Logistic regression

cs434

Logistic regression

- Recall the problem of regression
  - Learns a linear mapping from input vector $\mathbf{x}$ to a continuous output $y \in (-\infty, \infty)$
- Logistic regression can be viewed as extending linear regression to handle binary output y by warping the output of a linear function to the range between 0 and 1
- For convenience, we will assume $y \in \{0,1\}$ for this lecture

Logistic regression (cont.)

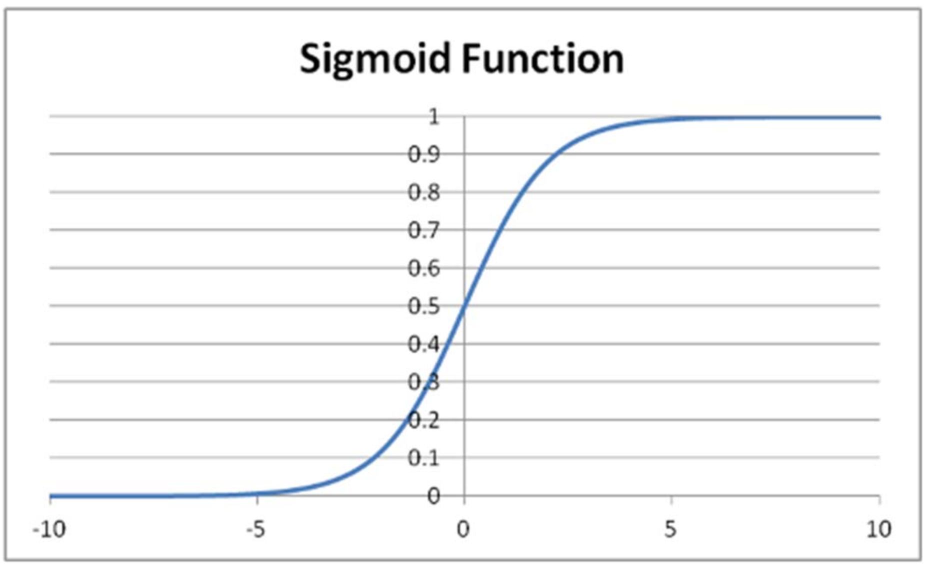
- Consider the linear regression function

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} = w_0 + w_1 x_1 + \cdots + w_m x_m$$

- We introduce a function g:

$$g(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

Referred to as the sigmoid function or logistic function



Logistic Regression Makes Probabilistic Prediction

- We interpret the output of logistic regression probabilistically:

$$g(\mathbf{w}^T\mathbf{x}) = P(y = 1|\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

i.e., probability of $y = 1$ given the input $\mathbf{x}$ and the model's parameter is $\mathbf{w}$

$$P(y = 0|\mathbf{x}; \mathbf{w}) = 1 - g(\mathbf{w}^T\mathbf{x})$$

$$= \frac{\exp(-\mathbf{w}^T\mathbf{x})}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

Logistic Regression forms a **linear** decision boundary

- We predict $y = 1$ if

$$p(y=1|\mathbf{x};\mathbf{w}) > p(y=0|\mathbf{x};\mathbf{w})$$

Predict $y = 1$ if

$$\frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} > 1$$

Predict $y = 1$ if

$$\log\frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = w_0 + w_1 x_1 + \cdots + w_m x_m > 0$$

Odds of y=1

Side Note:
the odds of an event are the quantity *p* / (1 – *p*), where *p* is the probability of the event
If I toss a fair dice, what are the odds that I will have a six?
LR assumes that the log odds of y=1 is a linear function of the input features

Learning **w** for logistic regression

- Given a set of training data points $(\mathbf{x}^{\boldsymbol{i}}, y^{i})$, $i = 1, \dots, n,$ the goal of learning is to find a weight vector $\mathbf{w}$ such that

$$g(\mathbf{w}^T\mathbf{x}) = P(y = 1|\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

  - is large (approaching 1) for positive ($y = 1$) training examples

  - is small (approaching 0) for negative ($y = 0$) training examples

Learning Objective for Logistic Regression

- Consider a general form of objective (to be minimized):

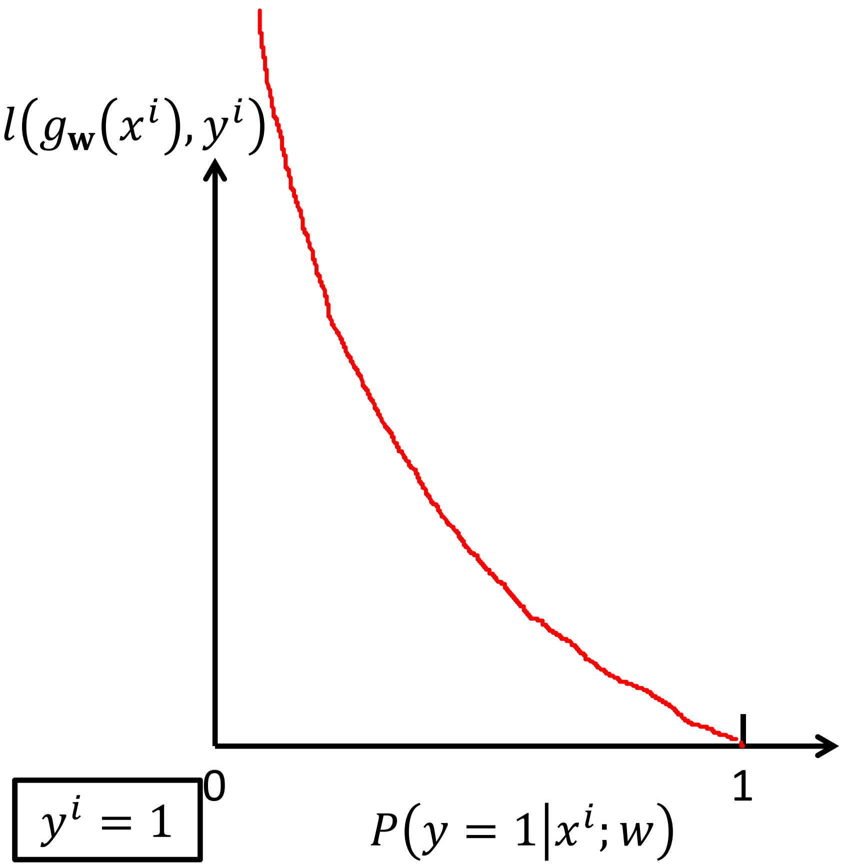
$$L(\mathbf{w}) = \sum_{i=1}^{n} l(g(\mathbf{w}^T\mathbf{x}^i), y^i)$$

- For logistic regression the loss function is:

$$l\left(g\left(\mathbf{w}^T\mathbf{x}^i\right), y^i\right) = \begin{cases} -\log g\left(\mathbf{w}^T\mathbf{x}^i\right) & \text{if } y^i = 1 \\ -\log\left(1 - g\left(\mathbf{w}^T\mathbf{x}^i\right)\right) & \text{if } y^i = 0 \end{cases}$$

Loss function for logistic regression

$$l\left(g\left(\mathbf{w}^T\mathbf{x}^i\right), y^i\right) = \begin{cases} -\log P\left(y=1 \middle| x^i; w\right) & \text{if } y^i = 1 \\ -\log\left(1 - P\left(y=1 \middle| x^i; w\right)\right) & \text{if } y^i = 0 \end{cases}$$

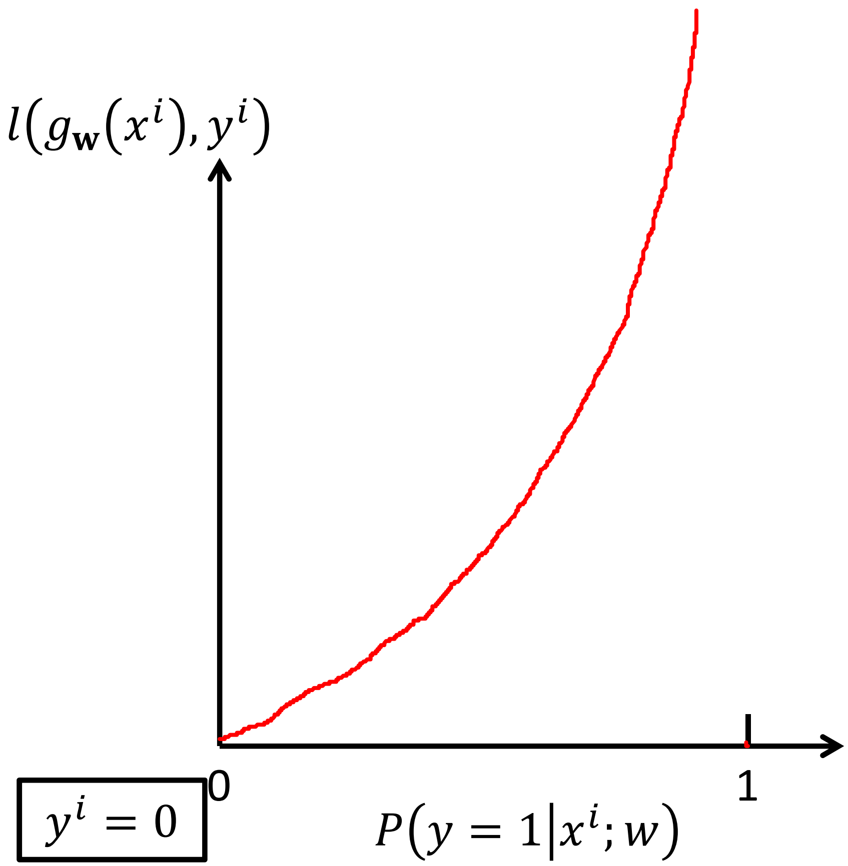


When $y^i = 1$,

- if we predict $P\left(y=1 \middle| x^i; w\right) = 1$, the loss is 0
- If we predict $P\left(y=1 \middle| x^i; w\right) = 0$, the loss is ∞

Loss function for logistic regression

$$l(g_{\mathbf{w}}(x^i), y^i) = \begin{cases} -\log P(y=1|x^i;w) & \text{if } y^i = 1 \\ -\log\left(1 - P(y=1|x^i;w)\right) & \text{if } y^i = 0 \end{cases}$$



When $y^i = 0$,

- if we predict $P(y=1|x^i;w) = 0$, the loss is 0
- If we predict $P(y=1|x^i;w) = 1$, the loss is ∞

Representing it compactly

$$l(g_{\mathbf{w}}(x^i), y^i) = \begin{cases} -\log P(y = 1|x^i; w) & \text{if } y^i = 1 \\ -\log\left(1 - P(y = 1|x^i; w)\right) & \text{if } y^i = 0 \end{cases}$$

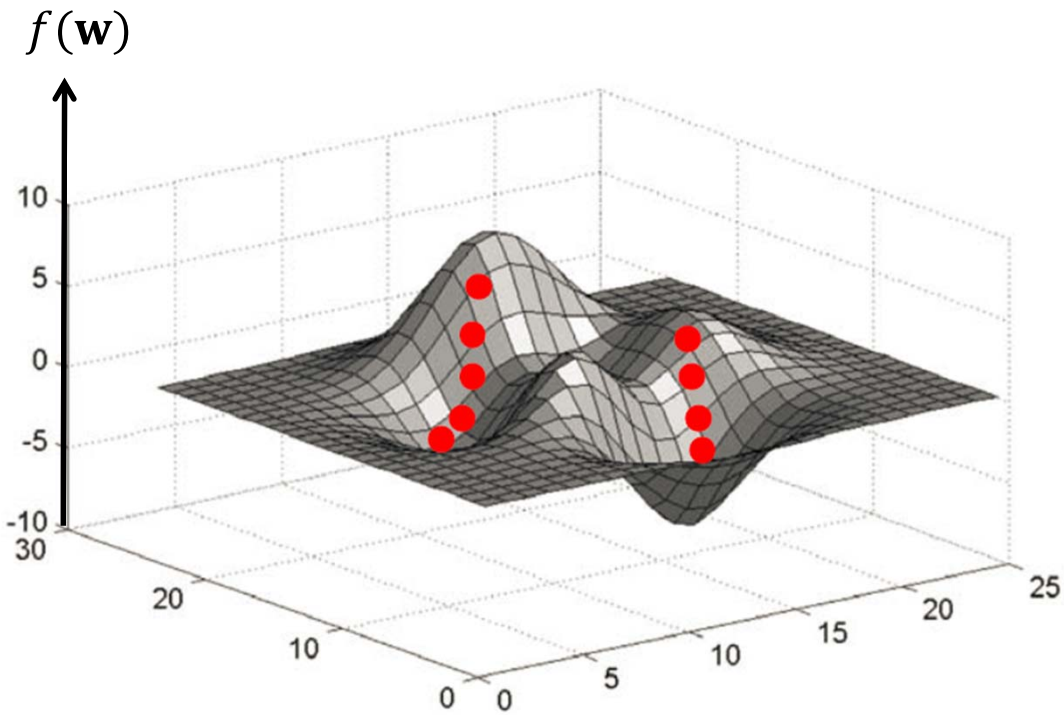
can be represented compactly as:

$$l(g_{\mathbf{w}}(x^i), y^i) =$$
$$-y^i \log P(y = 1|x^i; w) \; - \; (1 - y^i) \log\left(1 - P(y = 1|x^i; w)\right)$$

Minimizing $L(\mathbf{w})$

- Unfortunately this does not have a close form solution
  - You take the derivative, set it to zero, but no closed form solution
- Instead, we iteratively search for the optimal **w**
- Start with a random **w**, iteratively improve **w** (similar to Perceptron) by taking the negative gradient
- This is referred to as gradient descent

Gradient descent to minimize $L(\mathbf{w})$



1. Start from some initial guess $\mathbf{w}^0$

2. Find the direction of steepest descent – opposite of the gradient direction $\nabla f(\mathbf{w})$

3. Take a step toward that direction

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \nabla f(\mathbf{w}^t)$$

4. Repeat until no local improvement is possible

$(\ |\nabla f(\mathbf{w}^t)| \leq \epsilon)$

Different starting points may lead to different local minimum if objective is not convex (like what's shown in the above figure)

Gradient of $L(\mathbf{w})$

$$l(g(\mathbf{w}^T\mathbf{x}^i), y^i) = -y^i \log P(y=1|x^i; w) - (1-y^i)\log\left(1 - P(y=1|x^i;w)\right)$$

$$l(g(\mathbf{w}^T\mathbf{x}^i), y^i)$$

$$= -y^i \log g(\mathbf{w}^T\mathbf{x}^i) - (1-y^i)\log(1 - g(\mathbf{w}^T\mathbf{x}^i))$$

Useful fact: $g'(t) = t(1-t)$

$$\nabla\, g(\mathbf{w}^T\mathbf{x}^i) = g(\mathbf{w}^T\mathbf{x}^i)\left(1 - g(\mathbf{w}^T\mathbf{x}^i)\right)\mathbf{x}^i$$

$$\nabla l(g(\mathbf{w}^T\mathbf{x}^i), y^i) = (y^i - g(\mathbf{w}^T\mathbf{x}^i))\mathbf{x}^i$$

Online gradient descent for Logistic Regression

Note: y takes 0/1 here, not 1/-1

$$\text{Given : training examples } (\mathbf{x}^i, y^i),\ i = 1,\ldots,N$$

$$\text{Let } \mathbf{w} \leftarrow (0,0,0,\ldots,0)$$

$$\text{Repeat until convergence}$$

$$\quad \text{For every example } i$$

$$\qquad \hat{y}^i \leftarrow \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^i}}$$

$$\qquad \mathbf{w} \leftarrow \mathbf{w} + \eta \cdot (y^i - \hat{y}^i) \cdot \mathbf{x}^i$$

η is the learning rate or step size

Note the striking similarity between LR and perceptron

- Both learn a linear decision boundary
- The iterative algorithm takes very similar form

Daja vu?

Batch Learning for Logistic Regression

Note: y takes 0/1 here, not 1/-1

$$\textbf{Given : training examples } (\mathbf{x}^i, y^i),\ i = 1,\ldots,N$$

$$\textbf{Let } \mathbf{w} \leftarrow (0,0,0,\ldots,0)$$

Repeat until convergence

$$d \leftarrow (0,0,0,\ldots,0)$$

$$\textbf{For } i = 1 \textbf{ to } N \textbf{ do}$$

$$\hat{y}^i \leftarrow \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}^i}}$$

$$error = y^i - \hat{y}^i$$

$$d = d + error \cdot \mathbf{x}^i$$

$$\mathbf{w} \leftarrow \mathbf{w} + \eta d$$

Logistic regression: Summary

- LR uses the logistic function to warp the output of a linear function to between zero and one, which is interpreted as $P(y = 1|\mathbf{x}; \mathbf{w})$
- Learning aims to learn a vector **w** s.t. examples with $y = 1$ are predicted to have high $P(y = 1|\mathbf{x}; \mathbf{w})$ and vice versa
- Learn **w** iteratively using gradient descent
- Strong similarity with Perceptron
- Logistic regression learns a linear decision boundaries
  - By introducing nonlinear features (i.e., $x_1^2, x_2^2, x_1 x_2, \ldots$), can be extended to nonlinear boundary.