outrageously
AMBITIOUS

Module 4: Linear Models

Duke
PRATT SCHOOL of ENGINEERING

Types of Algorithms

Parametric algorithms

- Assumes a known form to model the input – output relationship
- Learns a fixed, pre-determined set of parameters/coefficients
- Can learn quickly and work well even on small data
- Constrained to the specified form, prone to underfitting

Types of Algorithms

Non-Parametric algorithms

- Does not make strong assumption about the form of the input-output relationship
- Highly flexible to model non-linear, complex data
- Can result in higher performance in prediction
- Require more data to train and are prone to overfitting

Supervised Learning Algorithms

	Simple	Complex
Parametric	Linear regression Ridge regression Lasso regression Logistic regression Naïve Bayes	Linear SVM Perceptron Neural networks*
Non-Parametric	Decision trees K Nearest Neighbors	AdaBoost Random forest

Regression

Classification

Both

Module 4 Objectives:

At the conclusion of this module, you should be able to:

1) Explain how linear regression works
2) Describe the differences between linear and logistic regression
3) Discuss the benefits and types of regularization

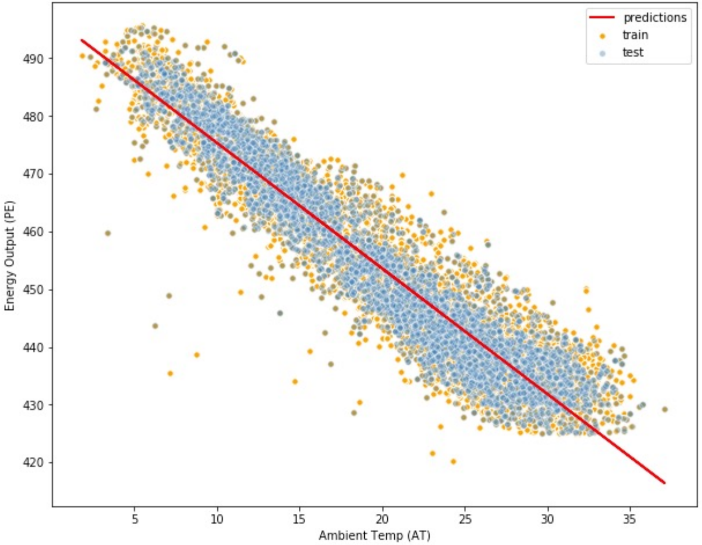
outraggeously AMBITIOUS

Linear Regression

Duke
PRATT SCHOOL of ENGINEERING

What is linear regression

Model which assumes linear relationships between features and targets, defined by a set of coefficients



Why linear regression?

- Forms the basis of more complex ML models
- Can be surprisingly effective if used properly
- Great first model to apply to get a benchmark
- Helps us understand relationships between inputs and outputs (feature and targets)

Simple vs. multiple linear regression

Simple linear regression

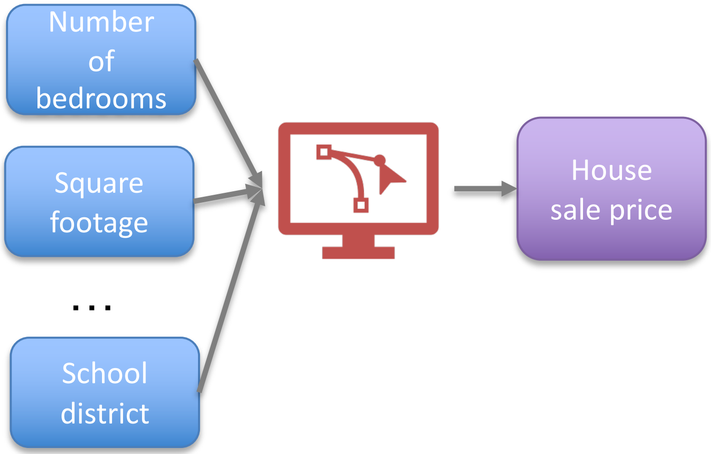
$$y = w_0 + w_1 x$$

Bias

Coefficient / weight

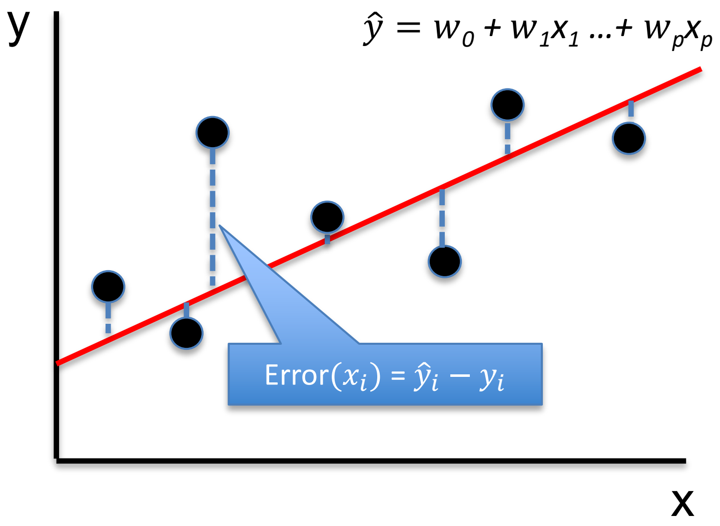
Simple vs. multiple linear regression

Multiple linear regression



$$y = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_p x_p$$

Estimating the parameters



We seek a model function that minimizes total error $\sum_{i=1}^{N}(\hat{y}_i - y_i)$, or alternatively the **Sum of Squared Error (SSE)** $\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$

Estimating the parameters

- SSE is our **cost function** (loss function)

$$Cost\ function\ J(w) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

- In modeling, we seek to find values for parameters w_o, $w_1 \ldots w_i$ which **minimize** our cost function
- We can use the training data to solve for the parameters that minimize the cost

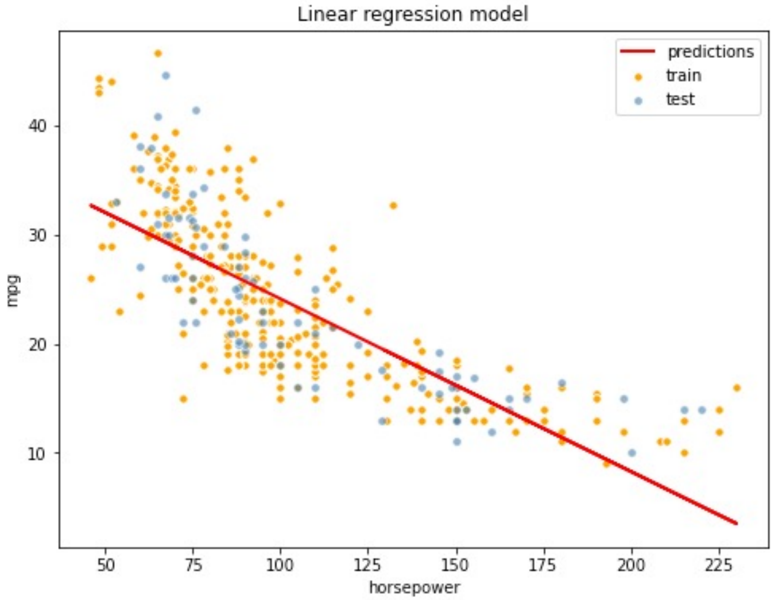
Non-linear relationships

- To model a nonlinear relationship, we can transform the feature by some nonlinear function to create a new feature:

$$z = x^a \text{ OR } z = log(x)$$

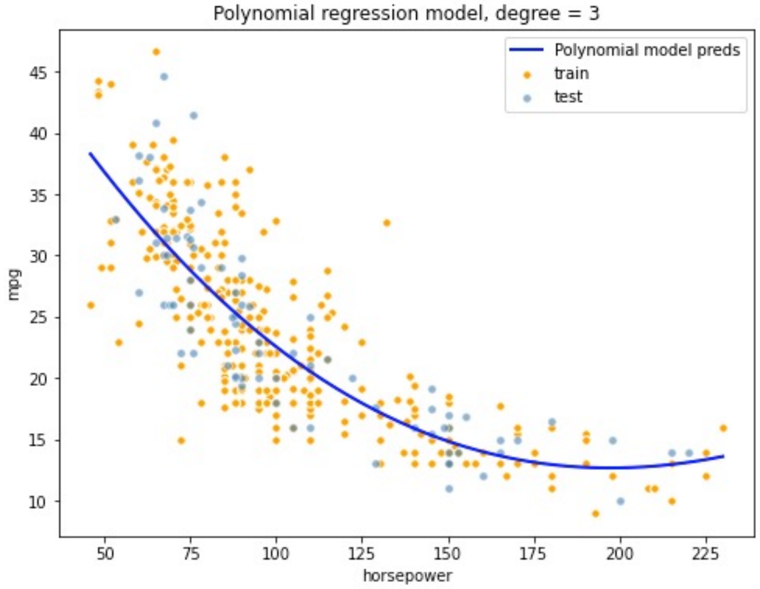
- We can then use our new feature *z* as an input to our model
- If we can model our target as a linear function of our new feature *z*, our model performance should be improved
- This is called **polynomial regression**

Example: Predicting fuel efficiency




```
MSE train: 24.430, test: 22.026
```

Example: Predicting fuel efficiency



MSE train: 19.728, test: 15.911

Regularization

Motivation for Regularization

- The training method we have been using tends to reward complexity / overfitting
- However, complex models have higher variance and thus may not predict as well on new data
- How can we build a regression model in a more balanced way?
  - We add a penalty factor to our cost function to penalize feature complexity

Regularization

Linear regression cost function:

$$J(w) = SSE = \sum_{i=1}^{N}(y_i - (w_0 + w_1 x_{i,1} + \cdots + w_p x_{i,p}))^2$$

- We add a penalty term that is a function of the sum of the coefficients
- Now, higher number or values of coefficients increases the cost function
- Minimizing this new cost function seeks optimal balance of fit and simplicity

Cost function with regularization:

$$J(w) = \sum_{i=1}^{N}(y_i - (w_0 + w_1 x_{i,1} + \cdots + w_p x_{i,p}))^2 + \lambda * Penalty(w_1 \ldots w_p)$$

λ controls strength of the penalty

LASSO & Ridge Regression

LASSO Regression

$$J(w) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |w_j|$$

- Forces coefficients to 0 if not relevant
  - Performs **feature selection** by removing unimportant features

LASSO & Ridge Regression

Ridge Regression

$$J(w) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} {w_j}^2$$

- Forces coefficients of irrelevant variables to be small but not to 0
  - Does not perform feature selection

Conclusion: Regularization

- Applying regularization will often give us a better model when we are dealing with complex data
- You might have a reason to prefer one method or the other:
  - Desire a simpler, more interpretable model -> LASSO
  - Complex relationship of target to many features with collinearity -> Ridge

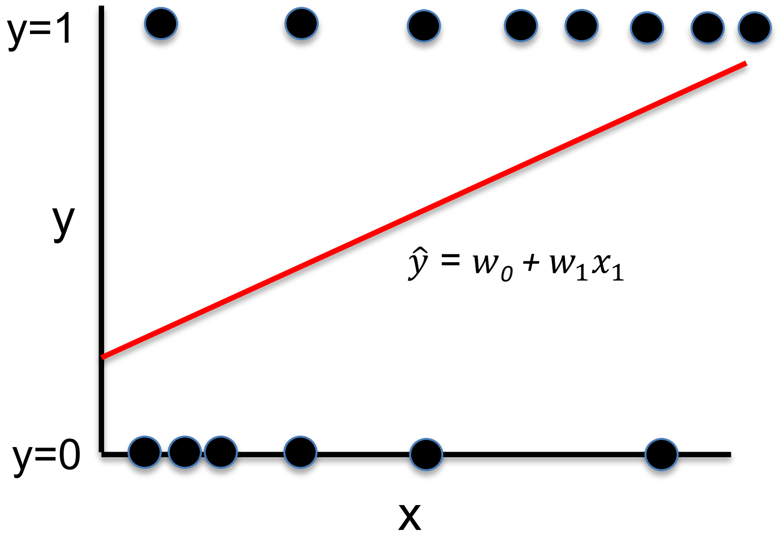
outrageously AMBITIOUS

Logistic Regression

Duke
PRATT SCHOOL *of* ENGINEERING

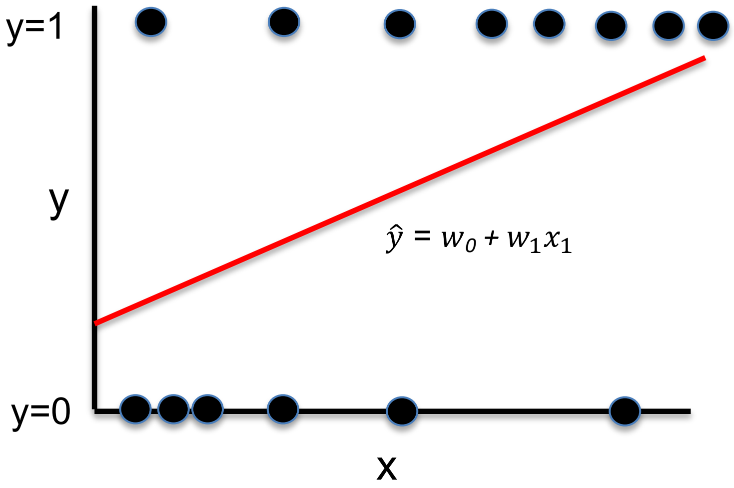
Let’s Tackle a Classification Problem

- We now want to predict a class (e.g. 0 or 1) rather than a numerical target
- We could use linear regression to do so



Problems

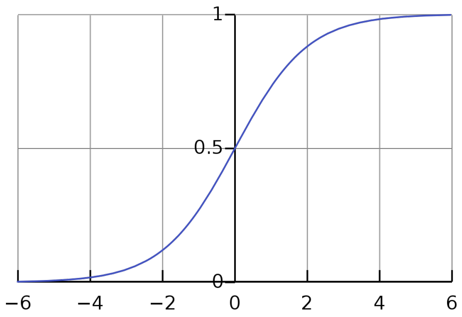
- The linear regression will almost always predict the wrong value
- How do we interpret predictions between 0 and 1?
- What about predictions greater than 1?



Solution: Predict the Probability y=1

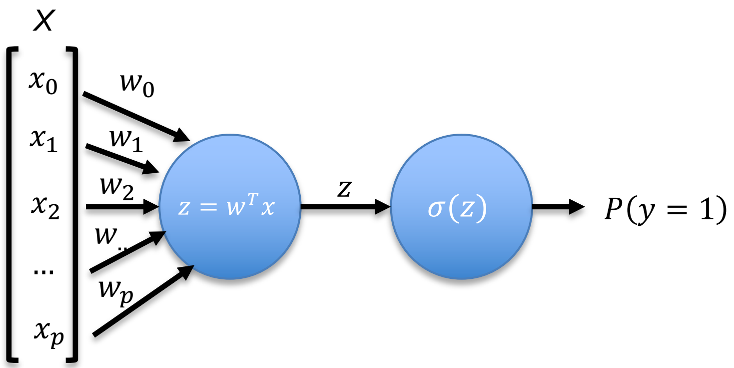
- Rather than predicting y, let's predict the probability P(y=1),
- To do so we need a function that predicts outputs between 0 and 1
- We use the **logistic/sigmoid** function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Solution: Predict the Probability y=1

- Desired model output is P(y=1)
- We use the sigmoid function to get outputs between 0 and 1
- As input to the sigmoid we provide the output of our linear regression (w_0+w_1x)



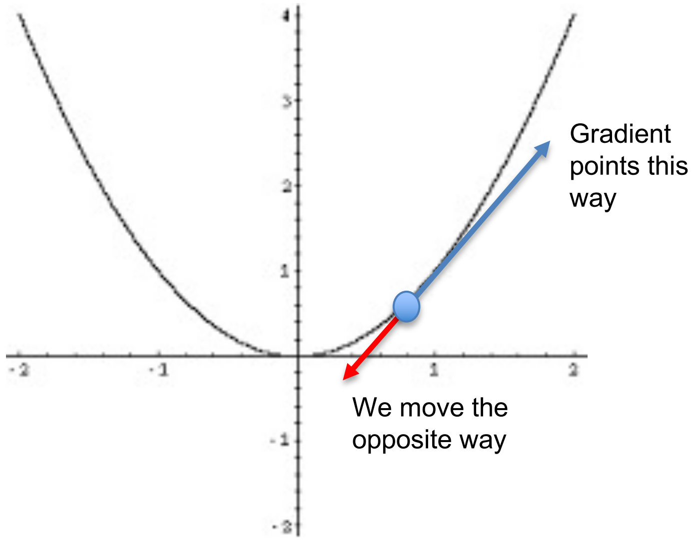
Estimating the parameters

To find the optimal values of $w_1 \ldots w_p$:

1. Define our cost function $J(w)$
2. Find the weight/coefficient values that minimize the cost function
    1. Calculate the derivative (gradient)
    2. Set the gradient equal to 0
    3. Solve for the coefficients using **gradient descent**

Gradient descent

- Suppose we want to minimize a function such as $y = x^2$
- We start at some point on the curve and move iteratively towards the minimum
  - Move in the direction opposite the gradient
  - Move by some small value (called the **learning rate** or η) multiplied by the gradient
- We continue until we find the minimum or reach a set number of iterations



Estimating the parameters

1. Define our cost function $J(w)$
2. Use gradient descent to find the values of the weights that minimize the cost
   - Calculate gradient of the cost function
   - Iteratively update the weights using gradient descent:

$$w_{t+1} = w_t - \eta * \nabla J(w_t)$$

   - Repeat until we reach a minimum cost

outrageously AMBITIOUS

Softmax Regression

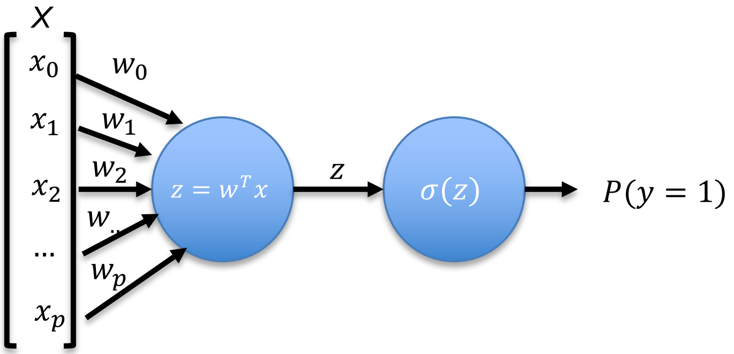
Duke PRATT SCHOOL of ENGINEERING

Predicting multiple classes

- Logistic regression function gave us the probability of the positive class
- But what if we have several classes?
- Instead of the sigmoid function, we use the **softmax function** to give us the probability of belonging to each class (normalized to sum to 1)

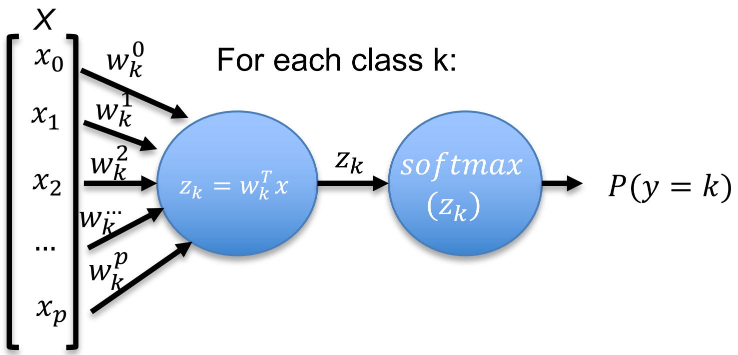
Binary (2 classes)

$$P(y_i = 1) = \sigma(w^T x)$$

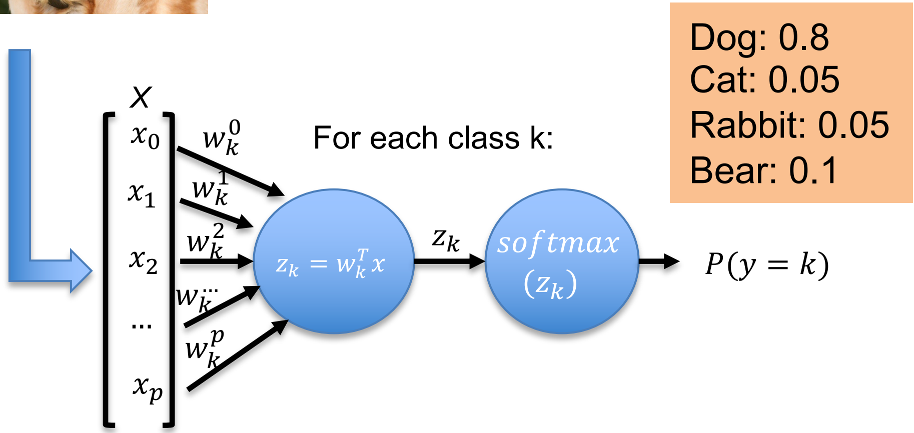


Multiclass

$$P(y_i = k) = softmax(w_k^T x)$$



Predicting multiple classes



outrageously
AMBITIOUS

Wrap-Up

Duke
PRATT SCHOOL *of* ENGINEERING

Wrap-Up: Linear Models

- The mathematical intuition behind linear models is the foundation of neural networks
- Linear models are a good starting point for modeling efforts
- Their capability is limited somewhat by their linear form