

Regression

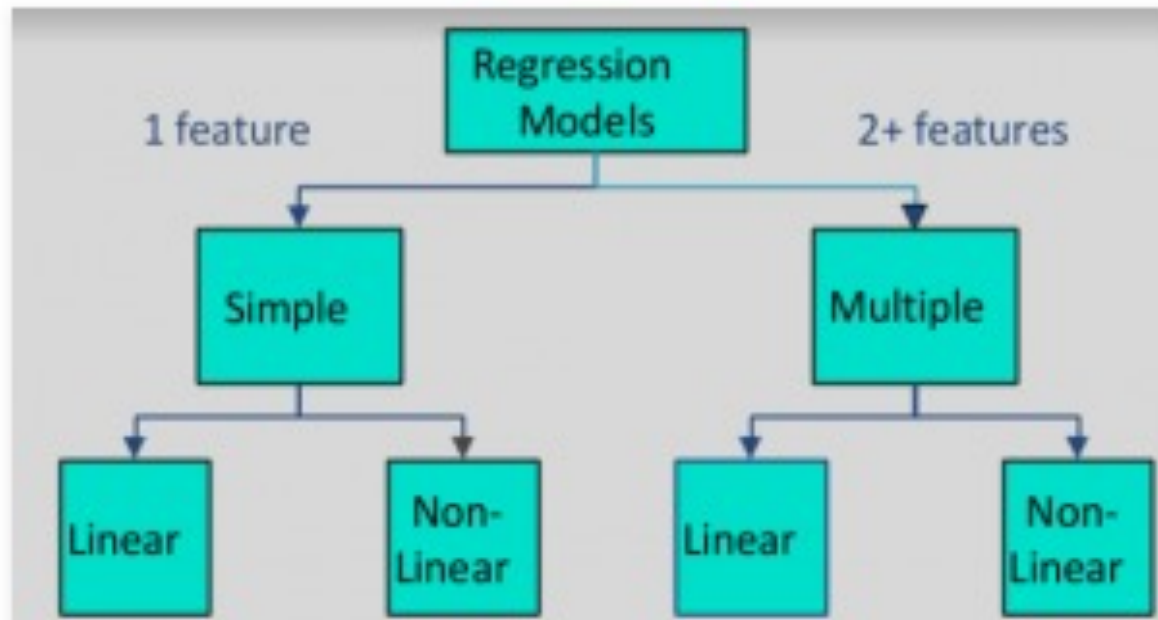


WQD7006
Machine Learning
Vimala Balakrishnan

Regression

- Regression is a method of modelling a target value based on independent predictors.
- Mostly used for forecasting and finding out cause and effect relationship between variables.
- Logistic regression (classification) versus linear regression (predict/forecast)
- **Classification** - seeks to identify the categorical class C associated with a given input vector x .
- **Regression** - seeks to identify (or estimate) a continuous variable y associated with a given input vector x .

Regression



Checkpoint

- **Which of the following is a regression task?**
- Predicting age of a person
- Predicting nationality of a person
- Predicting whether stock price of a company will increase tomorrow
- Predicting whether a document is related to sighting of UFOs?

Checkpoint

- **Which of the following is/are classification problem(s)?**
- Predicting the gender of a person by his/her handwriting style
- Predicting house price based on area
- Predicting whether monsoon will be normal next year
- Predict the number of copies a music album will be sold next month

Linear regression

Stock market



Weather prediction

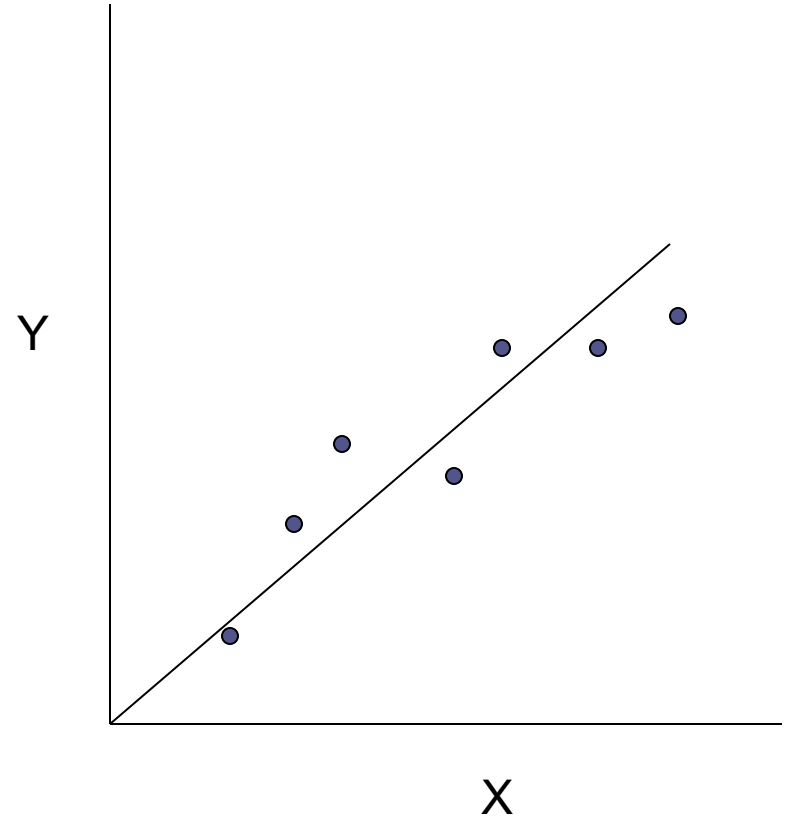


Temperature
72° F

Predict the temperature at any given location

Linear regression

- Given an input \mathbf{x} we would like to compute an output \mathbf{y}
- For example:
 - Predict height from age
 - Predict Google's price from Yahoo's price
 - Predict distance from wall from sensors



Simple Linear regression

- Assume that y and x are related with the following equation:

$$y = wx + \varepsilon$$

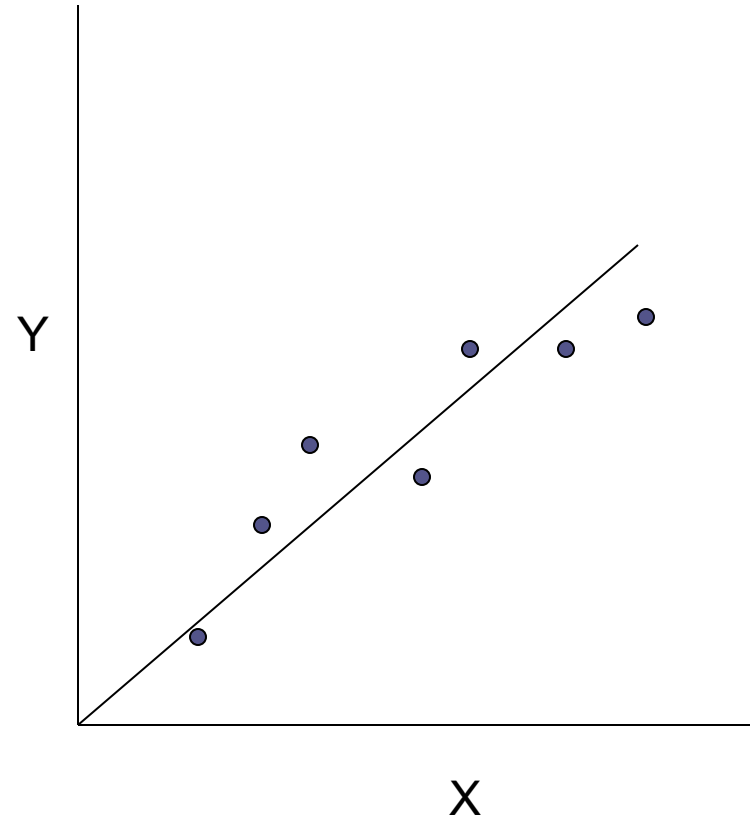
Dependent

What we are
trying to predict

Independent

Observed values

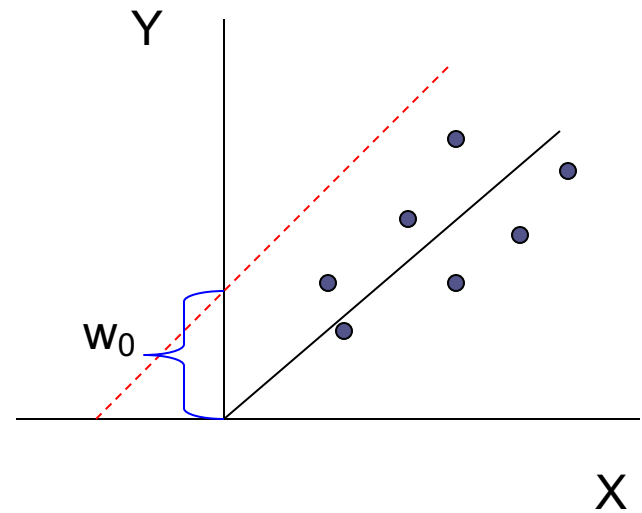
where w is a parameter and ε represents measurement or other noise



Assume starting at 0

Bias coefficient/intercept

- $y = w_0 + w_1x + \varepsilon$
- Can use least squares to determine w_0 , w_1

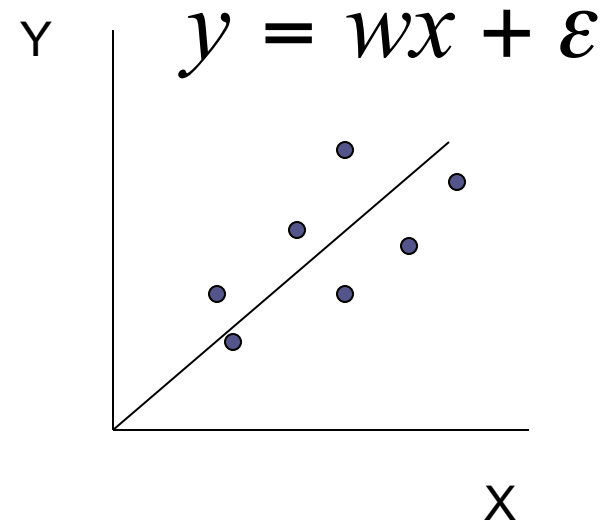


Linear regression

- 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):

$$\arg \min_w \sum_i (y_i - wx_i)^2$$

- Least squares -
 - minimizes squared distance between measurements and predicted line





Girls Love Fearlessly

Linear Regression

Three steps:

1. **G**uess – hypothesis, $h(x)$ or $(y = w_0 + w_1x_1 + \dots + w_kx_k)$

2. Calculate cost function (**L**oss) – “how wrong our randomly selected weights are” – Least Square

3. **F**ix mistake – optimization (Gradient Descent)

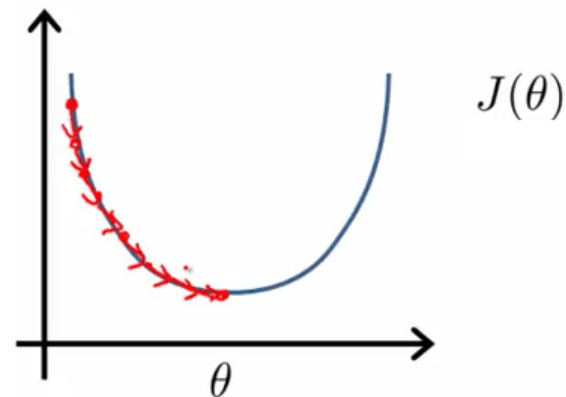
Cost Function

Sum of squared error over all training samples

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

... and minimize it:

$$\min_{\theta} J(\theta)$$



Fixing

Update each slope parameter until loss function = minimum

$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta)$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta)$$

...

$$\theta_n = \theta_n - \alpha \frac{\partial}{\partial \theta_n} J(\theta)$$

Simultaneously

Alpha

Learning rate

Derivative

Direction of moving

Multiple linear regression

- What if we have several inputs?
 - Stock prices for Yahoo, Microsoft and Ebay for the Google prediction task

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

stock price



Yahoo's stock price

Microsoft's stock price

Optimization

Gradient Descent

Finding the minimum



You're blindfolded, but you can see out of the bottom of the blindfold to the ground right by your feet. I drop you off somewhere and tell you that you're in a convex shaped valley and escape is at the bottom/minimum. How do you get out?

Gradient descent

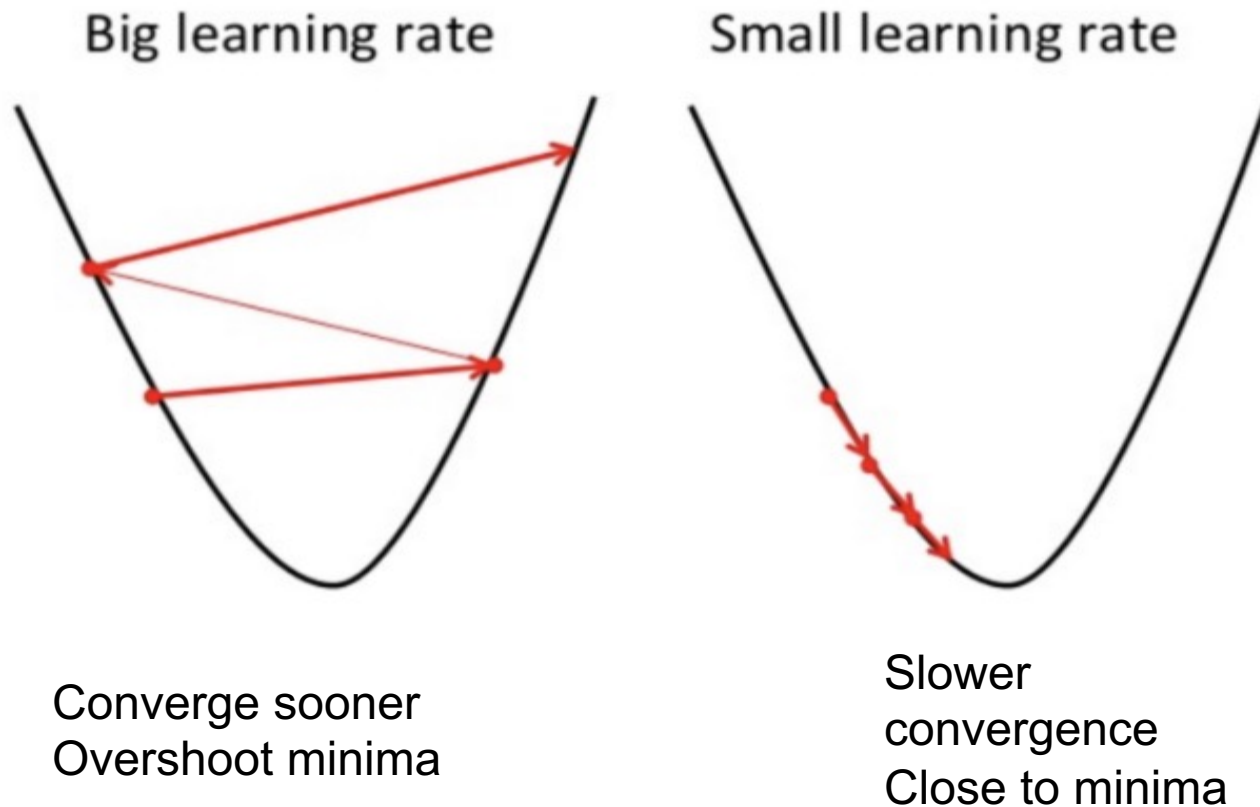
A process of optimizing the values of the coefficients by iteratively minimizing the error (cost function) of the model on the training data.

Start with random values for each coefficient, and iteratively change the values to reduce error/cost – GD

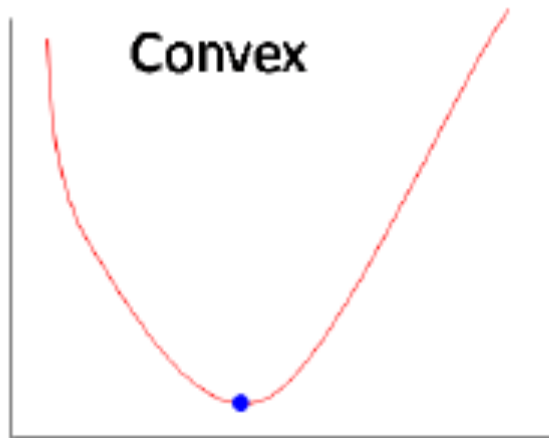
The sum of the squared errors are calculated for each pair of input and output values. A learning rate (alpha - selected) is used as a scale factor and the coefficients are updated in the direction towards minimizing the error/cost.

The process is repeated until a minimum sum squared error is achieved or no further improvement is possible

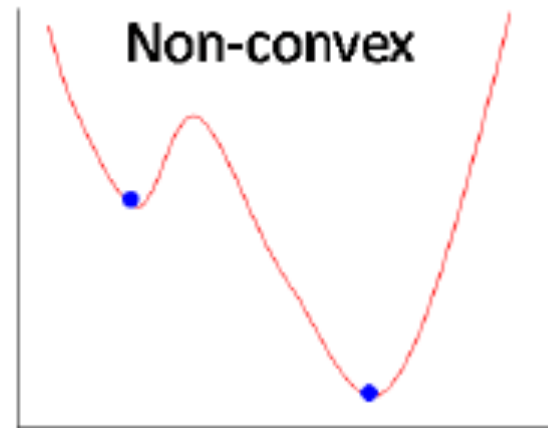
Gradient descent



Gradient descent

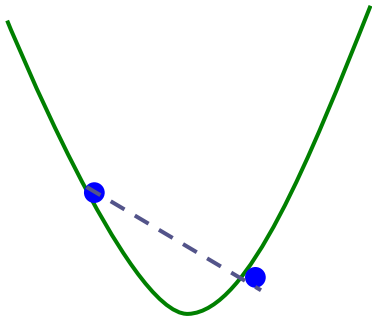


Linear regression



Gradient descent will reach a *global* optimum

Convexity



One definition: The line segment between any two points on the function is *above* the function

Mathematically, f is convex if for all x_1, x_2 :

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2) \quad \forall 0 < t < 1$$

the value of the
function at some
point between x_1 and
 x_2

the value at some
point on the **line
segment** between
 x_1 and x_2

Gradient Descent for Linear Regression

Learning algorithm:

- Initialize weights $\mathbf{w}=\mathbf{0}$
- For $t=1, \dots$ until convergence:
 - Predict for each example \mathbf{x}^i using \mathbf{w} :
$$\hat{y}^i = \sum_{j=0}^k w_j \phi_j(\mathbf{x}^i)$$
 - 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}
 - Update: $\mathbf{w} = \mathbf{w} - \lambda \mathbf{g}$
 - λ is the learning rate.

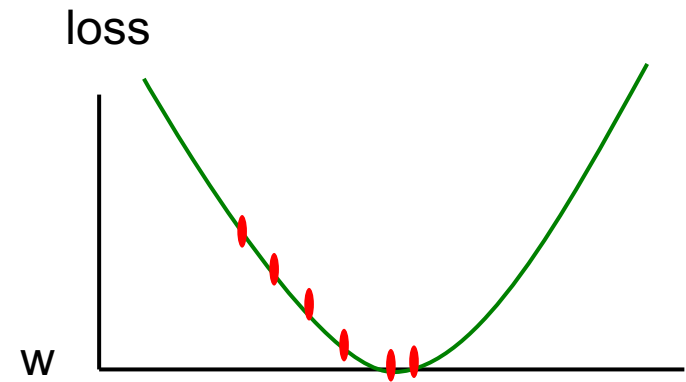
One concern

$$\operatorname{argmin}_{w,b} \sum_{i=1}^n \exp(-y_i(w \cdot x_i + b))$$

We're calculating this on the **training set**

We still need to be careful about overfitting!

The min w, b on the training set is generally NOT the min for the test set



Regression and Overfitting

A series of horizontal lines in teal and light blue colors, with varying lengths and thicknesses, extending from the left edge of the slide towards the right.

Regularization

Regularization

$$0 = b + \sum_{j=1}^n w_j f_j$$

Generally, we don't want huge weights

If weights are large, a small change in a feature can result in a large change in the prediction

Also gives too much weight to any one feature

Might also prefer weights of 0 for features that aren't useful

Common regularizers


sum of the weights

$$r(w, b) = \sum_{w_j} |w_j|$$

sum of the squared weights

$$r(w, b) = \sqrt{\sum_{w_j} |w_j|^2}$$

Squared weights penalizes large values more
Sum of weights will penalize small values more

$$\operatorname{argmin}_{w,b} \sum_{i=1}^n \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2} \|w\|^2$$
Two blue arrows originate from the text blocks below. One arrow points from the 'Loss function' text to the summation term of the equation. The other arrow points from the 'Regularizer' text to the $\frac{\lambda}{2} \|w\|^2$ term of the equation.

Loss function: penalizes examples where the prediction is different than the label

Regularizer: penalizes large weights

Regularization - weight decay

These seek to both minimize the sum of the squared error of the model on the training data (using ordinary least squares) but also to reduce the complexity of the model (like the number or absolute size of the sum of all coefficients in the model).

Two popular examples of regularization procedures for linear regression are:
[Lasso Regression](#) (L1): where Ordinary Least Squares is modified to also minimize the **absolute sum** of the coefficients

[Ridge Regression](#) (L2): where Ordinary Least Squares is modified to also minimize the **squared absolute sum** of the coefficients

These methods are effective to use when there is **collinearity** in your input values and ordinary least squares would overfit the training data.

L1 regularization

$$w_j = w_j + \eta y_i x_{ij} \exp(-y_i (w \cdot x_i + b)) - \eta \lambda \text{sign}(w_j)$$

learning rate
direction
to update

regularization

constant: how far from wrong

If w_j is positive, reduces by a constant

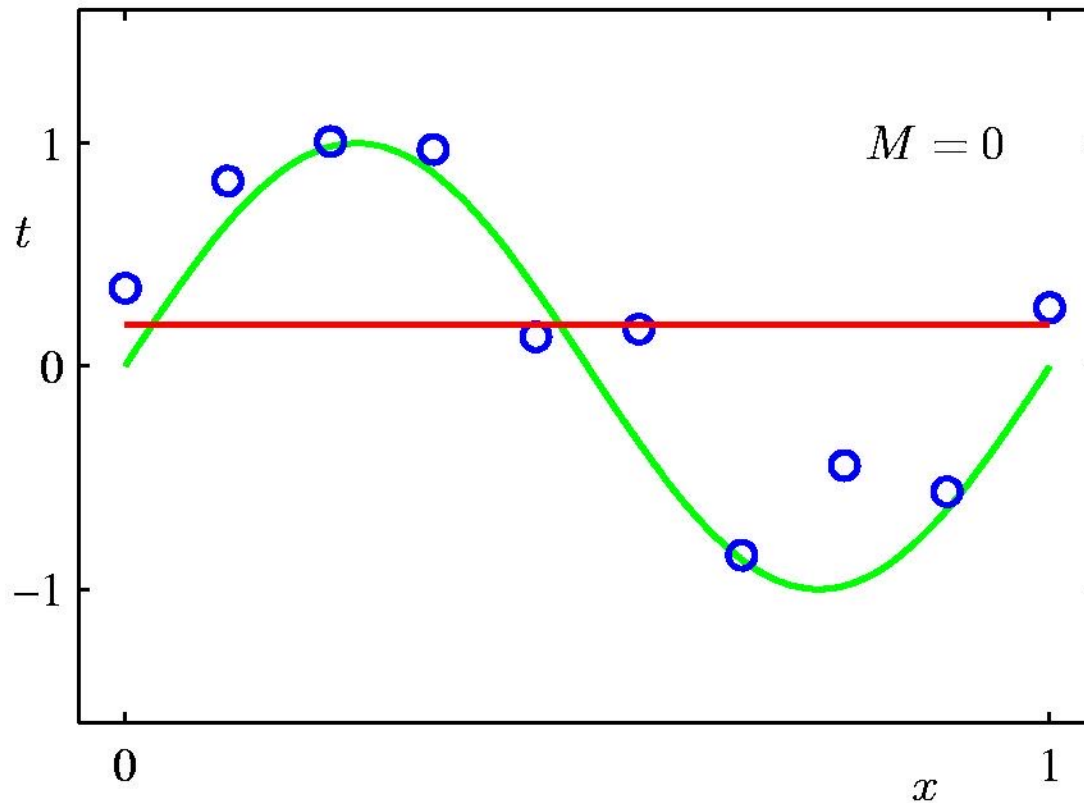
If w_j is negative, increases by a constant

moves w_j towards 0

regardless of magnitude

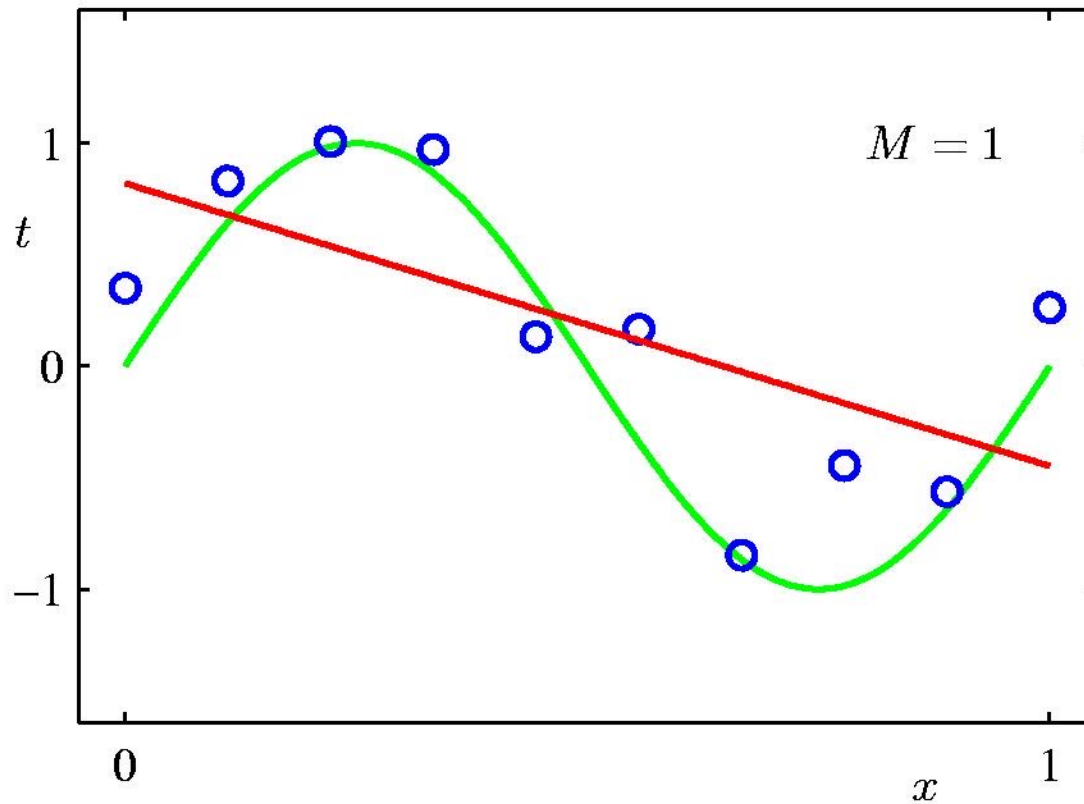
An example: polynomial basis vectors on a small dataset

0th Order Polynomial

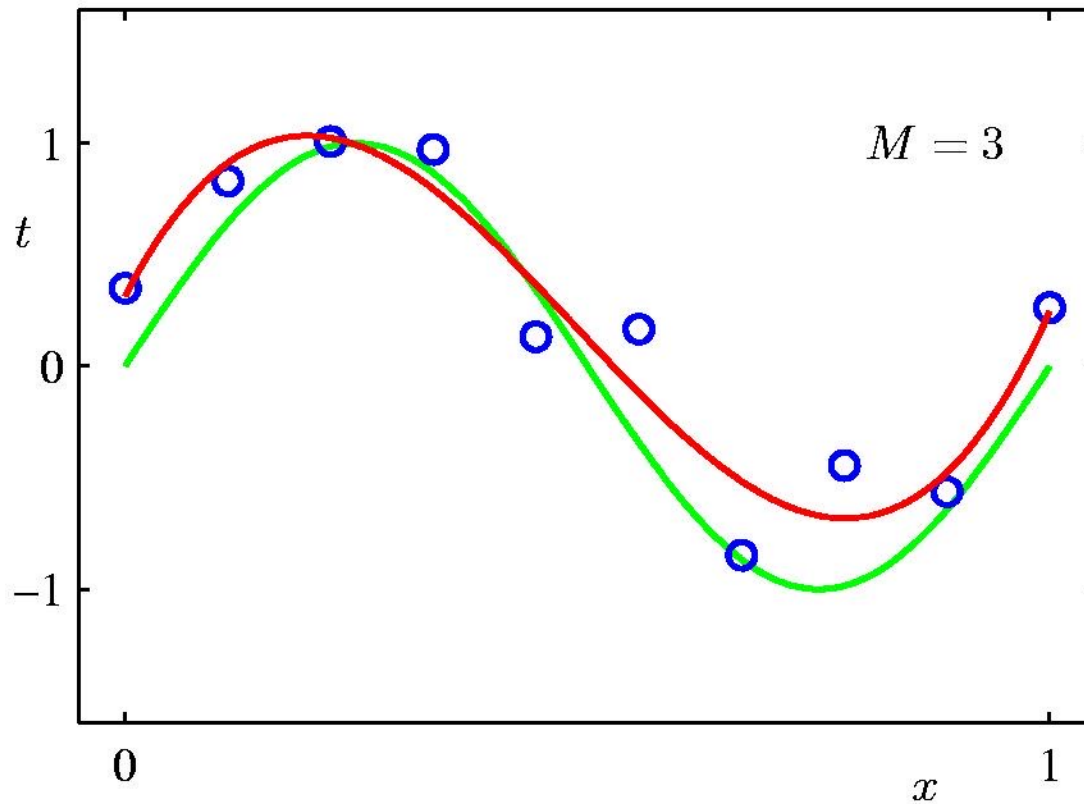


$n=10$

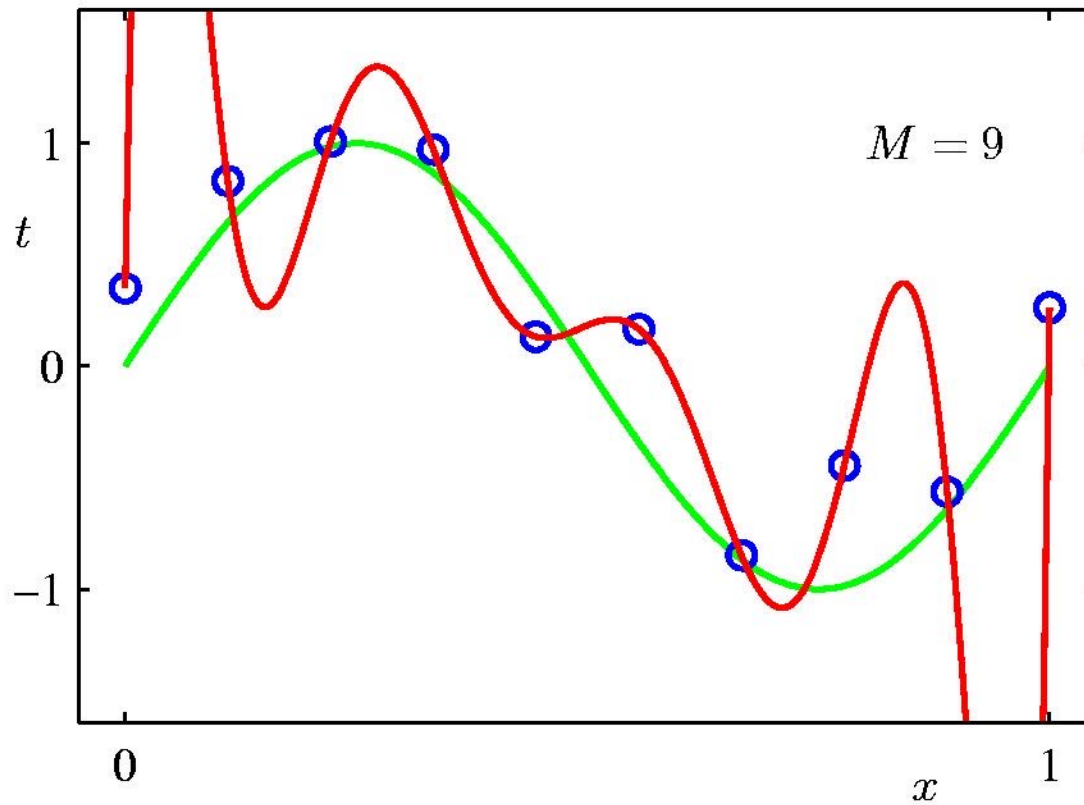
1st Order Polynomial



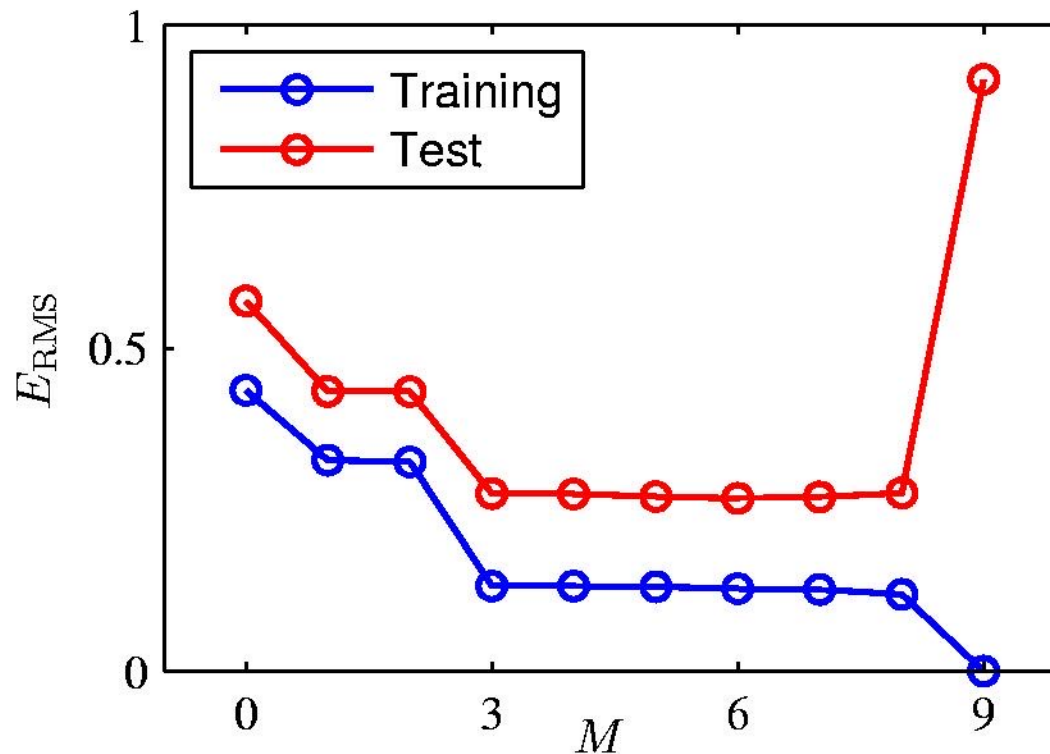
3rd Order Polynomial



9th Order Polynomial



Over-fitting



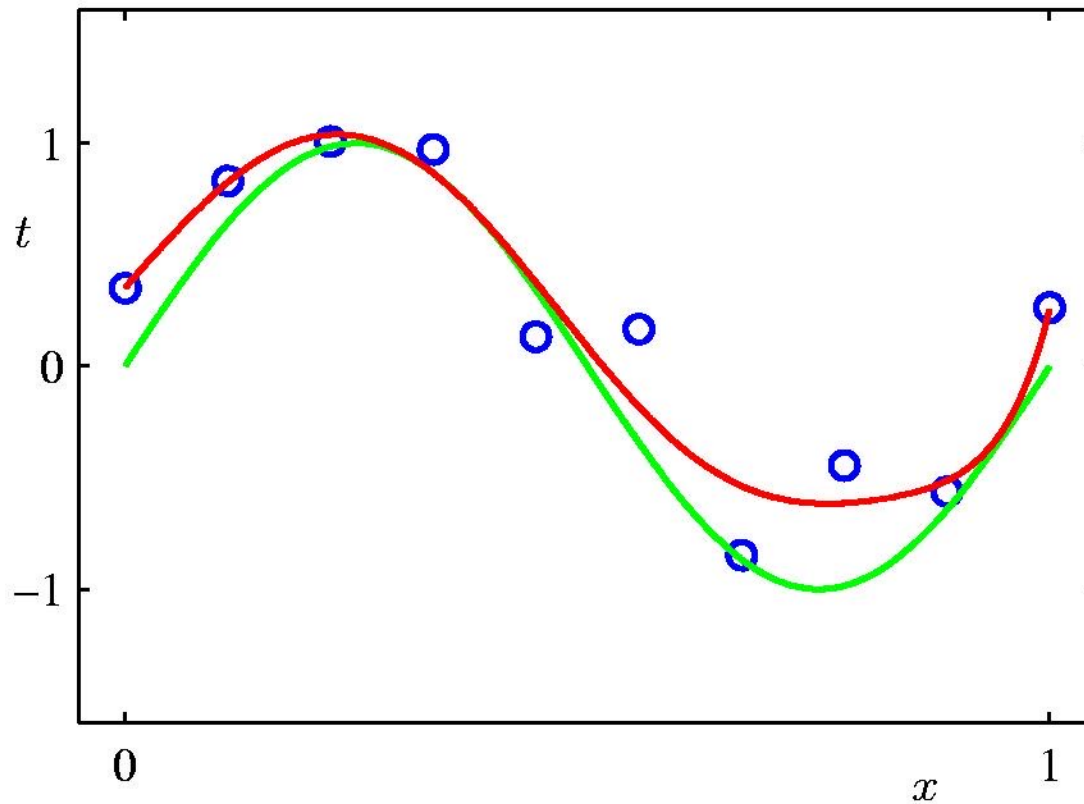
Root-Mean-Square (RMS) Error: $E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^*)/N}$

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

Regularization:

$$\ln \lambda = +.18$$



Polynomial Coefficients

	none	exp(18)	huge
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

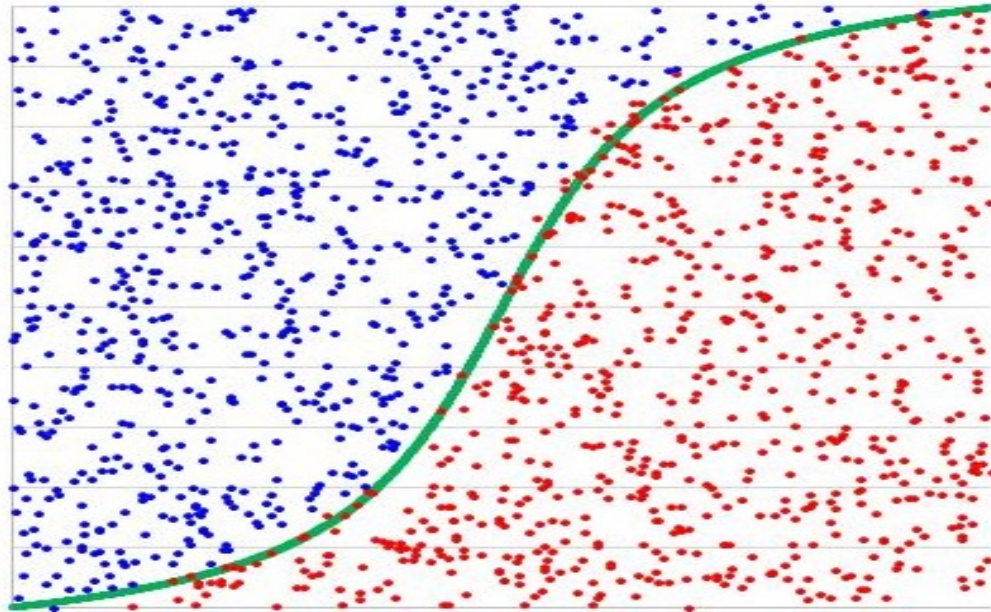
Summary

- Simplest regression algorithm
- Very fast
- Good at numerical data with lots of features
- Output from numerical continuous range
- Linear hypothesis
- Uses gradient descent

Logistic Regression

A series of horizontal lines in teal and light blue colors, some solid and some dashed, extending across the bottom of the slide.

Logistic Regression



Output is a category

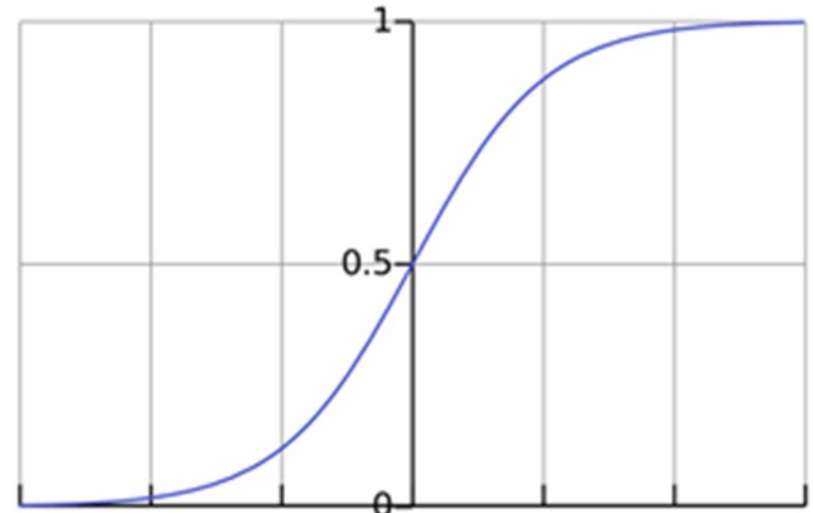
A classification algorithm that may have only two categories on the output, so it's binary classification

Sigmoid/Logistic Function

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$



Estimated probability that $Y = 1$ on input X

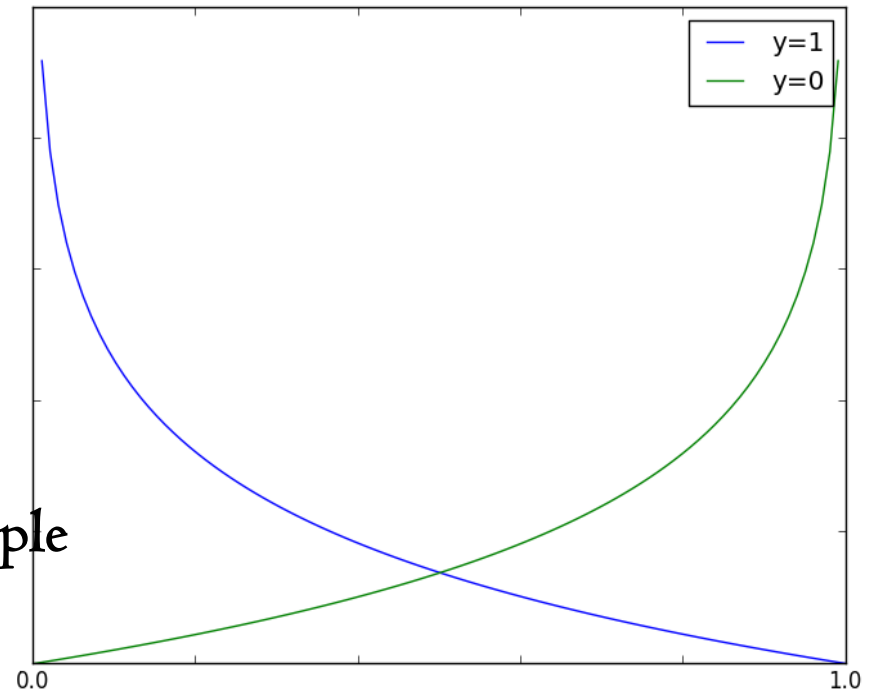


Loss function

$$\text{cost}(h_{\theta}(x), y) =$$

$$\begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

Cost for a single training data example



Complete loss function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right]$$

1. Uses the principle of maximum likelihood estimation.
2. We minimize it same way as with Linear Regression

Summary

- Classification algorithm
- Output is the binary value, either 1 or 0
- Relatively small number of predictors
- Uses logistics function for hypothesis
- Has the cost function that is convex
- Uses gradient descent for correcting the mistake