

CS 460200

**Introduction to
Machine Learning**

Regression

Instructor: Po-Chih Kuo

Roadmap

- Introduction and Basic Concepts
- Regression
- Bayesian Classifiers
- Decision Trees
- KNN
- Linear Classifier
- Neural Networks
- Deep learning
- Convolutional Neural Networks
 - Autoencoder
 - Adversarial
 - Transfer learning
 - ...
- RNN/Transformer
- Reinforcement Learning
- Model Selection and Evaluation
- Clustering
- Data Exploration & Dimensionality reduction

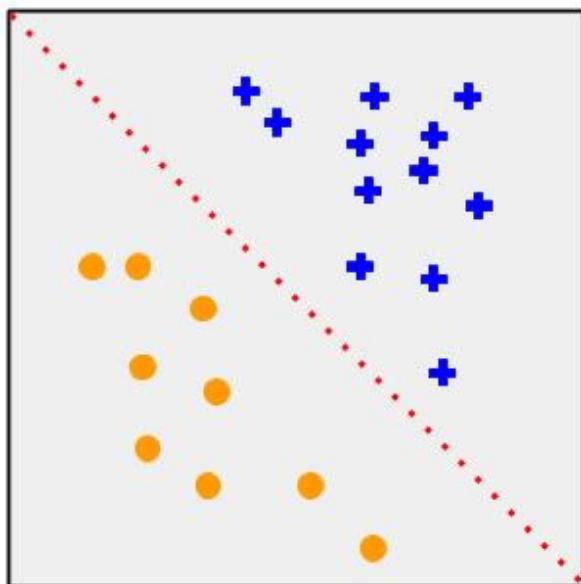


Supervised learning

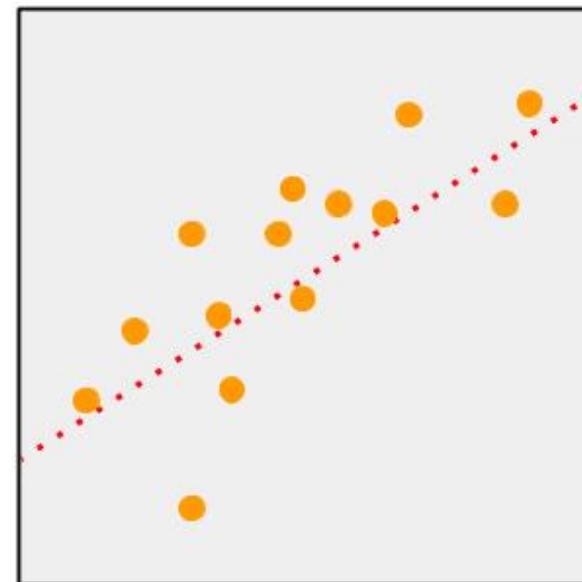
This week

- Regression vs Classification
- Linear regression
 - As matrix inversion (Ordinary Least Squares)
 - As optimization: Gradient descent
 - Batch gradient decent
 - Stochastic gradient descent
- Overfitting and regularization
- Autoregression
- Logistic regression (more like classification)

Regression vs Classification



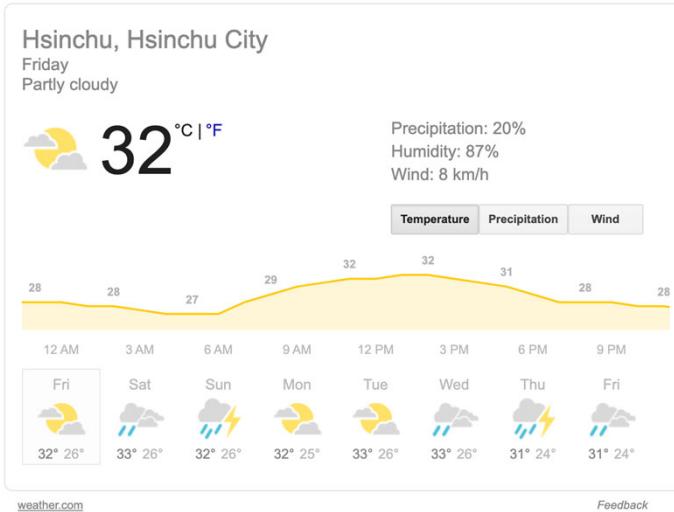
Classification



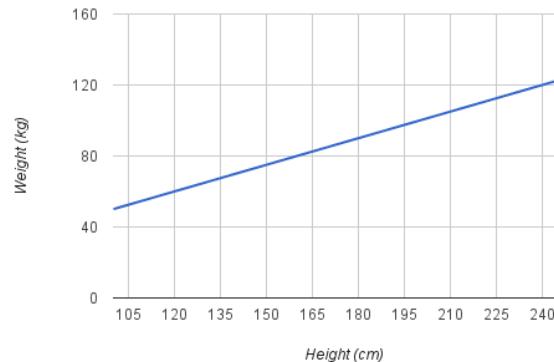
Regression

Regression examples

Weather

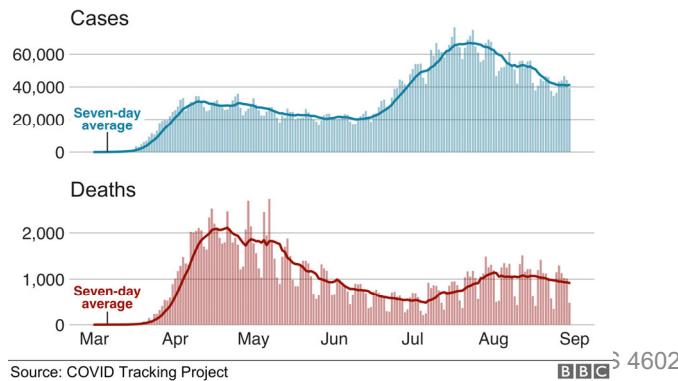


Weight vs. Height

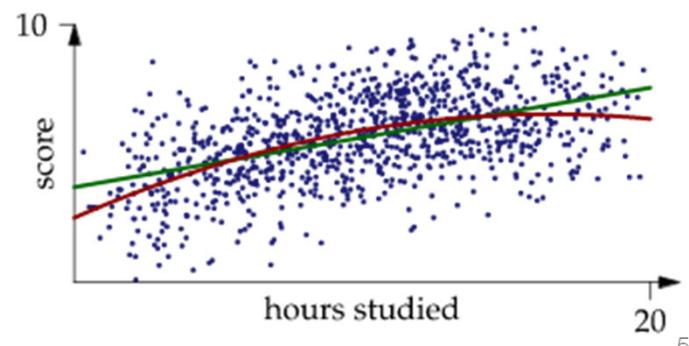


Disease case

Number of daily cases and deaths in the US



Score vs. Hours studied



Example: House price in the U.S

	X_i	Y_i
i	sqft	price
1	5650	221900
2	7242	538000
3	10000	180000
4	5000	604000
5	8080	510000
6	101930	1230000

Training data
 $\{(X_i, Y_i)\}_{i=1}^n$



Machine learning algorithm



Prediction function
 $f(X_i)$

Linear regression

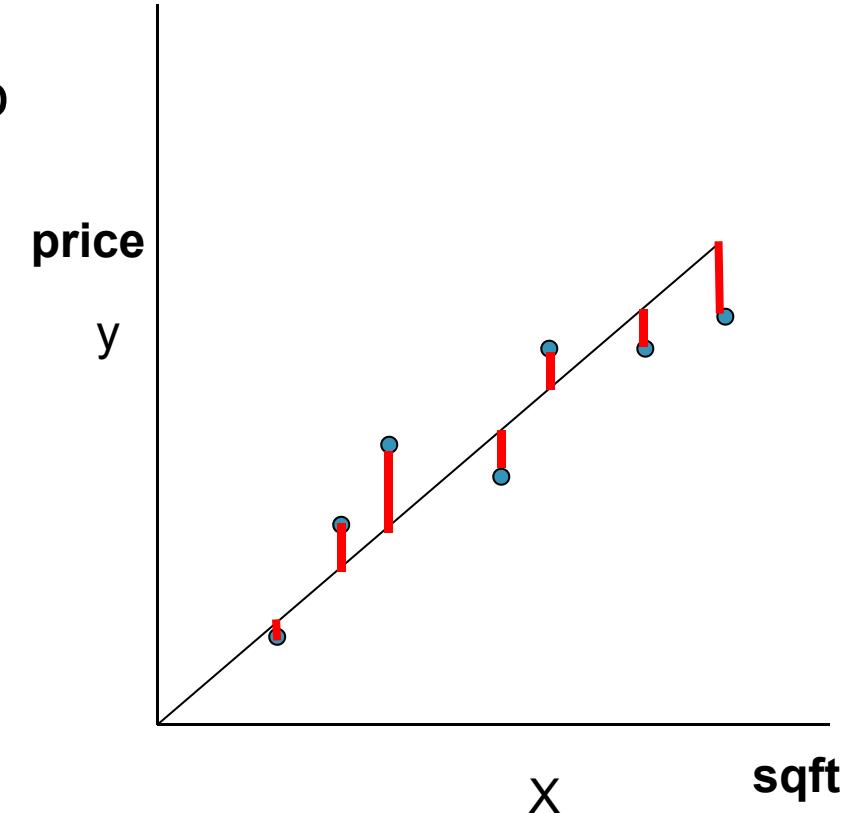
- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

What we are trying to predict



$$y = f(x) = wx + \varepsilon$$

where w is a parameter and ε represents measurement noise



Linear regression

- Our goal is to estimate w from a training data of $\langle x_i, y_i \rangle$ pairs
- Optimization goal: minimize squared error (least squares):

$$\operatorname{argmin}_w \sum_i (y_i - w x_i)^2$$

\downarrow
 $L(w)$

- Why least squares?

- Straightforward! minimizes squared distance between measurements and predicted line
- Has probabilistic interpretation: equivalent to maximum likelihood estimation.
(http://people.math.gatech.edu/~ecroot/3225/maximum_likelihood.pdf)
- The math is pretty

Solving linear regression

- To optimize – closed form:
- Let's take the derivative w.r.t. to w and set to 0:

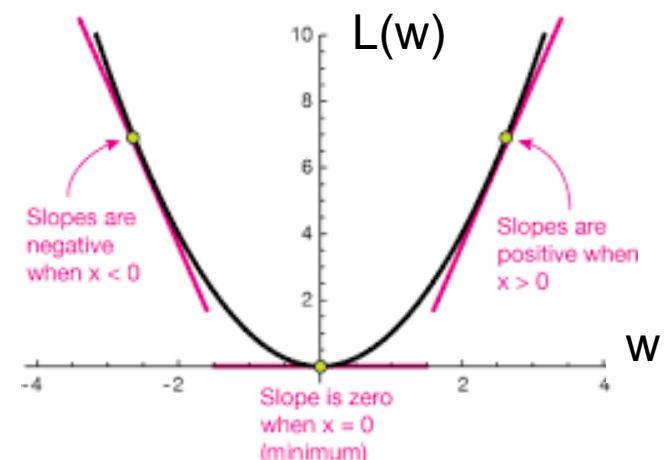
$$\frac{\partial}{\partial w} \sum_i (y_i - wx_i)^2 = 0$$

$$2 \sum_i -x_i(y_i - wx_i) = 0$$

$$2 \sum_i x_i(y_i - wx_i) = 0$$

$$2 \sum_i x_i y_i - 2 \sum_i w x_i x_i = 0$$

$$\sum_i x_i y_i = \sum_i w x_i^2$$



$$w = \frac{\sum_i x_i y_i}{\sum_i x_i^2} = \frac{COVAR(X,Y)}{VAR(X)}$$

if $\text{mean}(X)=\text{mean}(Y)=0$

Variance & Covariance

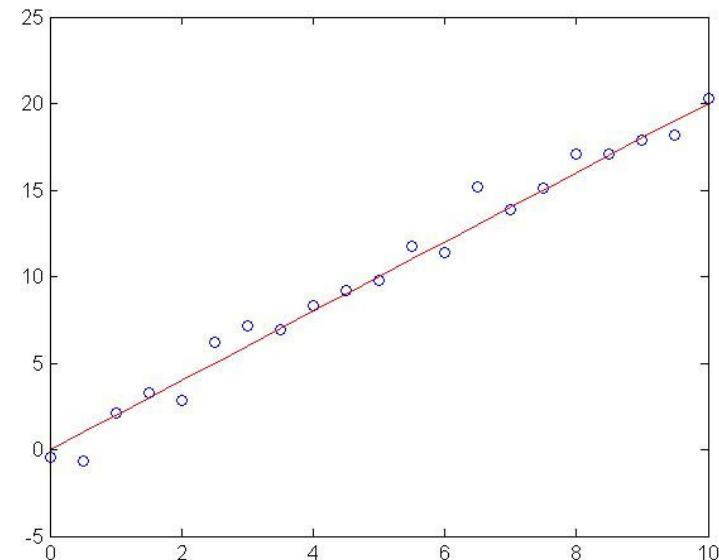
- Variance:
 - Measure of the deviation from the mean for points in one dimension
- Covariance:
 - Measure of how much each of the dimensions vary from the mean with respect to each other

$$\text{VAR}(\mathbf{X}) = \frac{1}{n-1} \sum_{i=1}^n (x_i - E(X))^2$$

$$\text{COV}(\mathbf{X}, \mathbf{Y}) = \frac{1}{n-1} \sum_{i=1}^n (x_i - E(X))(y_i - E(Y))$$

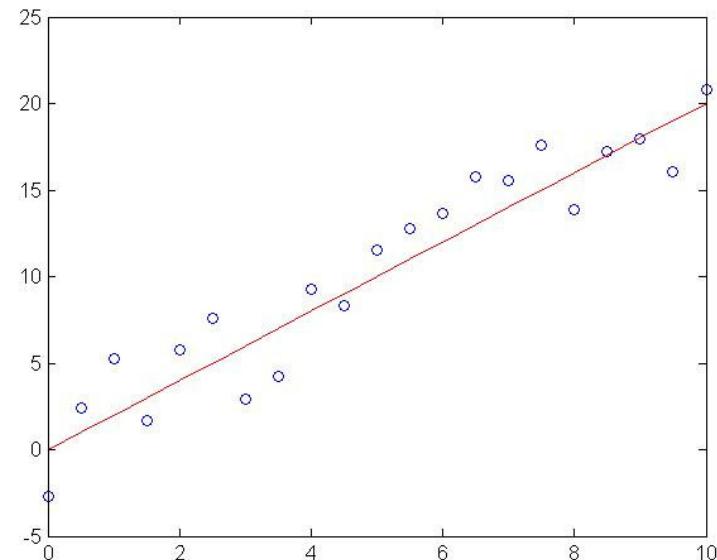
Example-1

- Generated: $w=2$
- Noise: $\text{std}=1$
- Solved: $w=2.03$



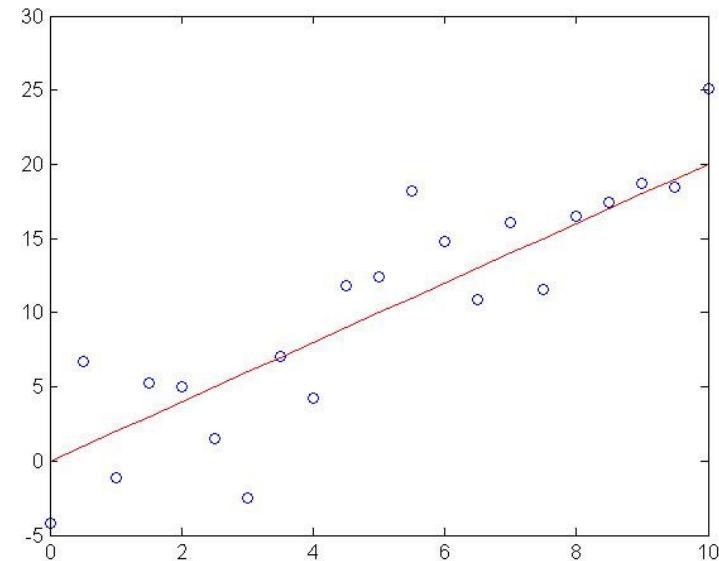
Example-2

- Generated: $w=2$
- Noise: $\text{std}=2$
- Solved: $w=2.05$



Example-3

- Generated: $w=2$
- Noise: std=4
- Solved: $w=2.08$

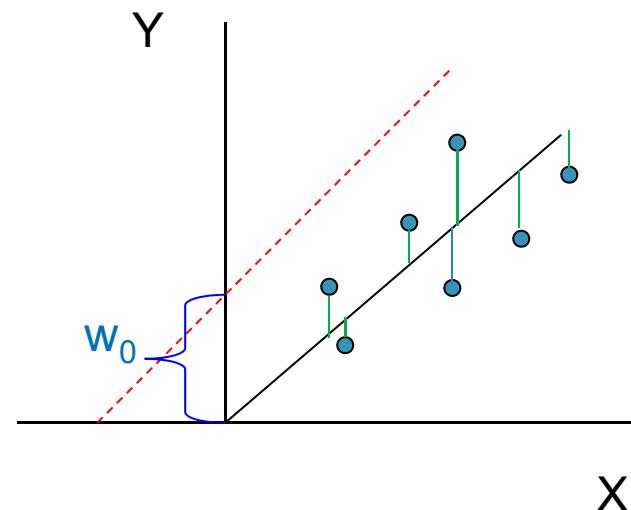


Intercept/Bias term

- So far, we assumed that the line passes through the origin. What if the line does not?
- Simply change the model to:

$$y = w_0 + w_1 x + \epsilon$$

- We can use least squares to determine w_0 , w_1

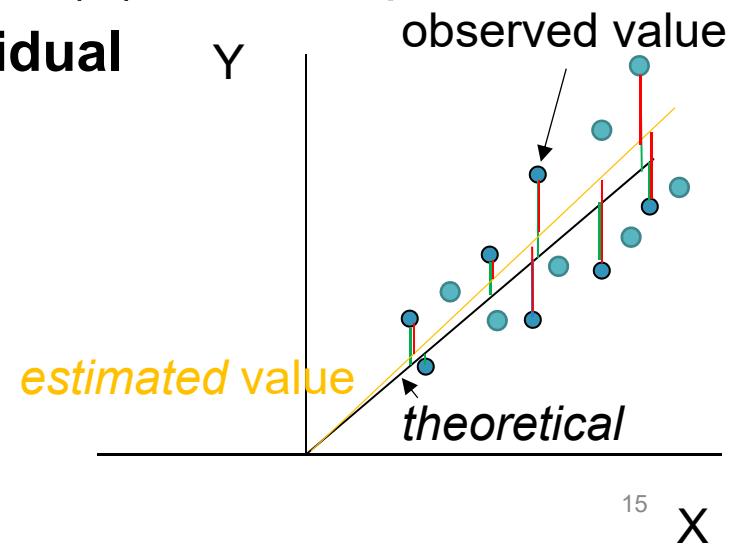


$$w_0 = \frac{\sum_i y_i - w_1 x_i}{n} \quad w_1 = \frac{\sum_i x_i (y_i - w_0)}{\sum_i x_i^2}$$

Error vs Residual

- The **error** of an observed value is the deviation of the observed value from the (unobservable) *true(theoretical)* value of a quantity of interest (for example, a population mean)
- The **residual** of an observed value is the difference between the observed value and the *estimated value* of the quantity of interest (for example, a sample mean).
- The error term accounts for the variation in the dependent variable(Y) that the independent variables(X) do not explain.
- Least square error -> Least square **residual**

$$y = wx + \varepsilon$$
$$\hat{y} = \textcolor{blue}{w}x$$
$$\operatorname{argmin}_w \sum_i (y_i - \hat{y}_i)^2$$
$$|y - \hat{y}|$$



- What if we have several inputs?

	X_1	X_2	X_3	X_4	Y
I	bedrooms	bathrooms	condition	sqft	price
1		3	1	3	5650 221900
2		3	2.25	3	7242 538000
3		2	1	3	10000 180000
4		4	3	5	5000 604000
5		3	2	3	8080 510000
6		4	4.5	3	101930 1230000

Multiple regression

“Multivariate” Regression :
more than one predictors (Y)
and more than one dependent
variable (X)

- This becomes a multiple regression problem
- Again, its easy to model:

$$y = w_0 + w_1x_1 + \dots + w_kx_k + \varepsilon$$

The diagram illustrates the components of a multiple regression equation. At the top, the equation is written as $y = w_0 + w_1x_1 + \dots + w_kx_k + \varepsilon$. Below the equation, there are three rectangular boxes with light blue borders. The first box contains the word "price", the second contains "sqft", and the third contains "condition". Three black arrows originate from the bottom of each box and point upwards towards the corresponding terms in the equation: the first arrow points to w_1x_1 , the second to w_2x_2 , and the third to w_3x_3 .

Non-linear basis functions

- So far we only used the observed values x_1, x_2, \dots
- However, linear regression can be applied in the same way to **functions** of these values
 - Eg: to add a term $w x_1 x_2$ add a new variable $z=x_1 x_2$ so each example becomes: x_1, x_2, \dots, z
 - As long as these functions can be directly computed from the observed values, the parameters are still **linear** in the data, and the problem remains a multiple linear regression problem:

$$y = w_0 + w_1 x_1^2 + \dots + w_k x_k^2 + \varepsilon$$

Linear in W, not necessarily in x

Non-Linear basis functions

- What type of functions can we use?
- A few common examples:
 - Polynomial:

$$\phi_j(x) = x^j$$

- Gaussian:

$$\phi_j(x) = \exp\left(-\frac{(x - \mu_j)^2}{2s^2}\right)$$

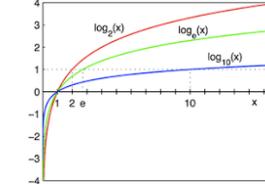
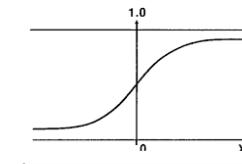
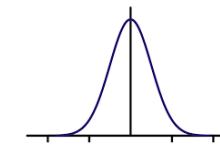
- Sigmoid:

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right) \quad \sigma(a) = \frac{1}{1 + \exp(-a)}$$

- Logs:

$$\phi(x) = \log(x + 1)$$

Similar idea in
SVM



Any function of the input values can be used. The solution for the parameters of the regression remains the same.

General linear regression problem

- Using our new notations for the basis function linear regression can be written as

$$y = \sum_{j=0}^n w_j \phi_j(x)$$

- Where $\phi_j(x)$ can be either x_j for multiple regression or one of the non-linear basis functions we defined
- $\phi_0(x)=1$ for the intercept term

Learning/Optimizing Multivariate Least Squares

Approach 1: Matrix Inversion

Approach 2: Gradient Descent

OLS (Ordinary Least Squares Solution)

OLS Assumption: The error term has a population mean of zero

predict with: $\hat{y}^i = \sum_j^n w_j \phi_j(\mathbf{x}^i)$

Goal: minimize the following loss function:

$$J_{\mathbf{x}, \mathbf{y}}(\mathbf{w}) = \sum_i \left(y^i - \hat{y}^i \right)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2$$



sum over n examples sum over $k+1$ basis vectors

OLS (Ordinary Least Squares Solution)

Goal: minimize the following loss function:

$$J_{\mathbf{x}, \mathbf{y}}(\mathbf{w}) = \sum_i (y^i - \hat{y}^i)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2$$

$$\begin{aligned} \frac{\partial}{\partial w_j} J(\mathbf{w}) &= \frac{\partial}{\partial w_j} \sum_i (y^i - \hat{y}^i)^2 \\ &= 2 \sum_i (y^i - \hat{y}^i) \frac{\partial}{\partial w_j} \hat{y}^i \\ &= 2 \sum_i (y^i - \hat{y}^i) \frac{\partial}{\partial w_j} \sum_j w_j \phi_j(\mathbf{x}^i) \\ &= 2 \sum_i (y^i - \hat{y}^i) \phi_j(\mathbf{x}^i) \end{aligned}$$

We see

$$J_{\mathbf{x}, \mathbf{y}}(\mathbf{w}) = \sum_i (y^i - \hat{y}^i)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = 2 \sum_i (y^i - \hat{y}^i) \phi_j(\mathbf{x}^i)$$

Now, let's consider i

Notation:

$$\mathbf{y} = \begin{pmatrix} y^1 \\ \cdots \\ \cdots \\ y^n \end{pmatrix} \quad \Phi = \left[\begin{array}{cccc} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \cdots & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \cdots & \phi_k(\mathbf{x}^2) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \cdots & \phi_k(\mathbf{x}^n) \end{array} \right] \quad \mathbf{w} = \begin{pmatrix} w_0 \\ \cdots \\ w_k \end{pmatrix}$$

$k+1$ basis vectors

n examples

k+1 basis vectors

$$\mathbf{y} = \begin{pmatrix} y^1 \\ \cdots \\ \cdots \\ y^n \end{pmatrix} \quad \Phi = \left(\begin{array}{cccc} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \cdots & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \cdots & \phi_k(\mathbf{x}^2) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \cdots & \phi_k(\mathbf{x}^n) \end{array} \right) = \left[\begin{array}{c} \vec{\phi}^1 \\ \cdots \\ \cdots \\ \vec{\phi}^n \end{array} \right] \quad \boxed{n \text{ examples}} \quad \mathbf{w} = \begin{pmatrix} w_0 \\ .. \\ w_k \end{pmatrix}$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = \left(\begin{array}{c} \frac{\partial}{\partial w_0} J(\mathbf{w}) = 2 \sum_i (y^i - \hat{y}^i) \phi_0(x^i) \\ \cdots \\ \frac{\partial}{\partial w_k} J(\mathbf{w}) = 2 \sum_i (y^i - \hat{y}^i) \phi_k(x^i) \end{array} \right) = \left(\begin{array}{c} \frac{\partial}{\partial w_0} J(\mathbf{w}) = 2 \sum_i (y^i \phi_0^i - \hat{y}^i \phi_0^i) \\ \cdots \\ \frac{\partial}{\partial w_k} J(\mathbf{w}) = 2 \sum_i (y^i \phi_k^i - \hat{y}^i \phi_k^i) \end{array} \right)$$

notation: $\phi_j^i \equiv \phi_j(\mathbf{x}^i)$

$$\text{recall } \hat{y}^i = \sum_j^n w_j \phi_j^i \\ = \vec{\phi}^i \mathbf{w}$$

$$\begin{aligned} & \left\{ \begin{array}{l} \frac{\partial}{\partial w_0} J(\mathbf{w}) = 2 \sum_i (y^i \phi_0^i - \hat{y}^i \phi_0^i) \\ \dots \\ \frac{\partial}{\partial w_k} J(\mathbf{w}) = 2 \sum_i (y^i \phi_k^i - \hat{y}^i \phi_k^i) \end{array} \right. \\ &= \left\{ \begin{array}{l} \frac{\partial}{\partial w_0} J(\mathbf{w}) = 2 \sum_i (y^i \phi_0^i - \phi^i \mathbf{w} \phi_0^i) \\ \dots \\ \frac{\partial}{\partial w_k} J(\mathbf{w}) = 2 \sum_i (y^i \phi_k^i - \phi^i \mathbf{w} \phi_k^i) \end{array} \right. \\ &= \left\{ \begin{array}{l} \frac{\partial}{\partial w_0} J(\mathbf{w}) = 2 \sum_i (\phi_0^i y^i - \phi_0^i \phi^i \mathbf{w}) \\ \dots \\ \frac{\partial}{\partial w_k} J(\mathbf{w}) = 2 \sum_i (\phi_k^i y^i - \phi_k^i \phi^i \mathbf{w}) \end{array} \right. \end{aligned}$$

n examples

$$\Phi^T = \begin{pmatrix} \phi_0(\mathbf{x}^1) & \phi_0(\mathbf{x}^2) & \cdots & \phi_0(\mathbf{x}^n) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_k(\mathbf{x}^1) & \phi_k(\mathbf{x}^2) & \cdots & \phi_k(\mathbf{x}^n) \end{pmatrix}$$

k+1 $\mathbf{y} = \begin{pmatrix} y^1 \\ \cdots \\ \cdots \\ y^n \end{pmatrix}$

$$\frac{\partial}{\partial w_0} J(\mathbf{w}) = 2 \sum_i (\phi_0^i y^i - \phi_0^i \phi^i \mathbf{w})$$

...

$$\frac{\partial}{\partial w_k} J(\mathbf{w}) = 2 \sum_i (\phi_k^i y^i - \phi_k^i \phi^i \mathbf{w})$$

$= 2\Phi^T \mathbf{y} - \dots$

<i>n examples</i>	<i>k+1 basis vectors</i>	
$\left(\begin{array}{cccc} \phi_0(\mathbf{x}^1) & \dots & \cdots & \phi_0(\mathbf{x}^n) \\ \phi_1(\mathbf{x}^1) & & & \phi_1(\mathbf{x}^n) \\ \vdots & \vdots & \dots & \vdots \\ \phi_k(\mathbf{x}^1) & & \cdots & \phi_k(\mathbf{x}^n) \end{array} \right)$	$\left(\begin{array}{cccc} \phi_0(\mathbf{x}^1) & & \cdots & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & & \cdots & \phi_k(\mathbf{x}^2) \\ \vdots & \vdots & \dots & \vdots \\ \phi_0(\mathbf{x}^n) & & \cdots & \phi_k(\mathbf{x}^n) \end{array} \right)$	$\left(\begin{array}{c} w_0 \\ \vdots \\ w_k \end{array} \right)$

$$\begin{aligned}
 \frac{\partial}{\partial w_0} J(\mathbf{w}) &= 2 \sum_i (\phi_0^i y^i - \phi_0^i \phi^i \mathbf{w}) \\
 &\quad \dots \\
 \frac{\partial}{\partial w_k} J(\mathbf{w}) &= 2 \sum_i (\phi_k^i y^i - \phi_k^i \phi^i \mathbf{w})
 \end{aligned}
 = \dots - 2 \Phi^T \Phi \mathbf{w}$$

k+1 basis vectors

$$\mathbf{w} = \begin{pmatrix} w_0 \\ .. \\ w_k \end{pmatrix} \quad \Phi = \left[\begin{array}{cccc} \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \cdots & \phi_k(\mathbf{x}^1) \\ \phi_0(\mathbf{x}^2) & \phi_1(\mathbf{x}^2) & \cdots & \phi_k(\mathbf{x}^2) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_0(\mathbf{x}^n) & \phi_1(\mathbf{x}^n) & \cdots & \phi_k(\mathbf{x}^n) \end{array} \right] \quad \boxed{n \text{ examples}}$$

$$\mathbf{y} = \begin{pmatrix} y^1 \\ \cdots \\ y^n \end{pmatrix}$$

$$\frac{\partial}{\partial w_0} J(\mathbf{w}) = 2 \sum_i (\phi_0^i y^i - \phi_0^i \phi^i \mathbf{w})$$

...

$$\frac{\partial}{\partial w_k} J(\mathbf{w}) = 2 \sum_i (\phi_k^i y^i - \phi_k^i \phi^i \mathbf{w})$$

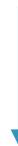
$$= 2\Phi^T \mathbf{y} - 2\Phi^T \Phi \mathbf{w} = 0$$

\downarrow
 $\mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$

OLS for general linear regression

$$w = \frac{\sum_i x_i y_i}{\sum_i x_i^2} = \frac{COVAR(X,Y)}{VAR(X)}$$

$$J(w) = \sum_i (y^i - w^T \phi(x^i))^2$$



Deriving w we get:

$$w = (\Phi^T \Phi)^{-1} \Phi^T y$$

k+1 entries vector

n entries vector

n by k+1 matrix

This solution is also known as '**pseudo inverse**'

Learning/Optimizing Multivariate Least Squares

Approach 1: Matrix Inversion

Approach 2: Gradient Descent

Gradient descent for linear regression

predict with: $\hat{y}^i = \sum_j^n w_j \phi_j(\mathbf{x}^i)$

Goal: minimize the following loss function:

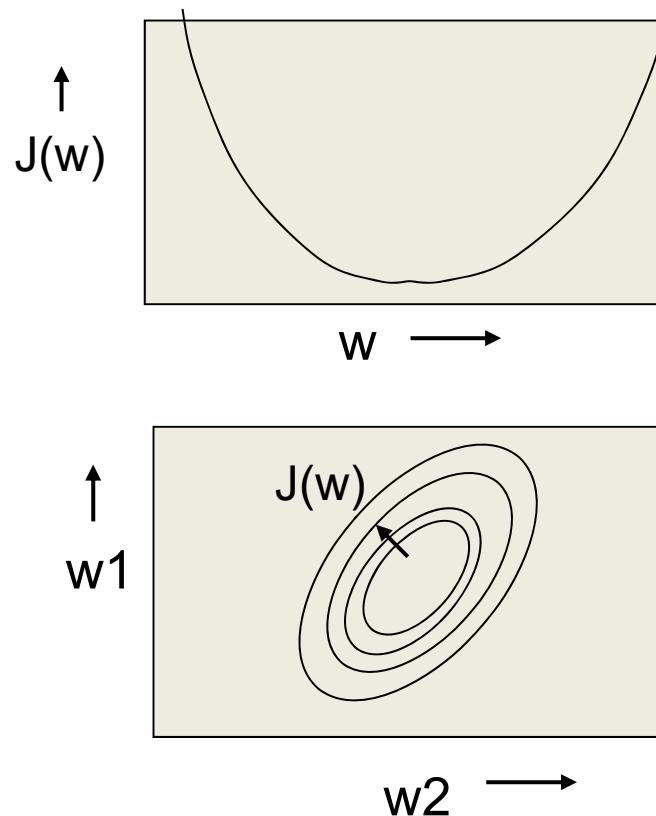
$$J_{\mathbf{x}, \mathbf{y}}(\mathbf{w}) = \sum_i \left(y^i - \hat{y}^i \right)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2$$



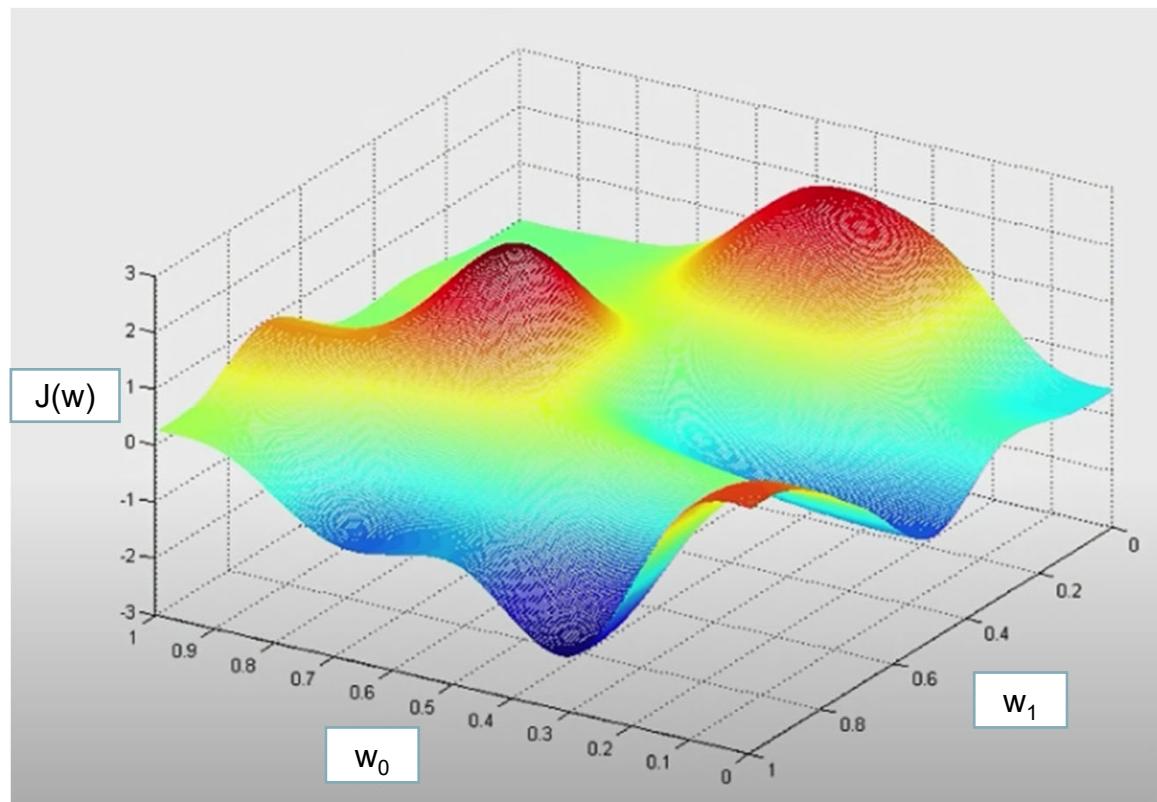
sum over n examples

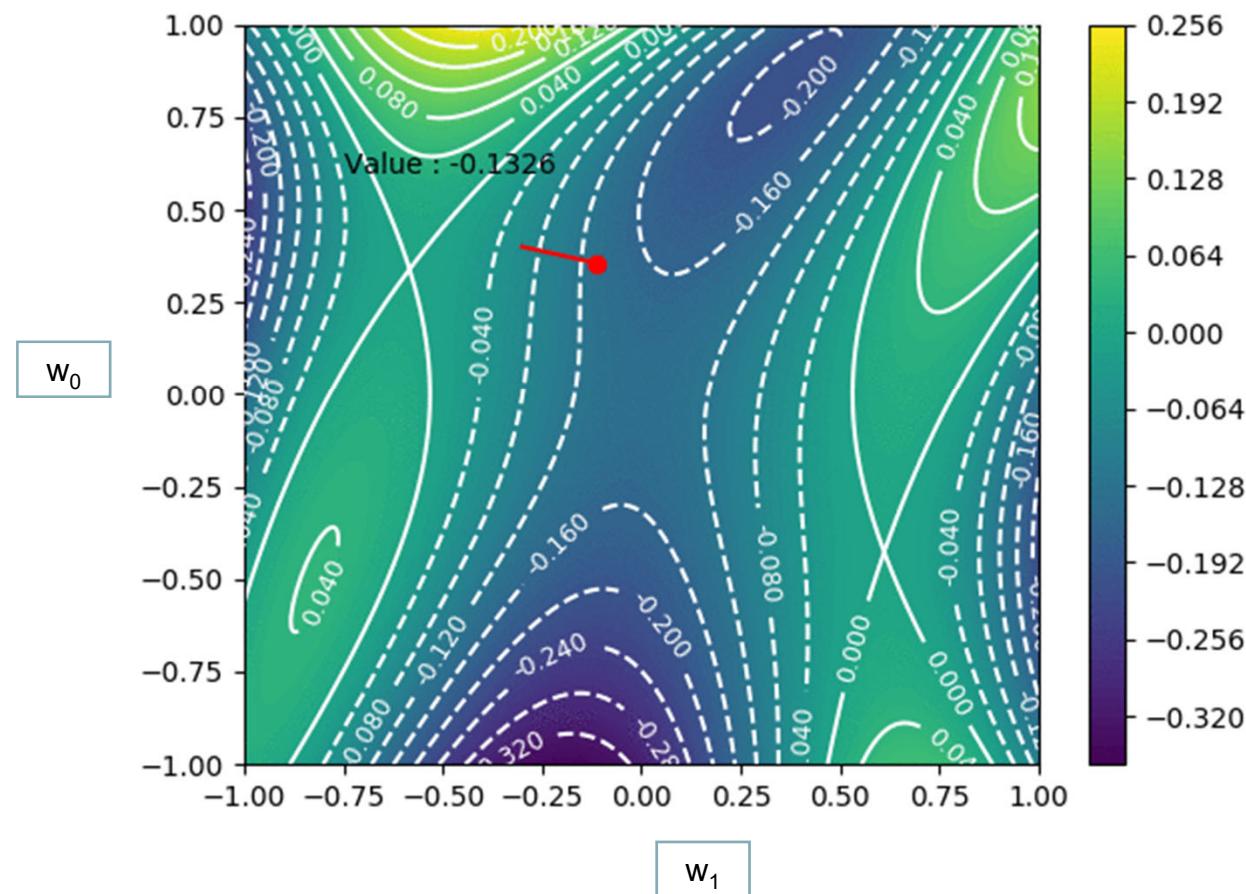
sum over $k+1$ basis vectors

Gradient descent



Gradient descent





Gradient descent for linear regression

Goal: minimize the following loss function:

$$J_{\mathbf{x}, \mathbf{y}}(\mathbf{w}) = \sum_i (y^i - \hat{y}^i)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2$$

$$\begin{aligned} \frac{\partial}{\partial w_j} J(\mathbf{w}) &= \frac{\partial}{\partial w_j} \sum_i (y^i - \hat{y}^i)^2 \\ &= 2 \sum_i (y^i - \hat{y}^i) \frac{\partial}{\partial w_j} \hat{y}^i \\ &= 2 \sum_i (y^i - \hat{y}^i) \frac{\partial}{\partial w_j} \sum_j w_j \phi_j(\mathbf{x}^i) \\ &= 2 \sum_i (y^i - \hat{y}^i) \phi_j(\mathbf{x}^i) \end{aligned}$$

Gradient descent for linear regression

Learning algorithm:

(1) Initialize weights $\mathbf{w} = \mathbf{0}$

(2) For $t=1, \dots$ until convergence:

(2-1) Predict for each example \mathbf{x}^i using \mathbf{w} : $\hat{y}^i = \sum_{j=0}^k w_j \phi_j(\mathbf{x}^i)$

(2-2) Compute gradient of loss: $\frac{\partial}{\partial w_j} J(\mathbf{w}) = -2 \sum_i (y^i - \hat{y}^i) \phi_j(\mathbf{x}^i)$
*This is a vector \mathbf{g}

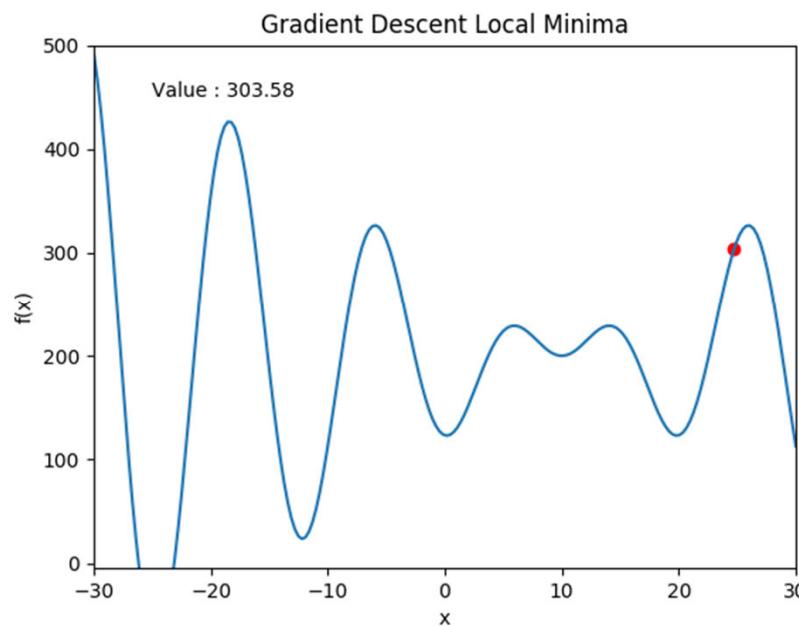
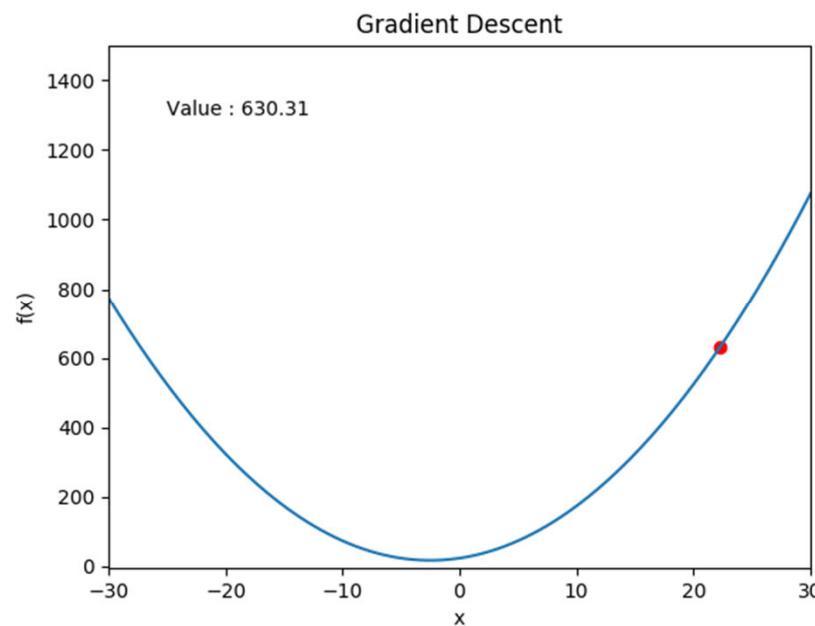
(2-3) Update: $\mathbf{w} = \mathbf{w} - \alpha \mathbf{g}$, α is the learning rate

$$\mathbf{w} = \mathbf{w} - \alpha \nabla J(\mathbf{w})$$

(bowl shaped)

Linear regression is a convex optimization problem

gradient descent can reach a *global* optimum



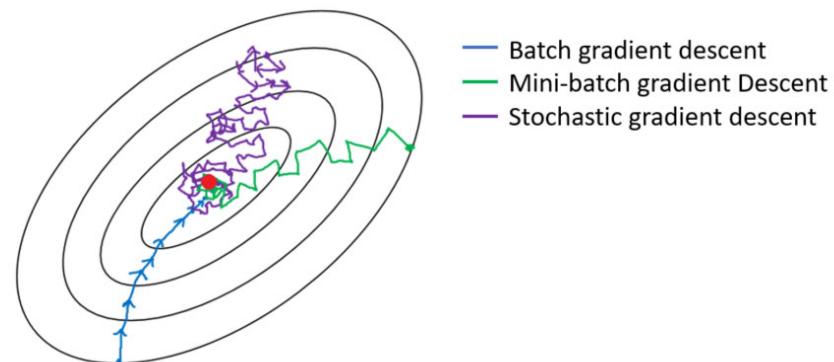
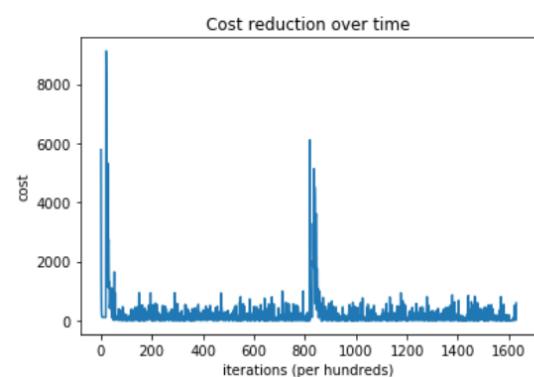
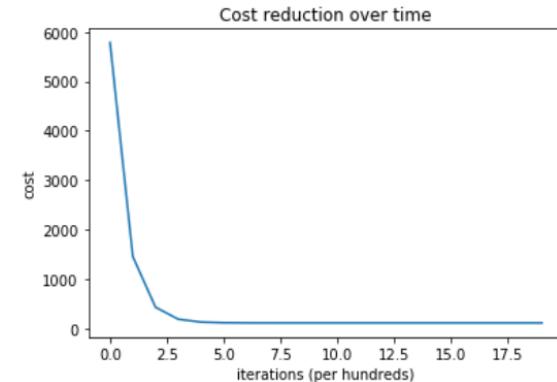
(<https://github.com/Shathra/gradient-descent-demonstration>)

Non-convex optimization

- NP-hard in general
- Deep learning

Stochastic gradient descent

- What we just see is “Batch” gradient decent
 - The gradient will be computed using the whole training data set
 - Since it uses the whole dataset at every iteration, hence, it makes the computation very, very slow especially when the data set is very large.
- Idea: rather than using the full gradient, just use one training example
 - Fast to compute $w = w - \alpha \nabla J(w; y_i)$



OLS versus gradient descent

$$J(\mathbf{w}) = \sum_i (y^i - \mathbf{w}^T \phi(x^i))^2 \quad \mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$$

Ordinary Least Squares (OLS) solution:

- + Very simple in programming
- Requires matrix inverse, which is expensive for a large matrix.

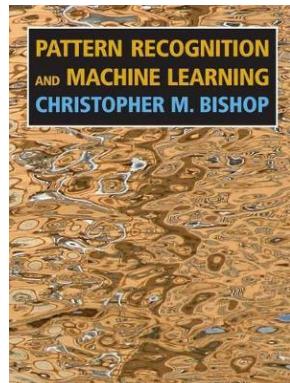
Gradient descent:

- + Fast for large matrices
- + Stochastic GD is very memory efficient
- + Easily extended to other cases
- Parameters to tune (how to decide convergence? what is the learning rate?)

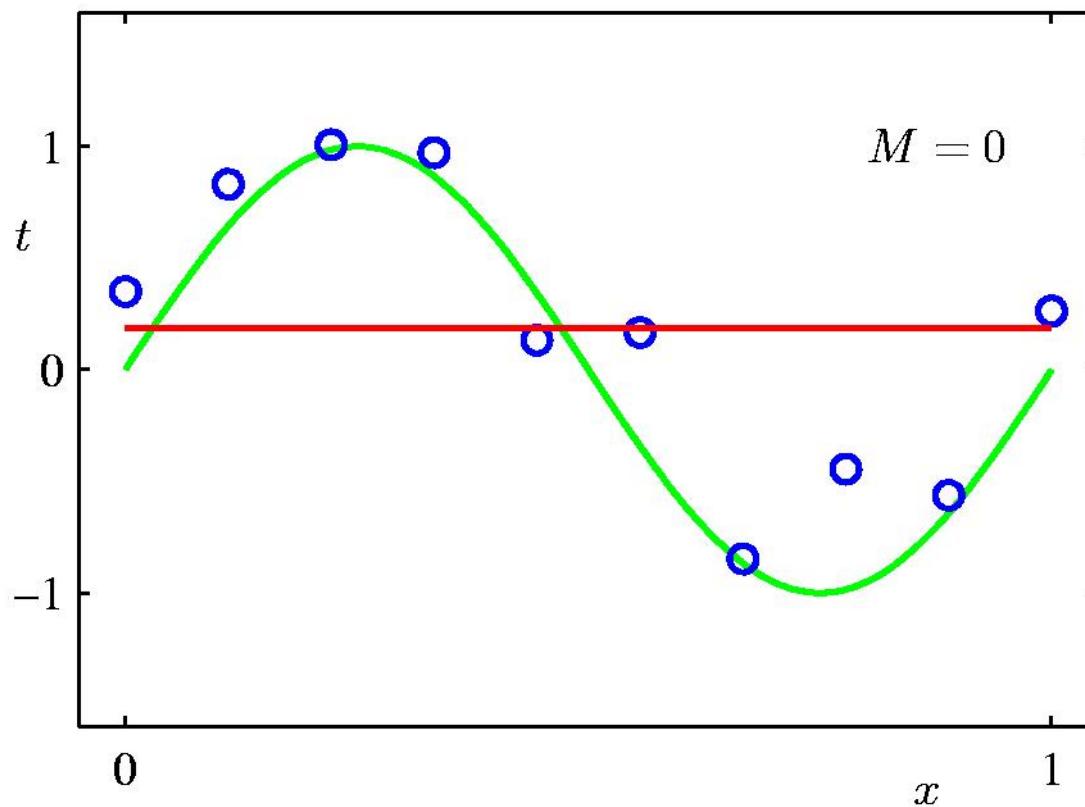
Overfitting and Regularization

An example: polynomial basis vectors on a small dataset

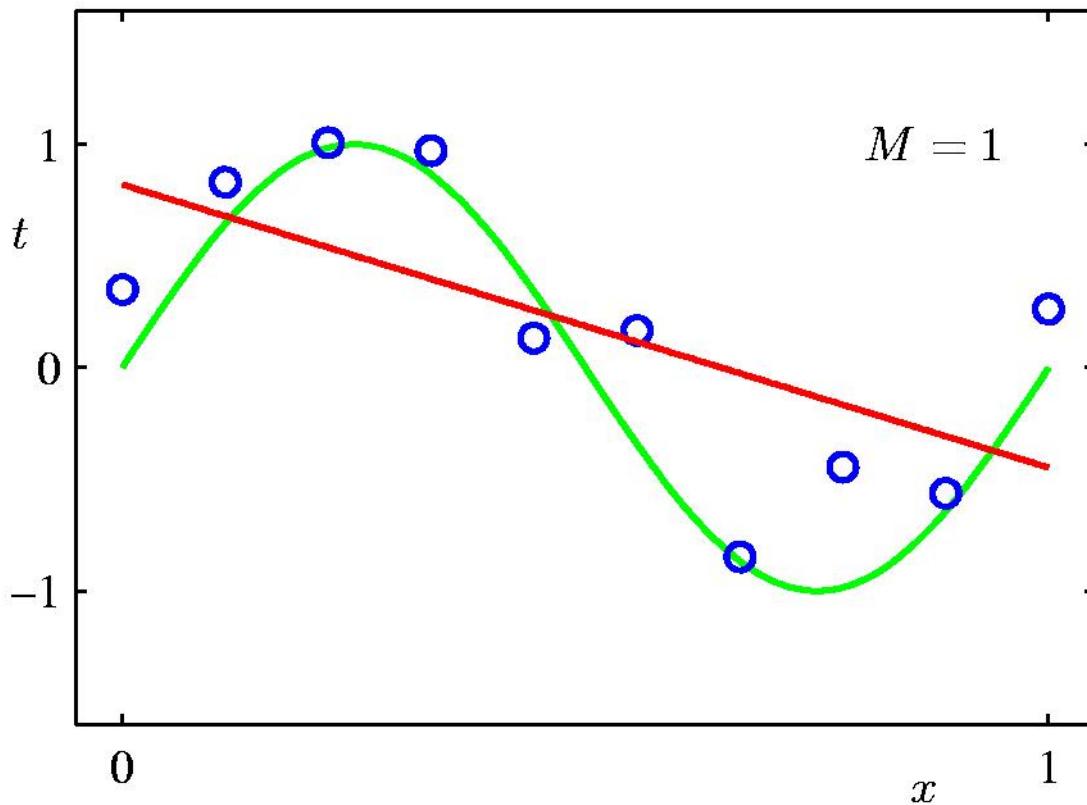
(From Bishop Ch 1)



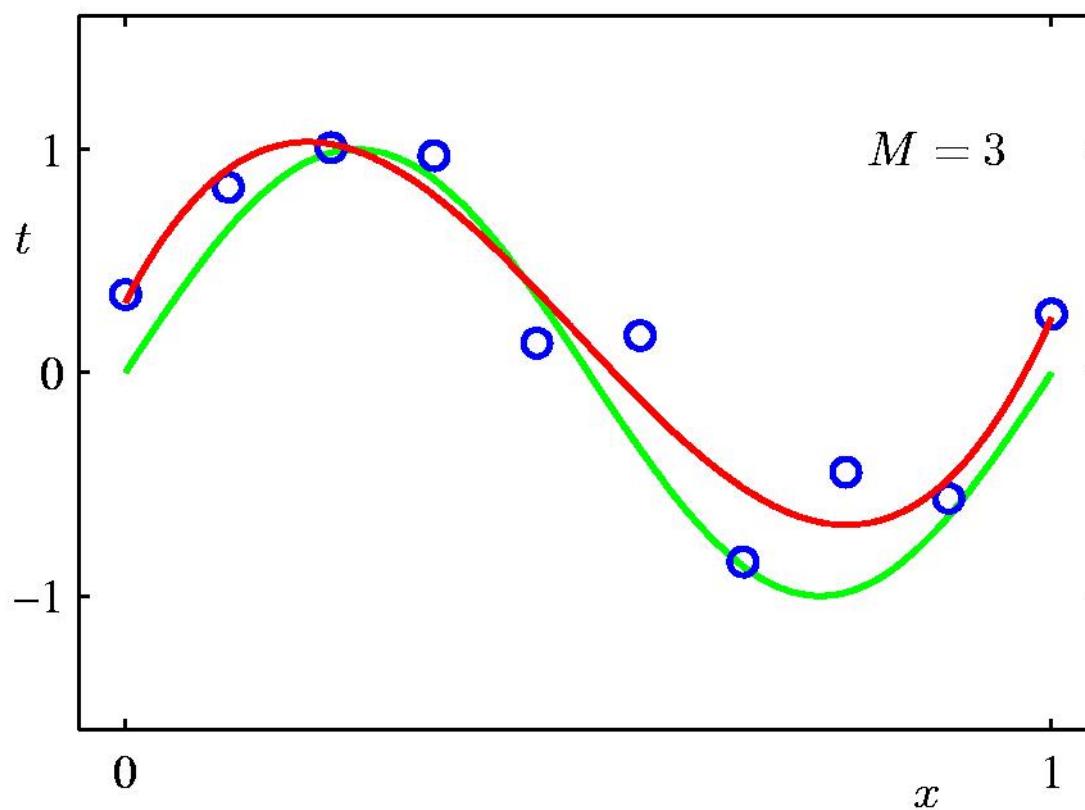
0th Order Polynomial



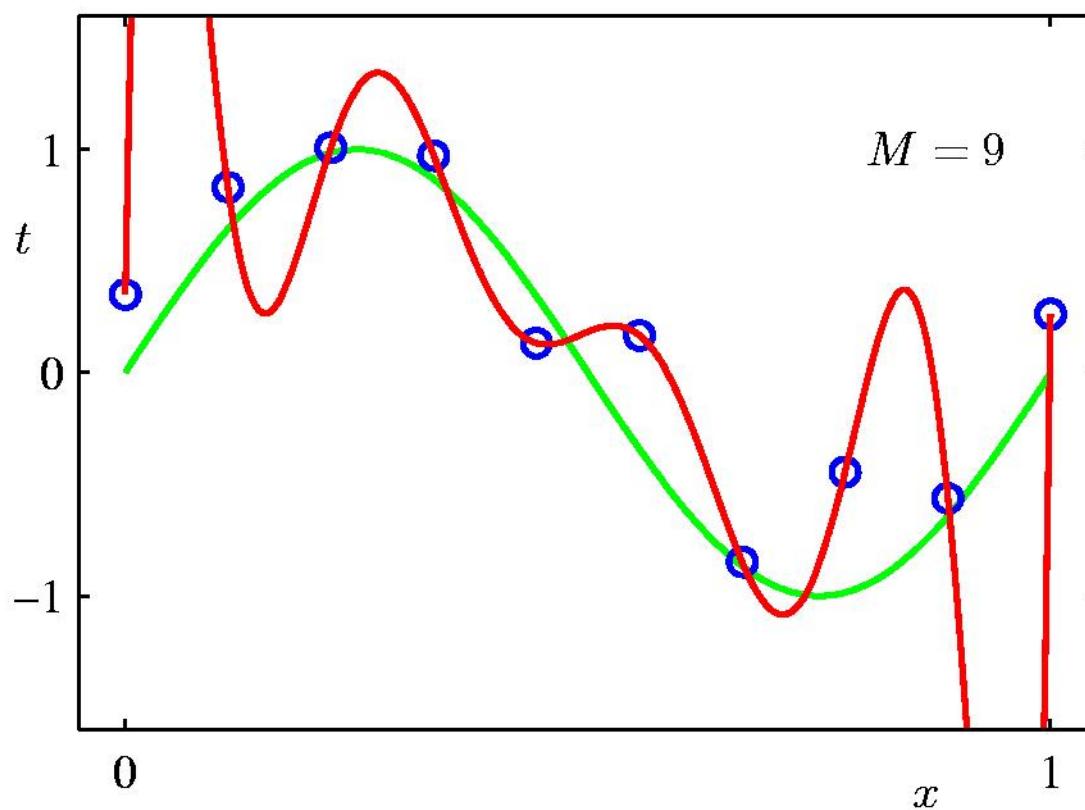
1st Order Polynomial



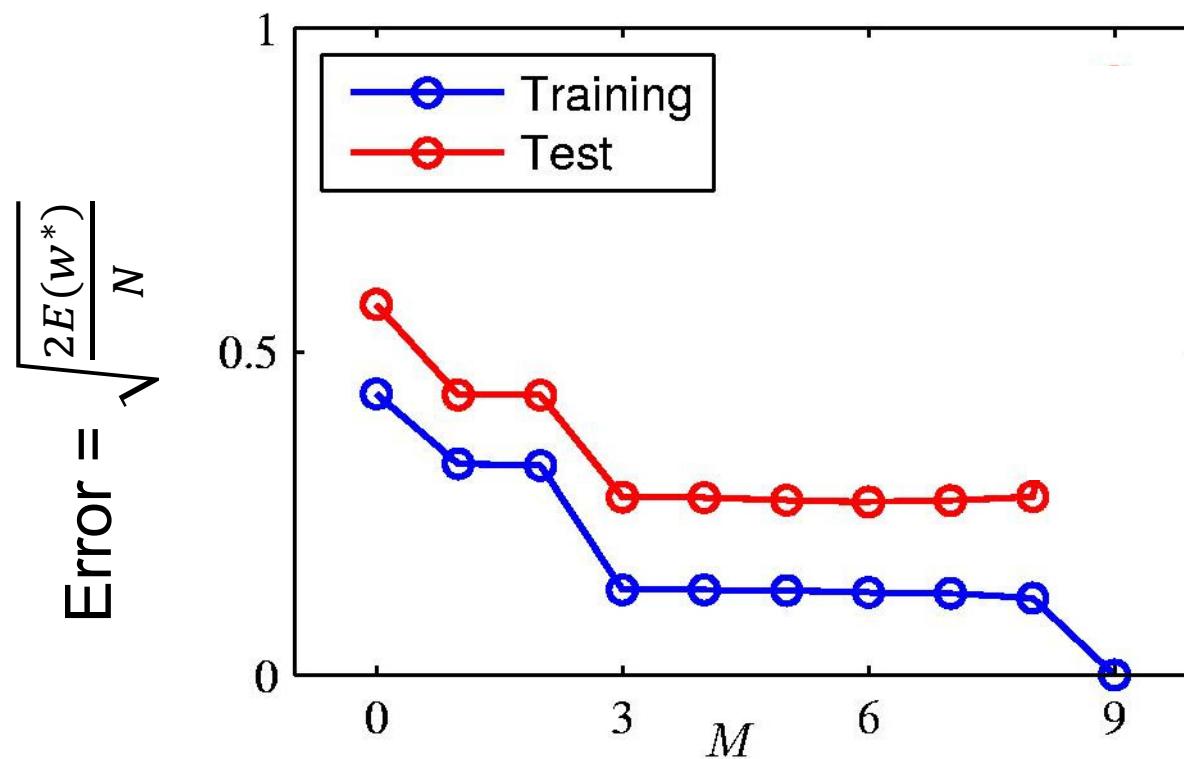
3rd Order Polynomial



9th Order Polynomial



Over-fitting



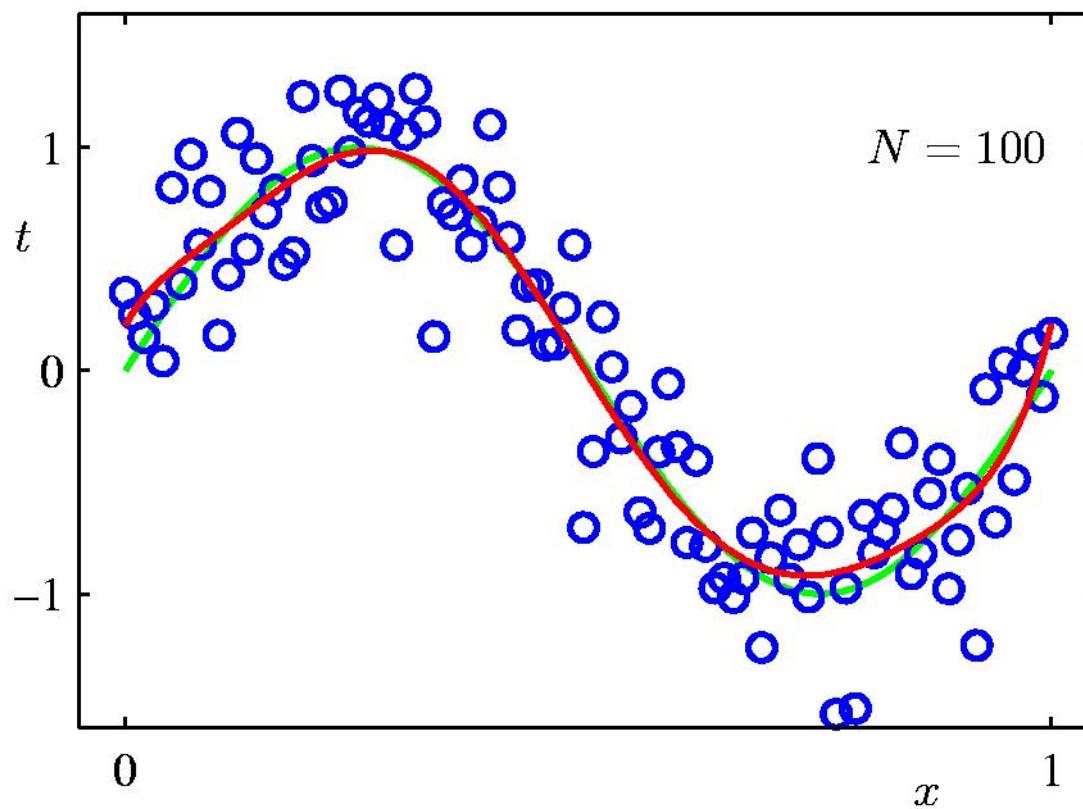
Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

Data Set Size:

9th Order Polynomial

$N = 100$



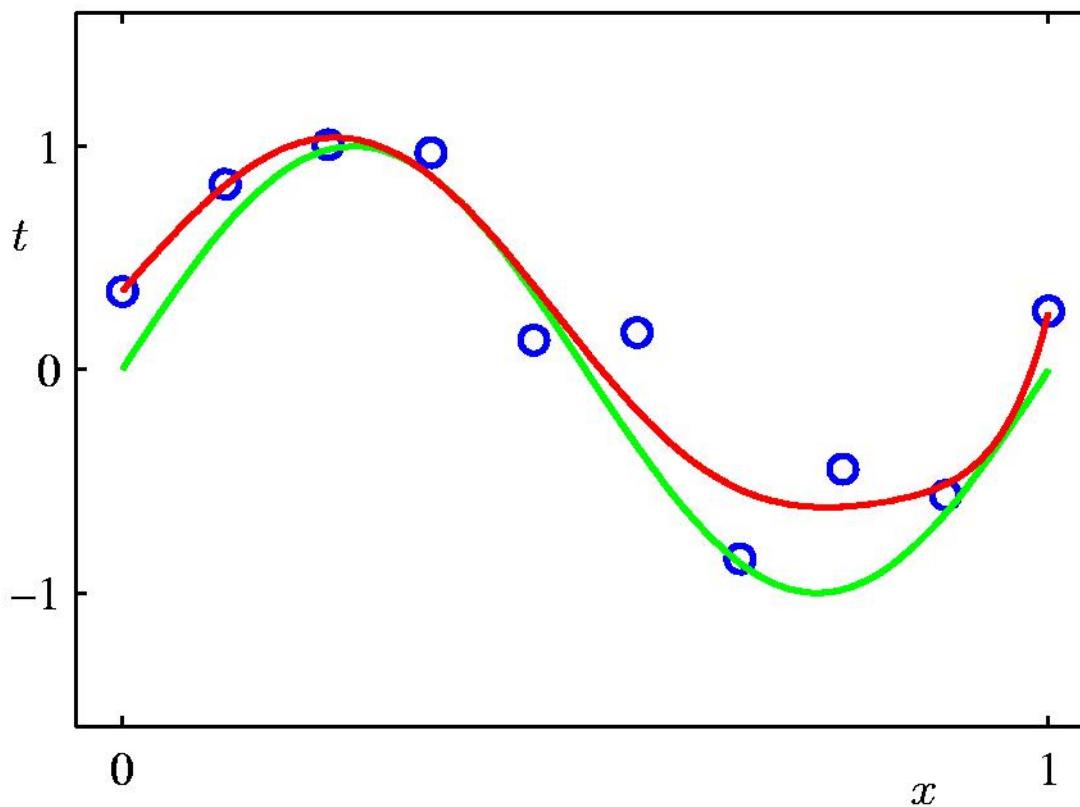
Regularization

Penalize large coefficient values

$$J_{\mathbf{x}, \mathbf{y}}(\mathbf{w}) = \frac{1}{2} \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

Regularization:

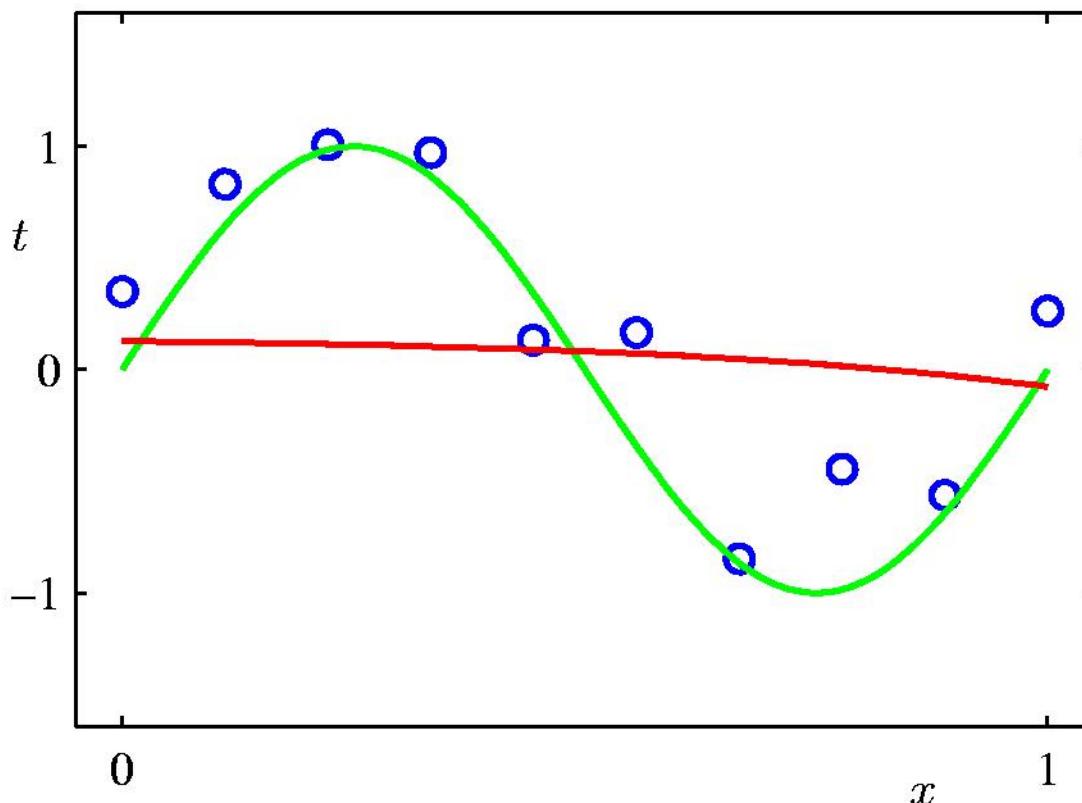
$$\ln \lambda = -18$$



Polynomial Coefficients

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232.37	4.74	-0.05
w_2^*	-5321.83	-0.77	-0.06
w_3^*	48568.31	-31.97	-0.05
w_4^*	-231639.30	-3.89	-0.03
w_5^*	640042.26	55.28	-0.02
w_6^*	-1061800.52	41.32	-0.01
w_7^*	1042400.18	-45.95	-0.00
w_8^*	-557682.99	-91.53	0.00
w_9^*	125201.43	72.68	0.01

Over Regularization:



Example in Google colab notebook:

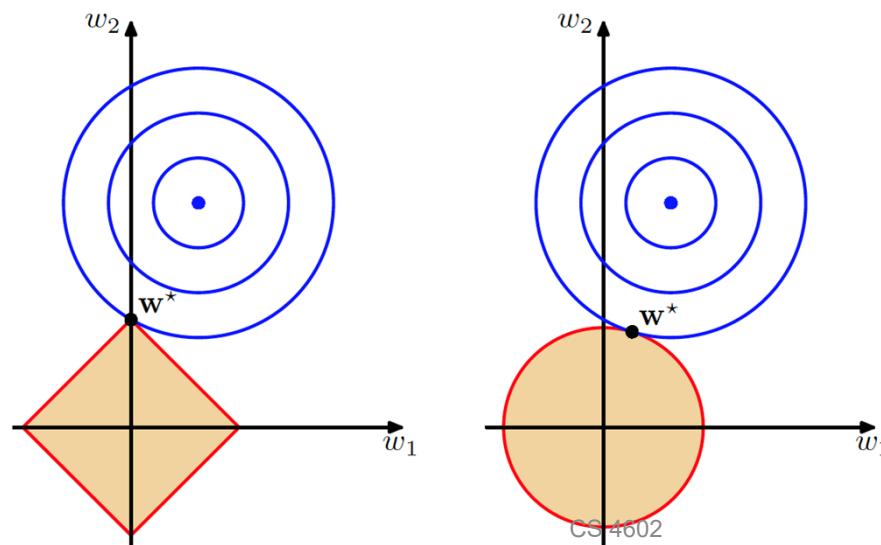
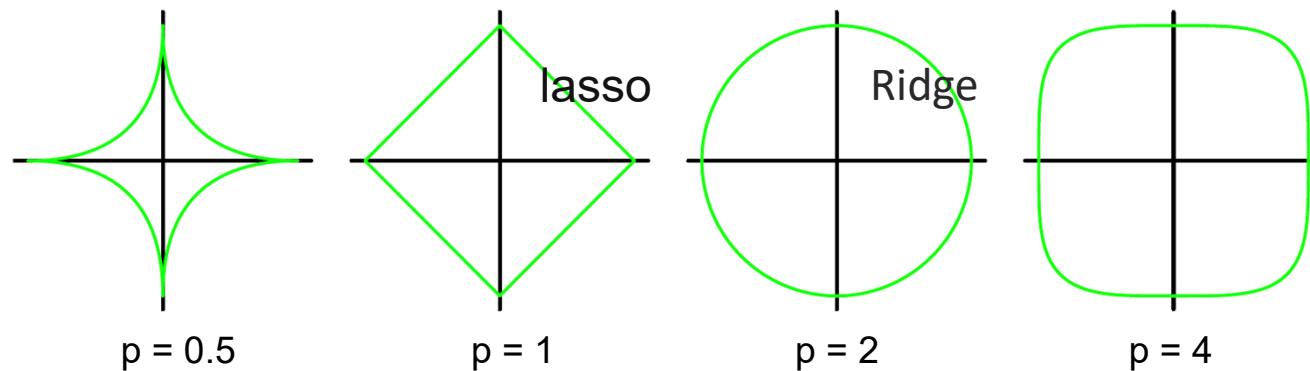
<https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.06-Linear-Regression.ipynb#scrollTo=xW7yPbeMBvwy>

$$\|x\|_p = \left(\sum_i |x_i|^p \right)^{1/p}$$

$$J(\mathbf{w}) = (\sum_i y^i - \mathbf{w}^\top \mathbf{x}^i)^2 + \lambda \|\mathbf{w}\|_p$$

$$\|x\|_2 = \sqrt{\left(\sum_i x_i^2 \right)} = \sqrt{x_1^2 + x_2^2 + \dots + x_i^2}$$

$$\|x\|_1 = \sum_i |x_i| = |x_1| + |x_2| + \dots + |x_i|$$



- What if we are going to forecast the future?

Autoregressive (AR) Modeling

- Used for forecasting
- Takes Advantage of Autocorrelation
 - 1st order - correlation between consecutive values
 - 2nd order - correlation between values 2 periods apart
- Autoregressive Model for p -th order:

$$Y_i = A_0 + A_1 Y_{i-1} + A_2 Y_{i-2} + \dots + A_p Y_{i-p} + \delta_i$$

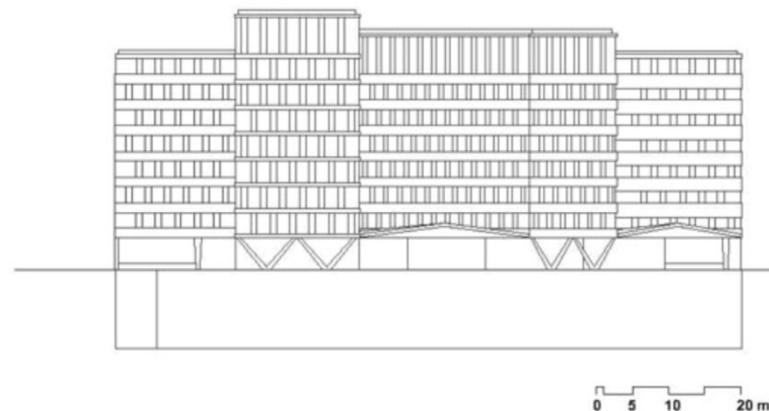
Random Error



Autoregressive Model: Example

The Office Concept Corp. has acquired a number of office units (in thousands of square feet) over the last 8 years. Develop the 2nd order Autoregressive models.

Year	Units
92	4
93	3
94	2
95	3
96	2
97	2
98	4
99	6



Autoregressive Model: Example Solution

- Develop the 2nd order table
- Run a regression model

	Coefficients
Intercept	3.5
X Variable 1	0.8125
X Variable 2	-0.9375

Year	Y_i	Y_{i-1}	Y_{i-2}
92	4	---	---
93	3	4	---
94	2	3	4
95	3	2	3
96	2	3	2
97	2	2	3
98	4	2	2
99	6	4	2
100	?	6	4

$$Y_i = 3.5 + 0.8125 Y_{i-1} - 0.9375 Y_{i-2}$$

$$Y_{100} = 3.5 + 0.8125 \times 6 - 0.9375 \times 4 = 4.625$$

AR family

- The Autoregressive Moving Average (ARMA) Model
 - The ARMA (p,q) model is a combination of AR (autoregressive) and MA (moving average) terms.
- The Autoregressive Integrated Moving Average (ARIMA) Model
 - The BJ methodology is based on the assumption that the underlying time series is stationary or can be made stationary by differencing it one or more times.
- AR + NN

***AR-Net: A SIMPLE AUTO-REGRESSIVE NEURAL NETWORK
FOR TIME-SERIES***

A PREPRINT

Oskar J. Tribe
Stanford University
tribebe@stanford.edu

Nikolay Laptev
Facebook, Inc.
nlaptev@fb.com

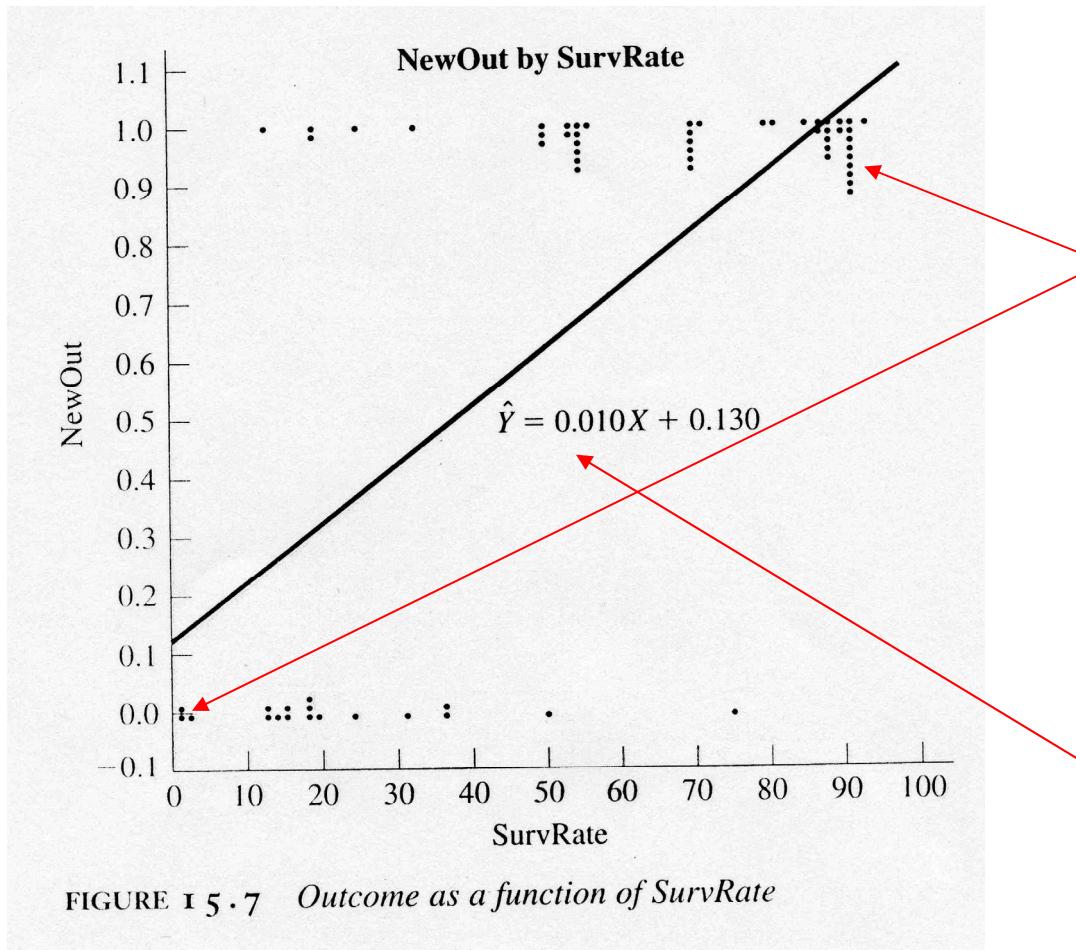
Ram Rajagopal
Stanford University
ramr@stanford.edu

December 2, 2019

Classification by regression?

- Linear regression is not suitable for classification problem.

Example



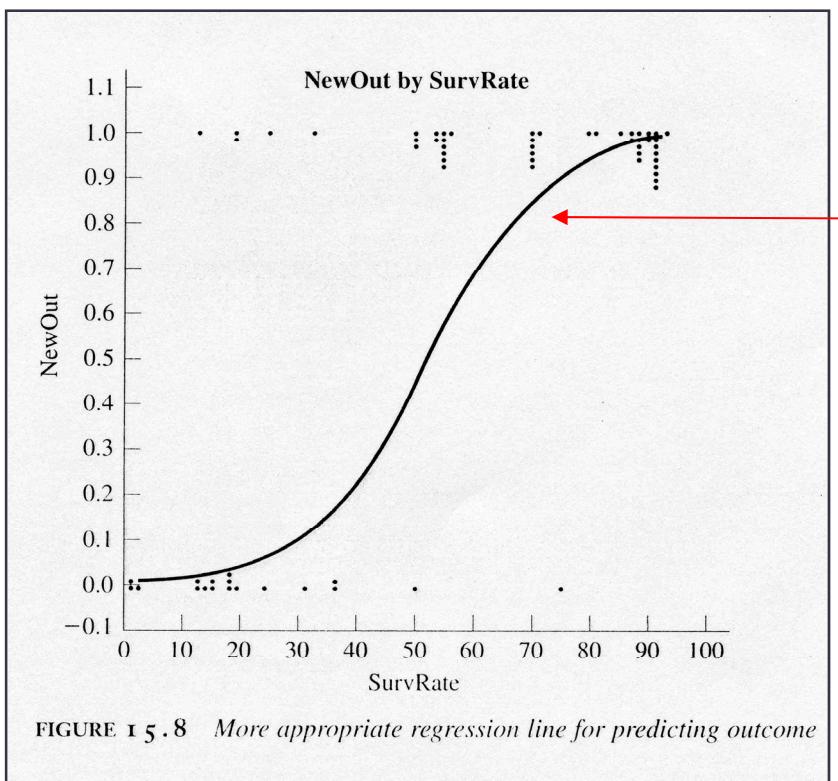
Observations:
For each value of SurvRate, the number of dots is the number of patients with that value of NewOut

Regression:
Standard linear regression

Problem: extending the regression line a few units left or right along the X axis produces predicted probabilities that fall outside of [0,1]

Logistic Regression: A better solution!

- To fit a curve to data in which the dependent variable is binary
- Typical application: Medicine
 - We might want to predict response to treatment, where we might code survivors as 1 and those who don't survive as 0



Regression Curve: Sigmoid function!
(Logistic function)
(bounded by asymptotes $y=0$ and $y=1$)

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

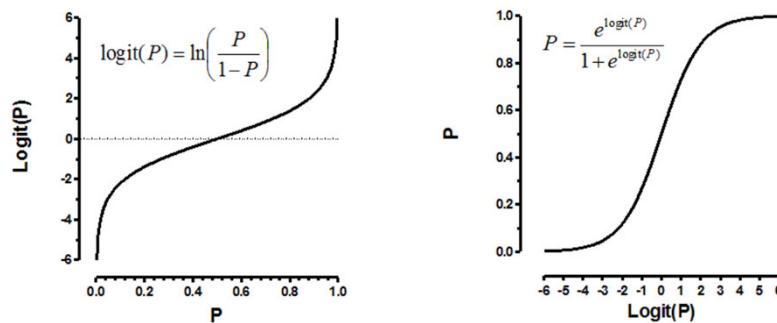
Logistic Regression

- In logistic regression, we seek a model:

$$\text{logit}(p) = \beta_0 + \beta_1 X$$

Y

- That is, **the log odds (logit)** is assumed to be linearly related to the independent variable X
- So, now we can focus on solving an ordinary (linear) regression!



Odds

- Given some event with probability p of being 1, the odds of that event are given by: $\text{odds} = p / (1-p)$
- Consider the following data

Testosterone	Delinquent		
	Yes	No	Total
Normal	402	3614	4016
High	101	345	446
	503	3959	4462

- The odds of being delinquent if you are in the Normal group are:

$$p_{\text{delinquent}} / (1 - p_{\text{delinquent}}) = (402/4016) / (1 - (402/4016)) = 0.1001 / 0.8889 = 0.111$$

Odds Ratio

- The odds of being not delinquent in the Normal group is the reciprocal of this:
 - $0.8999/0.1001 = 8.99$
- Now, for the High testosterone group
 - $\text{odds(delinquent)} = 101/345 = 0.293$
 - $\text{odds(not delinquent)} = 345/101 = 3.416$
- When we go from Normal to High, the odds of being delinquent nearly triple:
 - **Odds ratio:** $0.293/0.111 = 2.64$
 - 2.64 times more likely to be delinquent with high testosterone levels

$\text{logit}(p) = \ln(\text{odds})$


$$\text{logit}(p) = \beta_0 + \beta_1 x$$
$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

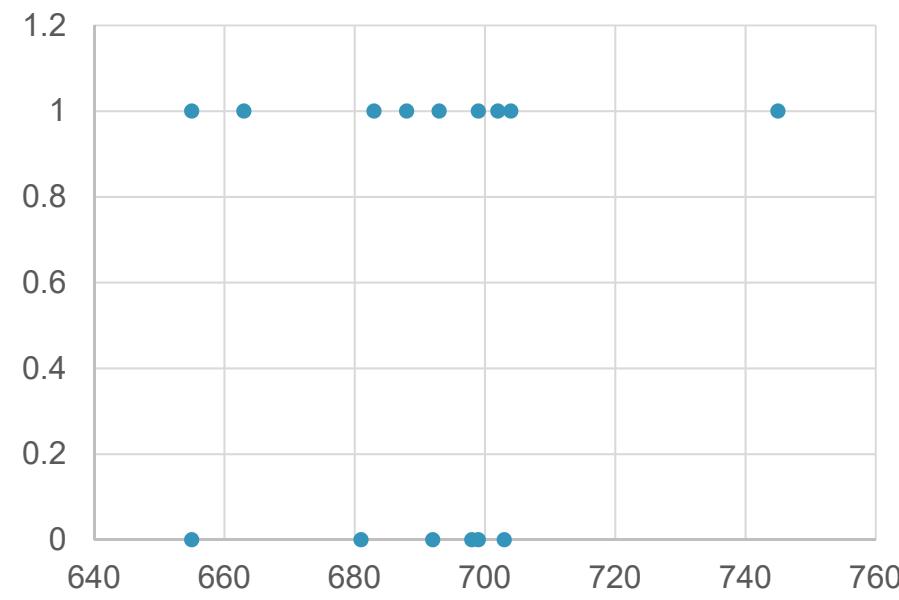
$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 x}$$

$$p = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

An example- Home Mortgage Loan

Credit Score (FICOscore)	Approved
655	0
692	0
681	0
663	1
688	1
693	1
699	0
699	1
683	1
698	0
655	1
703	0
704	1
745	1
702	1

Approved = 1
Not approved = 0



Binary Logistic Regression: Approve versus FICOscore

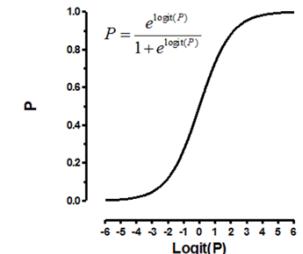
Coefficients

Term	Coef	SE Coef	95% CI	Z-Value	P-Value	VIF
Constant	-9.346	0.637	(-10.594, -8.097)	-14.67	0.000	
FICOscore	0.014634	0.000940	(0.012791, 0.016476)	15.56	0.000	1.00

$$\text{Sigmoid function: } S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

Regression equation:

$$\text{Estimated probability of being approved } \hat{p} = \frac{e^{\beta_0 + \beta_1 x_1}}{1 + e^{\beta_0 + \beta_1 x_1}}$$



FICOscore

$$\hat{p} = \frac{e^{-9.346 + 0.014634x_1}}{1 + e^{-9.346 + 0.014634x_1}}$$

Credit Score (FICOscore)	Approved
655	0
692	0
681	0
663	1
688	1
693	1
699	0
699	1
683	1
698	0
655	1
703	0
704	1
745	1
702	1

→ $\hat{p} = \frac{e^{-9.346+0.014634(655)}}{1+e^{-9.346+0.014634(655)}} = \frac{1.270321477}{2.270321477} = 0.559533744$

$$odds = \frac{0.559533744}{1 - 0.559533744}$$

Odds ratio for 655 to 745 = $\frac{\frac{0.825826101}{1-0.825826101}}{\frac{0.559533744}{1-0.559533744}} = \frac{4.74138838}{1.270321477}$
 $= 3.732431881$

→ $\hat{p} = \frac{e^{-9.346+0.014634(745)}}{1+e^{-9.346+0.014634(745)}} = \frac{4.74138838}{5.74138838} = 0.825826101$

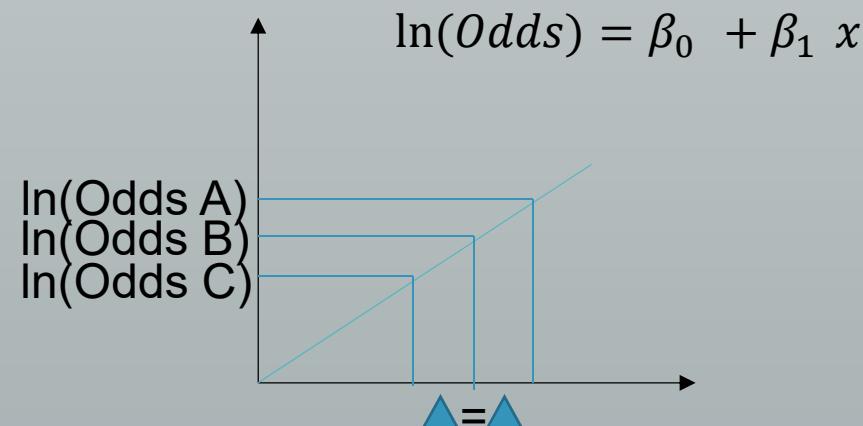
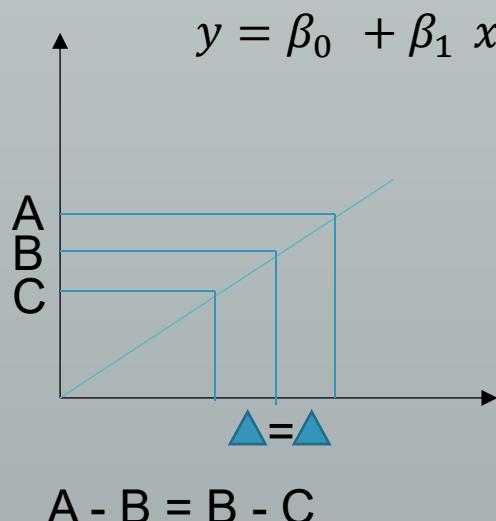
$$odds = \frac{0.825826101}{1 - 0.825826101}$$

$$\text{Odds ratio for 655 to 745} = \frac{\frac{0.825826101}{1-0.825826101}}{\frac{0.559533744}{1-0.559533744}} = \frac{4.74138838}{1.270321477}$$

$$= 3.732431881$$

$$\text{Odds ratio for 600 to 690} = \frac{\frac{0.679496743}{1-0.679496743}}{\frac{0.362252721}{1-0.362252721}} = \frac{2.120093094}{0.568019233}$$

$$= 3.732431881$$



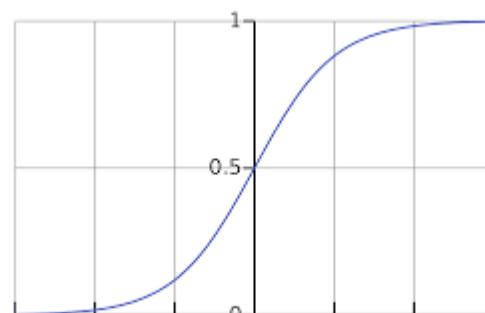
FICO	\hat{p}	$1 - \hat{p}$	ODDS
600	0.362253	0.637747	0.568019
610	0.396694	0.603306	0.657533
620	0.43219	0.56781	0.761154
630	0.468397	0.531603	0.881104
640	0.50494	0.49506	1.019957
650	0.54143	0.45857	1.180691
660	0.577481	0.422519	1.366756
670	0.612725	0.387275	1.582143
680	0.646827	0.353173	1.831472
690	0.679497	0.320503	2.120093
700	0.710497	0.289503	2.454198

$$+0 = \frac{0.568019}{0.568019} = 1$$

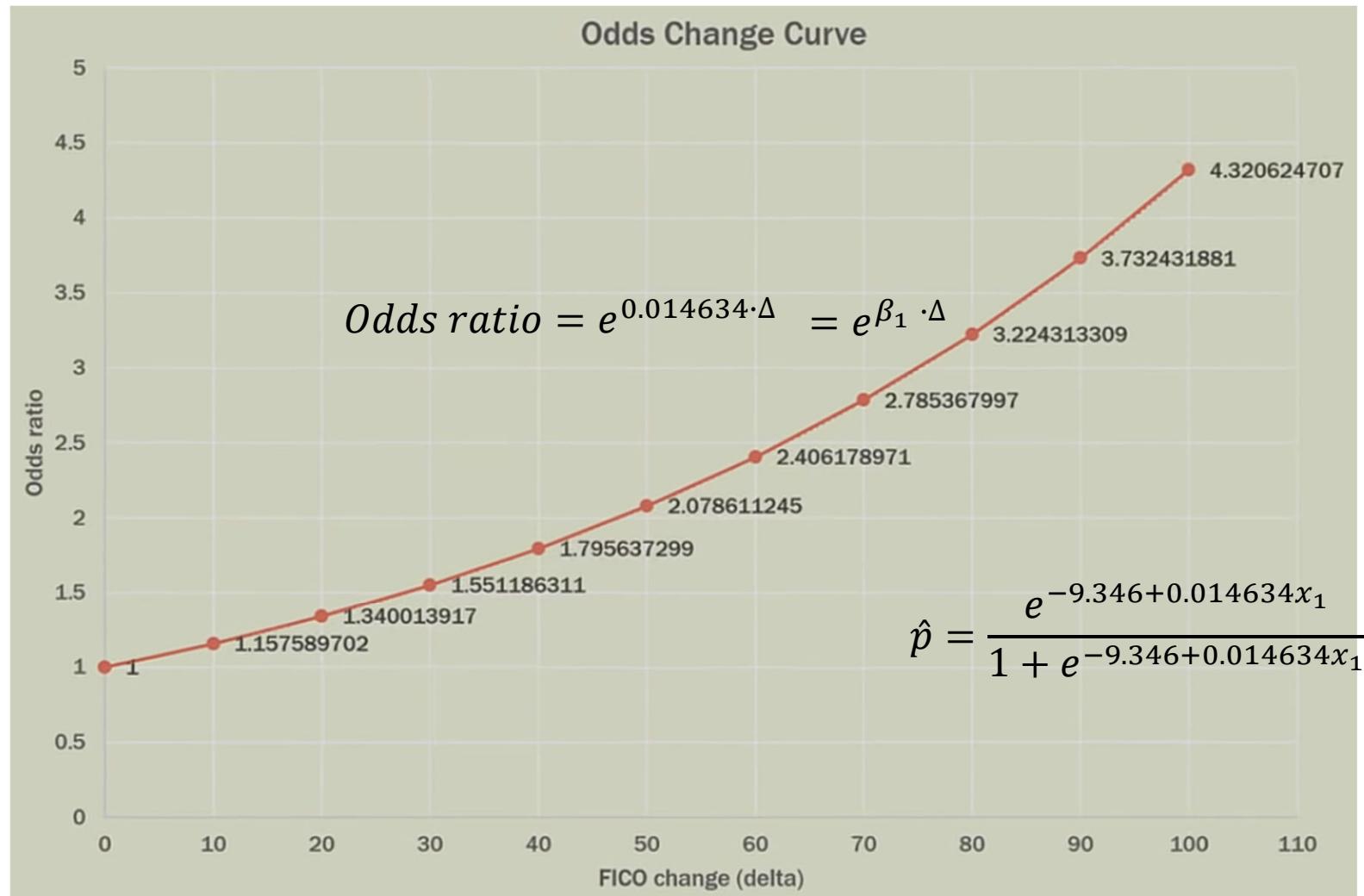
$$+10 = \frac{0.657533}{0.568019} = 1.157589702$$

$$+20 = \frac{0.761154}{0.568019} = 1.340013917$$

$$+30 = \frac{0.881104}{0.568019} = 1.551187548$$



630~640



Binary Logistic Regression: Approve versus FICOScore

Coefficients							
Term	Coef	SE Coef	95% CI	Z-Value	P-Value	VIF	
Constant	-9.346	CS 400637	(-10.594, -8.097)	-14.67	0.000		
FICOScore	0.014634	0.000940	(0.012791, 0.016476)	15.56	0.000	1.00	71

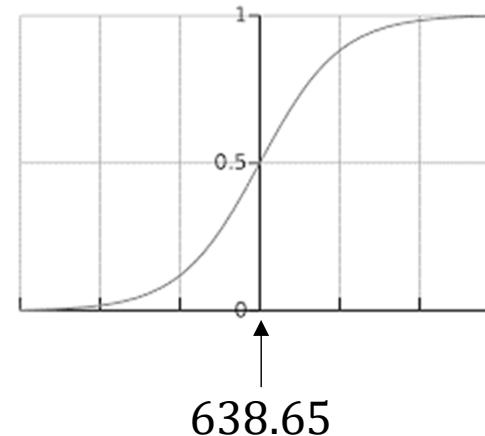
FICO for even odds (50/50)

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1$$

$$\ln\left(\frac{.5}{.5}\right) = -9.346 + 0.014634x_1$$

$$0 = -9.346 + 0.014634x_1$$

$$x_1 = 638.65$$



To have even odds for approve/disapprove, you will need to have FICO score = 639

FICO for even odds (50/50)

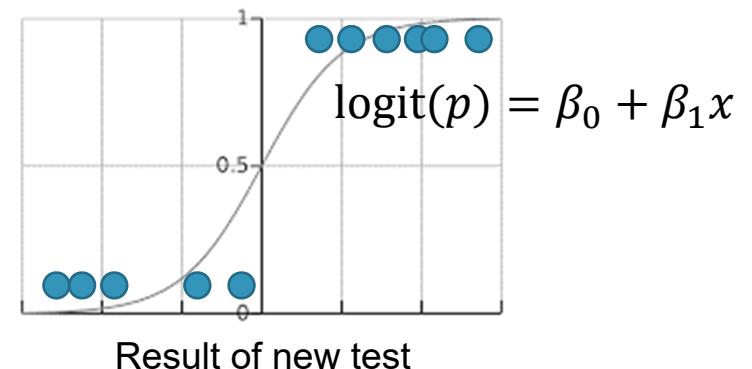
$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1$$

$$\beta_0 = -1, \beta_1 = 0.5$$

$$\ln\left(\frac{.5}{.5}\right) = -9.346 + 0.014634x_1$$

$$0 = -9.346 + 0.014634x_1$$

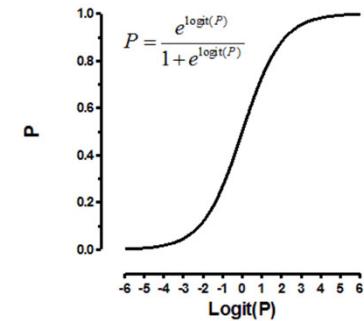
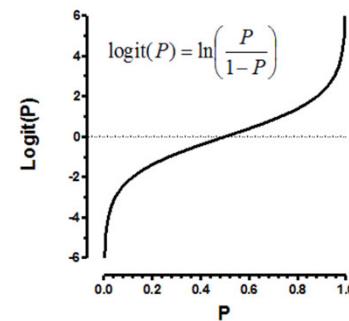
$$x_1 = 638.65$$



To have even odds for approve/disapprove, you will need to have FICO score = 639

Logit Transform

- p is the probability that the event Y occurs, $p(Y=1)$
 - [range=0 to 1]
- $p/(1-p)$ is the "odds"
 - [range=0 to ∞]
- $\ln[p/(1-p)]$: log odds, or "logit"
 - [range=- ∞ to + ∞]
- $\text{logit}(p) = \ln(\text{odds}) = \ln(p/(1-p))$

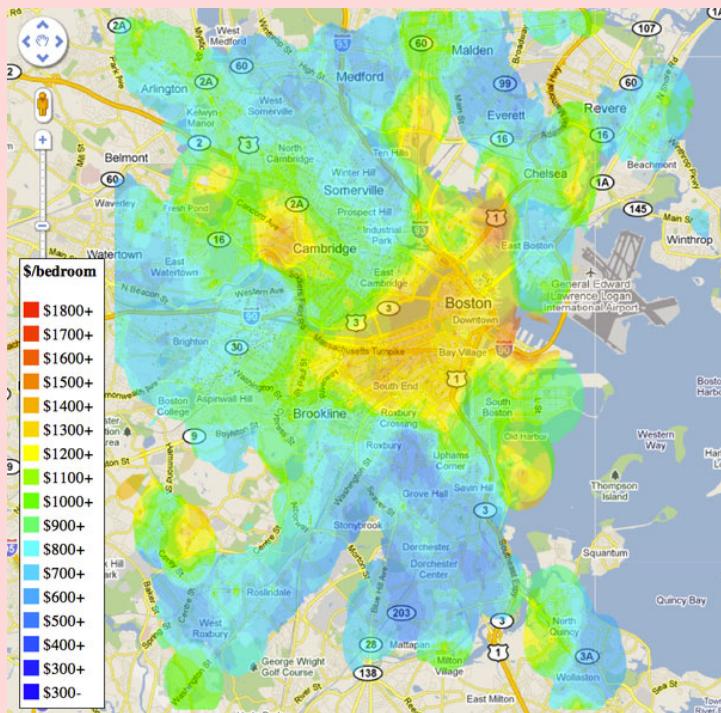


Logistic Regression: $\text{logit}(p) = \beta_0 + \beta_1 X$

We've learned

- Regression vs Classification
- Linear regression – another discriminative learning method
 - As matrix inversion (Ordinary Least Squares)
 - As optimization: Gradient descent
 - Batch gradient decent
 - Stochastic gradient descent
- Overfitting and regularization
- Autoregression
- Logistic regression (more like classification)

Boston Housing Price Prediction



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

CRIM per capita crime rate by town

ZN proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS proportion of non-retail business acres per town

CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

NOX nitric oxides concentration (parts per 10 million)

RM average number of rooms per dwelling

AGE proportion of owner-occupied units built prior to 1940

DIS weighted distances to five Boston employment centres

RAD index of accessibility to radial highways

TAX full-value property-tax rate per 10,000usd

PTRATIO pupil-teacher ratio by town

B $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town

LSTAT % lower status of the population

The screenshot shows a Kaggle notebook interface. On the left is a sidebar with navigation links: Create, Home, Competitions, Datasets, Models, Code, Discussions, Learn, More, Your Work, and Recently Viewed (which includes 'Boston house price ...', 'Pneumonia Detectio...', and 'Chest X-Ray Images...'). The main content area has a search bar at the top. Below it, a profile picture of the author, SHREAYAN CHAUDHARY, is shown with the text '4Y AGO · 102,048 VIEWS'. To the right are buttons for 'Edit My Copy' (highlighted in black) and '1227'. Below this, the title 'Boston house price prediction' is displayed in bold. Underneath the title, it says 'Python · Boston House Prices'. A horizontal navigation bar below the title includes tabs for Notebook, Input, Output, Logs, and Comments (18), with 'Notebook' being the active tab. Below the tabs, there's a 'Run' section showing '15.8s'. To the right of the run time is a button for 'Version 1 of 1'. Further down, there are three category tags: Beginner, Exploratory Data Analysis, and Regression. The main content area starts with a section titled 'Boston house price prediction' which describes the task of predicting house prices from a set of features. To the right of this content is a 'Table of Contents' sidebar with sections for 'Boston house price prediction' (which is expanded), 'Linear regression', and 'Random Forest Regressor'. There are also collapsed sections for 'Exploratory Data Analysis' and 'Conclusion'.

<https://www.kaggle.com/code/shreayan98c/boston-house-price-prediction>

Questions?

