# CS 383/613 – Machine Learning

## Logistic Regression

Slides adapted from material created by E. Alpaydin
Prof. Mordohai, Prof. Greenstadt, Pattern Classification (2nd Ed.),
Pattern Recognition and Machine Learning

# Objectives

- Logistic Regression

# Logistic Regression

- Logistic Regression is a terrible name!
  - It’s not regression at all!
  - It’s classification
- But it is extremely similar to linear regression
  - Hence it’s name!

# Logistic Regression

- With logistic regression we assume **binary classification**, and want to provide a probability for one of the classes (often referred to as the *positive* class, with $y = 1$) :

$$0 \leq P(y = 1|\boldsymbol{x}) \leq 1$$

- For logistic regression, we compute this value $P(y = 1|\boldsymbol{x})$ as:

$$P(y = 1|\boldsymbol{x}) = g(\boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{x}\boldsymbol{w}+b)}}$$

- Where :
  - $\boldsymbol{w}$ is a (column) vector of weights, one per feature of $\boldsymbol{x}$
  - $b$ is a *bias* weight

# Logistic Regression

$$P(y=1|\boldsymbol{x}) = g(\boldsymbol{x}) = \frac{1}{1+e^{-(\boldsymbol{x}\boldsymbol{w}+b)}}$$

- Note the similarity to *linear* regression:

$$g(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{w} + b$$

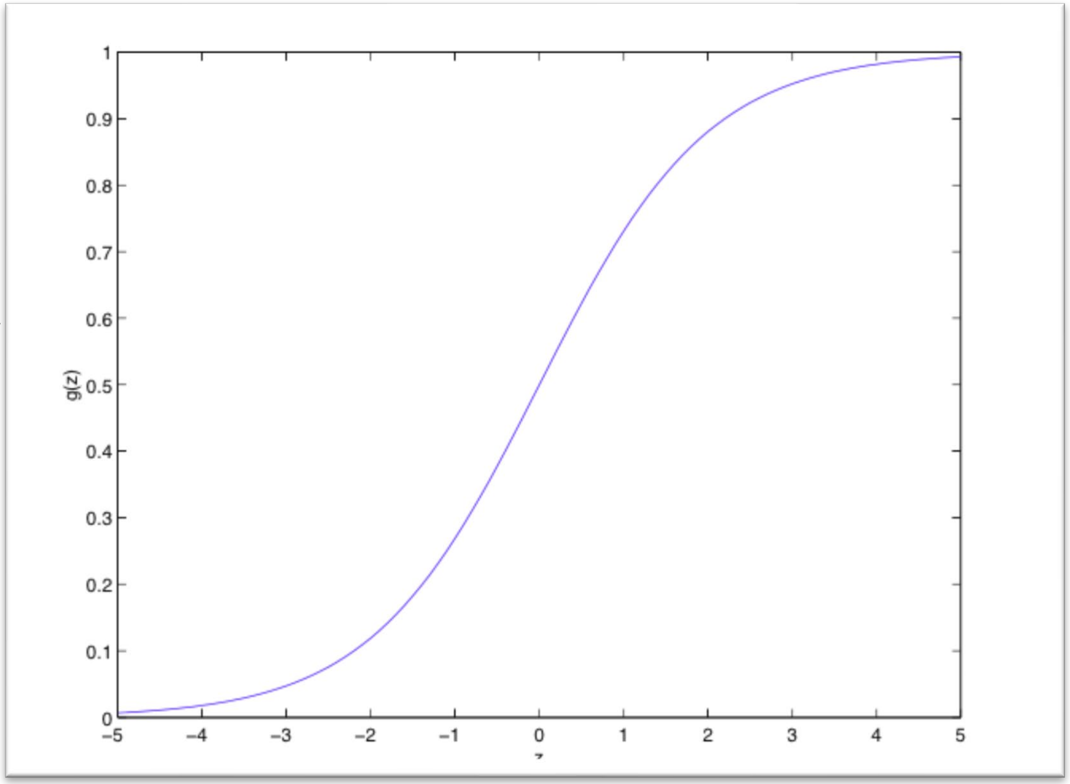
- So, just like with linear regression we can roll the bias weight into the weight vector by adding a bias feature to $\boldsymbol{x}$**!**
- If we do this, we have:

$$g(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{x}\boldsymbol{w}}}$$

# Logistic Regression

$$P(y = 1|\boldsymbol{x}) = g(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{xw}}}$$

- The function $g(z) = \frac{1}{1+e^{-z}}$ is called the *sigmoid* or *logistic* function (or logistic sigmoid)
  - Tends to 0 as $z$ decreases
  - Tends to 1 as $z$ increases
- In addition to resulting in values that can be interpreted as probabilities, it has the nice characteristic in that it's differentiable!



# Logistic Regression

- If we consider
  - $P(y = 1|\boldsymbol{x}) = g(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{xw}}}$
- Then we can compute the probability of being from the other class as:
  - $P(y = 0|\boldsymbol{x}) = 1 - g(\boldsymbol{x})$
- Ultimately, we want to find the weights $\boldsymbol{w}$ to minimize the classification error
  - Or conversely, to find the weights maximize the correct class likelihood

# Fit Weights Based on Maximum Likelihood

- Given a supervised observation $(\boldsymbol{x}, y)$, we'll let $\hat{y}$ be our prediction that $y = 1$
  - Which again, for logistic regression is computed as $\hat{y} = \frac{1}{1+e^{-\boldsymbol{xw}}}$
- We can then compute the **likelihood** that we are correct as
$$J = \ell(y|\boldsymbol{x}) = (\hat{y})^y (1 - \hat{y})^{(1-y)}$$
- So, what do we do with this likelihood $J$?
  - We want to maximize it!
- The process of finding the weights to maximize the likelihood is called….**maximum likelihood estimate (MLE)**

# Log Likelihood

$$J = \ell(y|\boldsymbol{x}) = (\hat{y})^{y}(1-\hat{y})^{(1-y)}$$

- To find the optimal weights we're going to use calculus.
  - Remember the nice property that the logistic sigmoid function is differentiable!
- But instead of taking the derivative of the likelihood (which deals with products and exponents), we can instead look to maximize the **log** of the likelihood!
- From the properties of logarithms
  - $\log_b(mn) = \log_b(m) + \log_b(n)$
  - $\log_b(m^n) = n \cdot \log_b(m)$
- So, applying the log to $J = \hat{y}^{y}(1-\hat{y})^{(1-y)}$ we get:

$$J = y\, ln(\hat{y}) + (1-y)\ln(1-\hat{y})$$

# Log Loss

$$J = y\,ln(\hat{y}) + (1-y)\ln(1-\hat{y})$$

- It is more common to minimize the negation of this, which is referred to as the *log loss:*

$$J = -(y\,ln(\hat{y}) + (1-y)\ln(1-\hat{y}))$$

- So, how do we use calculus to find the weights the minimize this?
- Recall that in general, when attempting to use calculus to find the weights to minimize (or maximize) and objective function, we'll use two approaches:
  - Direct
  - Iterative

# Log Loss

$$J = -(y\, ln(\hat{y}) + (1-y)\ln(1-\hat{y}))$$

- For this particular objective, function is there a direct solution?
  - No ☹
- So, we'll have to use the iterative approach!
- So lets compute the gradients $\frac{\partial J}{\partial w_j}$!

# Gradients for Log Loss

$$J = -(y\, ln(\hat{y}) + (1 - y)\ln(1 - \hat{y}))$$

- Since our objective function includes logs, recall how to take a derivative of a log:

$$\frac{\partial}{\partial x}(\ln x) = \frac{1}{x} \cdot \frac{\partial}{\partial x}(x)$$

- Therefore

$$\frac{\partial J}{\partial w_j} = -\left(\frac{y}{\hat{y}} \frac{\partial}{\partial w_j}(\hat{y}) + \frac{1-y}{1-\hat{y}} \frac{\partial}{\partial w_j}(1-\hat{y})\right) = \frac{1-y}{1-\hat{y}} \frac{\partial}{\partial w_j}(\hat{y}) - \frac{y}{\hat{y}} \frac{\partial}{\partial w_j}(\hat{y})$$

- But what is $\frac{\partial}{\partial w_j}\hat{y}$ when $\hat{y} = g(\boldsymbol{x})$ is the logistic sigmoid function?

# Gradients for Log Loss

$$\hat{y} = g(x) = \frac{1}{1 + e^{-\boldsymbol{xw}}}$$

- $\frac{\partial \hat{y}}{\partial w_j} = \frac{\partial}{\partial w_j} g(\boldsymbol{x}) = \frac{\partial}{\partial w_j}\left(\frac{1}{1+e^{-\boldsymbol{xw}}}\right) = \frac{\partial}{\partial w_j}(1 + e^{-\boldsymbol{xw}})^{-1}$
- $= -1\left(0 - e^{-\boldsymbol{xw}} x_j\right)(1 + e^{-\boldsymbol{xw}})^{-2} = \frac{e^{-\boldsymbol{xw}}}{(1+e^{-\boldsymbol{xw}})^2} x_j$
- $= \frac{1}{1+e^{-\boldsymbol{xw}}} \frac{e^{-\boldsymbol{xw}}}{1+e^{-\boldsymbol{xw}}} x_j$
- $= g(\boldsymbol{x})\left(1 - g(\boldsymbol{x})\right)x_j = \hat{y}(1 - \hat{y})x_j$

# Gradients for Log Loss

$$\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j}\left(-\left(y\, ln\left(\hat{y}\right) + (1-y)\ln\left(1-\hat{y}\right)\right)\right)$$

- From the previous slide we have

$$\frac{\partial \hat{y}}{\partial w_j} = \frac{\partial}{\partial w_j} g(\boldsymbol{x}) = \hat{y}(1-\hat{y})x_j$$

- And from earlier

$$\frac{\partial J}{\partial w_j} = \frac{1-y}{1-\hat{y}}\frac{\partial}{\partial w_j}(\hat{y}) - \frac{y}{\hat{y}}\frac{\partial}{\partial w_j}(\hat{y})$$

- Putting it all together (and simplifying) we get:

$$\frac{\partial J}{\partial w_j} = (\hat{y} - y)x_j$$

# Leveraging Linear Algebra

$$\boxed{\frac{\partial J}{\partial w_j} = (\hat{y} - y)x_j}$$

- We can ones again leverage linear algebra to compute a vector of the gradients and do batching! ☺
- Vector of the gradients for a single observation:

$$\frac{\partial J}{\partial \boldsymbol{w}} = \boldsymbol{x}^T(\hat{y} - y)$$

- Batching:

$$\frac{\partial J}{\partial \boldsymbol{w}} = \frac{1}{N} X^T(\hat{Y} - Y)$$

# Numeric Stability

- In ML, there's (at least) two places where *numeric stability* can come into account:
  1. Divide by zero
  2. Log(0)
- If at any point one of these are possible, it's common to add in a *numeric stability constant* to the area where this might happen (denominator or log).
- This is typically a small number like $\epsilon = 10^{-7}$
- With logistic regression, since $0 \leq \hat{y} \leq 1$, we see this possibility when we compute

$$J = -(y\, \boldsymbol{ln(\hat{y})} + (1-y)\, \boldsymbol{ln(1-\hat{y})})$$

- So, let's compute this more safely as:

$$J = -(y\, ln(\hat{y}+\epsilon) + (1-y)\ln(1-\hat{y}+\epsilon))$$

# Logistic Regression Example

- Let's classifying whether a person will buy a product or not

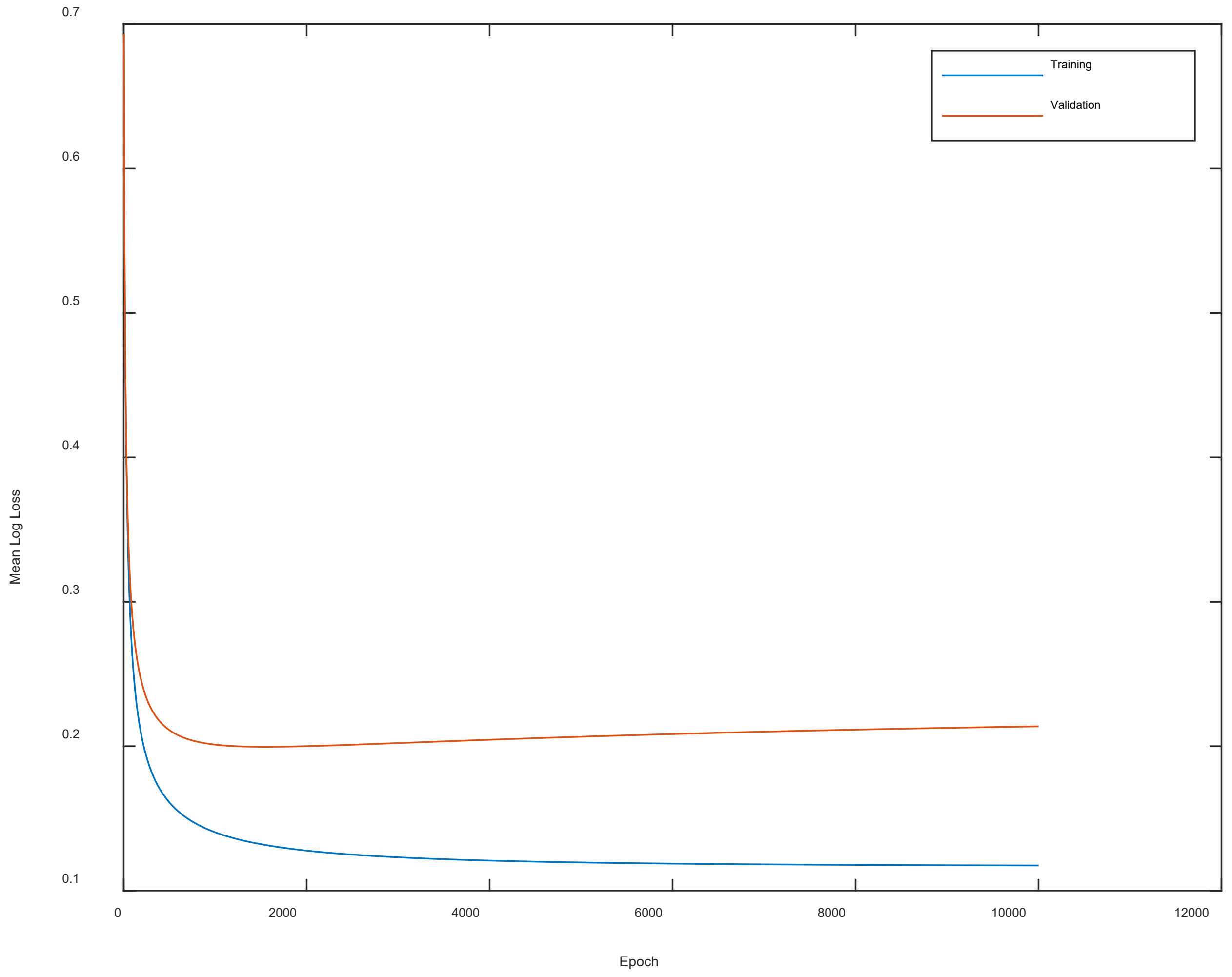
| | Y | X-Variables | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Obs. No. | Buy | Income | Is Female | Is Married | Has College | Is Pro-fessional | (Omit-ted Vari-ables) | Prev Child Mag | Prev Parent Mag |
| 1 | 0 | 24000 | 1 | 0 | 1 | 1 | ... | 0 | 0 |
| 2 | 1 | 75000 | 1 | 1 | 1 | 1 | ... | 1 | 0 |
| 3 | 0 | 46000 | 1 | 1 | 0 | 0 | ... | 0 | 0 |
| 4 | 1 | 70000 | 0 | 1 | 0 | 1 | ... | 1 | 0 |
| 5 | 0 | 43000 | 1 | 0 | 0 | 0 | ... | 0 | 1 |
| 6 | 0 | 24000 | 1 | 1 | 0 | 0 | ... | 0 | 0 |
| 7 | 0 | 26000 | 1 | 1 | 1 | 0 | ... | 0 | 0 |
| 8 | 0 | 38000 | 1 | 1 | 0 | 0 | ... | 0 | 0 |
| 9 | 0 | 39000 | 1 | 0 | 1 | 1 | ... | 0 | 0 |
| 10 | 0 | 49000 | 0 | 1 | 0 | 0 | ... | 0 | 0 |
| . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . |
| 654 | 0 | 10000 | 1 | 0 | 0 | 0 | ... | 0 | 0 |
| 655 | 1 | 75000 | 0 | 1 | 0 | 1 | ... | 0 | 0 |
| 656 | 0 | 72000 | 0 | 0 | 1 | 0 | ... | 0 | 0 |
| 657 | 0 | 33000 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| 658 | 0 | 58000 | 0 | 1 | 1 | 1 | ... | 0 | 0 |
| 659 | 1 | 49000 | 1 | 1 | 0 | 0 | ... | 0 | 0 |
| 660 | 0 | 27000 | 1 | 1 | 0 | 0 | ... | 0 | 0 |
| 661 | 0 | 4000 | 1 | 0 | 0 | 0 | ... | 0 | 0 |
| 662 | 0 | 40000 | 1 | 0 | 1 | 1 | ... | 0 | 0 |
| 663 | 0 | 75000 | 1 | 1 | 1 | 0 | ... | 0 | 0 |
| 664 | 0 | 27000 | 1 | 0 | 0 | 0 | ... | 0 | 0 |
| 665 | 0 | 22000 | 0 | 0 | 0 | 1 | ... | 0 | 0 |
| 666 | 0 | 8000 | 1 | 1 | 0 | 0 | ... | 0 | 0 |
| 667 | 1 | 75000 | 1 | 1 | 1 | 0 | ... | 0 | 0 |
| 668 | 0 | 21000 | 0 | 1 | 0 | 0 | ... | 0 | 0 |
| 669 | 0 | 27000 | 1 | 0 | 0 | 0 | ... | 0 | 0 |
| 670 | 0 | 3000 | 1 | 0 | 0 | 0 | ... | 0 | 0 |
| 671 | 1 | 75000 | 1 | 1 | 0 | 1 | ... | 0 | 0 |
| 672 | 1 | 51000 | 1 | 1 | 0 | 1 | ... | 0 | 0 |
| 673 | 0 | 11000 | 0 | 1 | 0 | 0 | ... | 0 | 0 |

`KidCreative.csv`

# Logistic Regression Example

- Make some design decisions:
  - Randomize data
  - Use 2/3 training, 1/3 validation
  - **Z-score features using training data**
  - Add a bias feature.
  - Initialize weights to random values in the range $\pm 10^{-4}$
  - Learning rate of $\eta = 0.1$
  - Since our equation is based on log loss, let's terminate when change in mean of log loss doesn't change more than $2^{-32}$ or 10,000 epochs have run
    - Recall the log loss of an example being correct is
$$-\left(y\ln(\hat{y}+\epsilon)+(1-y)\ln(1-\hat{y}+\epsilon)\right)$$
  - Let's do full batch learning

# Example




w =
4.7168
0.7976
0.5490
0.0849
-0.3296
-0.4418
-0.0377
0.1836
-0.0246
0.6571
0.3369
0.0472
0.8088
1.0618
0.1782
0.1871
-6.4500

# Example

- Choosing Class 1 if $P(y = 1|\boldsymbol{x}) \geq 0.5$ we get:

Class Priors
$P(y = 0) = 0.831$
$P(y = 1) = 0.169$

| Metric | Training Set | Validation Set |
|---|---|---|
| Precision | 0.865 | 0.776 |
| Recall | 0.842 | 0.776 |
| F-Measure | 0.853 | 0.776 |
| Accuracy | 0.951 | 0.902 |

# Dealing with Overfitting

- Since logistic regression is similar to the gradient based approach of linear regression, it’s techniques for dealing with overfitting are similar.
- General approaches:
  - Try to get more data for training to generalize better
  - Or get more data for training via cross-validation.
  - Don’t use all the features.
- Algorithm-specific approaches:
  - Use validation set to determine number of training epochs (perhaps stopping early).
  - Do *stochastic* gradient learning where each epoch uses a randomly selected subset (batch) from the training data.
  - Add some noise to our training, changing it each time.
  - Or we could add a regularization term to our objective function to penalize complexity.