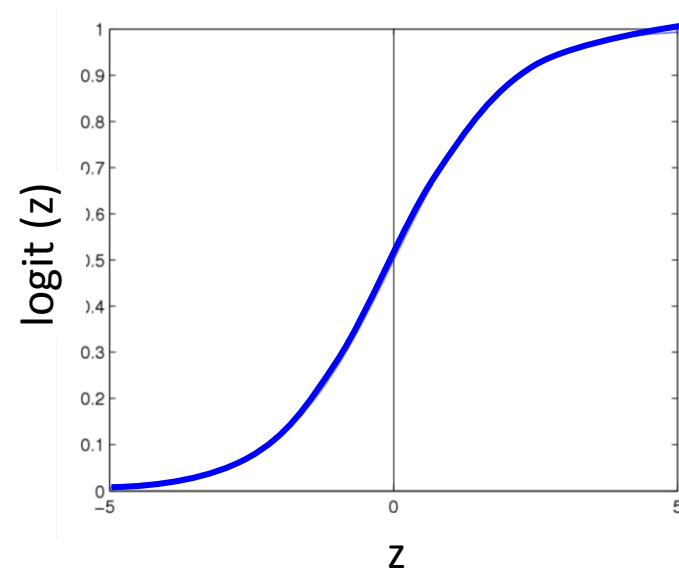
# Logistic Regression

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

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

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  $P(Y|X)$ :

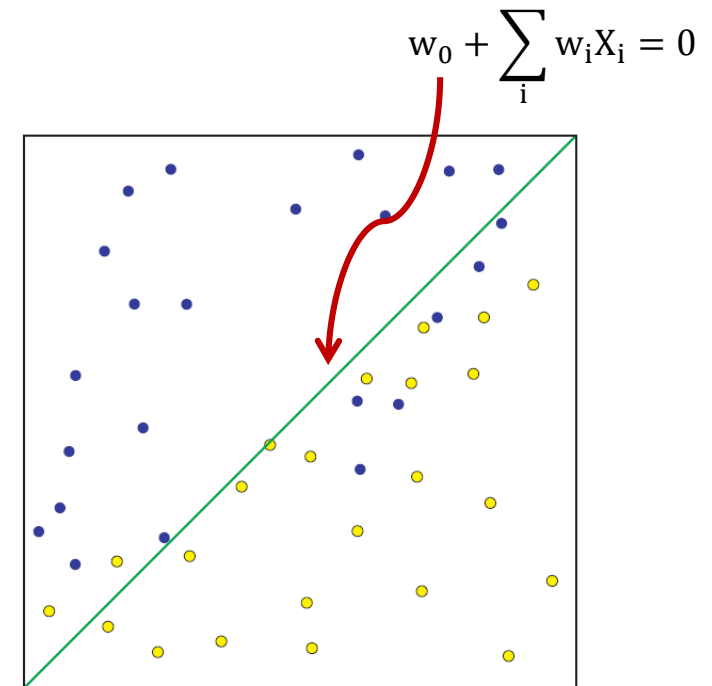
$$P(Y = 1|X) = \frac{1}{1 + \exp(-(w_0 + \sum_i w_i X_i))} = \frac{\exp(w_0 + \sum_i w_i X_i)}{\exp(w_0 + \sum_i w_i X_i) + 1}$$

Decision boundary:

$$P(Y = 1|X) > P(Y = 0|X) ?$$

$$w_0 + \sum_i w_i X_i > 0 ?$$

(Linear Decision Boundary)



# Logistic Regression is a Linear Classifier!

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

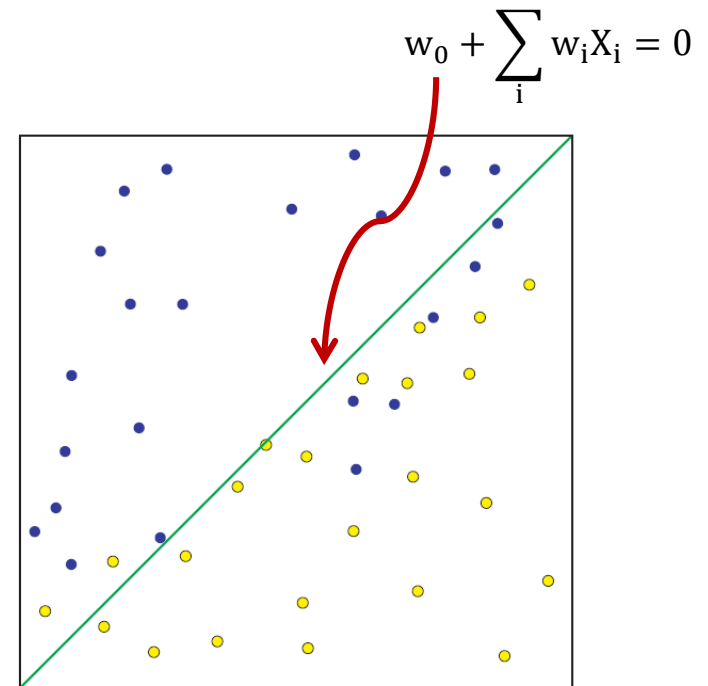
$$P(Y = 1|X) = \frac{1}{1 + \exp(-(w_0 + \sum_i w_i X_i))} = \frac{\exp(w_0 + \sum_i w_i X_i)}{\exp(w_0 + \sum_i w_i X_i) + 1}$$

Assumes a linear decision boundary: there are weights  $w_i$  s.t. when  $w_0 + \sum_i w_i X_i > 0$ , the example is more likely to be positive, and when this linear function is negative ( $w_0 + \sum_i w_i X_i < 0$ ) the example is more likely to be negative.

$$w_0 + \sum_i w_i X_i = 0, P(Y = 1|X) = \frac{1}{2}$$

$$w_0 + \sum_i w_i X_i \rightarrow \infty, P(Y = 1|X) \rightarrow 1$$

$$w_0 + \sum_i w_i X_i \rightarrow -\infty, P(Y = 1|X) \rightarrow 0$$



# Logistic Regression is a Linear Classifier!

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

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

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

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

$$\Rightarrow w_0 + \sum_i w_i X_i > 0 ?$$

# Training Logistic Regression

We'll focus on binary classification:

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

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

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:

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

But there's 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

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

**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

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

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

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

# Maximizing Conditional log Likelihood

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

**Good news:**  $l(\mathbf{w})$  is concave in  $\mathbf{w}$ . Local optimum = global optimum

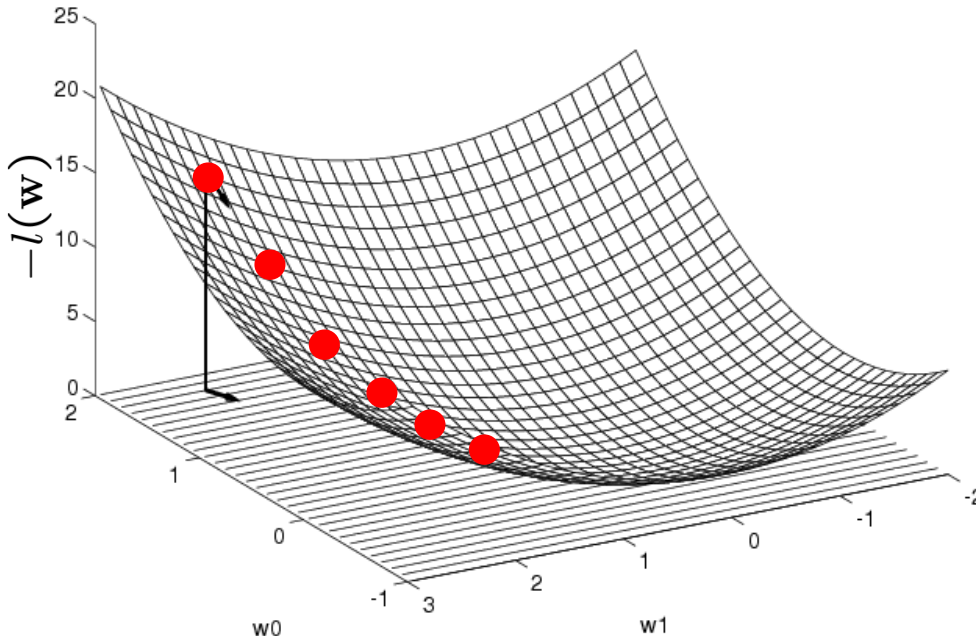
**Bad news:** no closed-form solution to maximize  $l(\mathbf{w})$

**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,  $\eta > 0$

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

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

# Gradient Ascent for Logistic Regression

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

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

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

$$w_i^{(t+1)} = w_i^{(t)} + \eta \sum_j x_i^j [y^j - \underbrace{\hat{P}(Y^j = 1 | \mathbf{x}^j, \mathbf{w}^{(t)})}_{\text{Predict what current weight thinks label Y should be}}]$$

repeat

Predict what current weight thinks label Y should be

look at actual labels of the examples, compare them to our current predictions, and then for each example  $j$  we multiply that difference by the feature value  $x_i^j$  and then add them up.

# Gradient Ascent for Logistic Regression

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

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

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

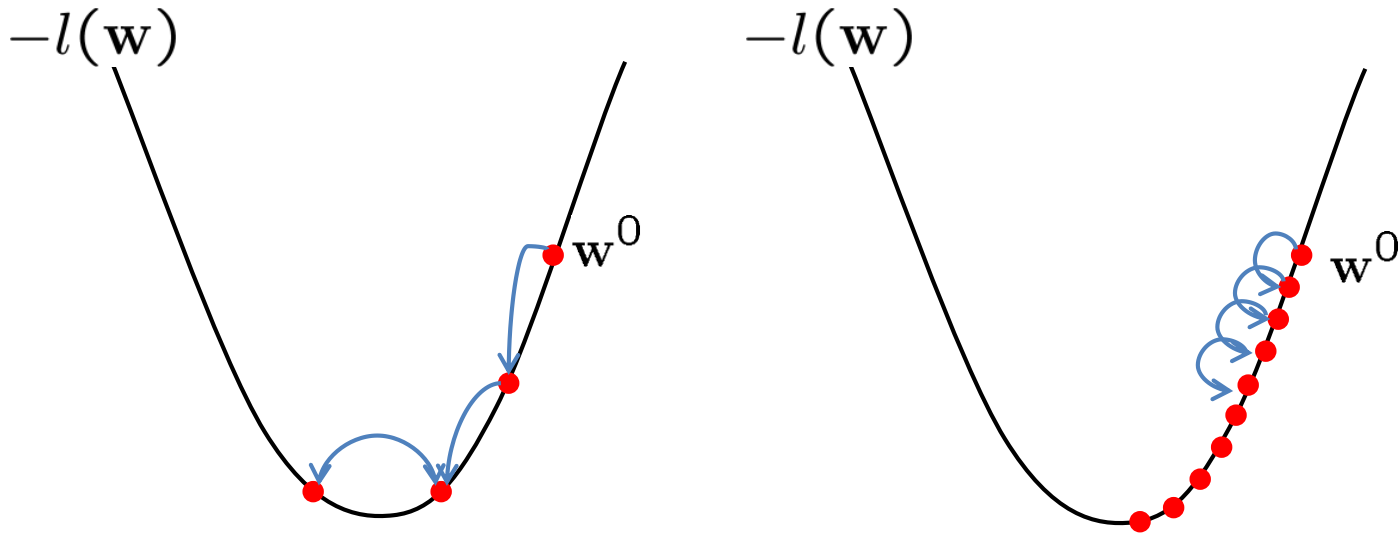
$$w_i^{(t+1)} = w_i^{(t)} + \eta \sum_j x_i^j [y^j - \underbrace{\hat{P}(Y^j = 1 | \mathbf{x}^j, \mathbf{w}^{(t)})}_{\text{Predict what current weight thinks label Y should be}}]$$

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 $\eta$



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

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



# What you should know

- LR is a linear classifier: decision rule is a hyperplane
- LR optimized by conditional likelihood
  - no closed-form solution
  - concave  $\Rightarrow$  global optimum with gradient ascent
  - Maximum conditional a posteriori corresponds to regularization