# Programming Machine Learning Applications

Lecture Six: Regression and Gradient Descent

Dr. Aleksandar Velkoski

Text Classification

Recommender Systems

## Review of Lecture Five

Regression

Gradient Descent

Discuss Code

# Lecture Six

## Regression

# What is Numeric Prediction

**(Numerical) prediction is similar to classification**

- construct a model
- use model to predict value for a given input

**Prediction is different from classification**

- Classification refers to predicting categorical class label
- Prediction models continuous-valued functions

**Major method for prediction: regression**

- model the relationship between one or more independent or predictor variables and a dependent or response variable

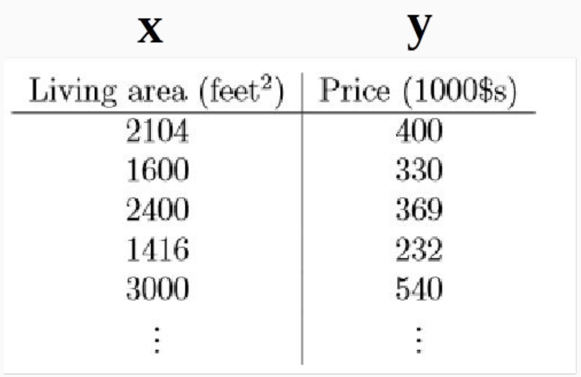
**Regression analysis**

- Linear and multiple regression
- Non-linear regression
- Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees

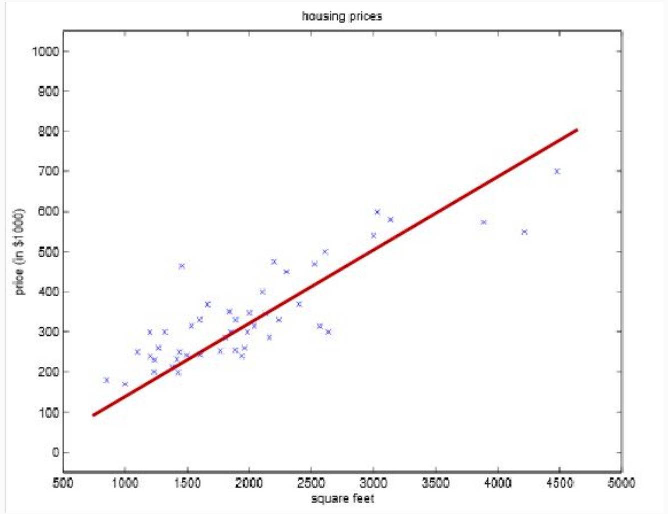
# Simple Linear Regression

**Simple linear regression:** involves a response variable y and a single predictor variable x

where $y = w_0 + w_1 x$



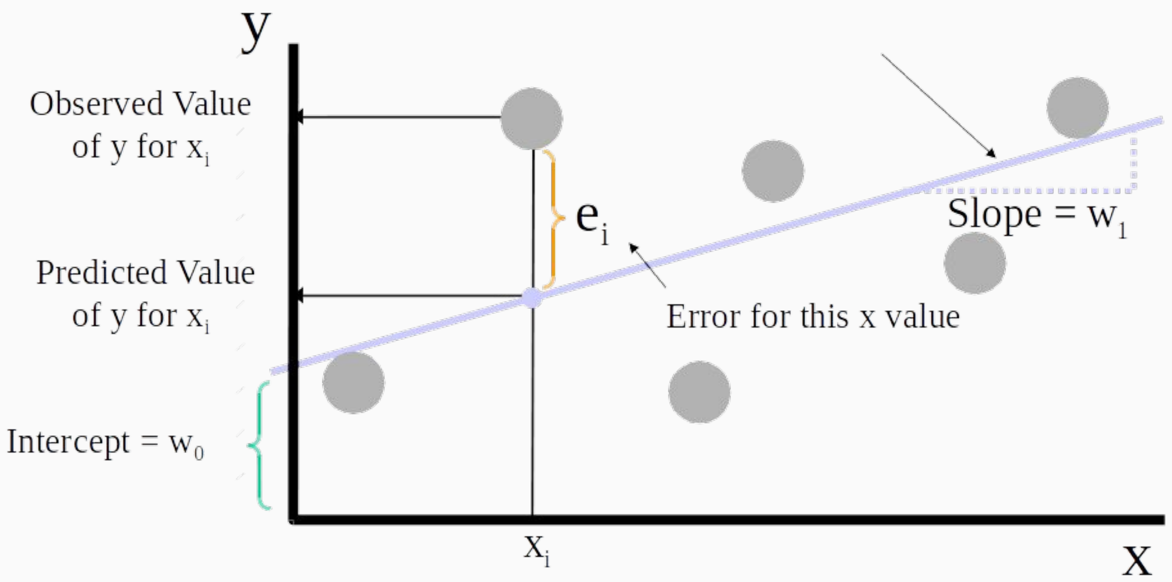
| **x** Living area (feet$^2$) | **y** Price (1000\$s) |
|---|---|
| 2104 | 400 |
| 1600 | 330 |
| 2400 | 369 |
| 1416 | 232 |
| 3000 | 540 |
| ⋮ | ⋮ |



**Goal**: Use data to estimate weights (parameters) $w_0$ and $w_1$ such that prediction error is minimized

# Simple Linear Regression

$$y = w_0 + w_1 x$$



## Simple Linear Regression

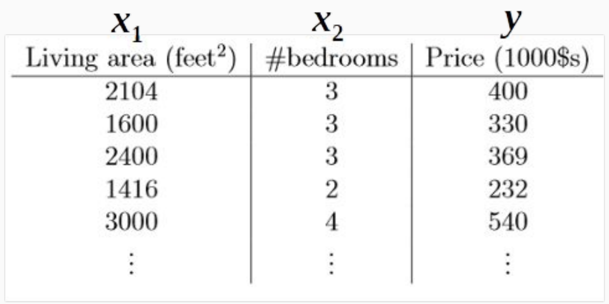
- **Method of least squares**:
  - Estimates the best-fitting straight line
  - $w_0$ and $w_1$ are obtained by minimizing the sum of the squared errors (a.k.a. residuals)

$$SSE = \sum_i e_i^2 = \sum_i (y_i - \hat{y}_i)^2$$
$$= \sum (y_i - (w_0 + w_1 x_i))^2$$

# Multiple Linear Regression

i **<u>Multiple linear regression</u>: involves more than one predictor variable**

- 4 Features represented as $x_1$, $x_2$, …, $x_d$
- 4 Training data is of the form $(\mathbf{x}^1, y^1)$, $(\mathbf{x}^2, y^2)$,…, $(\mathbf{x}^n, y^n)$
  (each $\mathbf{x}^j$ is a row vector in matrix $\mathbf{X}$, i.e. a row in the data)
- 4 For a specific value of a feature $\mathbf{x}_i$ in data item $\mathbf{x}^j$ we use: $x_i^j$
- 4 Ex. For 2-D data, the regression function is: $\hat{y} = w_0 + w_1 x_1 + w_2 x_2$

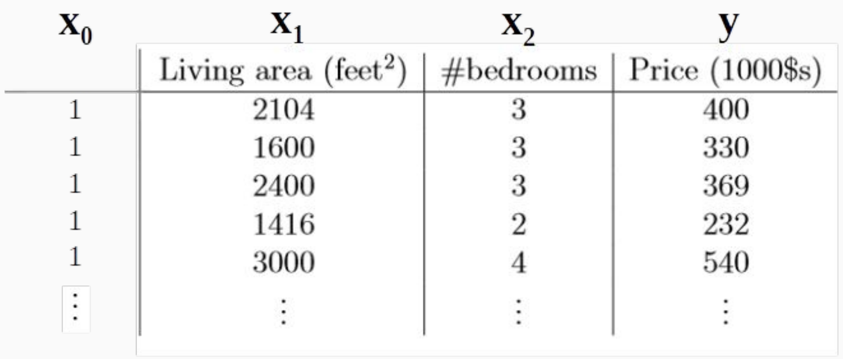


| $x_1$<br>Living area (feet$^2$) | $x_2$<br>#bedrooms | $y$<br>Price (1000\$s) |
|---|---|---|
| 2104 | 3 | 400 |
| 1600 | 3 | 330 |
| 2400 | 3 | 369 |
| 1416 | 2 | 232 |
| 3000 | 4 | 540 |
| ⋮ | ⋮ | ⋮ |

# Least Squares Generalization

i **Multiple dimensions**

4 To simplify add a new feature $x_0$ = **1** to feature vector **x**:



| $\mathbf{x}_0$ | $\mathbf{x}_1$ | $\mathbf{x}_2$ | $\mathbf{y}$ |
|---|---|---|---|
| | Living area (feet$^2$) | #bedrooms | Price (1000$s) |
| 1 | 2104 | 3 | 400 |
| 1 | 1600 | 3 | 330 |
| 1 | 2400 | 3 | 369 |
| 1 | 1416 | 2 | 232 |
| 1 | 3000 | 4 | 540 |
| ⋮ | ⋮ | ⋮ | ⋮ |

$$\hat{y} = f(x_0, x_1, \dots, x_d) = w_0 x_0 + \sum_{i=1}^{d} w_i x_i = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^{\mathrm{T}}.\mathbf{x}$$

## Least Squares Generalization

$$\hat{y} = f(x_0, x_1, \dots, x_d) = f(\mathbf{x}) = w_0 x_0 + \sum_{i=1}^{d} w_i x_i = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^{\mathrm{T}}.\mathbf{x}$$

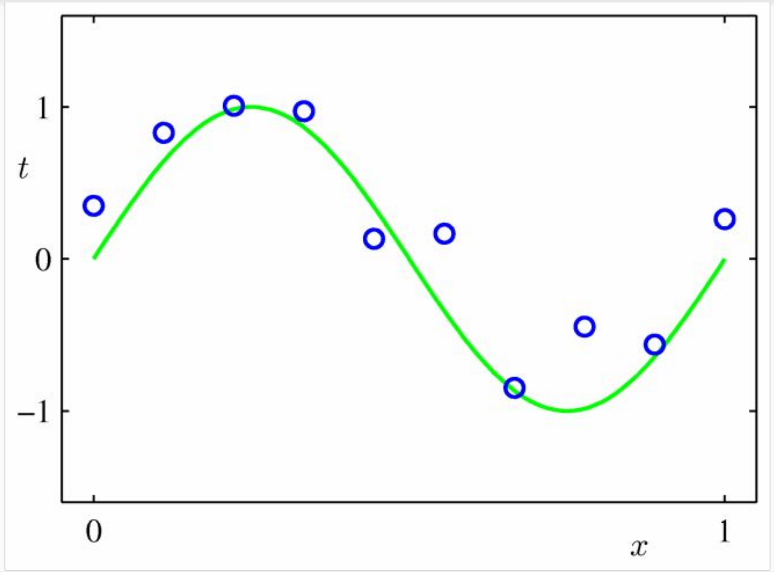
4 Calculate the error function (SSE) and determine **w**:

$$E(\mathbf{w}) = \left(\mathbf{y} - f(\mathbf{x})\right)^2 = \left[\mathbf{y} - \sum_{i=0}^{d} w_i \cdot x_i\right]^2 = \sum_{j=1}^{n} (y^i - \sum_{i=0}^{d} w_i \cdot x_i^j)^2$$

# Application of Linear Regression

- i **The inputs X for linear regression can be:**
  - 4 Original quantitative inputs
  - 4 Transformation of quantitative inputs, e.g. log, exp, square root, square, etc.
  - 4 Polynomial transformation
    - h example: $y = w_0 + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3$
  - 4 Dummy coding of categorical inputs
  - 4 Interactions between variables
    - h example: $x_3 = x_1 \cdot x_2$
- i **This allows use of linear regression techniques to fit much more complicated non-linear datasets.**

## Fitting Polynomial with Linear Model



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

# Regularization

- **Complex models (lots of parameters) are often prone to overfitting**
- **Overfitting can be reduced by imposing a constraint on the overall magnitude of the parameters (i.e., by including coefficients as part of the optimization process)**
- **Two common types of regularization in linear regression:**
  - $L_2$ regularization (a.k.a. ridge regression). Find **w** which minimizes:

$$\sum_{j=1}^{N}(y_j - \sum_{i=0}^{d} w_i \cdot x_i)^2 + \lambda \sum_{i=1}^{d} w_i^{\ 2}$$

    - λ is the regularization parameter: bigger λ imposes more constraint

  - $L_1$ regularization (a.k.a. lasso). Find **w** which minimizes:

$$\sum_{j=1}^{N}(y_j - \sum_{i=0}^{d} w_i \cdot x_i)^2 + \lambda \sum_{i=1}^{d} | w_i |$$

## Other Regression-Based Models

**Generalized linear models**

- Foundation on which linear regression can be applied to modeling categorical response variables
- Variance of y is a function of the mean value of y, not a constant
- Logistic regression: models the probability of some event occurring as a linear function of a set of predictor variables
- Poisson regression: models the data that exhibit a Poisson distribution

**Log-linear models  (for categorical data)**

- Approximate discrete multidimensional prob. distributions
- Also useful for data compression and smoothing

**Regression trees and model trees**

- Trees to predict continuous values rather than class labels

## Regression Trees & Model Trees

**Regression tree: proposed in CART system (Breiman et al. 1984)**

- CART: Classification And Regression Trees
- Each leaf stores a continuous-valued prediction
- It is the average value of the predicted attribute for the training instances that reach the leaf

**Model tree: proposed by Quinlan (1992)**

- Each leaf holds a regression model—a multivariate linear equation for the predicted attribute
- A more general case than regression tree

**Regression and model trees tend to be more accurate than linear regression when instances are not represented well by simple linear models**

## Evaluating Regression Models

- **Prediction Accuracy**
  - Difference between predicted scores and the actual results (from evaluation set)
  - Typically the accuracy of the model is measured in terms of variance (i.e., average of the squared differences)

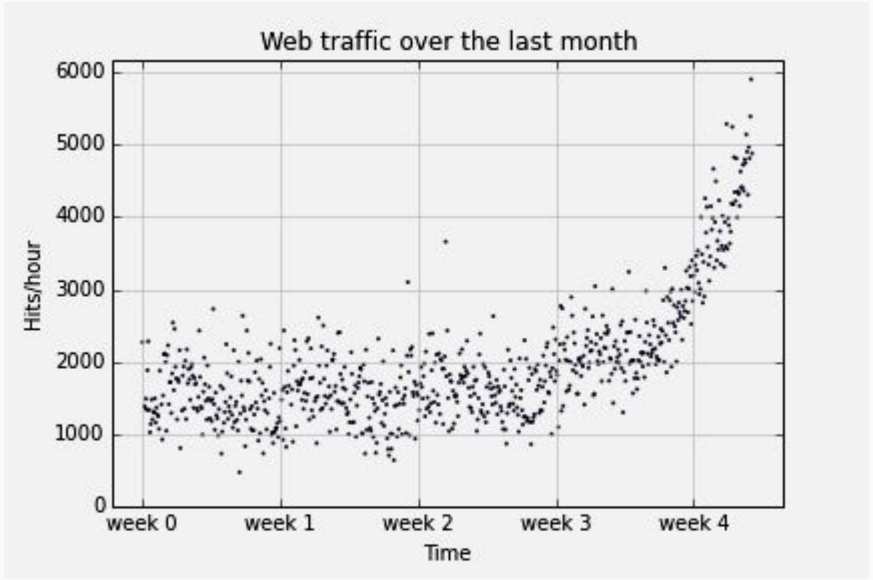
- **Common Metrics** ($p_i$ = predicted target value for test instance $i$, $a_i$ = actual target value for instance $i$)
  - **Mean Absolute Error:** Average loss over the test set

$$MAE = \frac{(p_1 - a_1) + \ldots + (p_n - a_n)}{n}$$

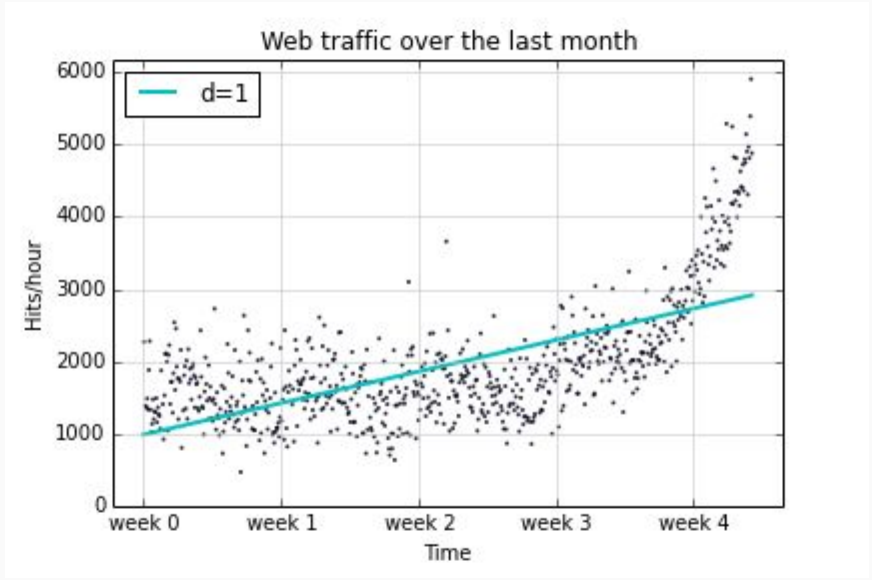
  - **Root Mean Squared Error**: compute the standard deviation (i.e., square root of the co-variance between predicted and actual ratings)

$$RMSE = \sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{n}}$$

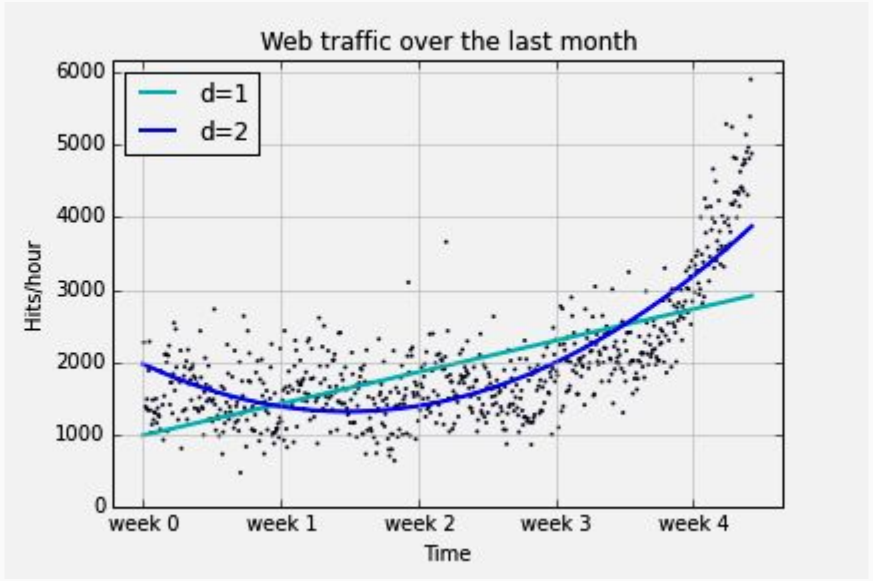
## Example: Web Traffic



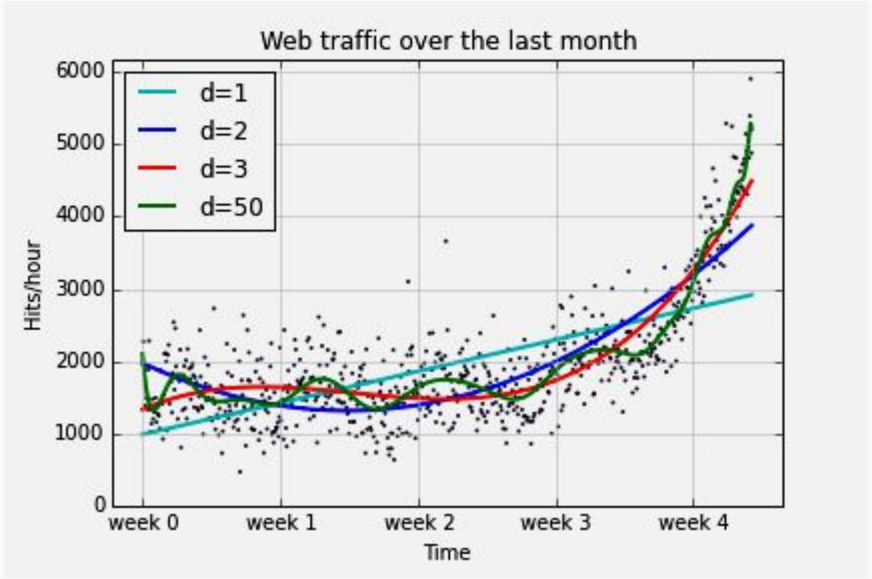
# Example: Web Traffic 1d Fit



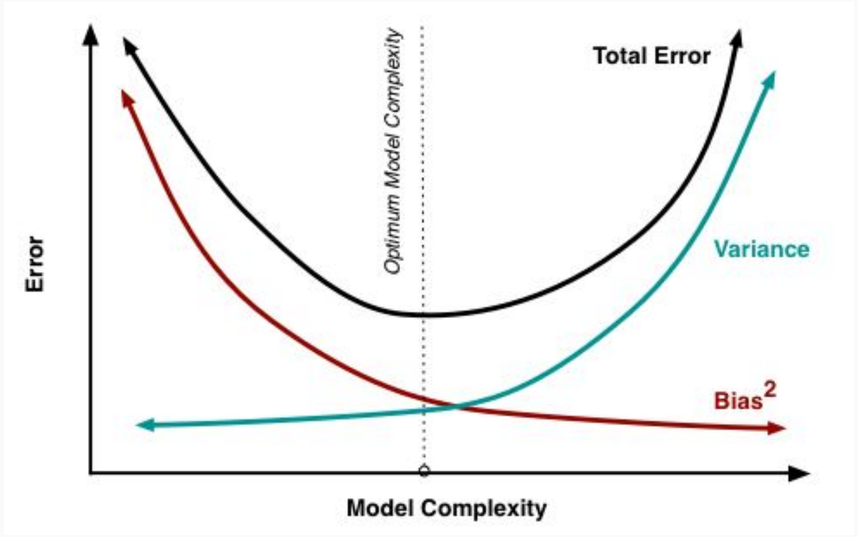
# Example: Web Traffic 2d Fit



## Example: Web Traffic Beyond 2d Fit

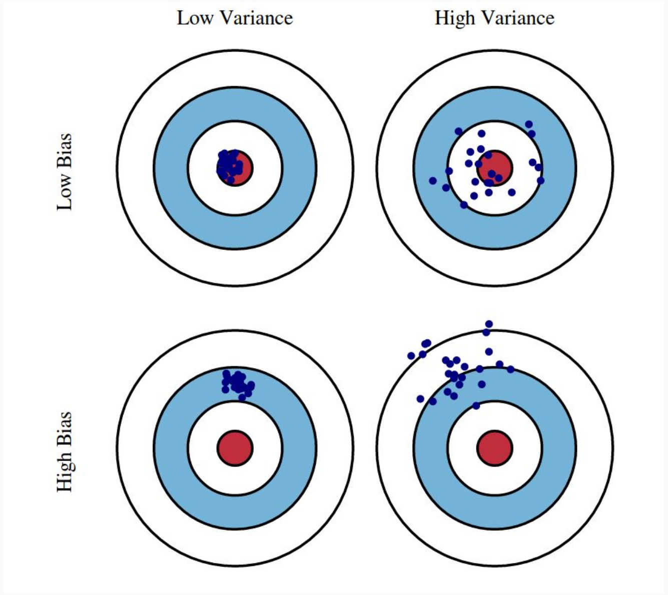


## Model Complexity



**Source**: http://www.cs.cornell.edu/courses/cs4780/2015fa/web/lecturenotes/lecturenote12.html

## Bias-Variance Tradeoff



**Source**: http://www.cs.cornell.edu/courses/cs4780/2015fa/web/lecturenotes/lecturenote12.html

# Bias-Variance Tradeoff

i **Possible ways of dealing with high bias**

- 4 Get additional features
- 4 More complex model (e.g., adding polynomial terms such as $x_1^2$, $x_2^2$ , $x_1.x_2$, etc.)
- 4 Use smaller regularization coefficient $\lambda$.
- 4 **Note:** getting more training data won't necessarily help in this case

i **Possible ways dealing with high variance**

- 4 Use more training instances
- 4 Reduce the number of features
- 4 Use simpler models
- 4 Use a larger regularization coefficient $\lambda$.

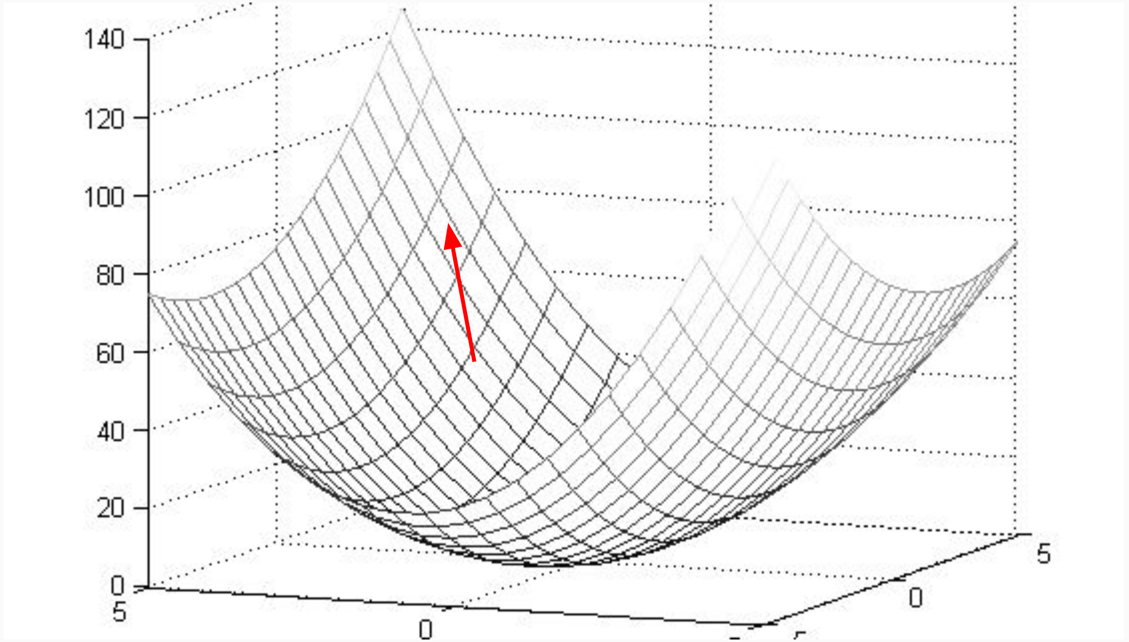
## Gradient Descent

# Gradient Descent Optimization

- **Linear regression can also be solved using Gradient Decent optimization approach**
- **GD can be used in a variety of settings to find the minimum value of functions (including non-linear functions) where a closed form solution is not available or not easily obtained**
- **Basic idea:**
  - Given an objective function $J(\mathbf{w})$ (e.g., sum of squared errors), with w as a vector of variables $w_0, w_1, \ldots, w_d$, iteratively minimize $J(\mathbf{w})$ by **finding the gradient** of the function surface in the variable-space and **adjusting the weights** in the opposite direction
  - The gradient is a vector with each element representing the slope of the function in the direction of one of the variables
  - Each element is the partial derivative of function with respect to one of variables

$$\nabla J(\mathbf{w}) = \nabla J(w_1, w_2, \ldots, w_d) = \left[ \frac{\partial f(\mathbf{w})}{\partial w_1} \quad \frac{\partial f(\mathbf{w})}{\partial w_2} \quad \cdots \quad \frac{\partial f(\mathbf{w})}{\partial w_d} \right]$$

# Gradient Descent Optimization



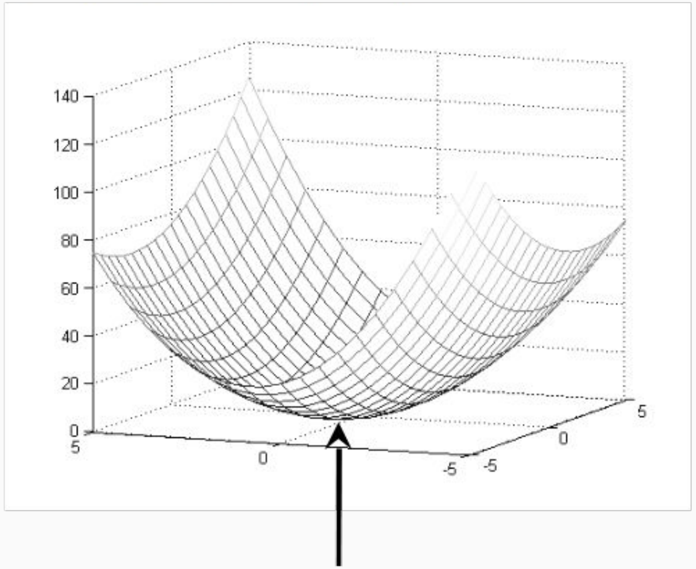
## Gradient is a Vector

$$\frac{\partial}{\partial \mathrm{m}} = \frac{2}{N} \sum_{i=1}^{N} -x_i(y_i - (mx_i + b))$$

$$\frac{\partial}{\partial \mathrm{b}} = \frac{2}{N} \sum_{i=1}^{N} -(y_i - (mx_i + b))$$

## Gradient Descent Optimization

Gradient vector points in the direction of the steepest ascent of function



## Gradient Descent Optimization

i **Finding minimum directly by closed form analytical solution often difficult or impossible**

- 4 Quadratic functions in many variables
  - h system of equations for partial derivatives may be ill-conditioned
  - h example: linear least squares fit where redundancy among features is high
- 4 Other convex functions
  - h global minimum exists, but there is no closed form solution
  - h example: maximum likelihood solution for logistic regression
- 4 Nonlinear functions
  - h partial derivatives are not linear
  - h example: $f( x_1, x_2 ) = x_1( \sin( x_1 x_2 ) ) + x_2^2$
  - h example: sum of transfer functions in neural networks

# Gradient Descent Optimization

- **Given an objective (e.g., error) function $E(\mathbf{w}) = E(w_0, w_1, \ldots, w_d)$**
- **Process (follow the gradient *downhill*):**
  1. Pick an initial set of weights (random): $\mathbf{w} = (w_0, w_1, \ldots, w_d)$
  2. Determine the descent direction: $-\nabla E(\mathbf{w}^t)$
  3. Choose a learning rate: $\eta$
  4. Update your position: $\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \cdot \nabla E(\mathbf{w}^t)$
  5. Repeat from 2) until stopping criterion is satisfied
- **Typical stopping criteria**
  - $\nabla E(\mathbf{w}^{t+1}) \sim 0$
  - some validation metric is optimized

Note: this step involves simultaneous updating of each weight $w_i$

# Gradient Descent Optimization

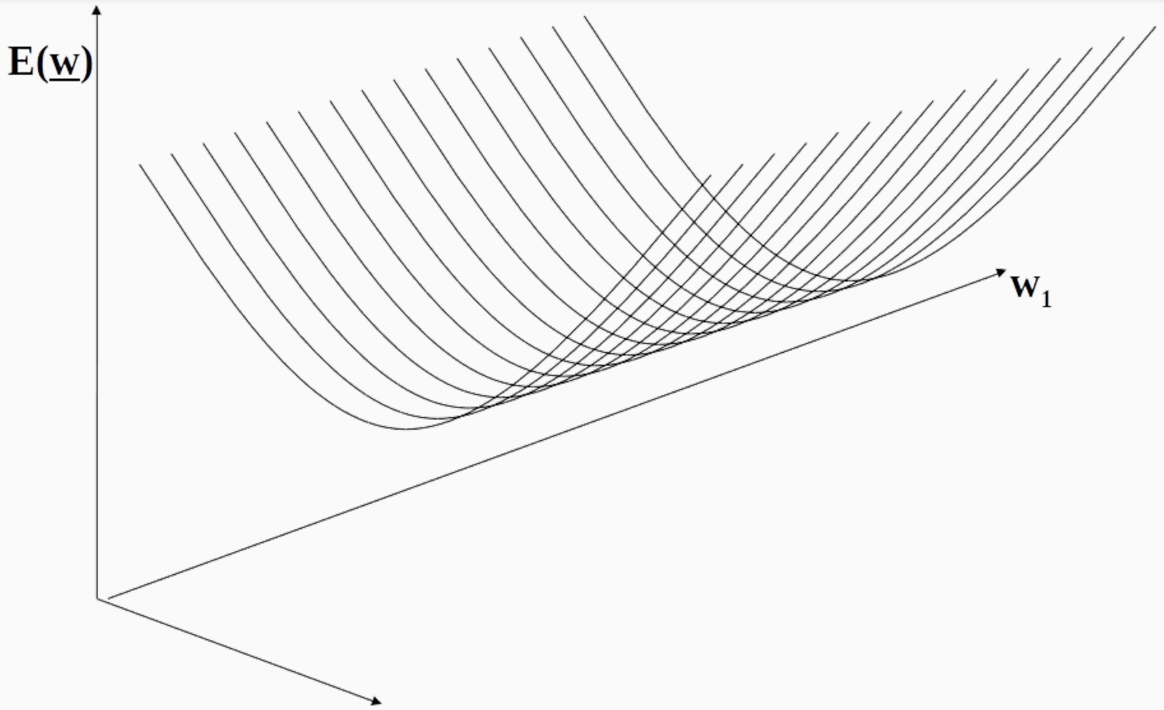
- **In Least Squares Regression:** $E(\mathbf{w}) = \left(\mathbf{y} - \sum_{i=0}^{d} w_i \cdot x_i\right)^2 = (\mathbf{y} - \mathbf{w}^{\mathrm{T}}.\mathbf{x})^2$

- **Process (follow the gradient *downhill*):**
  1. Select initial $\mathbf{w} = (w_0, w_1, \ldots, w_d)$
  2. Compute $-\nabla E(\mathbf{w})$
  3. Set $\eta$
  4. Update: $\boxed{\mathbf{w} := \mathbf{w} - \eta \cdot \nabla E(\mathbf{w})}$
  5. Repeat until $\nabla E(\mathbf{w}^{t+1}) \sim 0$

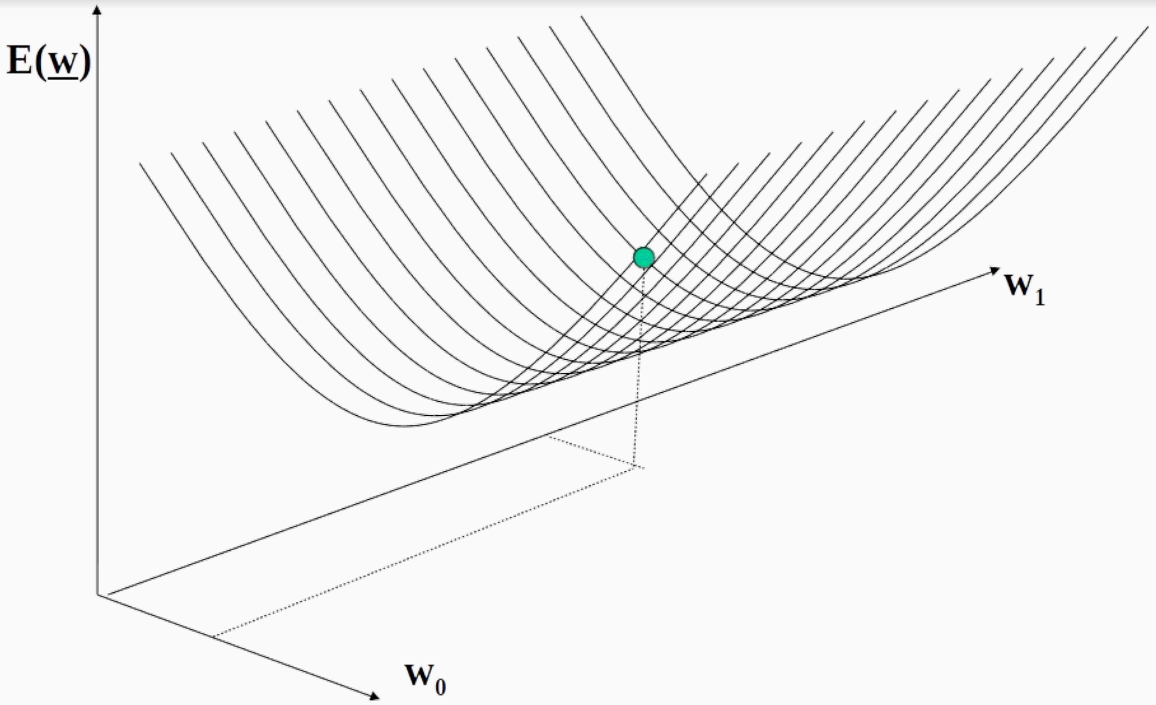
$$w_j := w_j - \eta \frac{1}{2n} \sum_{i=1}^{n} (\mathbf{w}^{\mathrm{T}}.\mathbf{x}^i - y^i) x_j^i$$

for $j = 0, 1, \ldots, d$

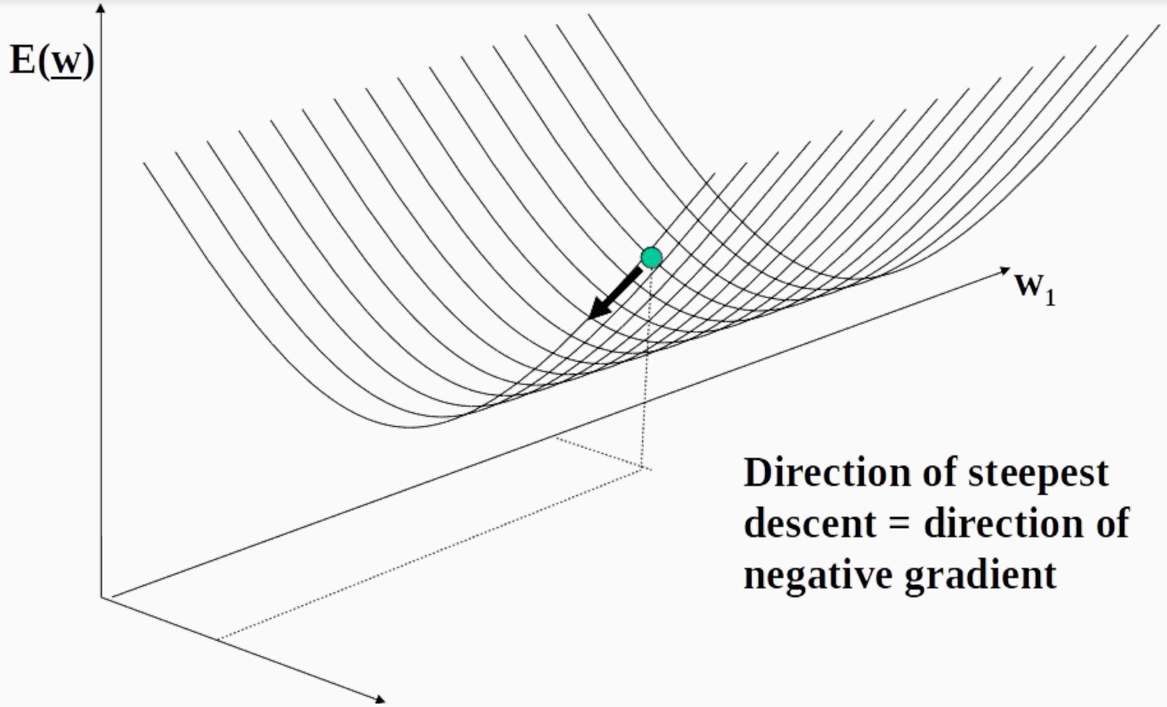
# Illustration



## Illustration

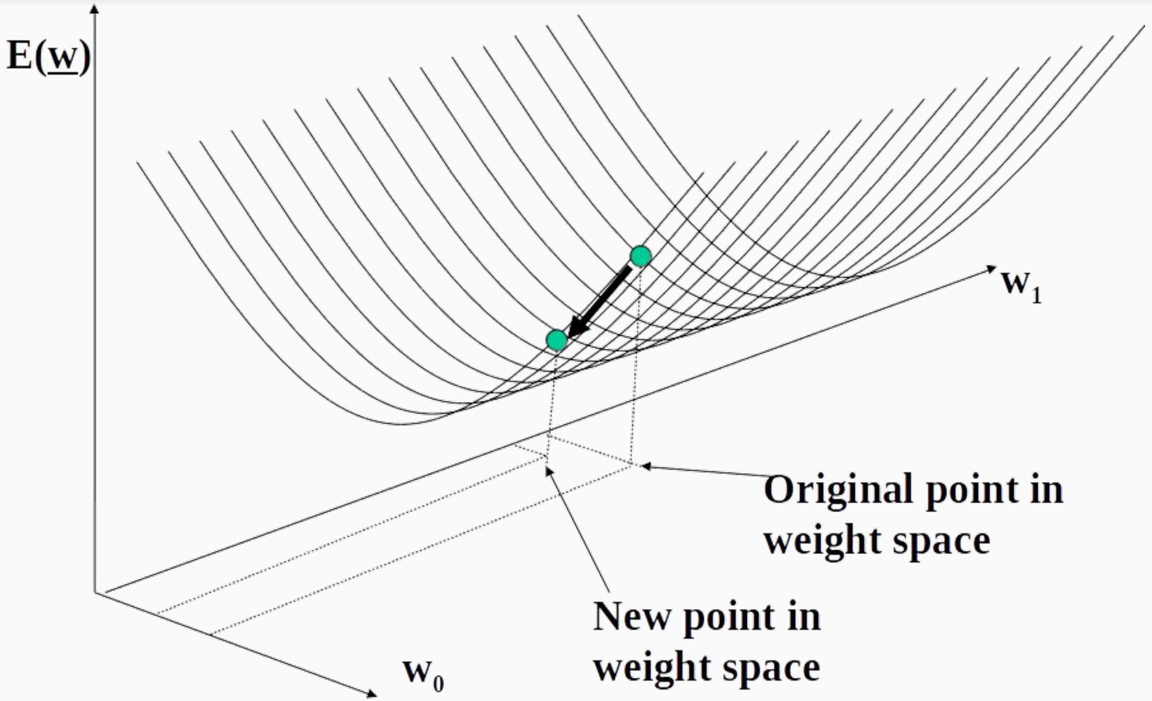


## Illustration



**Direction of steepest descent = direction of negative gradient**

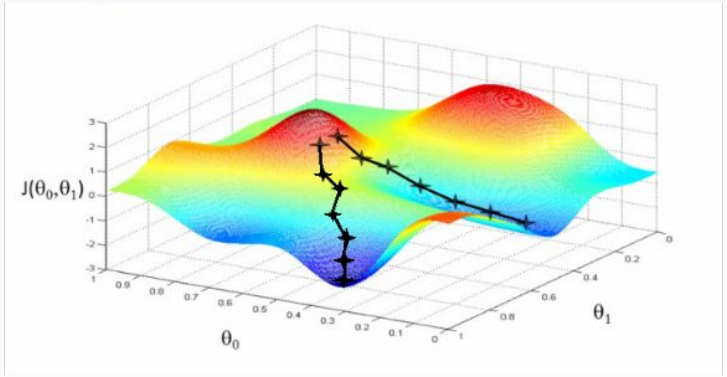
# Illustration



# Gradient Descent Optimization

i **Problems:**

- 4 Choosing step size (learning rate)
  - h too small $\rightarrow$ convergence is slow and inefficient
  - h too large $\rightarrow$ may not converge
- 4 Can get stuck on “flat” areas of function
- 4 Easily trapped in local minima



## Stochastic Gradient Descent

- **Application to training a machine learning model:**
  1. Choose one sample from training set: $x^i$
  2. Calculate objective function for that single sample: $(\mathbf{w}^{\mathrm{T}}.\mathbf{x}^i - y^i)^2$
  3. Calculate gradient from objective function:
  4. Update model parameters a single step based on gradient and learning rate:

$$w_j := w_j - \eta(\mathbf{w}^{\mathrm{T}}.\mathbf{x}^i - y^i)x_j^i \quad \text{for } j = 0, \ldots, d$$

  5. Repeat from 1) until stopping criterion is satisfied
- **Typically entire training set is processed multiple times before stopping**
- **Order in which samples are processed can be fixed or random.**

## Code

Wrapping-up the Lecture

## Questions

## What is the goal of Linear Regression?

## Describe two types of Regularization. What’s the intuition of each type?

Why is getting trapped in a local minima a problem in Gradient Descent?