

# Logistic Regression and MaxEnt

Wei Wang @ CSE, UNSW

April 1, 2020

# Generative vs. Discriminative Learning

- Generative models:

$$\Pr[y \mid \mathbf{x}] = \frac{\Pr[\mathbf{x} \mid y]\Pr[y]}{\Pr[\mathbf{x}]}$$
$$\propto \Pr[\mathbf{x} \mid y]\Pr[y] = \Pr[\mathbf{x}, y]$$

- The key is to model the **generative** probability:  $\Pr[\mathbf{x} \mid y]$ .
- Example: Naive Bayes.
- Discriminative models:
  - models  $\Pr[y \mid \mathbf{x}]$  directly as  $g(\mathbf{x}; \theta)$ .
  - Example: Decision tree, Logistic Regression.
- Instance-based Learning.
  - Example:  $k$ NN classifier.

# Linear Regression

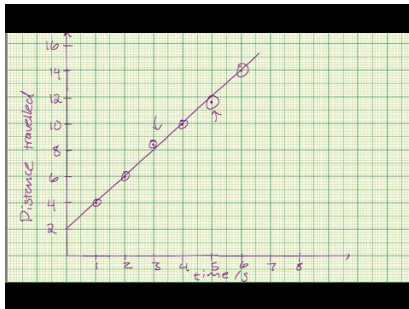


Figure: Linear Regression

## Task

- Input:  $(x^{(i)}, y^{(i)})$  pairs ( $1 \leq i \leq n$ )
- Preprocess: let  $\mathbf{x}^{(i)} = [1 \quad x^{(i)}]^\top$
- Output: The best  $\mathbf{w} = [w_0 \quad w_1]^\top$  such that  $\hat{y} = \mathbf{w}^\top \mathbf{x}$  **best** explains the observations

The criterion for “best”:

- Individual error:  $\epsilon_i = \hat{y}^{(i)} - y^{(i)}$
- Sum squared error:  $\ell = \sum_{i=1}^n \epsilon_i^2$

Find  $\mathbf{w}$  such that  $\ell$  is minimized.

# Minimizing a Function

## Taylor Series of $f(x)$ at point $a$

$$f(x) = \sum_{n=0}^{+\infty} \frac{f^{(n)}(a)}{n!} (x - a)^n \quad (1)$$

$$= f(a) + f'(a) \cdot (x - a) + \frac{f''(a)}{2} (x - a)^2 + o((x - a)^2) \quad (2)$$

- Intuitively,  $f(x)$  is almost  $f(a) + f'(a) \cdot (x - a)$  for all  $a$  if it is close to  $x$ .
- If  $f(x)$  has local minimum  $x^*$ , then
  - $f'(x^*) = 0$ , and
  - $f''(x^*) > 0$ .

Minimum of the local minima is the global minimum if it is smaller than the function values at all the boundary points.

- Intuitively,  $f(x)$  is almost  $f(a) + \frac{f''(a)}{2} (x - a)^2$  if  $a$  is close to  $x^*$ .

# Find the Least Square Fit for Linear Regression

$$\begin{aligned}\frac{\partial \ell}{\partial w_j} &= \sum_{i=1}^n 2\epsilon_i \frac{\partial \epsilon_i}{\partial w_j} = \sum_{i=1}^n 2\epsilon_i \frac{\partial \mathbf{w}^\top \mathbf{x}^{(i)}}{\partial w_j} \\ &= \sum_{i=1}^n 2\epsilon_i x_j^{(i)} = 2 \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)}) x_j^{(i)}\end{aligned}$$

By setting the above to 0, this essentially requires, **for all  $j$**

$$\sum_{i=1}^n \hat{y}^{(i)} x_j^{(i)} = \sum_{i=1}^n y^{(i)} x_j^{(i)}$$

what the model predicts

what the data says

# Find the Least Square Fit for Linear Regression

In the simple 1D case, we have only two parameters in  $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$

$$\sum_{i=1}^n (w_0 + w_1 x_1^{(i)}) x_0^{(i)} = \sum_{i=1}^n y^{(i)} x_0^{(i)}$$

$$\sum_{i=1}^n (w_0 + w_1 x_1^{(i)}) x_1^{(i)} = \sum_{i=1}^n y^{(i)} x_1^{(i)}$$

Since  $x_0^{(i)} = 1$ , they are essentially

$$\sum_{i=1}^n (w_0 + w_1 x_1^{(i)}) \cdot 1 = \sum_{i=1}^n y^{(i)} \cdot 1$$

$$\sum_{i=1}^n (w_0 + w_1 x_1^{(i)}) \cdot x_1^{(i)} = \sum_{i=1}^n y^{(i)} \cdot x_1^{(i)}$$

# Example

Using the same example in [https://en.wikipedia.org/wiki/Linear\\_least\\_squares\\_\(mathematics\)](https://en.wikipedia.org/wiki/Linear_least_squares_(mathematics))

$$\mathbf{X} = \begin{bmatrix} \text{—} & (x^{(1)})^\top & \text{—} \\ \text{—} & (x^{(2)})^\top & \text{—} \\ \text{—} & (x^{(3)})^\top & \text{—} \\ \text{—} & (x^{(4)})^\top & \text{—} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 6 \\ 5 \\ 7 \\ 10 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 6 \\ 5 \\ 7 \\ 10 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \end{bmatrix}$$



# Generalization to $m$ -dim

- Easily generalizes to more than 2-dim:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_m^{(1)} \\ 1 & \dots & \dots & \dots \\ 1 & x_1^{(i)} & \dots & x_m^{(i)} \\ 1 & \dots & \dots & \dots \\ 1 & x_1^{(n)} & \dots & x_m^{(n)} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_m \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(i)} \\ \dots \\ y^{(n)} \end{bmatrix}$$

- How to perform polynomial regression for one dimensional  $x$ ?
  - $\hat{y} = w_0 + w_1x + w_2x^2 \dots + w_mx^m$ .
  - Let  $x_j^{(i)} = (x_1^{(i)})^j \implies$  Polynomial least square fitting  
(<http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html>)

High-level idea:

- Observations (i.e., training data) are noisy
  - $P(y^{(i)} | \hat{y}^{(i)}) = f_i(\mathbf{w})$
- Any  $\mathbf{w}$  is possible, but some  $\mathbf{w}$  is most likely.
  - Assuming independence of training examples, the likelihood of the training dataset is  $\prod_i f_i(\mathbf{w})$ .
  - We shall choose the  $\mathbf{w}^*$  that **maximizes** the likelihood.
    - Maximum likelihood estimation (MLE)
    - If we also incorporate some prior on  $\mathbf{w}$ , this becomes Maximum Posterior Estimation (MAP)
    - If we assume some Gaussian prior on  $\mathbf{w}$ , this will add a  $\ell_2$  regularization term to the objective function.
- Many models and their variants can be deemed as different ways of estimating  $P(y^{(i)} | \hat{y}^{(i)})$

# Geometric Interpretation and the Closed Form Solution

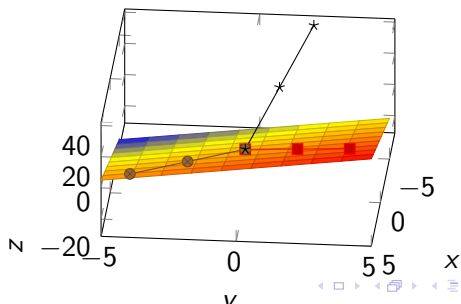
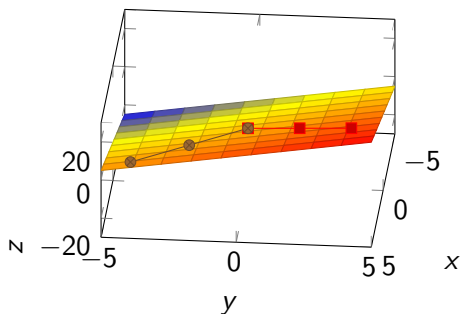
Find  $\mathbf{w}$  such that  $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2$  is minimized.

- What is  $\mathbf{X}\mathbf{w}$  when  $\mathbf{X}$  is fixed?
  - It is the hyperplane spanned by the  $d$  column vectors of  $\mathbf{X}$ .
- $\mathbf{y}$  in general is a vector outside the hyperplane. So the minimum distance is achieved when  $\mathbf{X}\mathbf{w}^*$  is exactly the projection of  $\mathbf{y}$  on the hyperplane. This means (denote  $i$ -th column of  $\mathbf{X}$  as  $X_i$ )

$$\left. \begin{array}{l} X_1^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \\ X_2^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \\ \dots\dots\dots \\ X_d^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0 \end{array} \right\} \implies \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$$

- $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^+ \mathbf{y}$  ( $\mathbf{X}^+$ : pseudo inverse of  $\mathbf{X}$ )

# Illustration



# Logistic Regression

Special case:  $y^{(i)} \in \{0, 1\}$ .

- Not appropriate to directly regress  $y^{(i)}$ .
- Rather, **model**  $y^{(i)}$  as the observed outcome of a Bernoulli trial with an unknown parameter  $p_i$
- How to model  $p_i$ 
  - We assume that  $p_i$  depends on  $\mathbf{x} \triangleq \mathbf{X}_{i\bullet} \implies$  rename  $p_i$  to  $p_{\mathbf{x}}$ .
  - Still hard to estimate  $p_{\mathbf{x}}$  reliably.
    - MLE:  $p_{\mathbf{x}} = \mathbf{E}[y = 1 \mid \mathbf{x}]$
  - What can we say about  $p_{\mathbf{x}+\epsilon}$  when  $p_{\mathbf{x}}$  is given?
- Answer: we impose a linear relationship between  $p_{\mathbf{x}}$  and  $\mathbf{x}$ 
  - What about a simple linear model  $p_{\mathbf{x}} = \mathbf{w}^T \mathbf{x}$  for some  $\mathbf{w}$ ? (Note: all points share the same parameter  $\mathbf{w}$ )
  - Problem: mismatch of the domains: vs
  - Solution: mean function / inverse of link function:

-1, 0, 1

- Solution: Link function  $g(\text{parameters}) \rightarrow \Re$

$$g(p) = \text{logit}(p) \triangleq \log \frac{p}{1-p} = \mathbf{w}^\top \mathbf{x} \quad (3)$$

- Equivalently, solve for  $p$ .

$$p = \frac{e^{\mathbf{w}^\top \mathbf{x}}}{1 + e^{\mathbf{w}^\top \mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}} = \sigma(\mathbf{w}^\top \mathbf{x}) \quad (4)$$

Where  $\sigma(z) = \frac{1}{1 + \exp(-z)}$ .

Recall that  $p_{\mathbf{x}} = \mathbf{E}[y = 1 \mid \mathbf{x}]$ .

- Decision boundary is  $p \geq 0.5$ .
  - Equivalent to whether  $\mathbf{w}^\top \mathbf{x} \geq 0$ . Hence, LR is a linear classifier.

# Learning the Parameter $\mathbf{w}$

- Consider a training data point  $\mathbf{x}^{(i)}$ .
  - Recall that the conditional probability ( $\Pr[y^{(i)} = 1 \mid \mathbf{x}^{(i)}]$ ) computed by the model is denoted by the shorthand notation  $p$  (which is a function of  $\mathbf{w}$  and  $\mathbf{x}^{(i)}$ ).
  - The likelihood of  $\mathbf{x}^{(i)}$  is  $\begin{cases} p & , \text{ if } y^{(i)} = 1 \\ 1 - p & , \text{ otherwise} \end{cases}$ , or equivalently,  $p^{y^{(i)}}(1 - p)^{1-y^{(i)}}$ .

- Hence, the likelihood of the whole training dataset is

$$L(\mathbf{w}) = \prod_{i=1}^n p(\mathbf{x}^{(i)})^{y^{(i)}} (1 - p(\mathbf{x}^{(i)}))^{1-y^{(i)}}.$$

- Log-likelihood is (assume  $\log \triangleq \ln$ )

$$\ell(\mathbf{w}) = \sum_{i=1}^n y^{(i)} \log p(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - p(\mathbf{x}^{(i)})) \quad (5)$$

# Learning the Parameter $\mathbf{w}$

- To maximize  $\ell$ , notice that it is concave. So take its partial derivatives

$$\begin{aligned}\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} &= \sum_{i=1}^n \left( y^{(i)} \frac{1}{p(\mathbf{x}^{(i)})} \frac{\partial p(\mathbf{x}^{(i)})}{\partial \mathbf{w}_j} + (1 - y^{(i)}) \frac{1}{1 - p(\mathbf{x}^{(i)})} \frac{\partial (1 - p(\mathbf{x}^{(i)}))}{\partial \mathbf{w}_j} \right) \\ &= \sum_{i=1}^n \left( \mathbf{x}^{(i)}_j y^{(i)} - \mathbf{x}^{(i)}_j p(\mathbf{x}^{(i)}) \right)\end{aligned}$$

- and set them to 0 essentially means, for all  $j$

$$\sum_{i=1}^n \hat{y}^{(i)} \cdot \mathbf{x}^{(i)}_j = \sum_{i=1}^n p(\mathbf{x}^{(i)}) \mathbf{x}^{(i)}_j = \sum_{i=1}^n y^{(i)} \cdot \mathbf{x}^{(i)}_j$$

what the model predicts

what the data says



# Understand the Equilibrium

- Consider one dimensional  $\mathbf{x}$ . The above condition is simplified to

$$\sum_{i=1}^n p^{(i)} x^{(i)} = \sum_{i=1}^n y^{(i)} x^{(i)}$$

- The RHS is essentially the sum of  $x$  values **only** for the training data in class  $Y = 1$ .
- The LHS says: if we use our learned model to assign a probability (of belonging to the class  $Y = 1$ ) for **every** training data, the LHS is the expected sum of  $x$  values.
- If this is still abstract, think of an example.

# Numeric Solution

- There is no closed-form solution to maximize  $\ell$ .
- Use the *Gradient Ascent* algorithm to maximize  $\ell$ .
- There are faster algorithms.

# (Stochastic) Gradient Ascent

- $\mathbf{w}$  is initialized to some random value (e.g.,  $\mathbf{0}$ ).
- Since the gradient gives the *steepest* direction to increase a function's value, we move a small step towards that direction, i.e.,

$$w_j \leftarrow w_j + \alpha \frac{\partial \ell(\mathbf{w})}{\partial w_j}, \text{ or}$$

$$w_j \leftarrow w_j + \alpha \sum_{i=1}^n (y^{(i)} - p(\mathbf{x}^{(i)})) \mathbf{x}^{(i)}_j$$

where  $\alpha$  (*learning rate*) is usually a small constant, or decreasing over the epochs.

- Stochastic version: using the gradient on a **randomly** selected training instance, i.e.,

$$w_j \leftarrow w_j + \alpha (y^{(i)} - p(\mathbf{x}^{(i)})) \mathbf{x}^{(i)}_j$$

- Gradient Ascent moves to the “right” direction a tiny step a time.  
Can we find a good step size?
- Consider 1D case: **minimize**  $f(x)$  and the current point is  $a$ .
  - $f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2}(x - a)^2$  for  $x$  near  $a$ .
  - To minimize  $f(x)$ , take  $\frac{\partial f(x)}{\partial x} = 0$ , i.e.,

$$\begin{aligned}\frac{\partial f(x)}{\partial x} &= 0 \\ \Leftrightarrow f'(a) \cdot 1 + \frac{f''(a)}{2} \cdot 2(x - a) \cdot 1 &= f'(a) + f''(a)(x - a) = 0 \\ \Leftrightarrow x = a - \frac{f'(a)}{f''(a)}\end{aligned}$$

- Can be applied to multiple dimension cases too  $\Rightarrow$  need to use  $\nabla$  (gradient) and Hess (Hessian).



# Regularization

- Regularization is another method to deal with **overfitting**.
  - It is designed to penalize large values of the model parameters.
  - Hence it *encourages* simpler models, which are less likely to overfit.
- Instead of optimizing for  $\ell(\mathbf{w})$ , we optimize  $\ell(\mathbf{w}) + \lambda R(\mathbf{w})$ .
  - $\lambda$  is a hyper-parameter that controls the strength of regularization.
    - It is usually determined by cross validating with a list of possible values (e.g., 0.001, 0.01, 0.1, 1, 10, ...)
    - Grid search:  
[http://scikit-learn.org/stable/modules/grid\\_search.html](http://scikit-learn.org/stable/modules/grid_search.html)
    - There are alternative methods.
  - $R(\mathbf{w})$  quantifies the “size” of the model parameters. Popular choices are:
    - $L_2$  regularization (**Ridge LR**)  $R(\mathbf{w}) = \|\mathbf{w}\|_2^2$
    - $L_1$  regularization (**Lasso LR**)  $R(\mathbf{w}) = \|\mathbf{w}\|_1$
    - $L_1$  regularization is more likely to result in sparse models.

# Generalizing LR to Multiple Classes

- LR can be generalized to multiple classes  $\Rightarrow$  **MaxEnt**.

$$\Pr[c | \mathbf{x}] \propto \exp(\mathbf{w}_c^\top \mathbf{x}) \quad \Rightarrow \quad \Pr[c | \mathbf{x}] = \frac{\exp(\mathbf{w}_c^\top \mathbf{x})}{Z}$$

- $Z$  is the normalization constant. 
- Let  $\mathbf{c}^*$  be the last class in  $C$ , then  $\mathbf{w}_{\mathbf{c}^*} = \mathbf{0}$ .
- Derive LR from MaxEnt. 
- Both belong to *exponential* or *log-linear* classifiers.

- Andrew Ng's note:  
<http://cs229.stanford.edu/notes/cs229-notes1.pdf>
- Cosma Shalizi's note: <http://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf>
- Tom Mitchell's book chapter:  
<https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf>