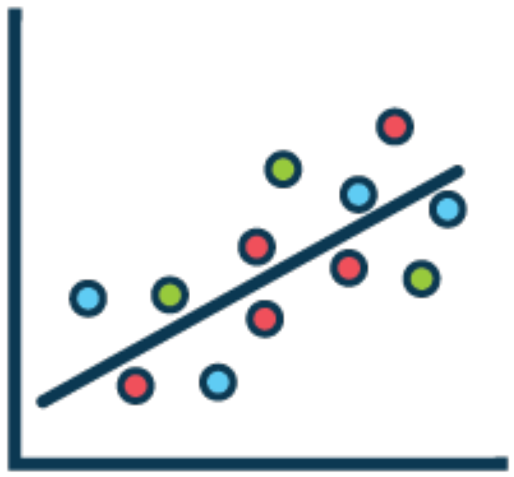


Lecture 7

Regression

Predicting Market Values of Houses

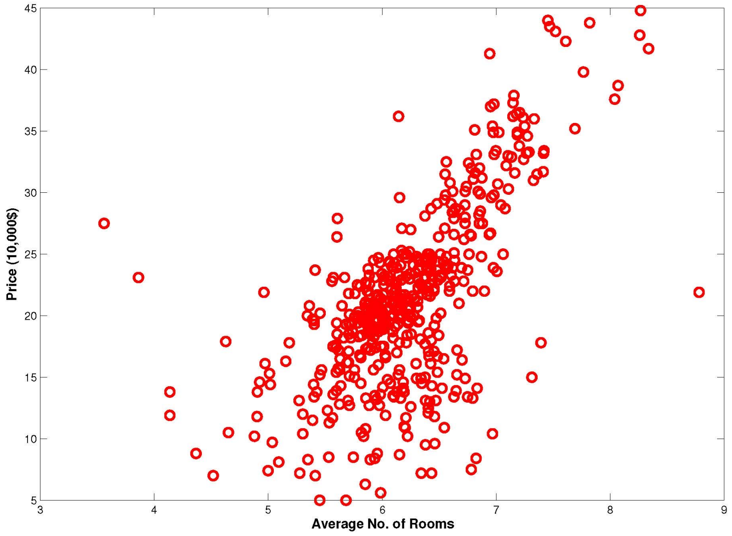
- Task
  - Predict the value of a house
  - Based on the number of rooms

- Data
  - Input: Number of rooms
  - Output/Target: Current value of the house
  - N examples (data points/observations)
  - Learn a model which "accurately" predicts the value of a house given number of rooms

	Number of Rooms	Market Value (10K dollars)
1.	3	40
2.	3	33
3.	3	36.9
4.	5	23.2
5.	4	45
. . .	. . .	. . .
N.	4	43.7

Predicting Market Values of Houses

- Another look at the data



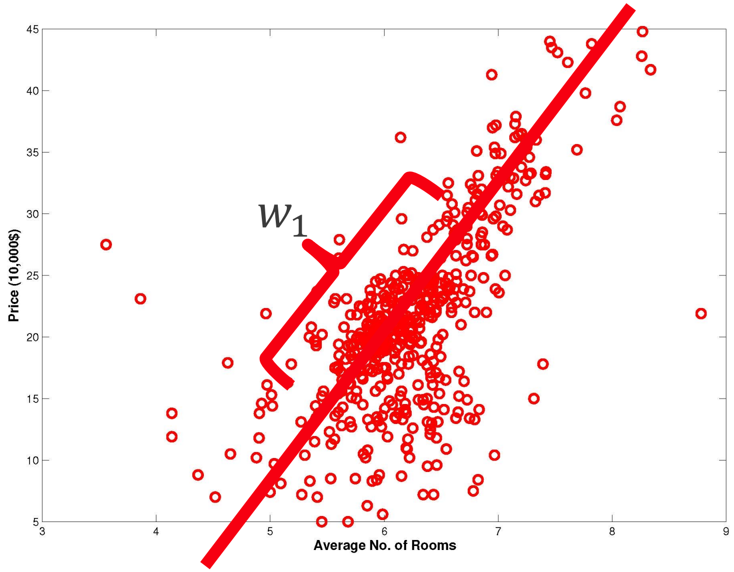
- X-Axis
  - Input Feature Values
- Y-Axis
  - Output / Target
- A real estate expert tells us that the model is a straight line

Predicting Market Values of Houses

- The model:

$$y = w_0 + w_1 x$$

- Notation:
  - x represents the number of rooms
  - y represents the market value of the house
  - w_0 is the intercept of the straight line
  - w_1 is the slope of the line
- Learning Task:
  - Find values of w_0 and w_1



Predicting Market Values of Houses

- For our dataset that has N observations we have N equations:

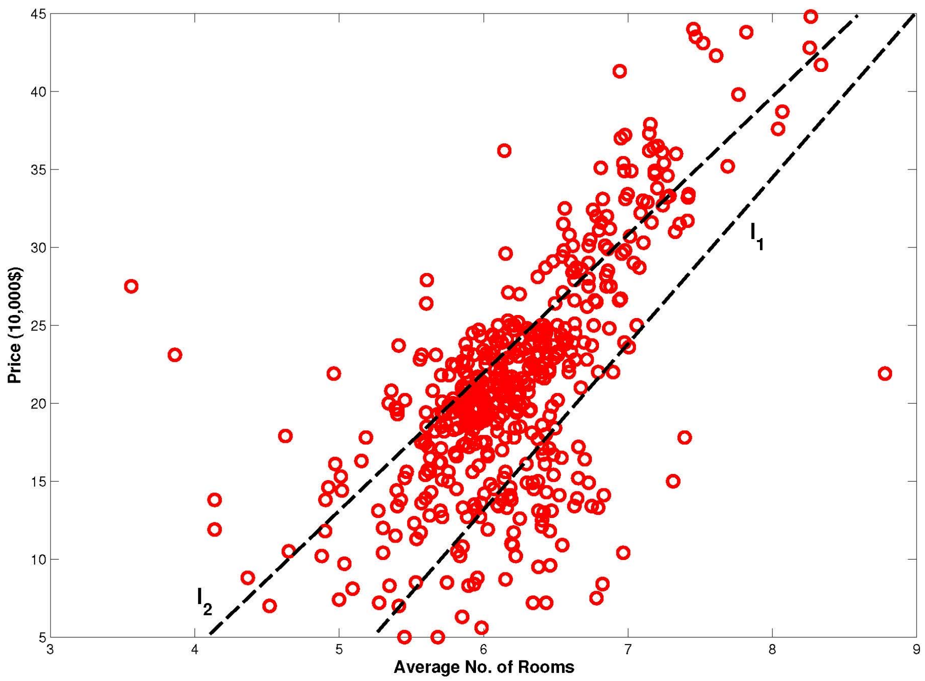
$$y_1 = w_0 + w_1 x_1$$
$$y_2 = w_0 + w_1 x_2$$

$$y_N = w_0 + w_1 x_N$$

- This is an overdetermined system
  - No solution satisfies all N equations
- Approximate Solution
  - Find values of parameters that define the best fit to data

Predicting Market Values of Houses

- How do we characterize the “goodness” of fit?
- If we are given a choice between l_1 and l_2, which one should we choose?
- Both lines have an associated error on the entire dataset



Loss Function

- For the i -th observation a given line predicts a value

$$\hat{y}_i = w_0 + w_1 x_i$$

- The error is then defined as:

$$error = (\hat{y}_i - y_i)^2 = (w_0 + w_1 x_i - y_i)^2$$

- For the dataset we can define the sum of squared errors (SSE):

$$E(w) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{N} (w_0 + w_1 x_i - y_i)^2$$

The Optimal Solution

- We can find the line that best fits the data by minimizing the SSE on the dataset:

$$\arg\min_{w_0, w_1} \sum_{i=1}^{N} (w_0 + w_1 x_i - y_i)^2$$

- How to minimize this function?
    - Calculate the derivative with respect to w
    - Set the derivative to zero
    - Solve for w

Linear Regression Assumptions (under OLS)

- Simple linear regression fits any numeric feature pair, but might not be ideal. It assumes x and y have specific relationships.

- A simple linear regression model $\hat{y} = b_0 + b_1 x$ assumes:
  - **Linearity**: The relationship between the independent and dependent variables is linear.
  - **Independence**: Residuals (errors) are independent of each other.
  - **Homoscedasticity**: Residuals have constant variance.
  - **Normality**: Residuals are approximately normally distributed.
  - **No Multicollinearity**: Independent variables are not highly correlated with each other.

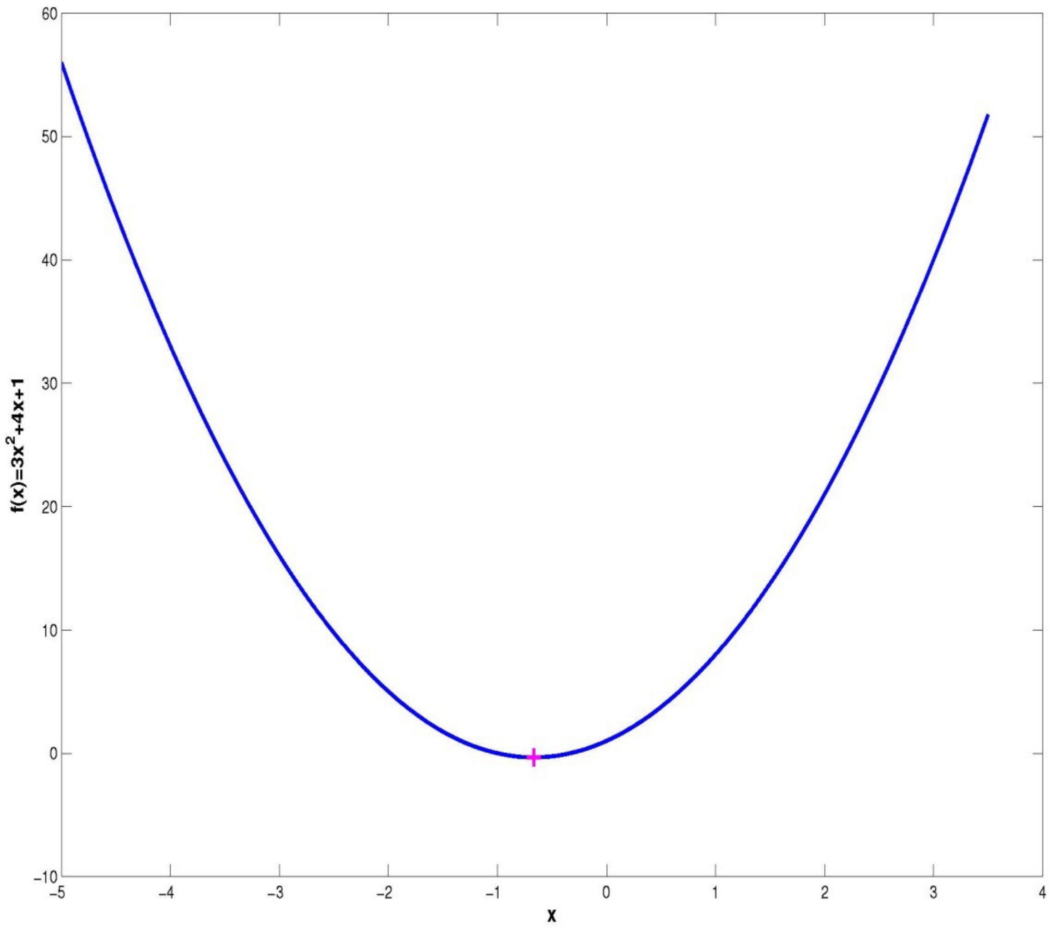
- Before simple linear regression, assess x and y's linearity via scatter plot.

Background

Numerical Optimization

Minimizing a function

- Consider the function
  - $f(x) = 3x^2 + 4x + 1$
  - Find x^*: $\forall x \; f(x^*) \leq f(x)$
- Approach-1: Analytical Solution



Optimizing analytically

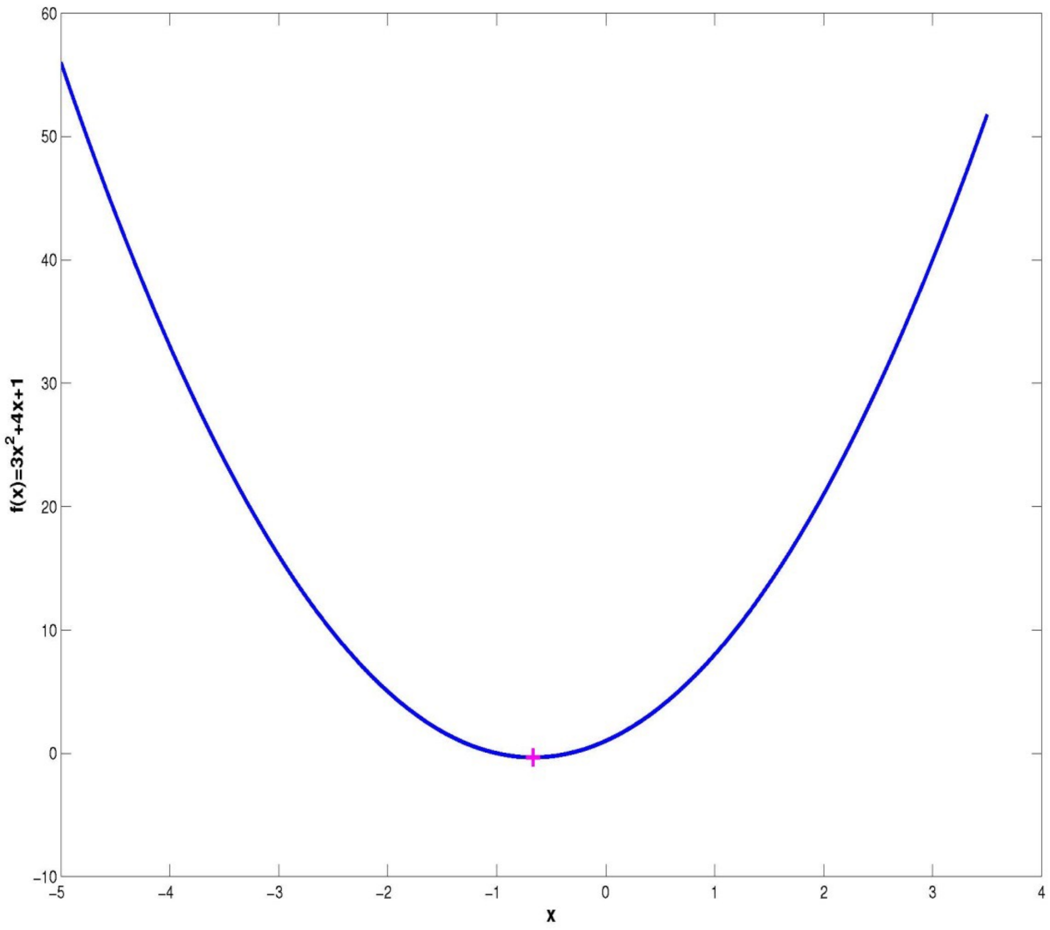
$$f(x) = 3x^2 + 4x + 1$$

$$\frac{df}{dx} = 6x + 4$$

$$6x + 4 = 0$$

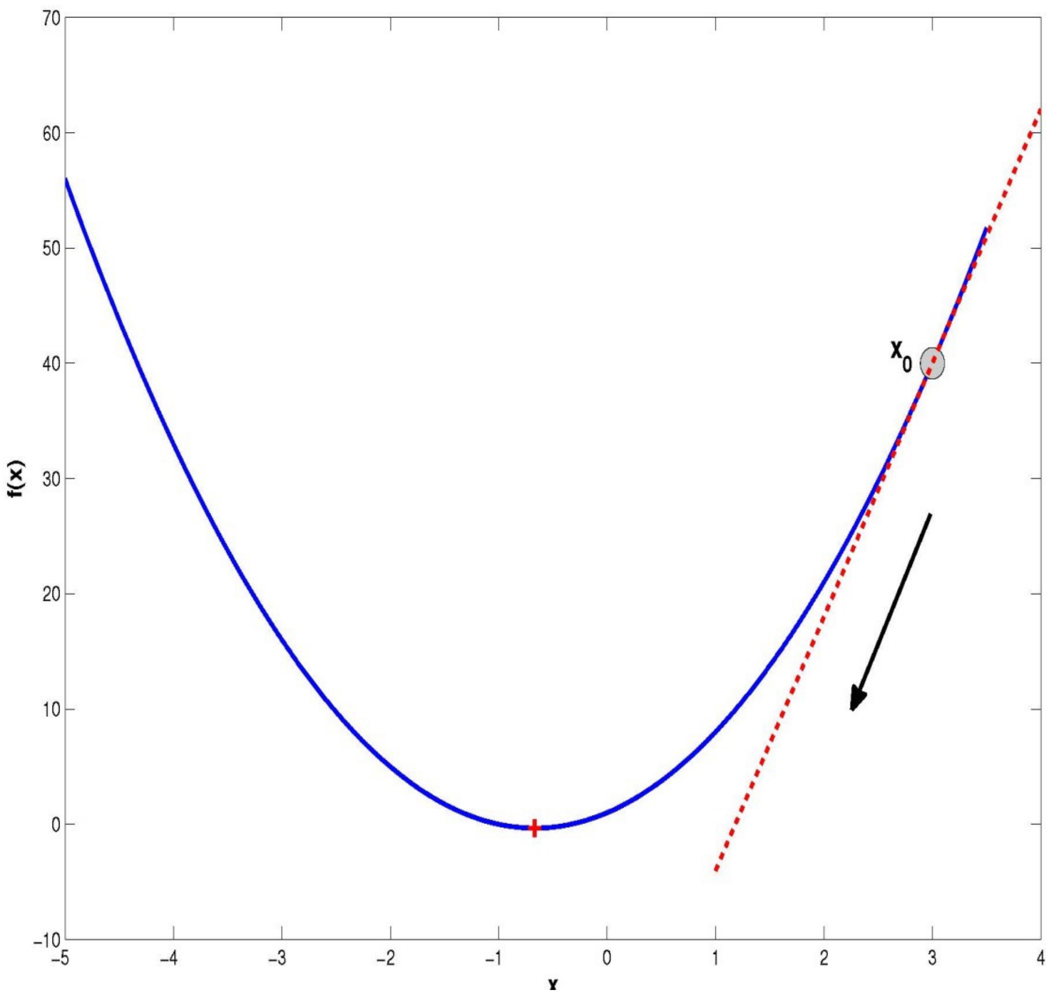
$$6x = -4$$

$$x = \frac{-2}{3}$$



Minimizing a function

- Approach-2: Iterative Solution
  - Begin at x_0
  - Iteratively reach x^*
  - Use the derivative at x_0
  - Defines the slope of the tangent line
  - Moving in the direction of negative slope we can reach the minimum
  - Method of steepest descent



Gradient Vector

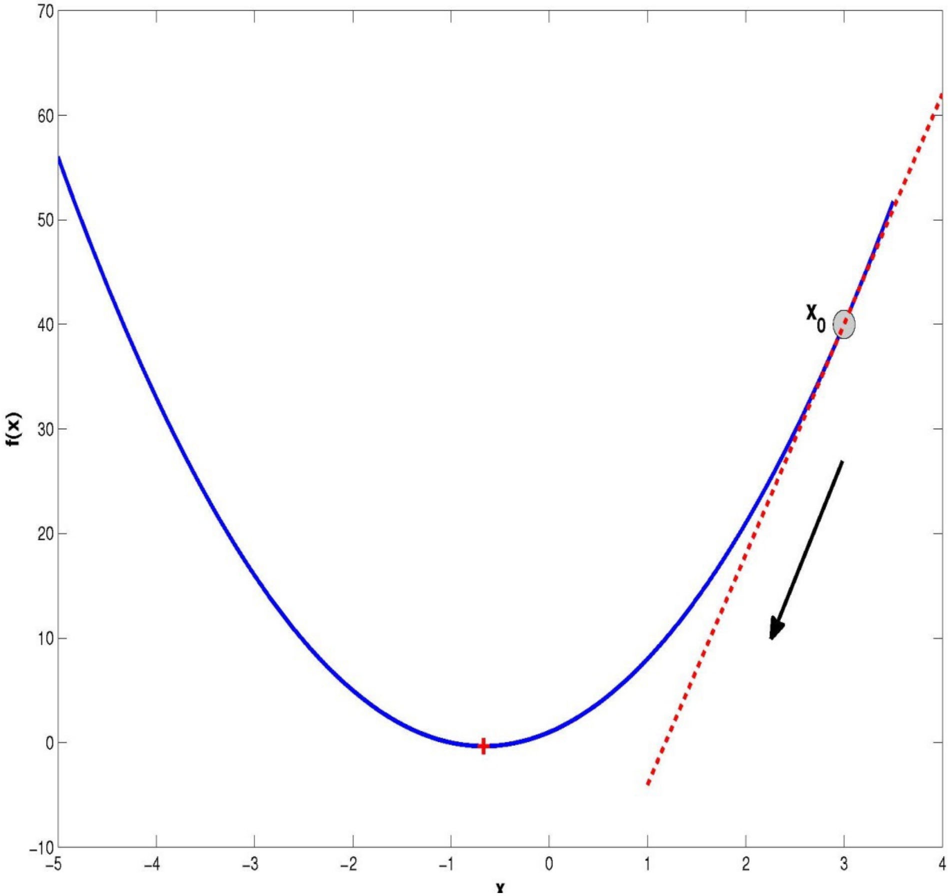
- How about a function of more than one variable?
  - Example: $f(x_1, x_2) = x_1 + 5x_2 - 19$

- Our regression model also had two parameters w_0 and w_1

- For functions of m variables we can use the gradient vector which specifies the direction of maximum increase

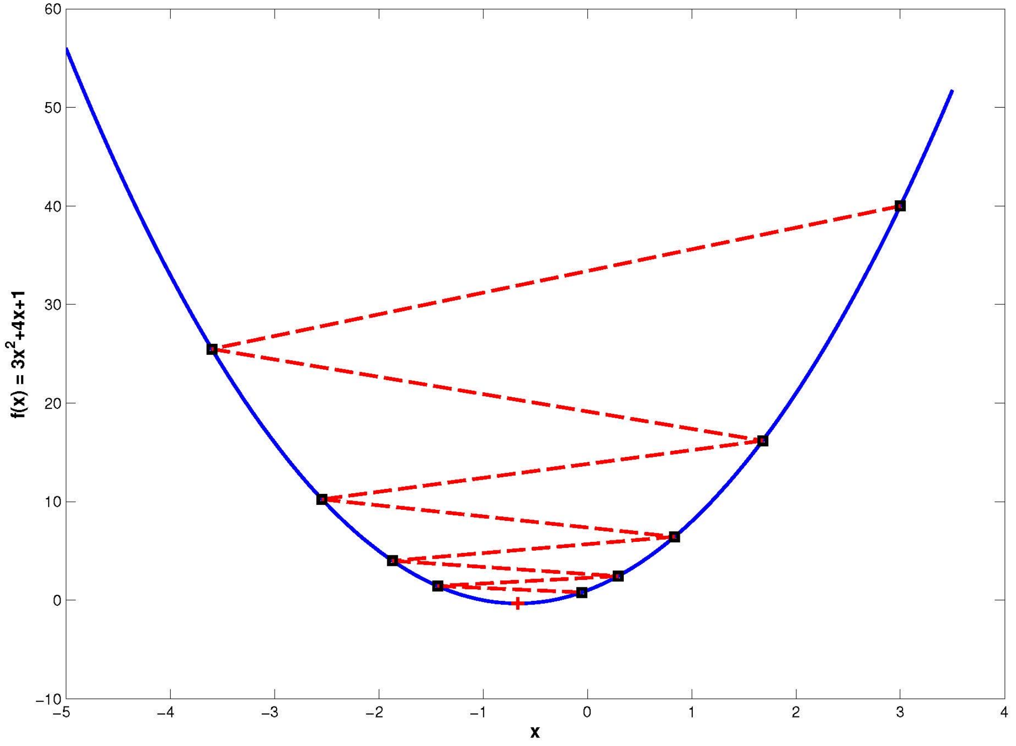
- $\nabla_{\mathrm{x}} f = \left[\frac{\delta f}{\delta x_1} \frac{\delta f}{\delta x_2} \dots \dots \frac{\delta f}{\delta x_m}\right]^T$

Gradient Descent Method



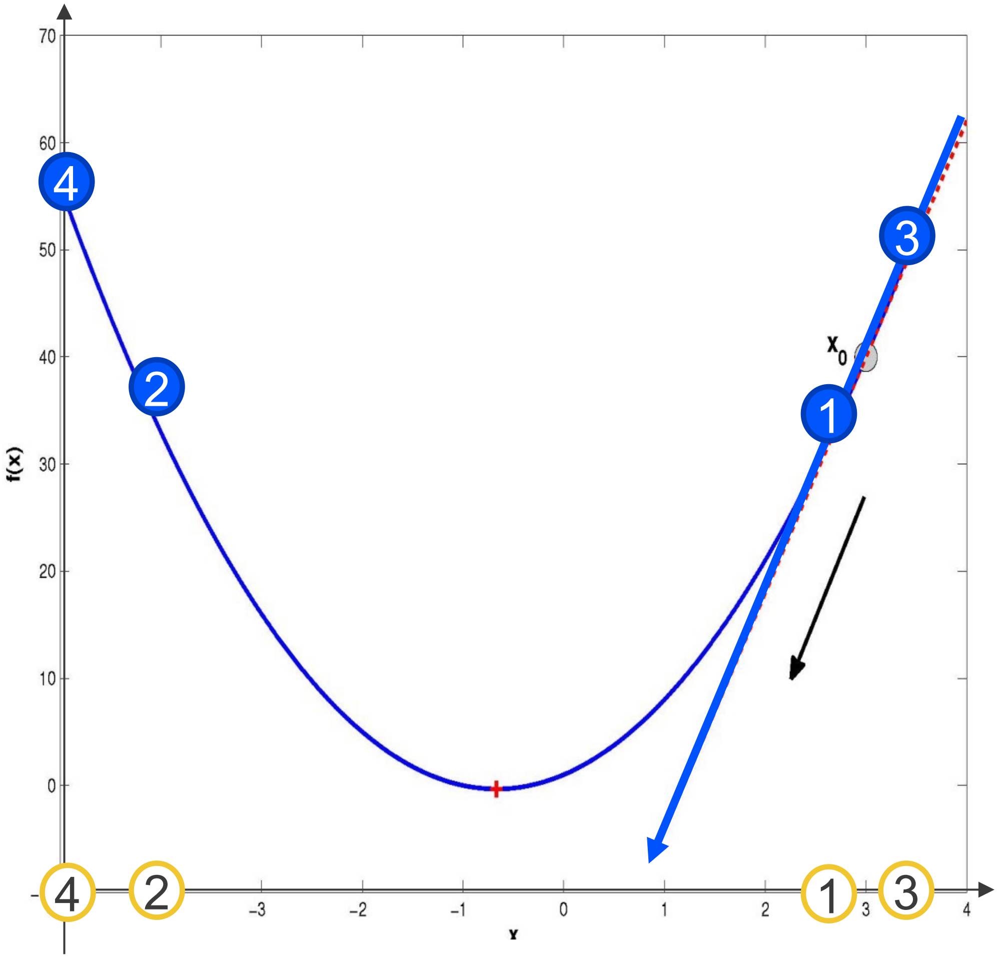
Data: Starting point: x_0, Learning Rate: η
Result: Minimizer: x^*
while *convergence criterion not satisfied* **do**
  estimate the gradient at x_i: $\nabla f(x_i)$;
  calculate the next point: $x_{i+1} = x_i - \eta \nabla f(x_i)$
end

Gradient Descent Method

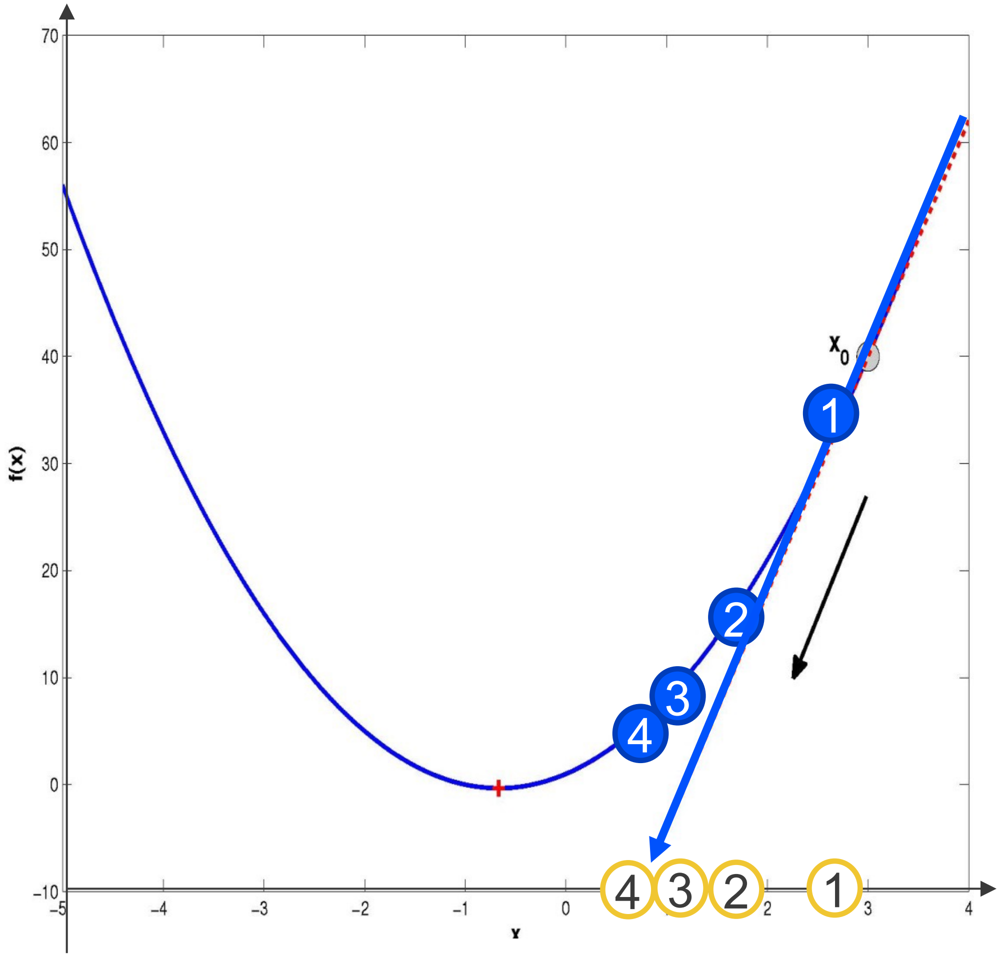


- Gradient vector defines the direction
- How much should we move?
  - Learning rate defines this distance
- Why learning rate matters?
  - Convergence depends on the learning rate:
  - Too small learning rate: long time to converge
  - Large learning rate: oscillate between intermediate values

Learning rate is too large: Overshoots



Learning rate is too small: Stalls



Regression

An Iterative Solution

Predicting Market Values of Houses

- Recall:

$$y = w_0 + w_1 x$$

- Notation:
  - x represents the number of rooms
  - y represents the market value of the house
  - w_0 is the intercept of the straight line
  - w_1 is the slope of the line

- Learning Task:
  - Find values of w_0 and w_1 by minimizing

$$E(w_0, w_1) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{N} (w_0 + w_1 x_i - y_i)^2$$

Predicting Market Values of Houses

- Analytical solution for one variable

$$w_0 = \bar{y} - w_1\bar{x}$$

$$w_1 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

$$w_1 = \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(x)}$$

Multiple Features

- Let's say we get some more information about the houses such as:
  - Crime rate in the locality
  - Level of pollution, etc.
  - How can we generalize our model to include an arbitrary number of features?

- Expand the model:

$$y_i = w_0 + \sum_{k=1}^{m} w_k x_{ik}$$

- Estimate all the m+1 coefficient simultaneously from data

Setup

- Regressor:
  - Create a constant feature which is always 1

$$y_i = w_0(1) + \sum_{k=1}^{m} w_k x_{ik} = \sum_{k=0}^{m} w_k x_{ik} = \mathrm{w}^T \mathrm{x}_i$$

- Matrix Notation:

$$\mathrm{X} = \begin{bmatrix} 1 & x_{11} \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ 1 & x_{N1} \cdots & x_{Nm} \end{bmatrix} \quad , \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad , \quad w = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

Gradient Descent For Linear Regression

- Re-write the optimization problem:

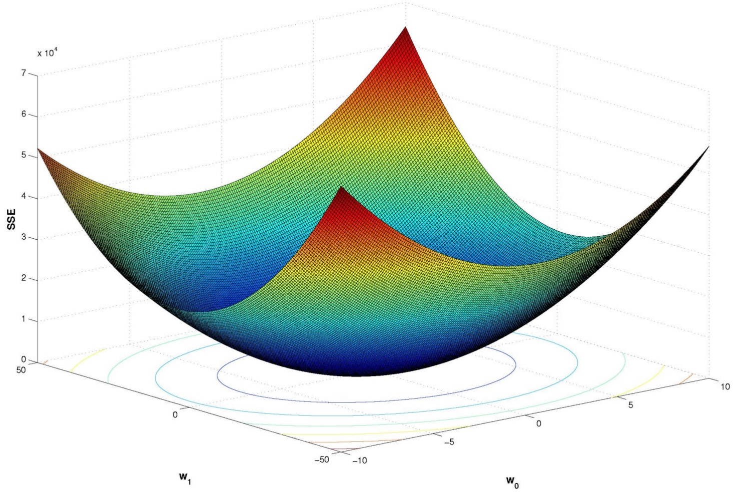
$$w^* = \arg\min_{w} J(w) = \frac{1}{2\mathrm{N}} \sum_{i=1}^{N} (w^T \mathrm{x}_i - y_i)^2$$

- Calculate gradient $\nabla_w J$:

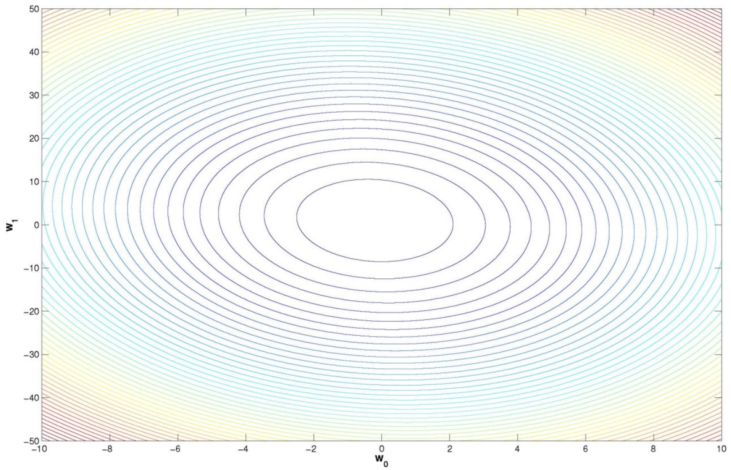
$$w_0 : \frac{\partial}{\partial w_0} J(w) = \frac{1}{N} \sum_{i=1}^{N} (w^T \mathrm{x}_i - y_i)$$

$$w_1 : \frac{\partial}{\partial w_1} J(w) = \frac{1}{N} \sum_{i=1}^{N} (w^T \mathrm{x}_i - y_i) . x_i$$

Predicting Market Values of Houses

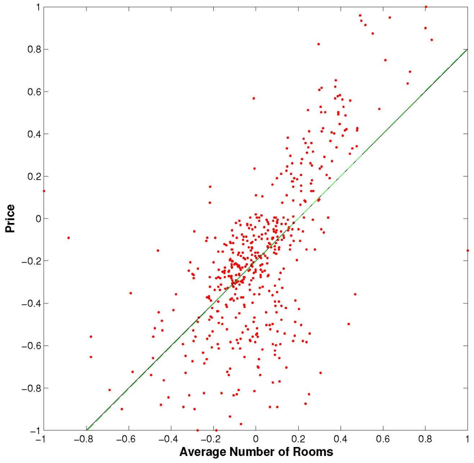


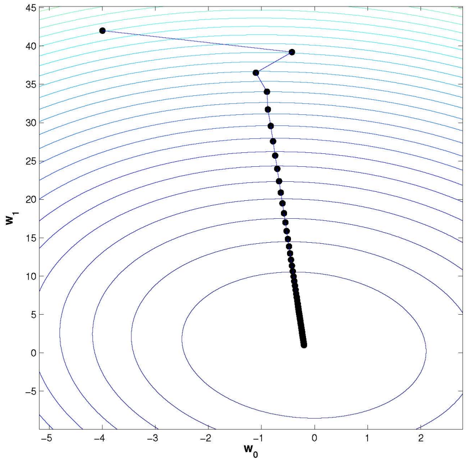
Plot of the sum of squared errors

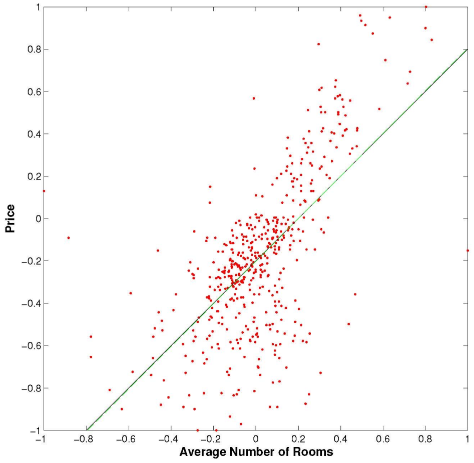


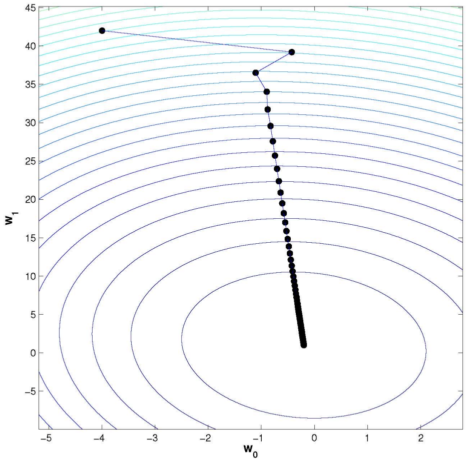
Contour plot of the SSE

Predicting Market Values of Houses









Analytical solution

- Given target vector y, and data in matrix X

$$y_i = w_0 + \sum_{i=1}^{N} w^T x_i$$

- Becomes

$$y = Xw$$

  - Where we introduce an extra “bias term” into w and x
- We can derive a closed-form solution:

$$X^T(y = Xw)$$
$$X^T y = X^T X w$$
$$(X^T X)^{-1}(X^T y = X^T X w)$$
$$(X^T X)^{-1} X^T y = w$$
$$w = (X^T X)^{-1} X^T y$$

Regression

Pitfalls

Linear Regression

- We found the solution by minimizing the squared error on training data
  - We want the model to generalize: have low error on unseen (test) data

- Parametric form of the regression function
  - Remember: Linear in parameters
  - Can transform input features
  - Do we learn a straight line, polynomial ??

- What are the impacts of these assumptions / design choices on the learned regression model?

What is the test set?

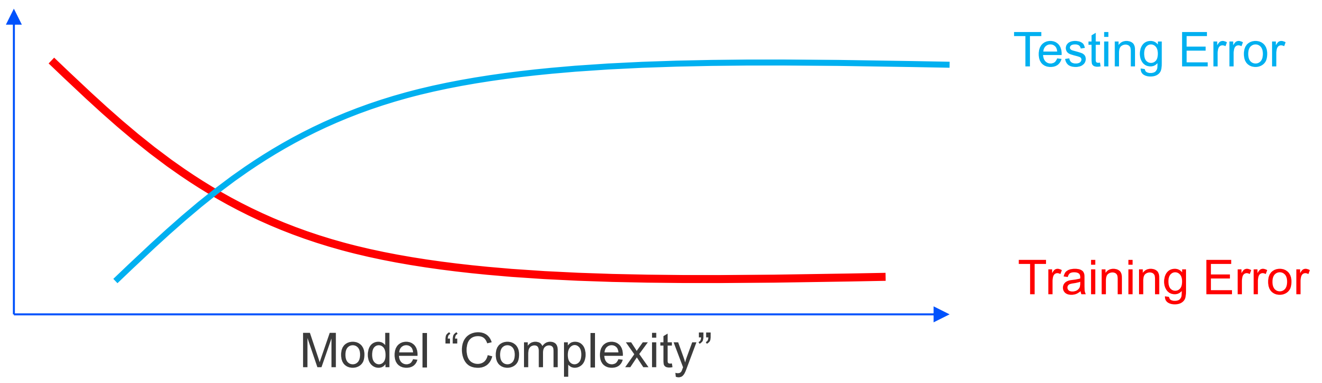
- Training Set:
  - Same distribution
  - Model look at this to train
    - Calculate stats
    - Estimate parameters
    - Check
  - Not a good indicator of true performance

- Test Set
  - Same Distribution
  - Model only sees this to evaluate not to train
  - Better indicator of true performance

Overfitting

- The model learns idiosyncrasies (noise) in training data, resulting in poor generalization
  - Low training error, order of magnitudes higher test error

- When does this happen?
  - Less training data relative to the model parameters
  - Model is too complex and fits the training too well
  - Example: degree-N polynomial exactly fits N+1 data points



Example

- Input data sampled from the unit interval [0,1]
- Target generated by $f(x) = 7.5\sin(2.5\pi x)$ with added random Gaussian noise $\epsilon \sim N(0, \sigma^2)$.

$$y_i = f(x_i) + \epsilon$$

- Training dataset has N instances (x_i, y_i)

Polynomial Regression

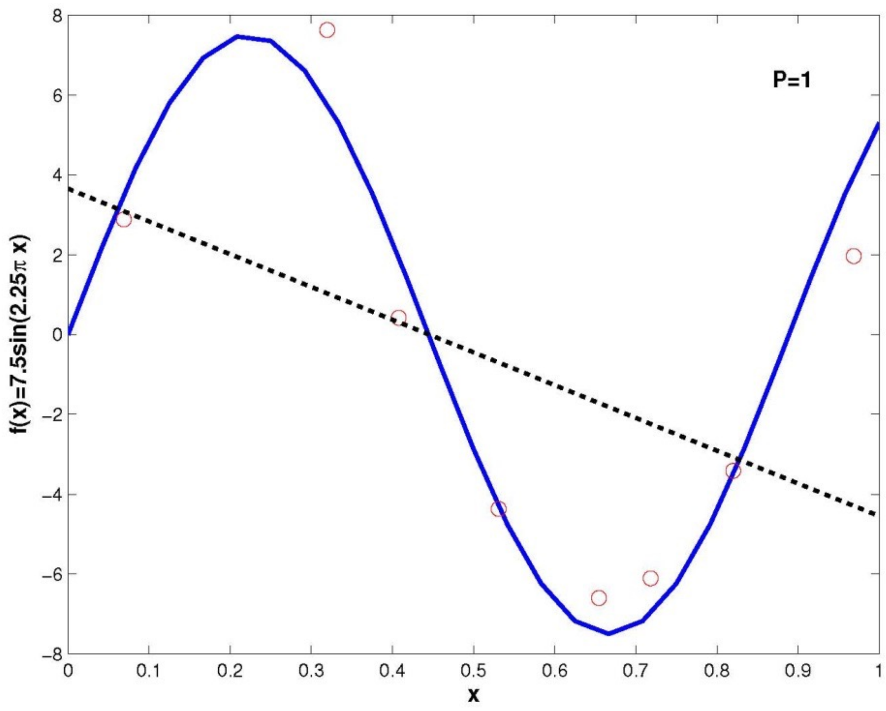
- Fit a k-order polynomial instead of a line
- Transform the data and train linear regression
    - Expand all combinations of the features up to degree K
    - K = 2, x1, x2, x1^2, x2^2, x1x2
    - K = 3, x1, x2, x1^2, x2^2, x1x2, x1^3, x1x2^2, x1x2x3, x2^3, x1^2*x2, …

x1	x2	y
1	2	y1
3	4	y2
7	3	y3
2	4	y4

x1	x2	x1^2	x2^2	x1x2	y
1	2	1	4	2	y1
3	4	9	16	12	y2
7	3	49	9	21	y3
2	4	4	16	8	y4

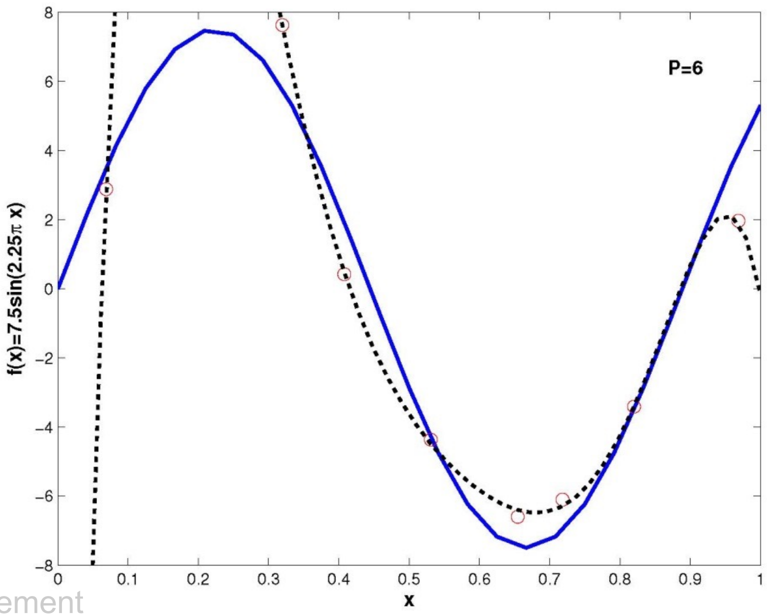
Polynomial Degree 1

- Polynomial order-1 (straight line)
- Large training error (distance of black line to red circles)
- Underfits the data and fails to adequately represent the target function (shown in blue)



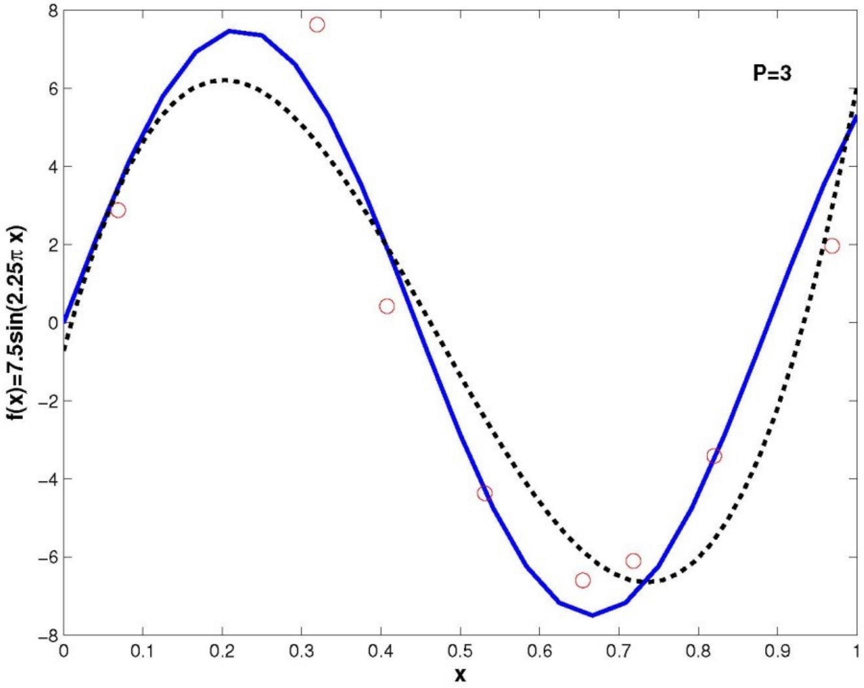
Polynomial Degree 6

- Polynomial order-6
- Negligible training error (distance of black line to red circles)
- Overfits the data and fits the training perfectly but is very different from the target function (shown in blue)
- Small changes in training data will cause the learned function to change drastically

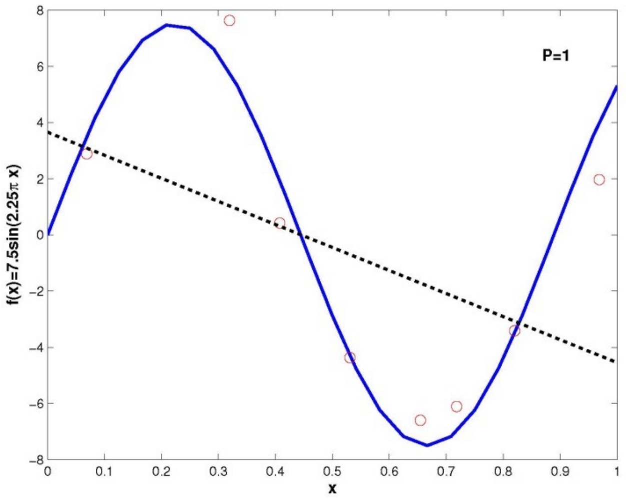


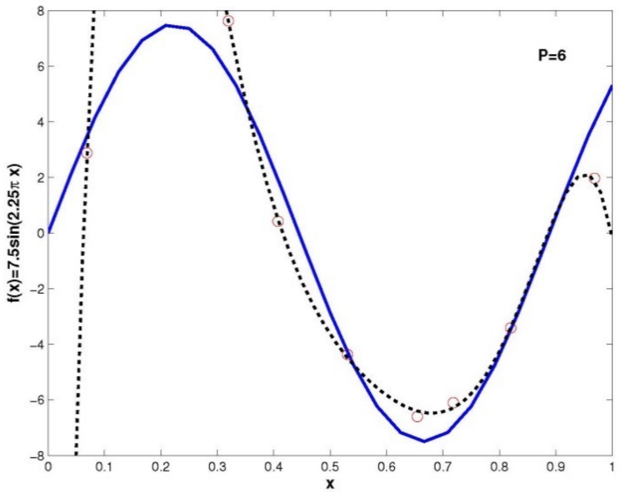
Polynomial Degree 3

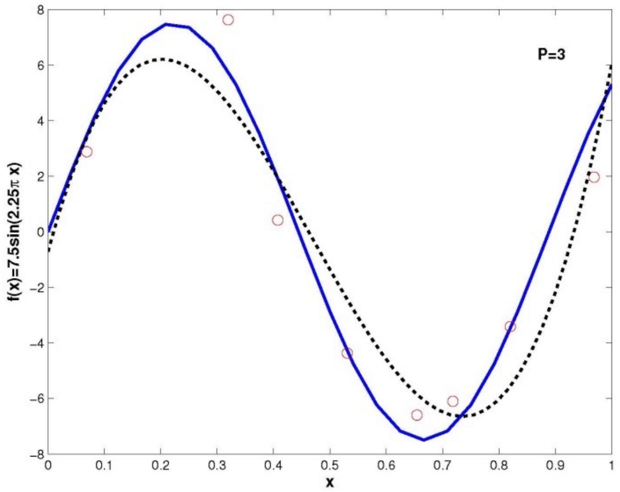
- Polynomial order-3
- Some training error (distance of black line to red circles)
- Does not fit the training data perfectly but closely resembles the target function



Which Fits Better k= 1, 3, or 6?







Avoiding Overfitting

- More training data
  - Always works, maybe difficult/costly to acquire in real world settings

- Cross Validation
  - Tune parameters by partitioning the training data into training and test data

- Regularization
  - Mathematical framework for constraining the complexity of the model

Avoiding Overfitting

- More complex models lead to larger magnitude of model parameters
- Slight perturbations in input features lead to large changes in the output/prediction
- To avoid overfitting in linear regression we should discourage large parameter values

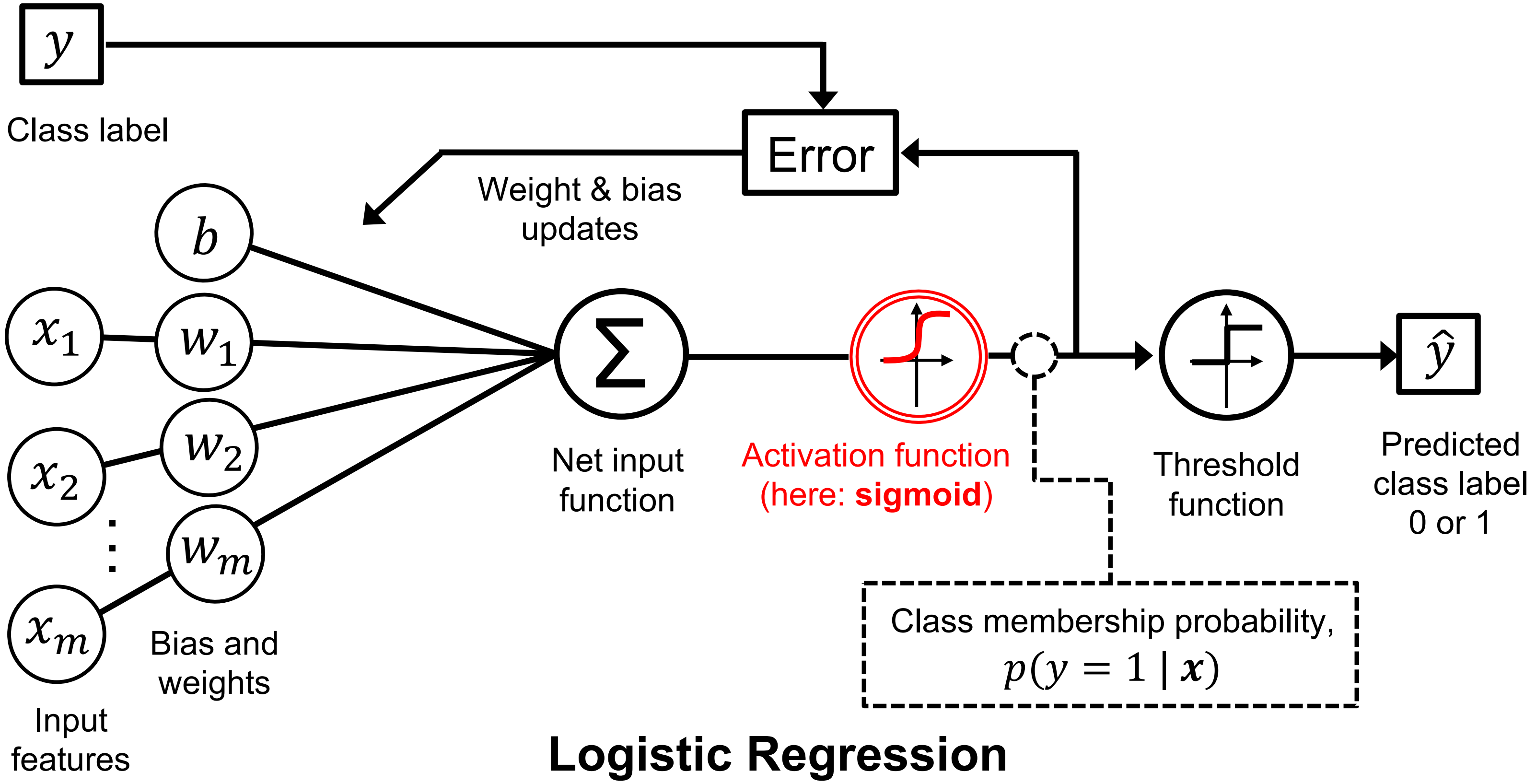
P=1	P=2	P=3	P=6	P=11	
4.03	7.05	6.94	9.67	-9.30	w0
-6.79	-67.39	17.90	71.16	349.12	w1
	55.93	-133.76	-88.63	-3007.89	w2
		118.92	-666.91	13686.70	w3
			1394.59	-33618.18	w4
			-697.61	35776.97	w5
			-7.12	2947.08	w6
				-28450.26	w7
				-1590.43	w8
				21449.25	w9
				-3644.63	w10
				-3888.54	w11

Model Complexity

Logistic Regression

Binary Classification Algorithm

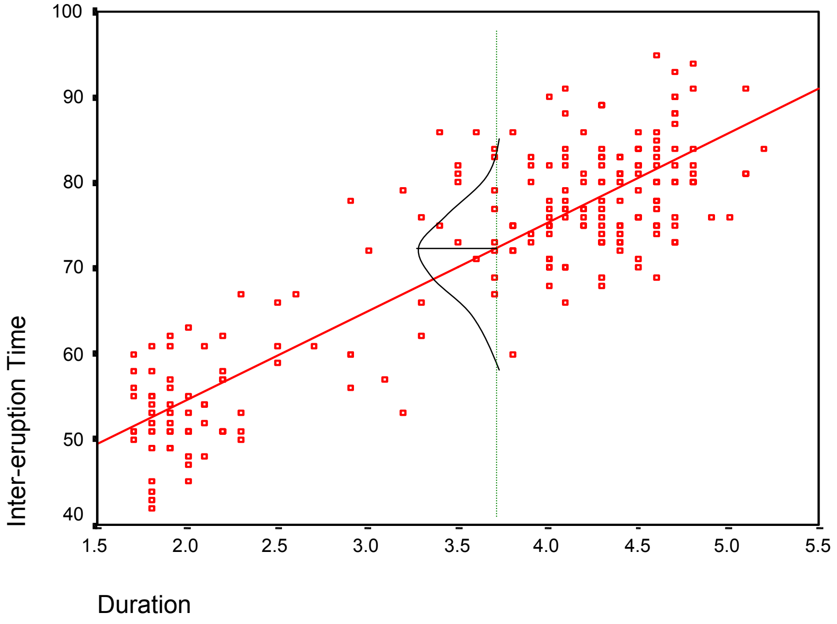
Logistic Regression



Source: Raschka, Liu, and Mirjalili. *Machine Learning with PyTorch and Scikit-Learn, Ch 3*

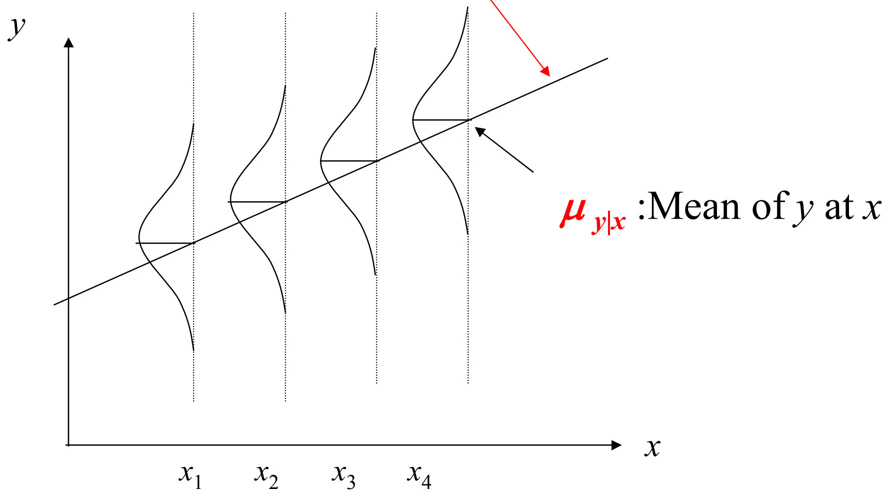
Linear Regression

In Simple Linear Regression
(Response variable is quantitative)



In Simple Linear Regression

$$\hat{y} = \hat{\alpha} + \hat{\beta} \cdot x$$



Linear Regression

Linear Regression Model for Quantitative Response Variable

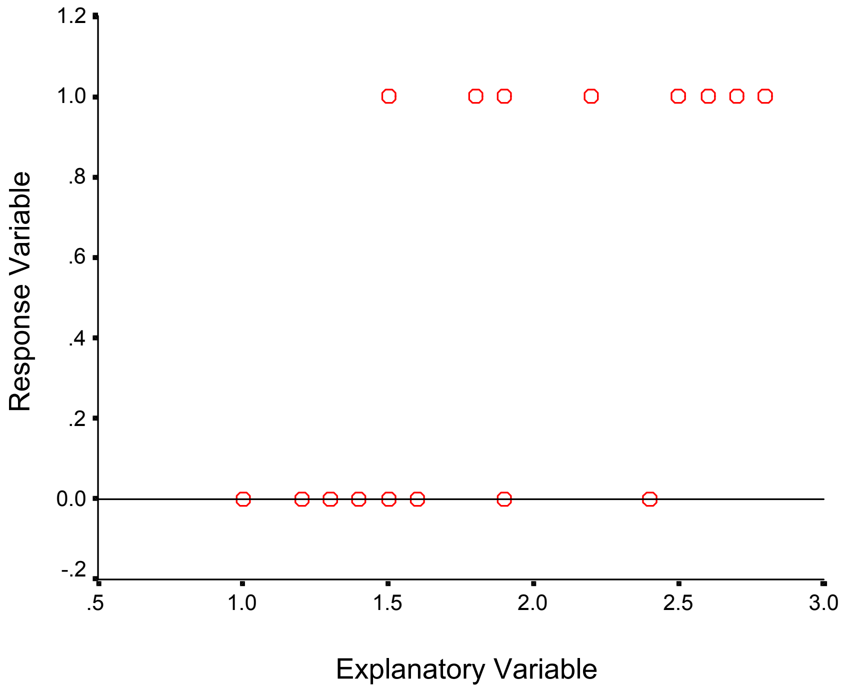
$$Y = \mu_{y|x} + \varepsilon$$

Models the mean response

$$\mu_{y|x} = \alpha + \beta X$$

Can we model a binary response variable by Logistic Regression?

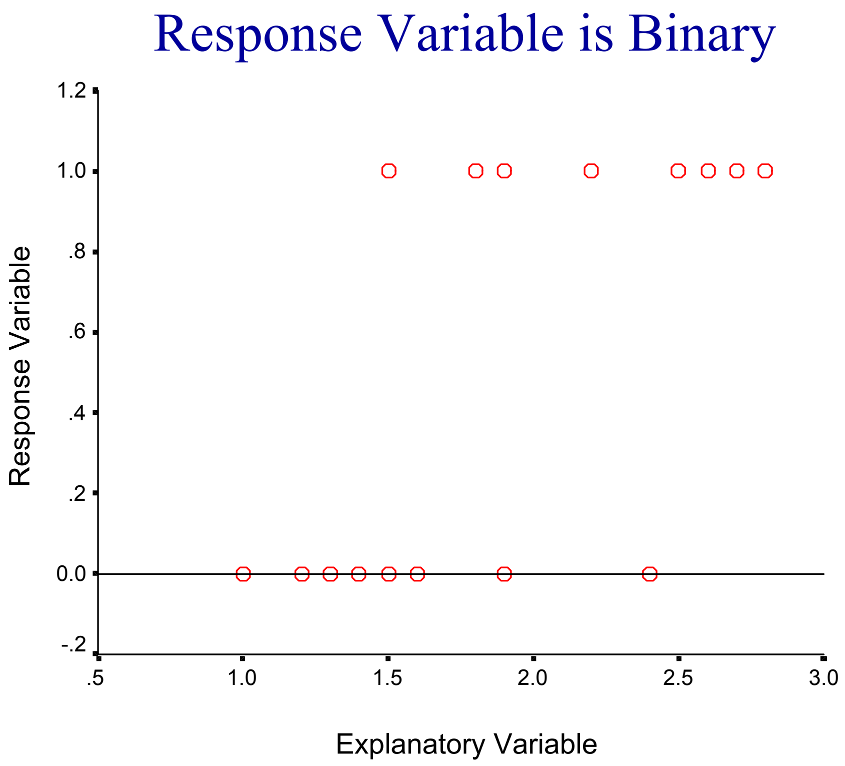
Response Variable is Binary



Linear Regression

Can we model a binary response variable by Logistic Regression?

- Doesn't matter that the targets are actually binary. Treat them as continuous values.
- What are the problems?



Regression Model with Binary Outcomes

Since the outcome is either 1 or 0, it can be modeled by Bernoulli distribution.

$$Y \sim \textbf{Bernoulli}$$

Y	**Probability**
1	$P(Y=1) = p$ or $P(Y=1 \mid X) = p$
0	$P(Y=0) = 1-p$ or $P(Y=0 \mid X) = 1-p$

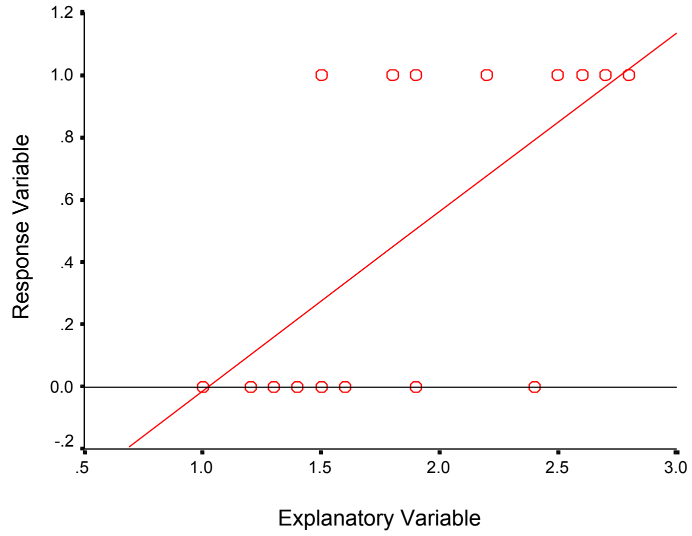
The **mean of Bernoulli distribution** is **the probability of success**: $\mu_{y|x} = \boxed{p = ????}$

(We want to model the probability of getting 1, or {Y = 1}, at a given X.)

Regression Model with Binary Outcomes

How to model the probability of success?

Can we fit a line for p and x using linear regression?



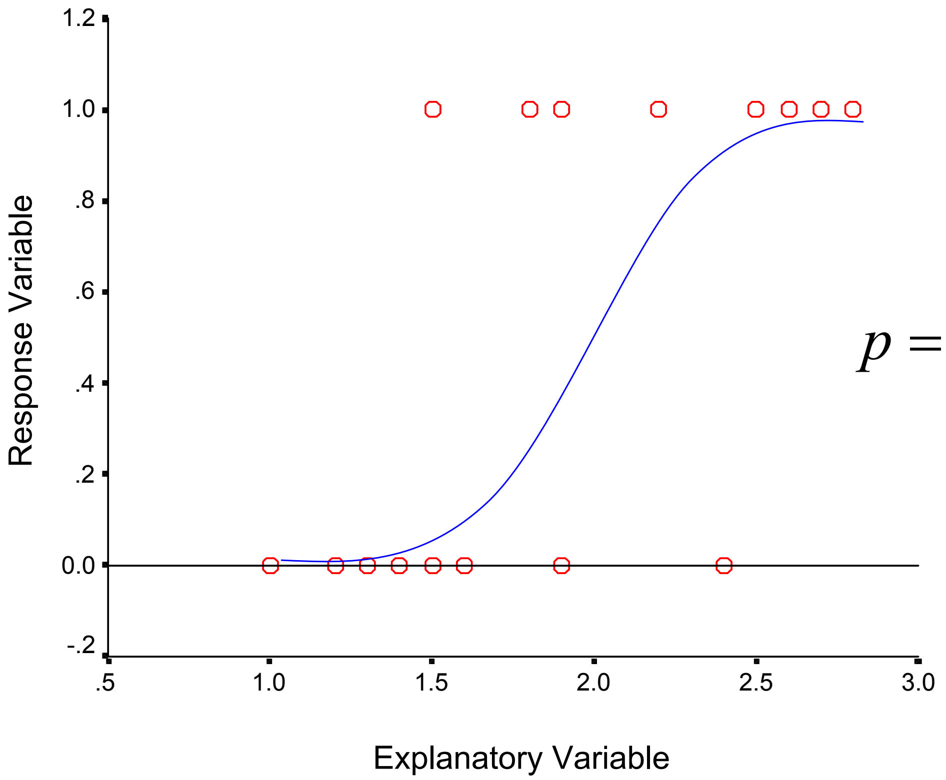
Problems

1. Non-normal error terms: ε
2. Non-constant error variance: $\sigma^2(\varepsilon)$
3. Constraints of response function: $0 \leq \mu_{y|x} = p \leq 1$

Fitting a Logistic Function

To model the probability of success:

Fitting a Logistic Function (Sigmoidal Curve)

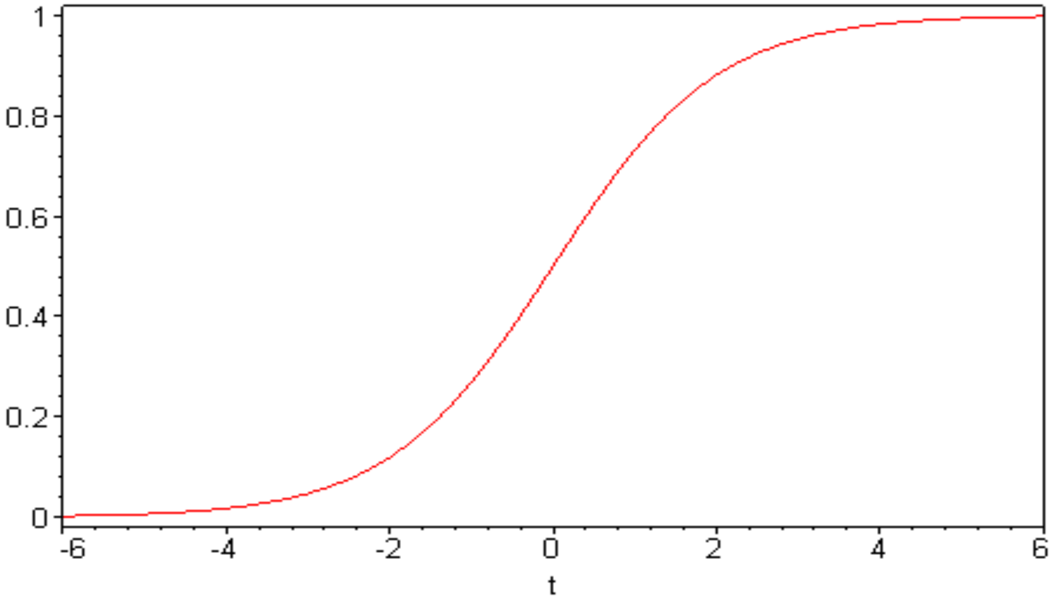


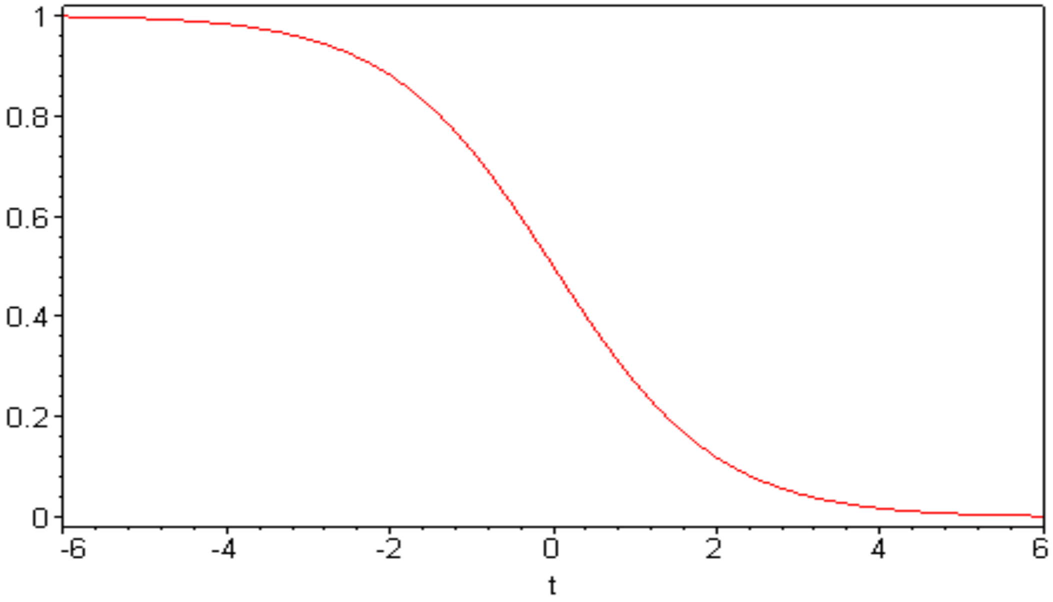
$$p = \frac{e^{\alpha + \beta \cdot x}}{1 + e^{\alpha + \beta \cdot x}}$$

Fitting a Logistic Function

$$f(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

$$f(x) = \frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^x}$$





Simple Logistic Regression

The probability of success (probability of $Y = 1$ given x):

$$p = \frac{e^{\alpha+\beta\cdot x}}{1+e^{\alpha+\beta\cdot x}} = \frac{1}{1+e^{-\alpha-\beta\cdot x}}$$

Properties:

Sigmoidal (S-shaped)

Monotonic (Increasing or decreasing)

Linearizable (Logit transformation)

Different form of the logistic function:

Logit Transformation (Logit link)

The natural log. of the odds in favor of success at x:

$$\boxed{\ln\left[\frac{p}{1-p}\right] = \alpha + \beta\cdot x}$$

Logit response function

(The logarithm of the odds is linearly related to x.)

Simple Logistic Regression

The odds in favor of success ($Y=1$) at x are

Odds
$$\frac{p}{1-p} = e^{\alpha+\beta\cdot x}$$

Probability

Solving for p $\Rightarrow$ $$p = \frac{e^{(\alpha+\beta\cdot x)}}{1+e^{(\alpha+\beta\cdot x)}}$$

Logistic Regression; Terminology

- The odds refers to the ratio of the probability of success (p) to the probability of failure (1-p).

$$\frac{p}{1-p}$$

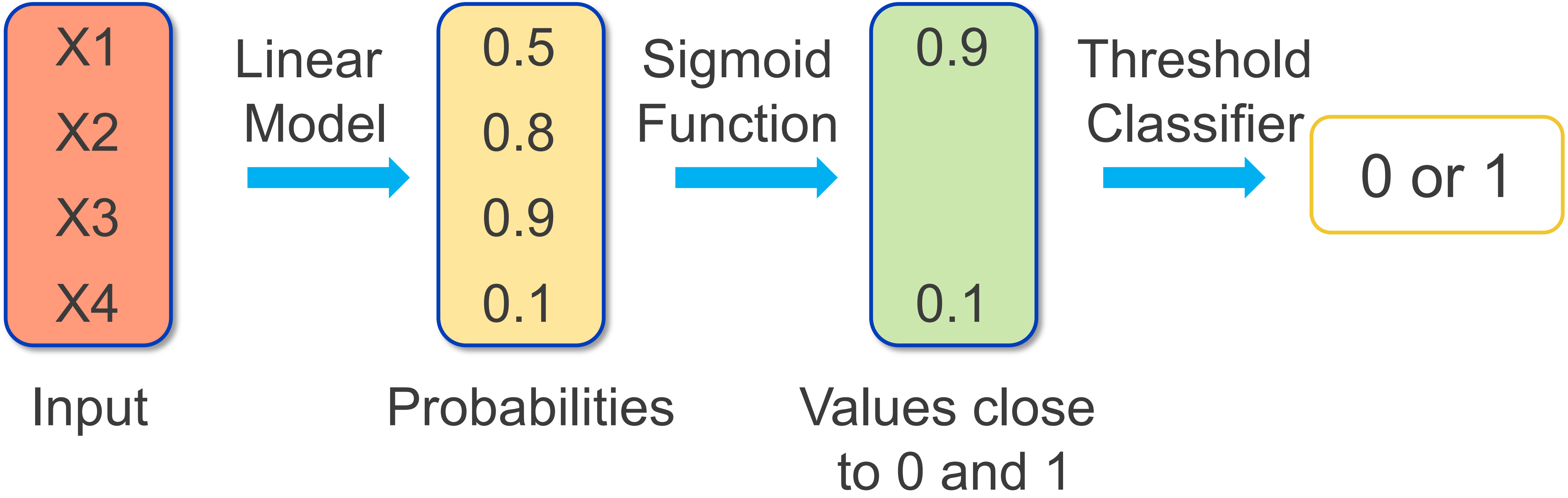
- The log-odds are the inverse of the sigmoid function.

$$\ln\left(\frac{p}{1-p}\right)$$

- The Sigmoid function is a function that maps logistic or multinomial regression output (log odds) to probabilities, returning a value between 0 and 1.

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

Logistic Regression Steps



Logistic Regression; Parameters Estimation

There are two common approaches:

- Maximum a Posteriori (MAP) Estimate

- Maximum Likelihood Estimate (MLE)
    - MLE is a special case of MAP when *prior* is uninformative.

Logistic Regression; MAP Approach

- We absorb the parameter b into w through an additional constant dimension.
- We treat w as a random variable and can specify a prior belief distribution over it (i.e. $\mathrm{w} = \mathcal{N}(0, \sigma^2)$)
- Our goal in MAP is to find the most likely model parameters given the data:

$$P(\mathrm{w}|Data) \propto P(Data|\mathrm{w})P(\mathrm{w})$$

$$\underset{\mathrm{w}}{\operatorname{argmax}}[logP(Data|\mathrm{w})P(\mathrm{w})] = \underset{\mathrm{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \mathrm{w}^T \mathrm{x}}\right) + \lambda \mathrm{w}^T \mathrm{w}$$

where λ is a linear function of $\frac{1}{2\sigma^2}$.

Logistic Regression; MLE Approach

- In MLE we choose parameters that maximize the conditional data likelihood.
- The conditional data likelihood is the probability of the observed values of Y in the training data conditioned on the values of X.
- We choose the parameters that maximize this function:

$$\log\left(\prod_{i=1}^{n} P(y_i|\mathrm{x}_i;\mathbf{w},\mathrm{b})\right) = -\sum_{i=1}^{n}\log\left(1+e^{-y_i(\mathbf{w}^T\mathrm{x}+b)}\right)$$

$$\mathbf{w},\mathrm{b} = \underset{\mathbf{w},b}{\operatorname{argmax}} -\sum_{i=1}^{n}\log\left(1+e^{-y_i(\mathbf{w}^T\mathrm{x}+b)}\right)$$

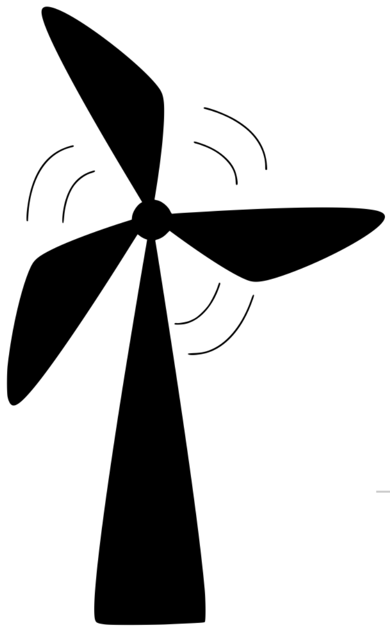
$$= \underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{n}\log\left(1+e^{-y_i(\mathbf{w}^T\mathrm{x}+b)}\right)$$

We need to estimate the parameters **w**,b.

Logistic Regression; MLE and MAP

- Both MAP and MLE have no closed form solutions in general.
    - But in 2015 a group of researchers showed that a closed-form solution exists when all predictors are categorical. If the design matrix X is coded in a particular way, with orthogonal binary variables, then there is a closed form solution. See this article for more details.

- Any gradient-based optimizer can optimize the loss functions in MAP and MLE approaches.



Case Studies

Linear Regression: Energy Consumption

Logistic Regression: Customer Churn

Next Lecture

Model Performance Evaluation