

Machine Learning (CE 40477)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

September 21, 2024



1 Course Overview

2 Introduction

3 Supervised Learning

4 Optimization

5 Polynomial Regression

6 References

1 Course Overview

2 Introduction

3 Supervised Learning

4 Optimization

5 Polynomial Regression

6 References

Introduction to Machine Learning

Course Overview

- **Supervised Learning**
 - **Unsupervised Learning**
 - **Neural Networks**
 - **Computer Vision**
 - **Natural Language Processing**
 - **Contrastive Language-Image Pretraining**

Supervised Learning

Learn to predict outcomes using labeled data.

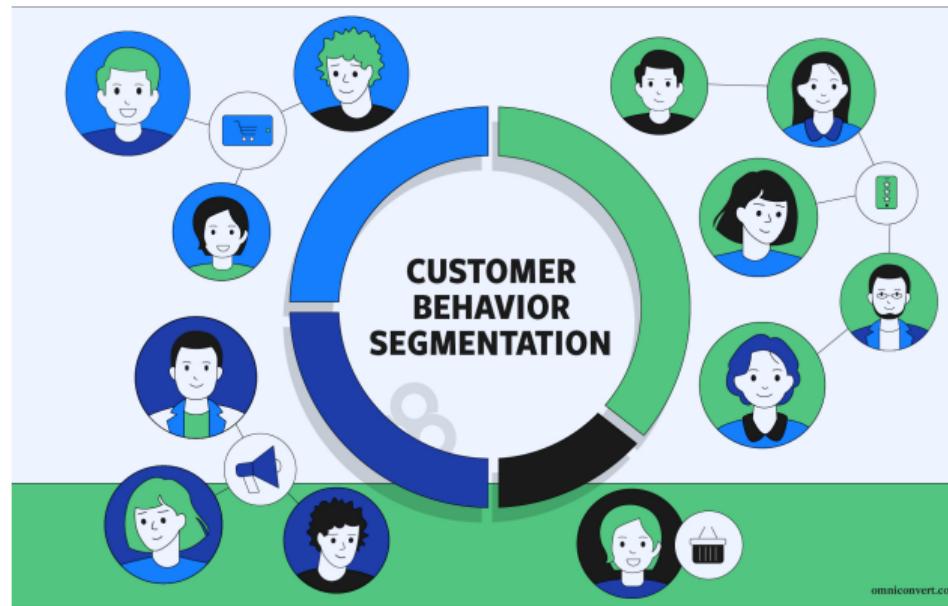
- Real estate agencies predict house prices using historical data.
 - Features like size, location, and number of rooms are used.



Learn More: Predicting House Prices with Linear Regression

Unsupervised Learning

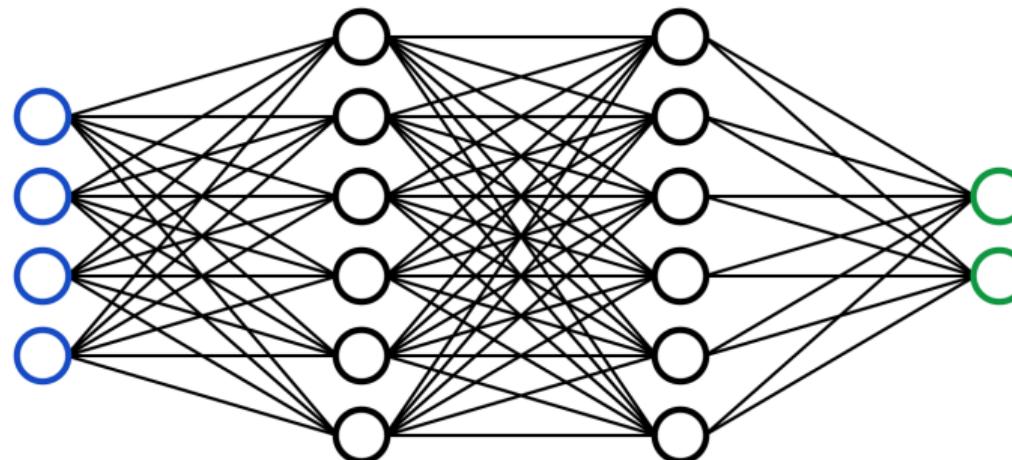
- Find patterns in unlabeled data



Example: Customer segmentation, news clustering.

Neural Networks

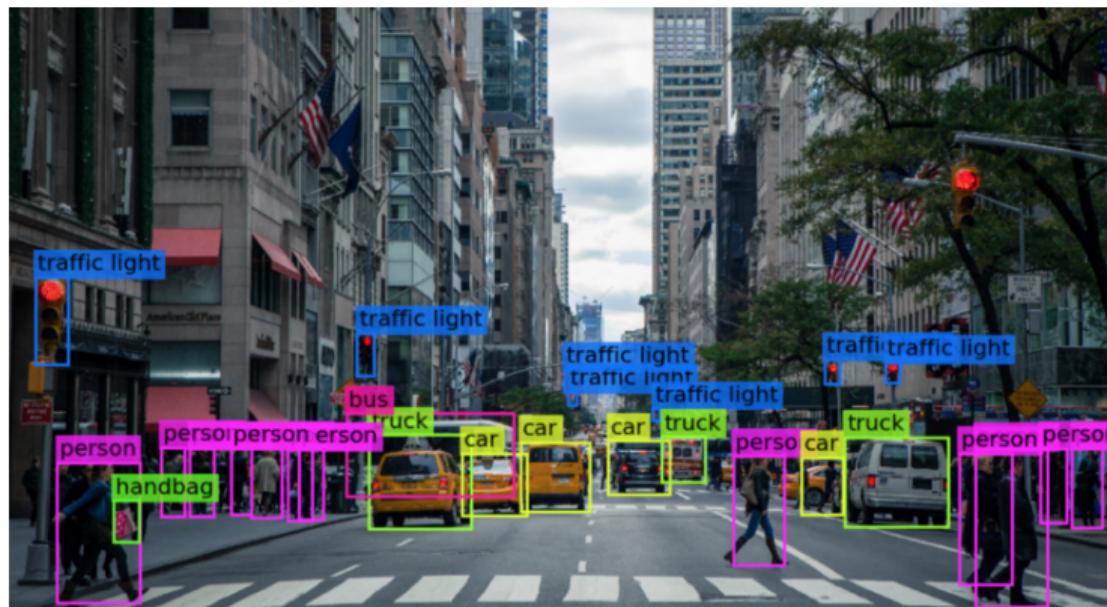
- Mimic the human brain to solve complex tasks



Example: Facial recognition, voice assistants.

Computer Vision

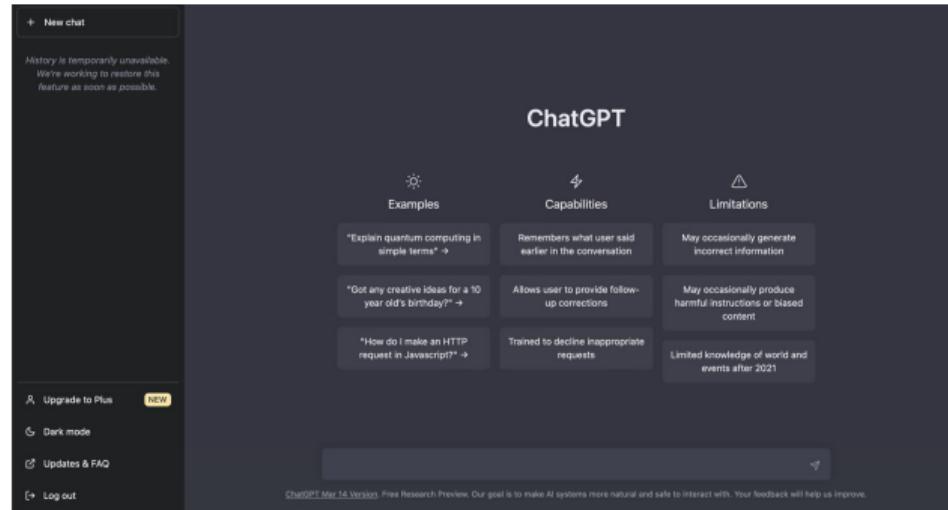
- Enable machines to see and interpret images



Example: Factory quality control, medical image analysis.

Natural Language Processing (NLP)

- Understand and generate human language



Example: Chat-bots, language translation.

Contrastive Language-Image Pretraining (CLIP)

- Connecting text and images.

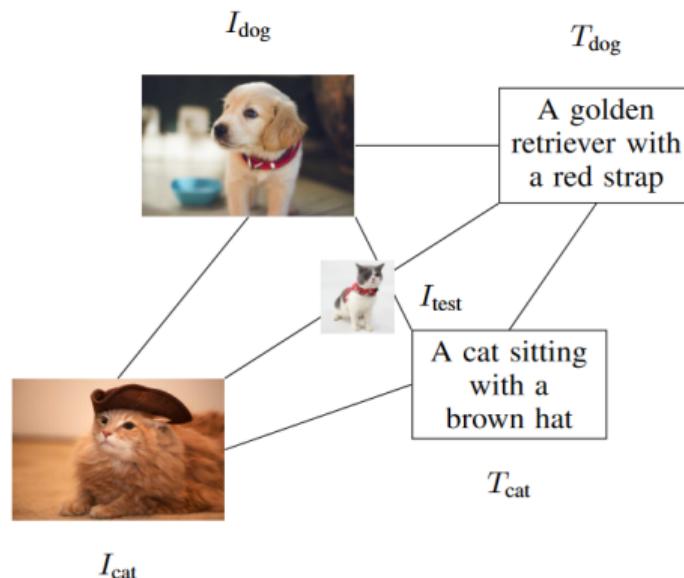


Figure adapted from [6]

1 Course Overview

2 Introduction

3 Supervised Learning

4 Optimization

5 Polynomial Regression

6 References

Definition of Machine Learning (ML)

- **Machine Learning:** A field of study that enables computers to learn from data without being explicitly programmed.
 - Involves constructing algorithms that generalize patterns from data.
 - Focuses on predicting outcomes, classification, or uncovering hidden structures.

Tom M. Mitchell's Definition of Machine Learning

- A well-known definition of machine learning comes from **Tom M. Mitchell**:
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Definition of Machine Learning (cont.)

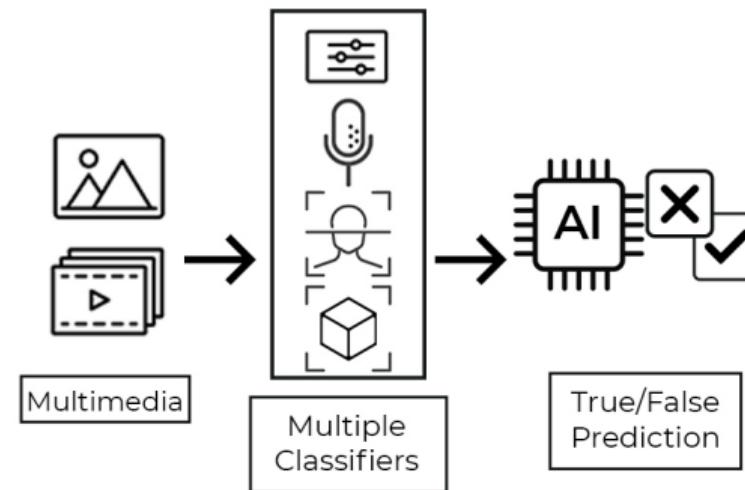
- **Goal:** Develop models that make accurate predictions based on past data.
 - Formal definition: Given a task T , performance measure P , and experience E :

Learning Problem = (T, P, E)

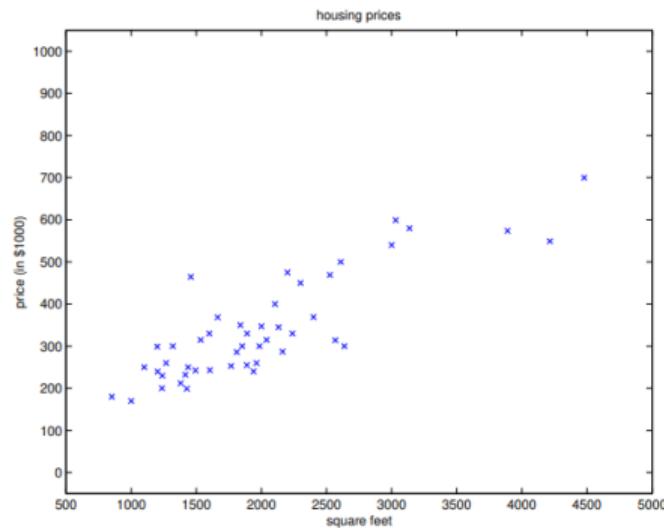
- Example: Predicting house prices based on previous data.

Example Usage of ML

- **Real-world examples:**
 - Fake News Detection (classification problem)
 - Predicting House Prices (regression problem)
 - Self-driving cars (real-time decision making)
- Application domains: finance, healthcare, robotics, etc.



Example Usage of ML: Home Price



Living area (feet ²)	Price (1000\$)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

Example Usage of ML: Applicant approval

- Applicant form as the input.
- Output: approving or denying the request.

age	23 years
gender	male
annual salary	\$30,000
years in residence	1 year
years in job	1 year
current debt	\$15,000
...	...

Paradigms of ML

- **Supervised learning** (regression, classification)
 - predicting a target variable for which we get to see examples.
- **Unsupervised learning**
 - revealing structure in the observed data
- **Reinforcement learning**
 - partial (indirect) feedback, no explicit guidance
 - given rewards for a sequence of moves to learn a policy and utility functions
- **Other paradigms:** semi-supervised learning, active learning, online learning, etc.

1 Course Overview

2 Introduction

3 Supervised Learning

Linear regression

Analytical solution

4 Optimization

5 Polynomial Regression

6 References

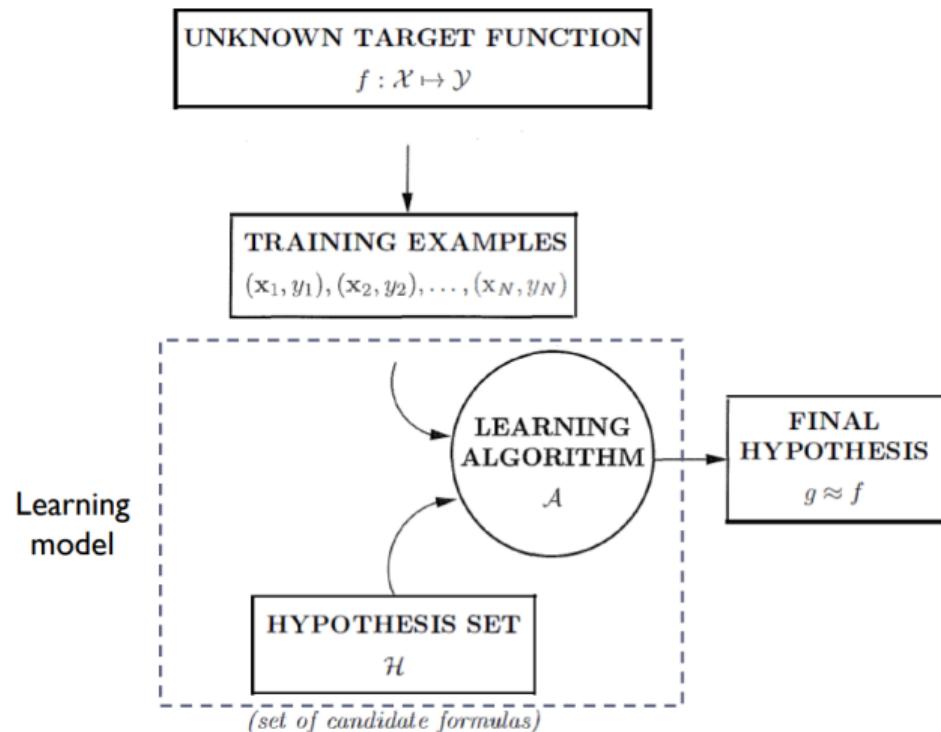
Supervised Learning

- **Definition:** A form of machine learning where the model learns from labeled data $\{(x_i, y_i)\}$ to predict an output y given an input x .
- **Goal:** Estimate a function $f: \mathbb{R}^D \rightarrow \mathbb{R}$, such that $y = f(x) + \epsilon$, where ϵ is noise.

Components of (Supervised) Learning

- **Unknown target function:** $f: \mathcal{X} \rightarrow \mathcal{Y}$
 - Input space: \mathcal{X}
 - Output space: \mathcal{Y}
 - **Training data:** $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$
 - **Pick a formula** $g: \mathcal{X} \rightarrow \mathcal{Y}$ **that approximates the target function** f
 - selected from a set of hypotheses \mathcal{H}

Components of (Supervised) Learning (cont.)



Supervised Learning: Regression vs. Classification

- **Regression:** predict a continuous target variable
 - E.g., $y \in [0, 1]$
- **Classification:** predict a discrete target variable
 - E.g., $y \in \{1, 2, \dots, C\}$

1 Course Overview

2 Introduction

③ Supervised Learning

Linear regression

Analytical solution

4 Optimization

5 Polynomial Regression

6 References

Solution Components - Learning Model

- The **Learning Model** consists of:
 - **Hypothesis Set:** Defines the possible functions $\mathcal{H} = \{h(x, \theta) | \theta \in \Theta\}$, where $h(x, \theta)$ represents candidate functions and θ is the learning parameters of problem.
 - **Learning Algorithm:** Find $\theta^* \in \Theta$ such that $h(x, \theta^*) \approx f(x)$.
- Both work together to map inputs x to outputs y with minimized error.
- In other words, θ^* is best parameters to predict outputs using chosen hypothesis.

Hypothesis Space Overview

- **Hypothesis (h):** A mapping from input space \mathcal{X} to output space \mathcal{Y} .
- **Linear Regression Hypothesis:**

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + \cdots + w_D x_D = \mathbf{w}^\top \mathbf{x}$$

- **Input Vector \mathbf{x} :**

$$\mathbf{x} = [x_0 = 1, x_1, x_2, \dots, x_D]$$

- **Parameter Vector \mathbf{w} :**

$$\mathbf{w} = [w_0, w_1, w_2, \dots, w_D]$$

Linear Hypothesis Representation

- **Linear Hypothesis Space:**
 - **Simplest form:** Linear combination of input features.

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^D w_i x_i$$

- **Linear Hypothesis Examples:**
 - **Single Variable:** $h_{\mathbf{w}}(x) = w_0 + w_1 x$
 - **Multivariate:** $h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_D x_D$

Understanding Cost Functions

- In **hypothesis space**, we select a function $h(x; \mathbf{w})$ to approximate the true relationship between input x and output y .
- The objective is to minimize the difference between predicted values $h(x)$ and actual values y .
- This difference is quantified using **cost functions**, which guide us in choosing the optimal hypothesis.

What is a Cost Function?

- A **cost function** measures how well the hypothesis $h(x; \mathbf{w})$ fits the training data.
- In regression problems, the most common error function is the **Squared Error (SE)**:

$$SE: \left(y^{(i)} - h(x^{(i)}; \mathbf{w}) \right)^2$$

- Cost function should measure all predictions. Thus a choice could be **Sum of Squared Errors (SSE)**:

$$J(\mathbf{w}) = \sum_{i=1}^N \left(y^{(i)} - h(x^{(i)}; \mathbf{w}) \right)^2$$

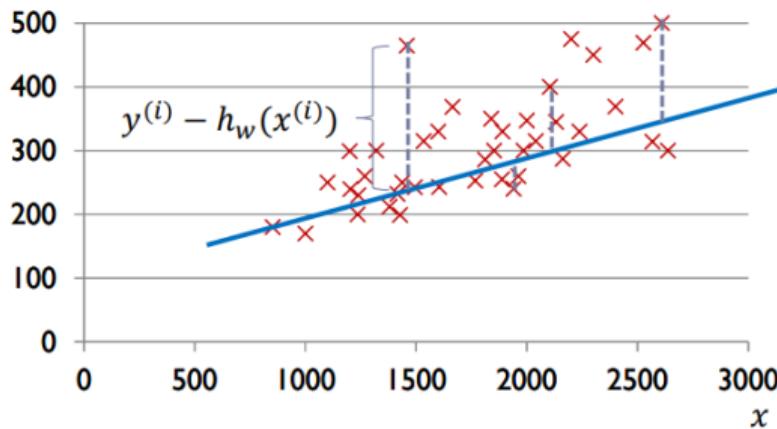
- **Objective:** Minimize the cost function to find the best parameters \mathbf{w} .

SSE: Sum of Squared Errors

- **SSE** is widely used due to its simplicity and differentiability.
- Intuitively, it represents the squared distance between predicted and true values.
- Penalizes larger errors more severely than smaller ones (due to the square).
- For linear regression, it can be written as:

$$SSE = \sum_{i=1}^N \left(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)} \right)^2$$

How to measure the error



$$J(w) = \sum_{i=1}^n \left(y^{(i)} - h_w(x^{(i)}) \right)^2$$

$$= \sum_{i=1}^n \left(y^{(i)} - w_0 - w_1 x^{(i)} \right)^2$$

1 Course Overview

2 Introduction

3 Supervised Learning

Linear regression

Analytical solution

4 Optimization

5 Polynomial Regression

6 References

The learning algorithm

- **Objective:** Choose \mathbf{w} so as to minimize the $J(\mathbf{w})$
- **The learning algorithm:** optimization of the cost function
 - Explicitly taking the cost function derivative with respect to the w_i 's, and setting them to zero.
 - Parameters of the best hypothesis for the training set:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} J(\mathbf{w})$$

Cost function optimization: univariate

$$J(w) = \sum_{i=1}^n \left(y^{(i)} - w_0 - w_1 x^{(i)} \right)^2$$

- **Necessary conditions for the “optimal” parameter values:**

$$\frac{\partial J(w)}{\partial w_0} = 0$$

$$\frac{\partial J(w)}{\partial w_1} = 0$$

Optimality conditions: univariate

$$\frac{\partial J(w)}{\partial w_0} = \underbrace{\sum_{i=1}^n y^{(i)}}_Y - \underbrace{\sum_{i=1}^n w_0}_n - \underbrace{\sum_{i=1}^n w_1 x^{(i)}}_{\sum_{i=1}^n (w_1 x^{(i)})} = 0$$

$$J(w) = \sum_{i=1}^n \left(y^{(i)} - w_0 - w_1 x^{(i)} \right)^2 \Rightarrow (Y - w_1 X)/n = w_0$$

$$\frac{\partial J(w)}{\partial w_1} = \sum_{i=1}^n 2 \left(y^{(i)} - w_0 - w_1 x^{(i)} \right) x^{(i)} = 0$$

$$\Rightarrow n \sum_{i=1}^n w_1 x^{(i)} y^{(i)} - YX + X^2 w_1 - n \sum_{i=1}^n w_1 x^{(i)} x^{(i)} = 0$$

$$\frac{\partial J(w)}{\partial w_0} = \sum_{i=1}^n 2 \left(y^{(i)} - w_0 - w_1 x^{(i)} \right) = 0 \Rightarrow w_0 = \frac{n \sum_{i=1}^n w_1 x^{(i)} y^{(i)} - XY}{n \sum_{i=1}^n w_1 x^{(i)} x^{(i)} - X^2}$$

- A system of 2 linear equations

Linear regression example: TV Advertising and Sales

This is a real-world example of how businesses can use linear regression to make decisions about marketing budgets. The following table shows the amount of TV advertising budget spend and respective average sales of houses in Boston:

TV Advertising Spend (\$1000)	Sales (Units)
230.1	22.1
44.5	10.4
17.2	9.3
151.5	18.5
180.8	12.9

Table 1: TV Advertising Spend vs. Sales Dataset

Learn More: Advertising Sales Dataset

Linear regression example: TV Advertising and Sales (cont.)

- In this problem, Sales per TV advertising is need thus the amount spend is considered as input x and sales is considered as output y .
- Using linear regression, cost function can be written as:

$$J(w) = \sum_{i=1}^5 \left(y^{(i)} - w_0 - w_1 x^{(i)} \right)^2$$

- Applying necessary conditions for the optimal parameters:

$$\frac{\partial J(w)}{\partial w_1} = \sum_{i=1}^5 2 \left(y^{(i)} - w_0 - w_1 x^{(i)} \right) (-x^{(i)}) = 0 \implies 110863w_1 + 624.1w_0 - 10843.04 = 0$$

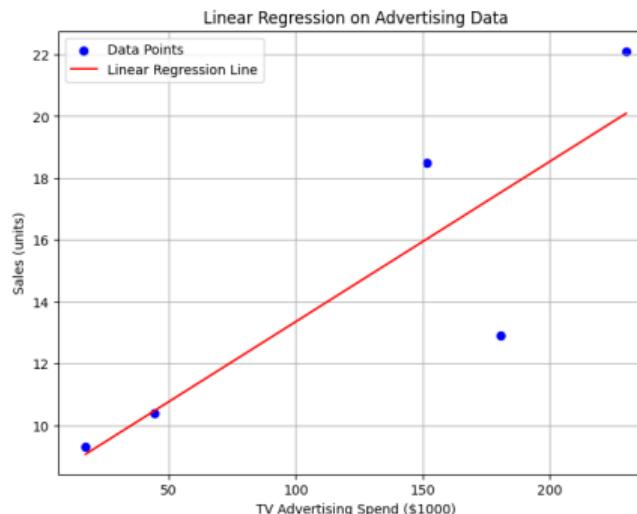
$$\frac{\partial J(w)}{\partial w_0} = \sum_{i=1}^5 2 \left(y^{(i)} - w_0 - w_1 x^{(i)} \right) (-1) = 0 \implies 624.1w_1 + 5w_0 - 73.2 = 0$$

Linear regression example: TV Advertising and Sales (cont.)

- Solving the system of two equations described, we get:

$$w_1 \approx 0.052$$

$$w_0 \approx 8.18$$



Cost function optimization: multivariate

- Remembering **SSE cost function** of multivariate linear regression

$$J(w) = \sum_{i=1}^n \left(y^{(i)} - h_w(x^{(i)}) \right)^2 = \sum_{i=1}^n \left(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)} \right)^2$$

- We usually write matrix form of the problem as follows:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ 1 & x_1^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

In which $x_m^{(i)}$ indicates m 'th feature of data point i

Cost function optimization: multivariate

- Using the matrix forms suggested, we can rewrite cost function:

$$J(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

- Explicitly taking the cost function derivative with respect to the \mathbf{w} , and setting them to zero:

$$\begin{aligned}\nabla_{\mathbf{w}} J(\mathbf{w}) &= -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ \nabla_{\mathbf{w}} J(\mathbf{w}) = 0 \implies \mathbf{X}^T \mathbf{X}\mathbf{w} &= \mathbf{X}^T \mathbf{y} \implies \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\end{aligned}$$

$$\begin{aligned}J(\mathbf{w}) &= \frac{1}{2n} \left(\mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} \right) \\ \nabla_{\mathbf{w}} J(\mathbf{w}) &= -2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w} \\ \nabla_{\mathbf{w}} J(\mathbf{w}) = 0 \implies \mathbf{X}^T \mathbf{X}\mathbf{w} &= \mathbf{X}^T \mathbf{y} \\ (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{X}\mathbf{w} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ \mathbf{w} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ &= \frac{1}{n} \mathbf{X}^T \mathbf{X}^{-1} \mathbf{X}^T \mathbf{y} \\ &= \frac{1}{n} \mathbf{y}^T \mathbf{X} \\ &= \mathbf{y}^T \mathbf{X} = \mathbf{X}^T \mathbf{y} \\ &= (\mathbf{X}^T \mathbf{y})^T \mathbf{X}\end{aligned}$$

Analytical solution: multivariate

- $(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ is called pseudo-inverse of matrix \mathbf{X}
- The matrix \mathbf{X} is often not square yet not invertible.
- The pseudo-inverse can be computed for any matrix, regardless of its shape.

Computational limitations of analytical solution

- **Scalability:** Analytical solutions do not scale well with very large datasets, making them impractical for big data applications.
- **Finding the inverse of a matrix:**
 - Simplest way of finding inverse: Gaussian elimination, having complexity of $O(n^3)$.
 - Other methods: LU decomposition, which also has a complexity of $O(n^3)$ but is more stable.
 - Numerical methods: Iterative methods like Conjugate Gradient, which can be more efficient for large, sparse matrices.

Practical limitations of analytical solution

- **Online learning:** Data observations are arriving in a continuous stream
 - Predictions must be made before seeing all of the data.
 - Analytical solutions require all data to be available upfront, which is not feasible in online learning scenarios.
 - Analytical methods are not adaptable to new data without re-computing the entire solution.

Cost function optimization

- **Another approach,**
 - Start from an initial guess and iteratively change w to minimize $J(w)$.
 - The gradient descent algorithm
 - **Steps:**
 - Start from w^0
 - Repeat:
 - Update w^t to w^{t+1} in order to reduce J
 - $t \leftarrow t + 1$
- until we hopefully end up at a minimum.

1 Course Overview

2 Introduction

3 Supervised Learning

4 Optimization

Gradient descent

Stochastic gradient descent

5 Polynomial Regression

6 References

1 Course Overview

2 Introduction

3 Supervised Learning

4 Optimization

Gradient descent

Stochastic gradient descent

5 Polynomial Regression

6 References

Gradient Descent

- In each step, takes steps proportional to the negative of the gradient vector of the function at the current point w^t :

$$w^{t+1} = w^t - \eta \nabla J(w^t)$$

- $J(w)$ **decreases fastest** if one goes from w^t in the direction of $-\nabla J(w^t)$
- Assumption:** $J(w)$ is defined and differentiable in a neighborhood of a point w^t
- Gradient ascent** takes steps proportional to (the positive of) the gradient to find a local maximum of the function.
- Continue to find:

$$w^* = \arg \min_w J(w)$$

Gradient Descent (cont.)

- Minimize $J(w)$

$$w^{t+1} = w^t - \eta \nabla_w J(w^t)$$

$$\nabla_w J(w) = \begin{bmatrix} \frac{\partial J(w)}{\partial w_1} \\ \vdots \\ \frac{\partial J(w)}{\partial w_d} \end{bmatrix}$$

$$J(w_0, w_1, w_2, \dots, w_D) = \frac{\nabla J(w_0, w_1, \dots, w_D)}{J(w_0, w_1, \dots, w_D)} = \begin{pmatrix} \frac{\partial J}{\partial w_0} \\ \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_D} \end{pmatrix}$$

Gradient Descent

Random \rightarrow W₀, W₁, ..., W_r

for steps in 100

$$\mathbf{w} = \mathbf{w} - \alpha \nabla$$

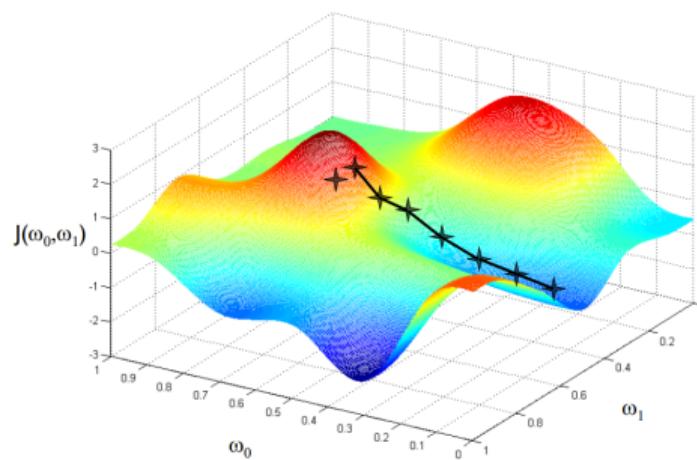
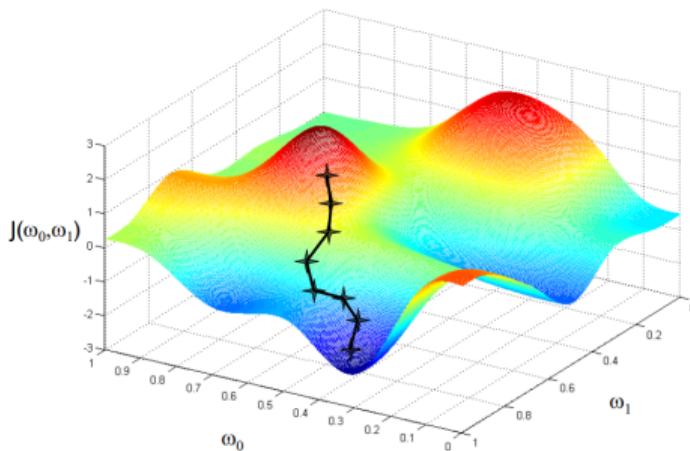
Learning rate

- If η is small enough, then $J(w^{t+1}) \leq J(w^t)$.
 - η can be allowed to change at every iteration as η_t .

Gradient descent disadvantages

- **Local minima problem**
- **However, when J is convex, all local minima are also global minima \Rightarrow gradient descent can converge to the global solution.**

Problem of gradient descent with non-convex cost functions



Figures adapted from slides of Andrew Ng, Machine Learning course, Stanford.

Cost function optimization

- Weight update rule having $h_w(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ is as follows:

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \eta \sum_{i=1}^n (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}) \mathbf{x}^{(i)}$$

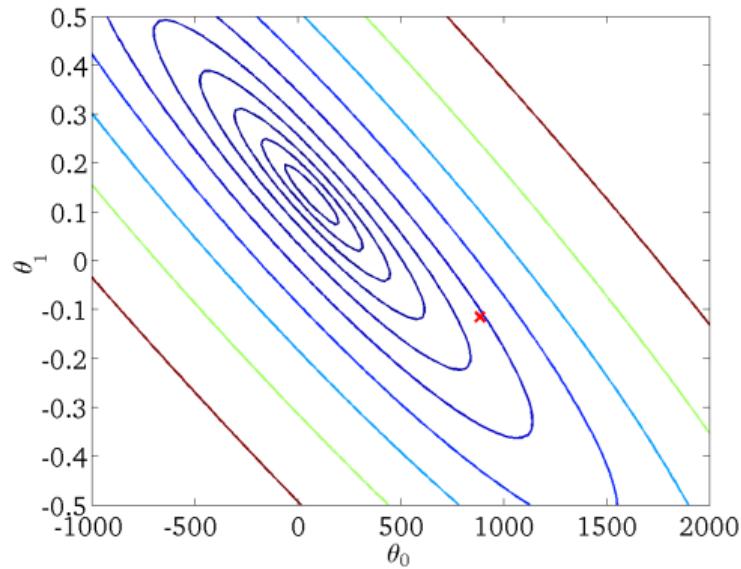
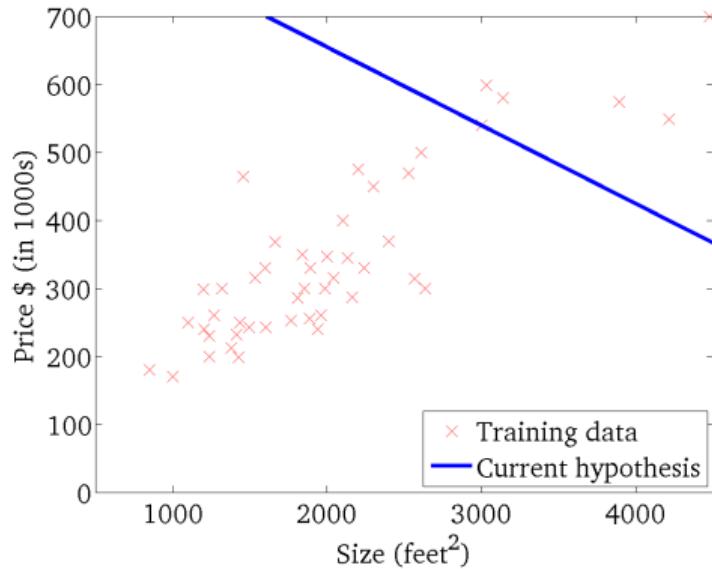
→ Gradient Descent

← Gradient Ascent

- η : too small ⇒ gradient descent can be slow.
- η : too large ⇒ gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

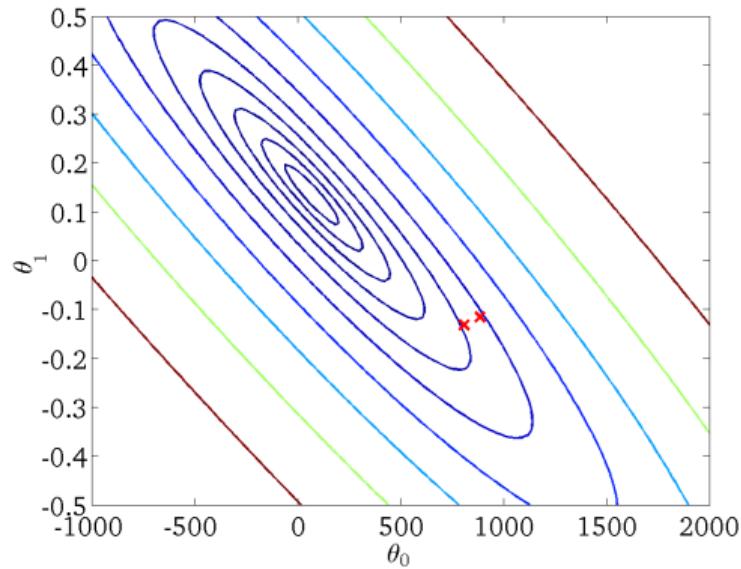
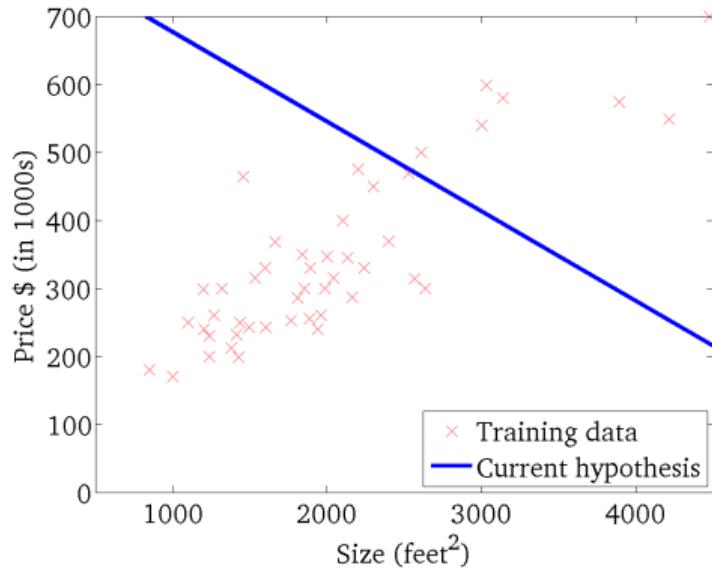
by
rate

Gradient descent overview



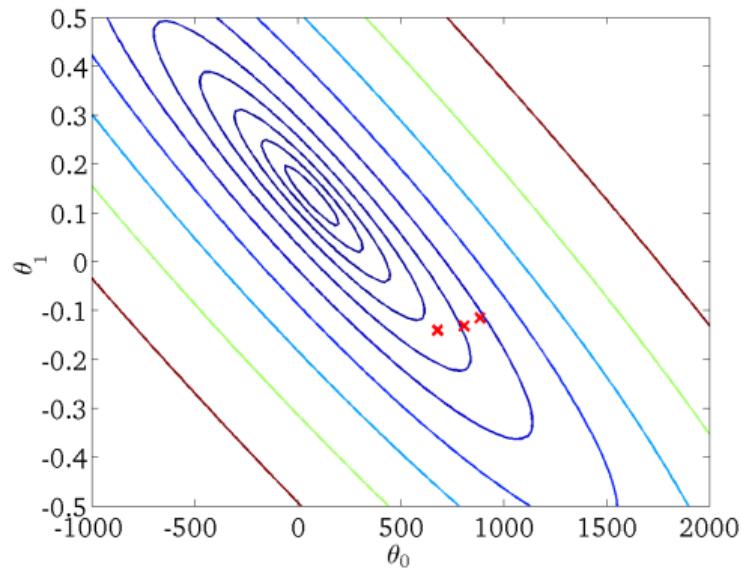
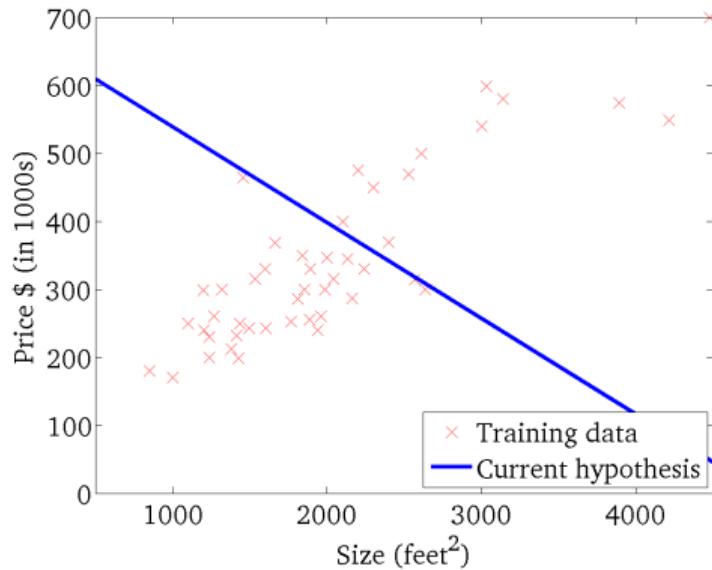
Figures adapted from slides of Andrew Ng, Machine Learning course, Stanford.

Gradient descent overview (cont.)



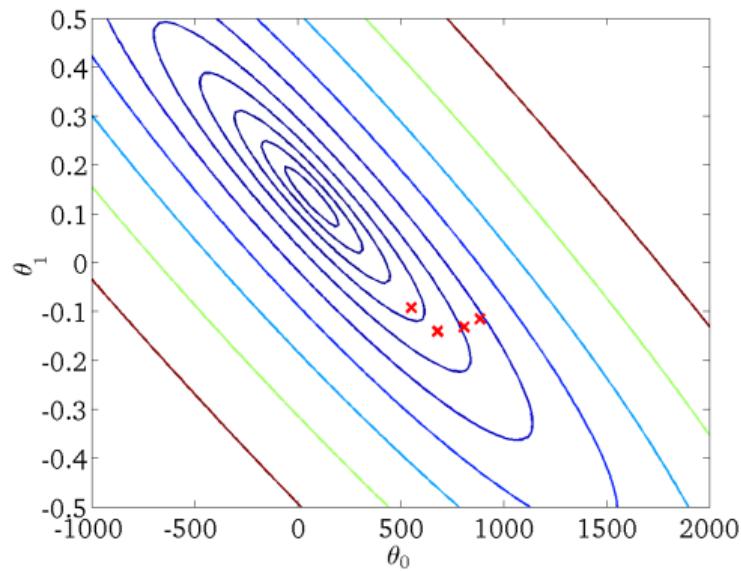
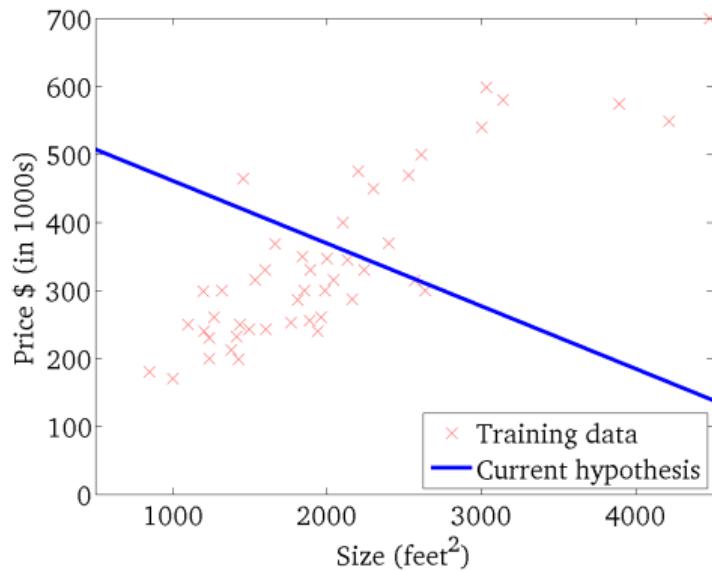
Figures adapted from slides of Andrew Ng, Machine Learning course, Stanford.

Gradient descent overview (cont.)



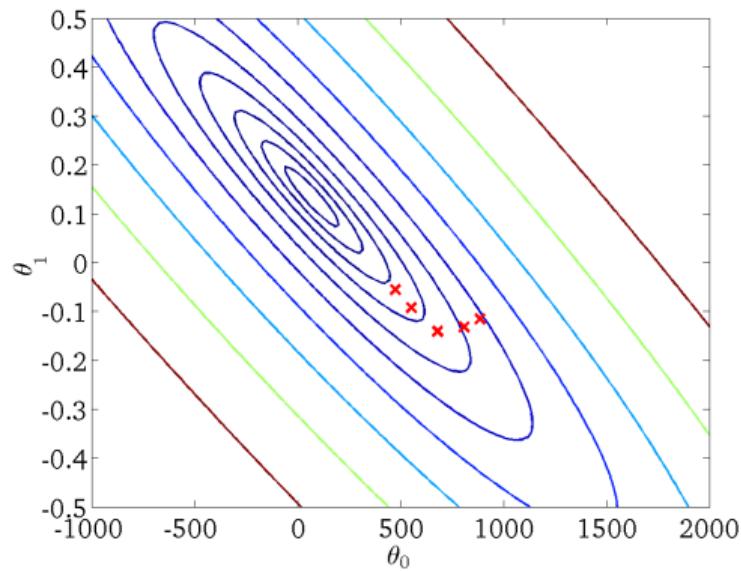
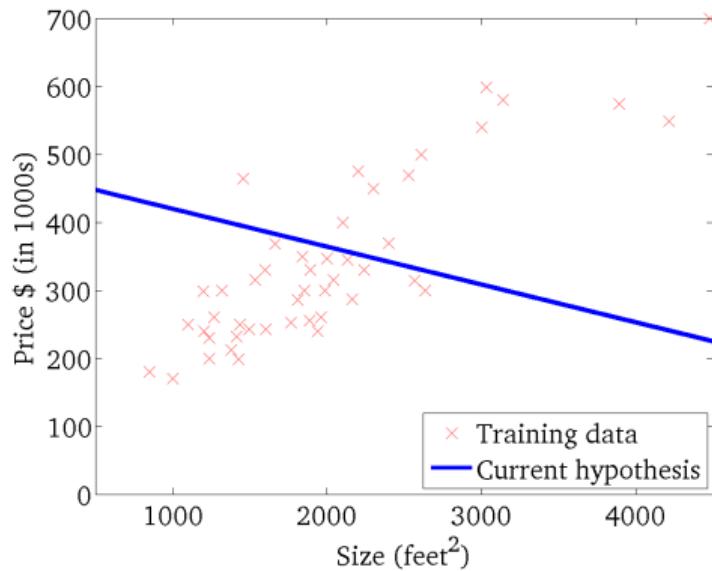
Figures adapted from slides of Andrew Ng, Machine Learning course, Stanford.

Gradient descent overview (cont.)



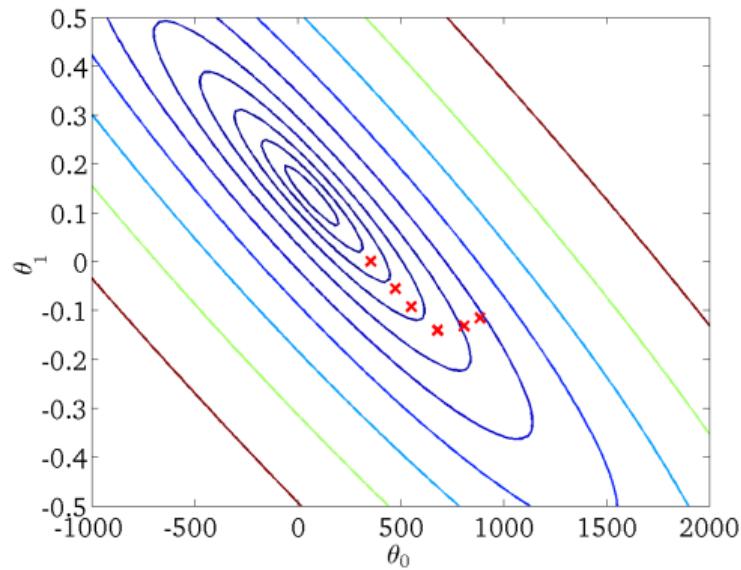
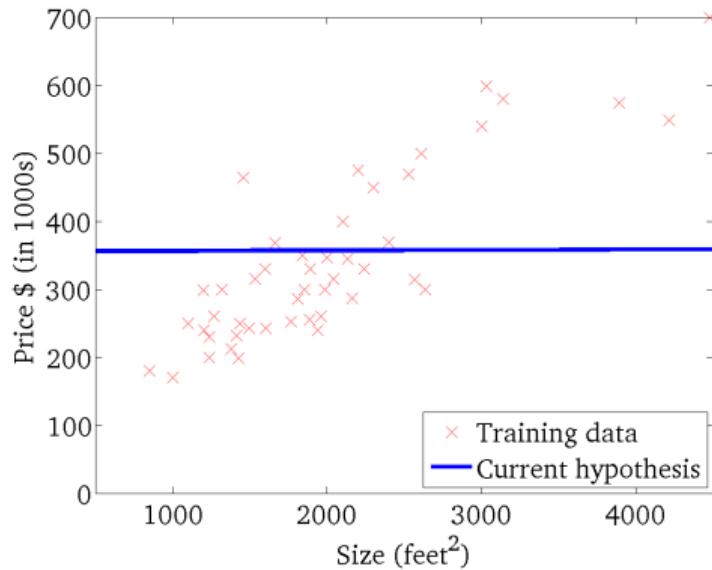
Figures adapted from slides of Andrew Ng, Machine Learning course, Stanford.

Gradient descent overview (cont.)



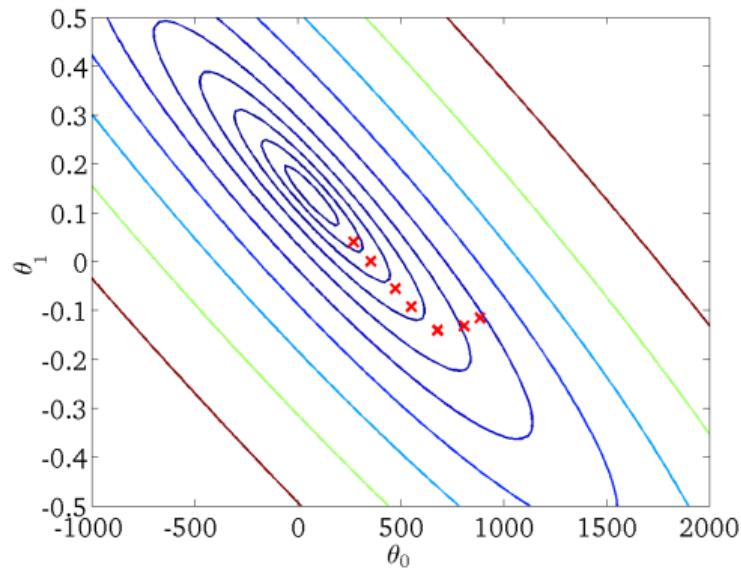
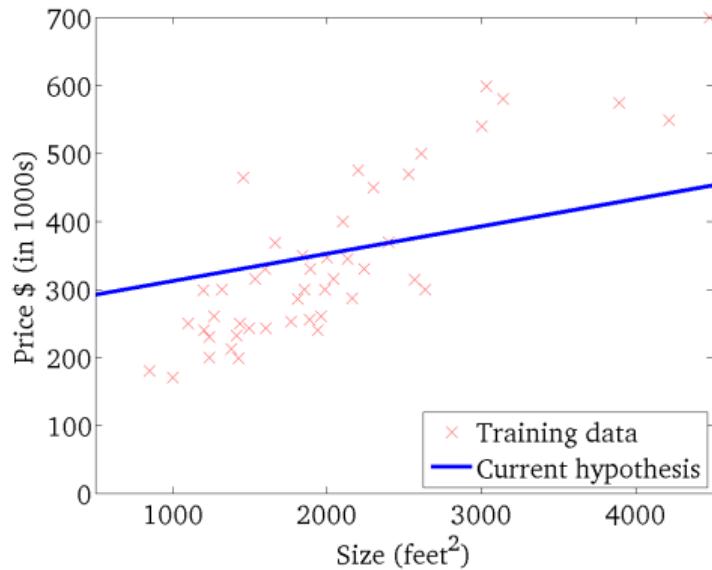
Figures adapted from slides of Andrew Ng, Machine Learning course, Stanford.

Gradient descent overview (cont.)



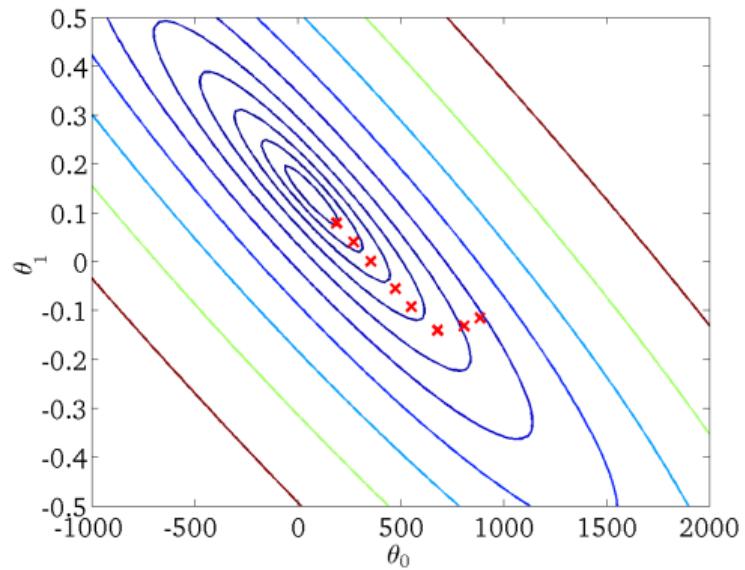
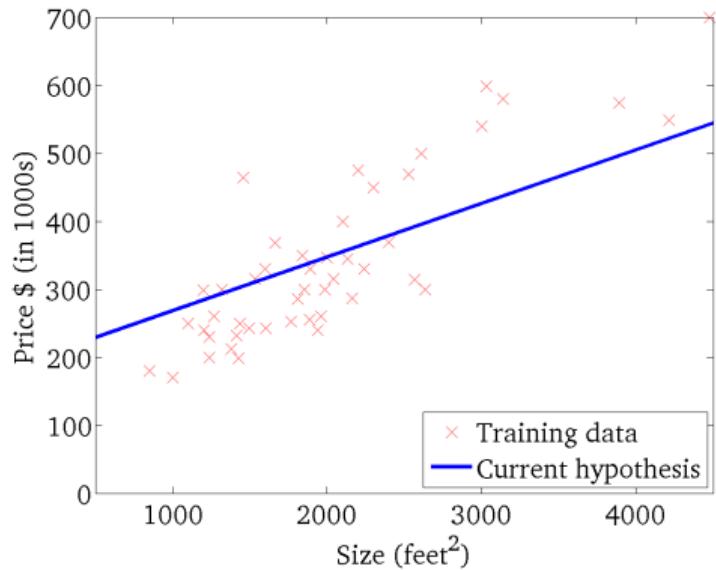
Figures adapted from slides of Andrew Ng, Machine Learning course, Stanford.

Gradient descent overview (cont.)



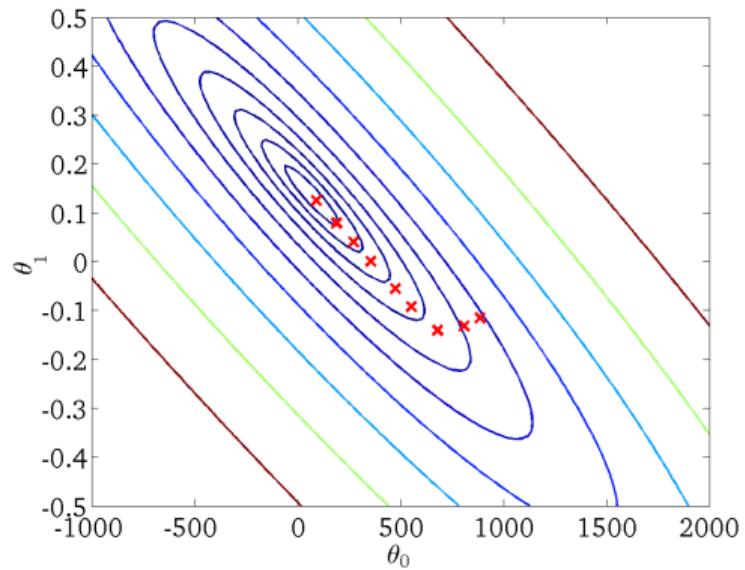
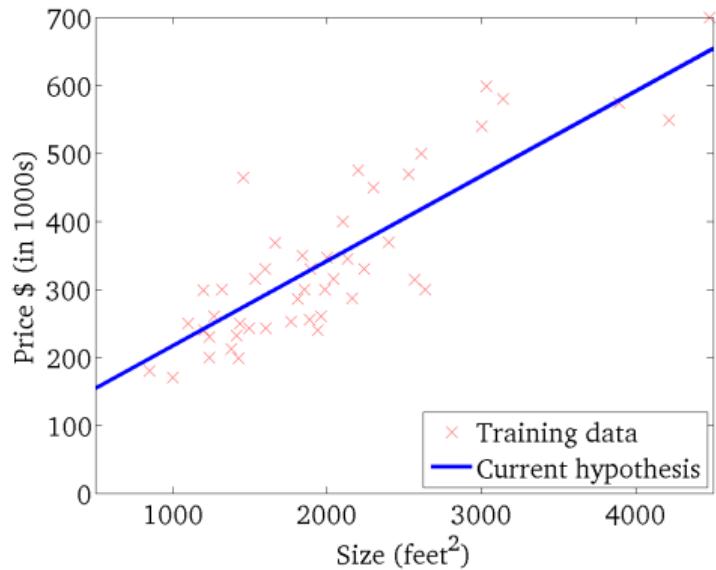
Figures adapted from slides of Andrew Ng, Machine Learning course, Stanford.

Gradient descent overview (cont.)



Figures adapted from slides of Andrew Ng, Machine Learning course, Stanford.

Gradient descent overview (cont.)



Figures adapted from slides of Andrew Ng, Machine Learning course, Stanford.

1 Course Overview

2 Introduction

3 Supervised Learning

4 Optimization

Gradient descent

Stochastic gradient descent

5 Polynomial Regression

6 References

Variants of gradient descent

- **Batch gradient descent:** processes the entire training set in one iteration
 - It can be computationally costly for large data sets and practically impossible for some applications (such as online learning).
- **Mini-batch gradient descent:** processes small, random subsets (mini-batches) of the training set in each iteration
 - Balances the efficiency of batch gradient descent.
- **Stochastic gradient descent:** processes one training example per iteration
 - Updates the model parameters more frequently, which can lead to faster convergence.

Stochastic gradient descent

- **Example:** Linear regression with SSE cost function

$$J(\mathbf{w}) = \sum_{i=1}^n J^{(i)}(\mathbf{w})$$

$$J^{(i)}(\mathbf{w}) = \left(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)} \right)^2$$

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla_{\mathbf{w}} J^{(i)}(\mathbf{w})$$

$$\implies \mathbf{w}^{t+1} = \mathbf{w}^t + \eta \left(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)} \right) \mathbf{x}^{(i)}$$

In which $x^{(i)}$ indicates the i 'th observation arrived.

Stochastic gradient descent: online learning

- Often, stochastic gradient descent gets close to the minimum much faster than batch gradient descent.
- Note however that it may never converge to the minimum, and the parameters will keep oscillating around the minimum of the cost function;
 - In practice, most of the values near the minimum will be reasonably good approximations to the true minimum.

1 Course Overview

2 Introduction

3 Supervised Learning

4 Optimization

5 Polynomial Regression

6 References

Limitations of linear regression

- It is possible that the best fitted line is still far off the real pattern of samples:

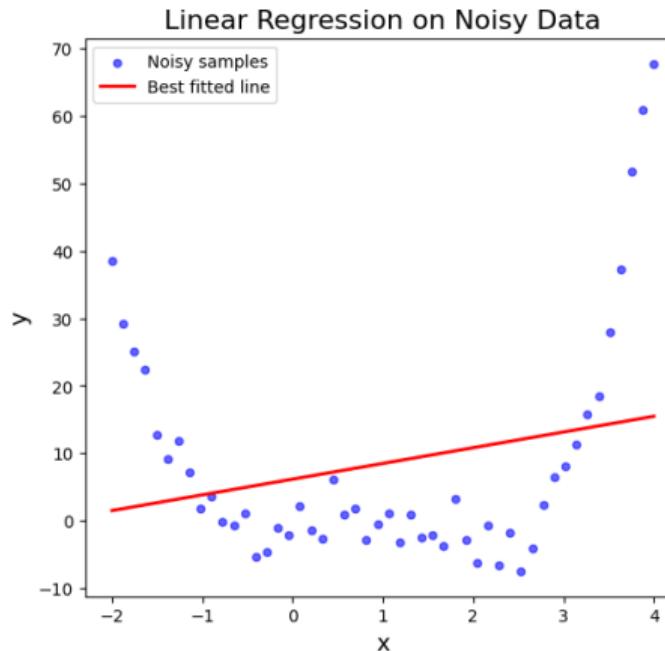


Figure 1: No line can be fitted to generalize on these samples

Beyond Linear Regression

- How can we extend linear regression to model non-linear relationships?
 - **Transforming Data Using Basis Functions:**
 - Basis functions allow us to transform the original features into a new feature space.
 - Common basis functions include polynomial and Gaussian functions.
 - **Learning a Linear Regression on Transformed Features:**
 - By applying linear regression to the transformed feature vectors, we can model complex, non-linear relationships.
 - This approach maintains the simplicity and interpretability of linear regression while extending its flexibility.

Beyond Linear Regression: Polynomial Regression

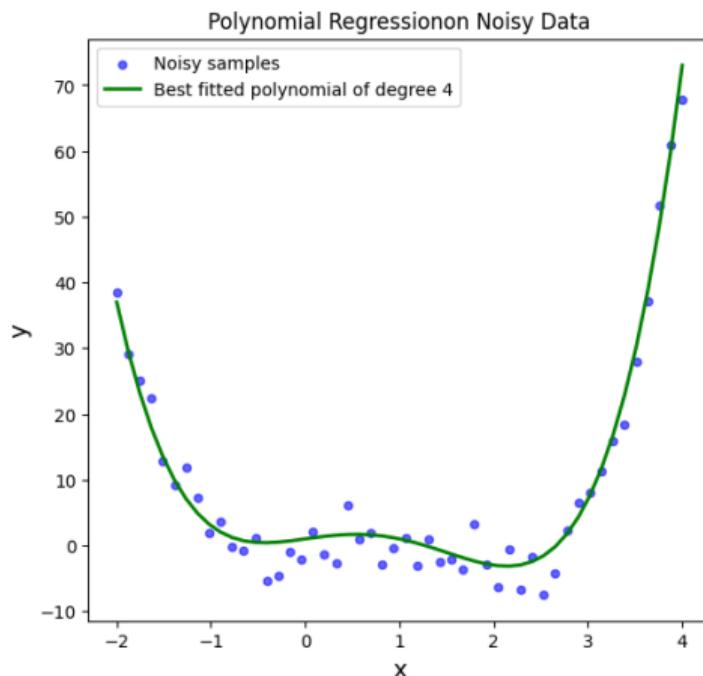


Figure 2: Best fitted polynomial of degree 4 can generalize well on samples

Beyond Linear Regression: change of basis

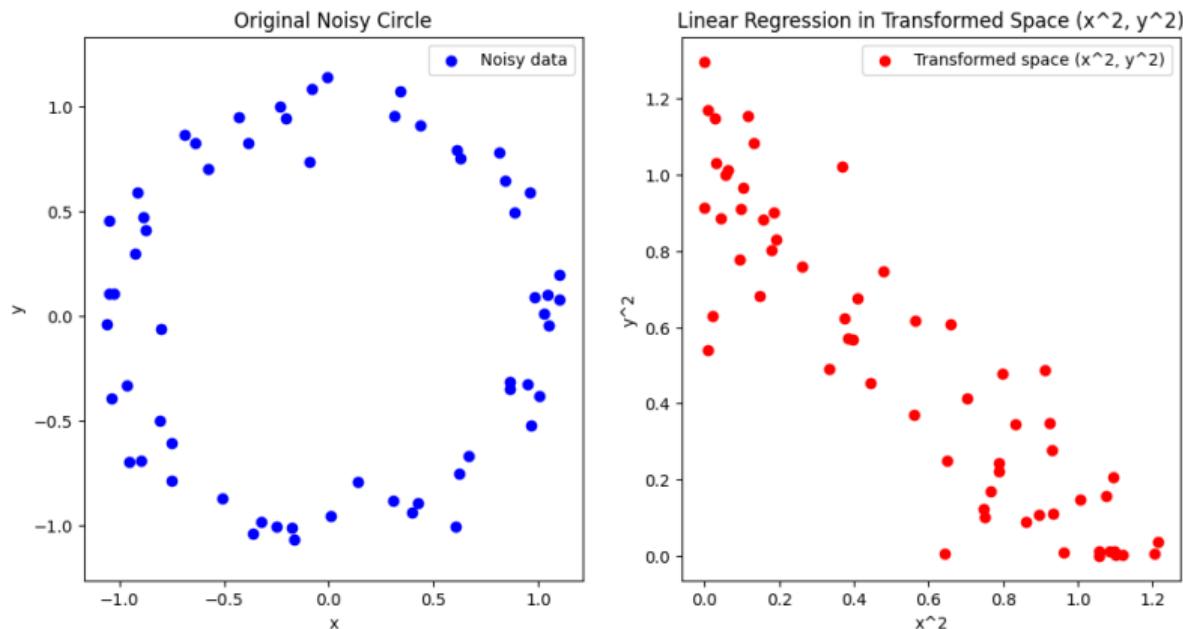


Figure 3: Changing the basis of $[1, x, y]$ to $[1, x^2, y^2]$, we can use linear regression

Polynomial regression: Univariate

- **Polynomial Regression Hypothesis:** m'th order regression

$$h(x; \mathbf{w}) = w_0 + w_1 x^1 + \cdots + w_{m-1} x^{m-1} + w_m x^m$$

- **Objective:** Fit a polynomial of degree m to data points.

Polynomial Regression: Univariate (cont.)

- Similar to what we did for univariate linear regression, we can define:

$$\mathbf{X}' = \begin{bmatrix} 1 & x^{(1)^1} & \dots & x^{(1)^m} \\ 1 & x^{(2)^1} & \dots & x^{(2)^m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x^{(n)^1} & \dots & x^{(n)^m} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} \hat{w}_0 \\ \hat{w}_1 \\ \vdots \\ \hat{w}_m \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

In which $x^{(i)}$ indicates the i -th data point.

Polynomial regression analytical solution: Univariate

Rewriting the SSE cost function using matrix form we have:

$$J(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}'\mathbf{w}\|_2^2$$

Analytical solution: It has closed form solution as

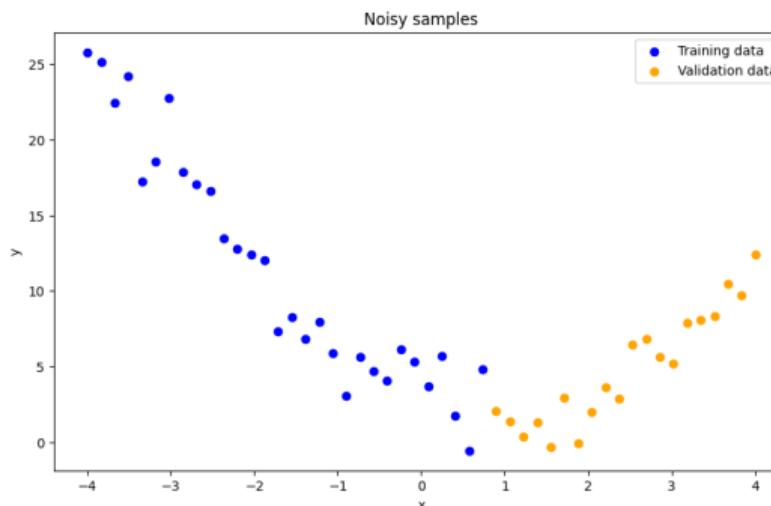
$$\hat{\mathbf{w}} = (\mathbf{X}'^T \mathbf{X}')^{-1} \mathbf{X}'^T \mathbf{y}$$

Training and Validation

- To better distinct between linear regression and polynomial regression, we will show that linear model can not generalize well.
- Assume the samples are split into two subsets. **Train dataset** which is used to train a regression model. As well as a **Validation dataset** to find the best regression model for an application.
- If a model can generalize well on validation set, it can be a good candidate.

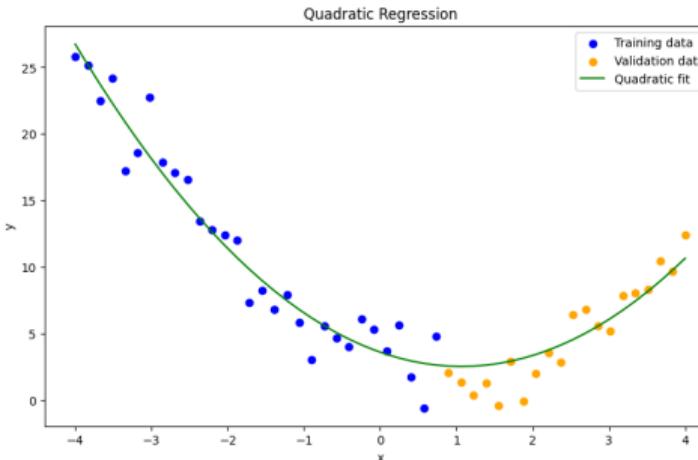
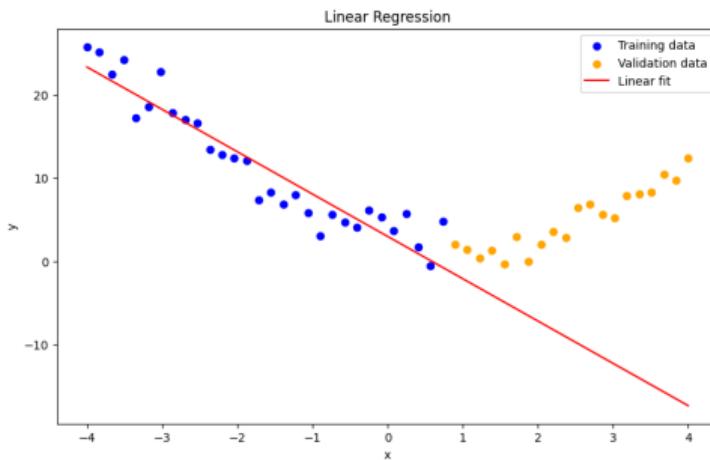
Polynomial regression: example

- Consider the following noisy samples



Splitting samples to train and validation sets

Quadratic regression vs Linear regression



Fitting both quadratic and linear regression shows the power of polynomial regression on generalizing for more complex patterns.

1 Course Overview

2 Introduction

3 Supervised Learning

4 Optimization

5 Polynomial Regression

6 References

Contributions

- **This slide has been prepared thanks to:**
 - Arshia Gharooni
 - Mahan Bayaghhi

- [1] C. M., *Pattern Recognition and Machine Learning*.
Information Science and Statistics, New York, NY: Springer, 1 ed., Aug. 2006.
- [2] M. Soleymani Baghshah, “Machine learning.” Lecture slides.
- [3] A. Ng and T. Ma, *CS229 Lecture Notes*.
- [4] T. Mitchell, *Machine Learning*.
McGraw-Hill series in computer science, New York, NY: McGraw-Hill Professional, Mar. 1997.
- [5] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data: A Short Course*.
New York, NY: AMLBook, 2012.
- [6] S. Goel, H. Bansal, S. Bhatia, R. A. Rossi, V. Vinay, and A. Grover, “CyCLIP: Cyclic Contrastive Language-Image Pretraining,” *ArXiv*, vol. abs/2205.14459, May 2022.

Machine Learning (CE 40477)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

September 30, 2024



1 Generalization

2 Probabilistic regression

3 References

1 Generalization

2 Probabilistic regression

3 References

Generalization Overview

Main Idea: The ability of a model to perform well on unseen data

- **Training Set:** $D = \{(x_i, y_i)\}_{i=1}^n$
 - **Test Set:** New data not seen during training
 - **Cost Function:** Measures how well the model fits data

$$J(w) = \sum_{i=1}^n (y^{(i)} - h_w(x^{(i)}))^2$$

- **Objective:** Minimize the cost function on unseen data (generalization error)

Expected Test Error

Definition: Expected performance on unseen data

- Test data sampled from the same distribution $p(x, y)$

$$J(w) = \mathbb{E}_{p(x,y)}[(y - h_w(x))^2]$$

- Approximate using test set $\hat{J}(w)$
 - Generalization error is the gap between training and test performance.

Training vs Test Error

Key Concept: Training error measures fit on known data, test error on unseen data

- **Training (empirical) error:**

$$J_{\text{train}}(w) = \frac{1}{n} \sum_{i=1}^n \left(y^{(i)} - h_w(x^{(i)}) \right)^2 \quad \text{Approximate Error}$$

- **Test error:**

$$J_{\text{test}}(w) = \frac{1}{m} \sum_{i=1}^m \left(y_{\text{test}}^{(i)} - h_w(x_{\text{test}}^{(i)}) \right)^2 \quad \text{Generalization error}$$

- **Goal:** Minimize the test error (generalization).

Overfitting Definition

Concept: A model fits the training data well but performs poorly on the test set

$$J_{\text{train}}(w) \ll J_{\text{test}}(w)$$

- Causes: Model too complex, high variance
- Consequence: Captures noise in training data, fails on unseen data

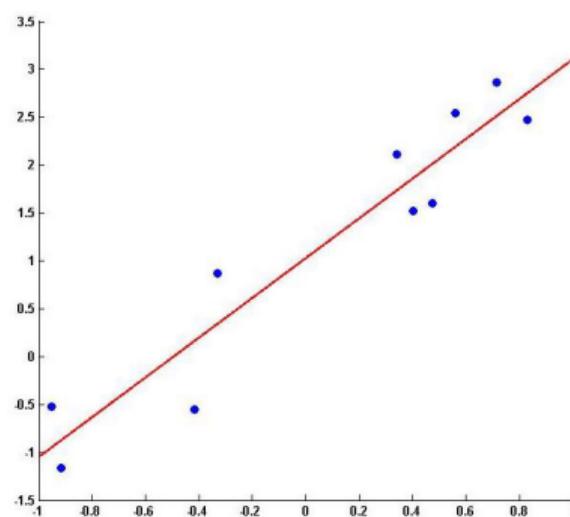
Underfitting Definition

Concept: The model is too simple and cannot capture the structure of the data

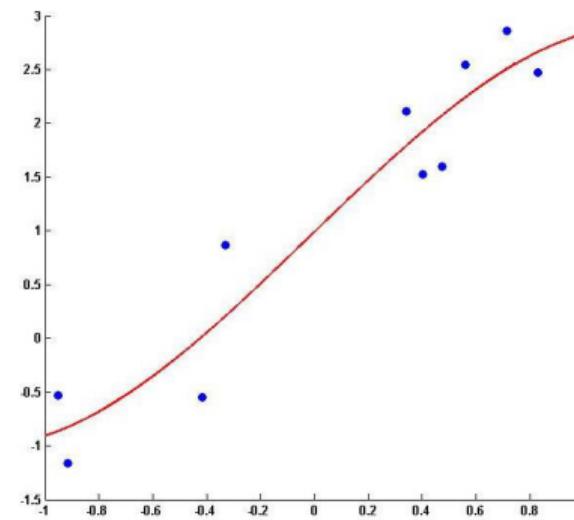
$$J_{\text{train}}(w) \approx J_{\text{test}}(w) \gg 0$$

- Causes: Model lacks complexity, high bias
- Consequence: Poor fit on both training and test data

Generalization: polynomial regression

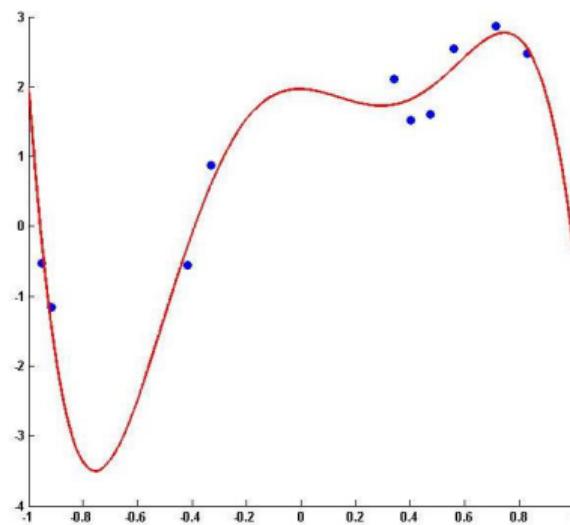


Degree of 1

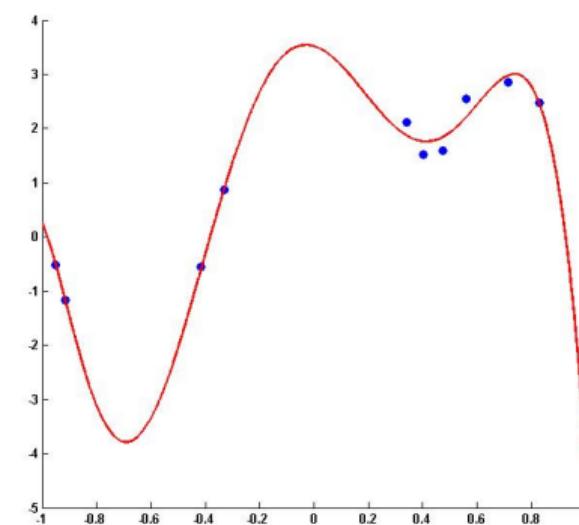


Degree of 3

Overfitting: polynomial regression

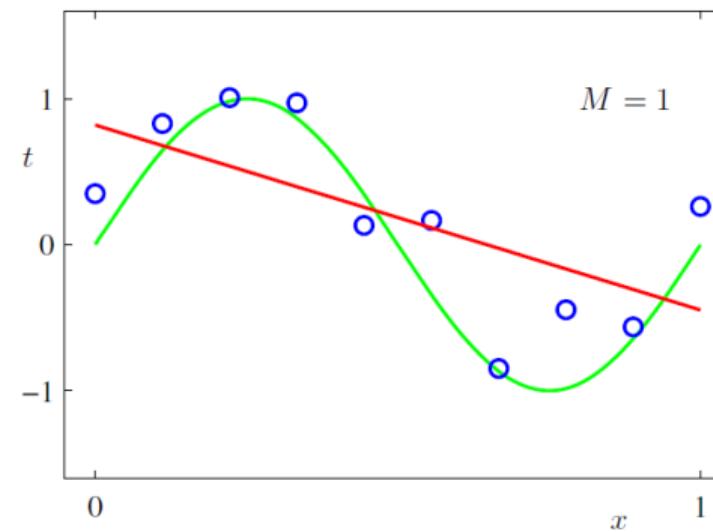
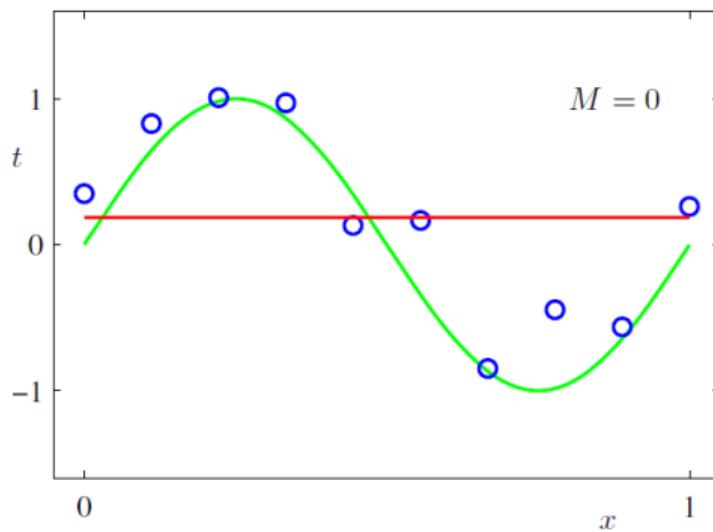


Degree of 5

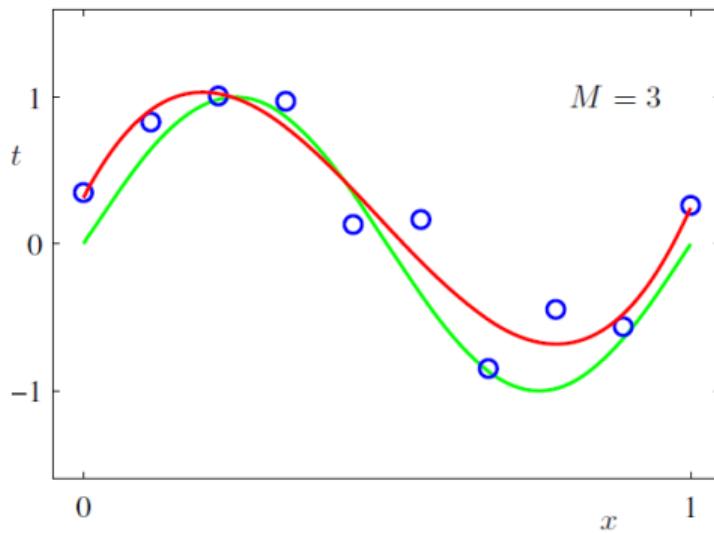
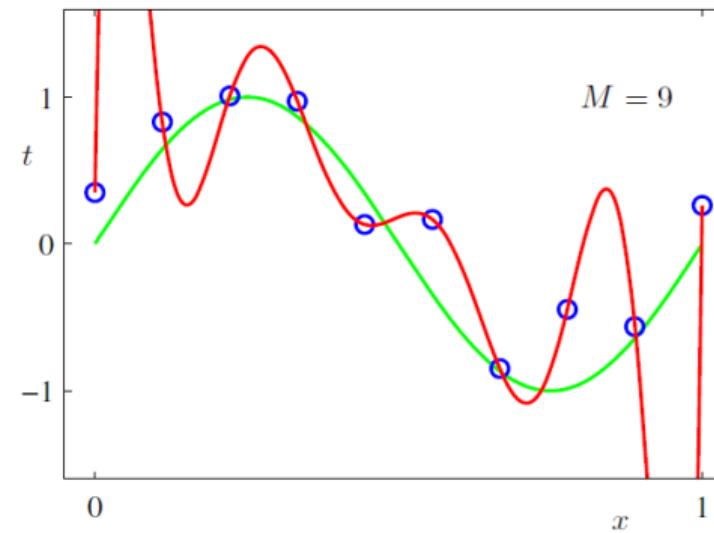


Degree of 7

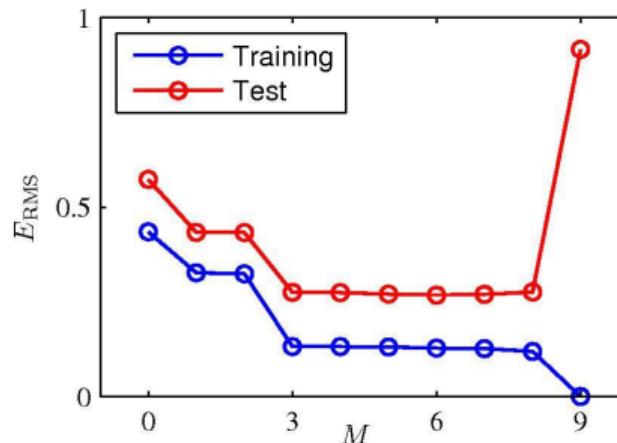
Polynomial regression with various degrees: example



Polynomial regression with various degrees: example (cont.)

 $M = 3$  $M = 9$

Root mean squared error



$$E_{RMS} = \sqrt{\frac{\sum_{i=1}^n (y^{(i)} - f(\mathbf{x}^{(i)}; \mathbf{w}))^2}{n}}$$

Bias-Variance Decomposition

Generalization error decomposition:

$$\mathbb{E}[(y - h_w(x))^2] = (\text{Bias})^2 + \text{Variance} + \text{Noise}$$

- **Bias:** Error due to simplifying assumptions in the model

$$\text{Bias}(x) = \mathbb{E}[h_w(x)] - f(x)$$

- **Variance:** Sensitivity of the model to training data

$$\text{Variance}(x) = \mathbb{E}[(h_w(x) - \mathbb{E}[h_w(x)])^2]$$

- **Noise:** Irreducible error from the inherent randomness in data

Bias-Variance Decomposition Proof

Assume $f(x)$ is the ground truth and observation y is a noisy observation $y = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We start with the definition of the expected squared error, which is:

$$\begin{aligned}\mathbb{E}_{data} \left[(\hat{f}(x) - y)^2 \right] &= \mathbb{E}_{data} \left[(\hat{f}(x) - f(x) + \epsilon)^2 \right] \\ &= \mathbb{E} \left[(\hat{f}(x) - f(x))^2 - 2\epsilon(\hat{f}(x) - f(x)) + \epsilon^2 \right]\end{aligned}$$

Since we assume the noise ϵ has zero mean and variance σ^2 , the term $\mathbb{E}[\epsilon] = 0$, and thus:

$$\mathbb{E}[\epsilon^2] = \sigma^2$$

Since $\mathbb{E}[\epsilon] = 0$ and ϵ is independent of the parenthesis, we can write:

$$\mathbb{E}[-2\epsilon(\hat{f}(x) - f(x))] = 0$$

Bias-Variance Decomposition Proof (cont.)

Now, we decompose the squared difference $(\hat{f}(x) - f(x))^2$ as follows:

$$\mathbb{E} \left[(\hat{f}(x) - f(x))^2 \right] = \mathbb{E} \left[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)] + \mathbb{E}[\hat{f}(x)] - f(x))^2 \right]$$

Expanding this further:

$$= \mathbb{E} \left[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2 \right] + \mathbb{E} \left[(\mathbb{E}[\hat{f}(x)] - f(x))^2 \right] + 2\mathbb{E} \left[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]) (\mathbb{E}[\hat{f}(x)] - f(x)) \right]$$

Since $\mathbb{E}[\epsilon A] = \mathbb{E}[\epsilon] \mathbb{E}[A]$, A and ϵ are independent and $\mathbb{E}[\epsilon] = 0$ we have $\mathbb{E}[\epsilon A] = 0$ thus:

$$\mathbb{E}[\mathbb{E}[\hat{f}(x)] - \hat{f}(x)] = \mathbb{E}[\hat{f}(x)] - \mathbb{E}[\hat{f}(x)] = 0$$

Bias-Variance Decomposition Proof (cont.)

Thus, the expected squared error becomes:

$$\mathbb{E}_{\text{data}} \left[(\hat{f}(x_n) - y)^2 \right] = \text{Variance} + \text{Bias}^2 + \sigma^2$$

where:

- **Variance** is $\mathbb{E} \left[(\hat{f}(x) - \mathbb{E} [\hat{f}(x)])^2 \right]$
- **Bias** is $\mathbb{E} [\hat{f}(x)] - f(x)$
- **Noise** is σ^2

High Bias in Simple Models

Explanation: Simple models, such as linear regression, often underfit

$$h_w(x) = w_0 + w_1 x$$

- Bias remains large even with infinite data

$$\text{Bias}^2 \gg \text{Variance}$$

- Leads to large generalization error

High Variance in Complex Models

Explanation: Complex models tend to overfit

$$h_w(x) = w_0 + w_1x + w_2x^2 + \cdots + w_mx^m$$

- Variance dominates when the model is too complex

Variance \gg Bias

- Fits noise, leading to high test error

Bias-Variance Tradeoff

Tradeoff: Balancing between bias and variance is key for optimal performance

- Low complexity: High bias, low variance
- High complexity: Low bias, high variance

Regularization

Purpose: Prevent overfitting by penalizing large weights

$$J_{\lambda}(w) = J(w) + \lambda R(w)$$

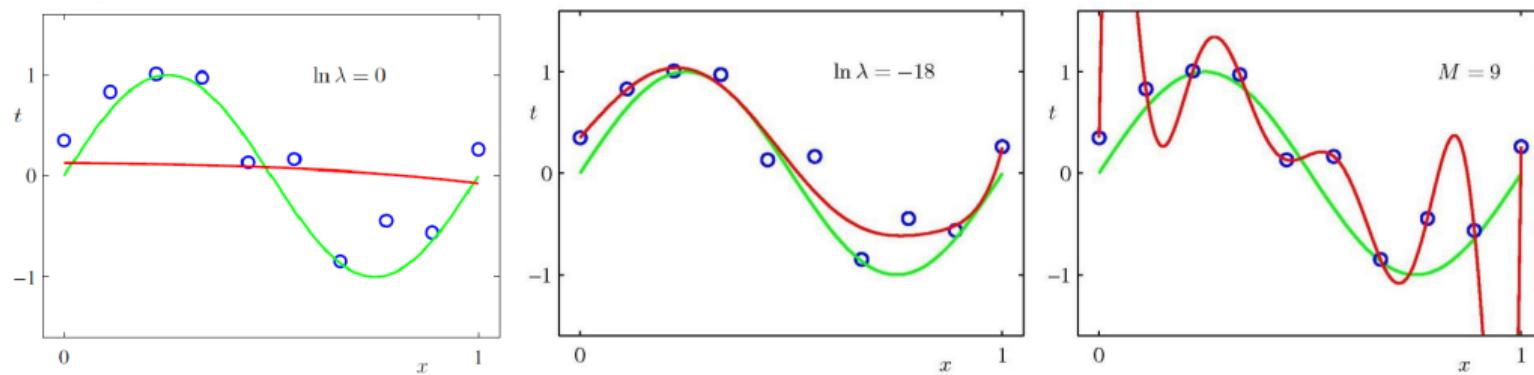
- Common regularizers: L1 and L2 norms
- λ controls the balance between fit and simplicity

Effect of Regularization Parameter λ

Balancing Fit and Complexity:

$$J_{\lambda}(w) = J(w) + \lambda \sum_{j=1}^m w_j^2 = J(w) + \lambda \mathbf{w}^T \mathbf{w}$$

- Large λ : Forces smaller weights, reduces complexity, increases bias, decreases variance
- Small λ : Allows larger weights, increases complexity, reduces bias, increases variance

Effect of Regularization parameter λ 

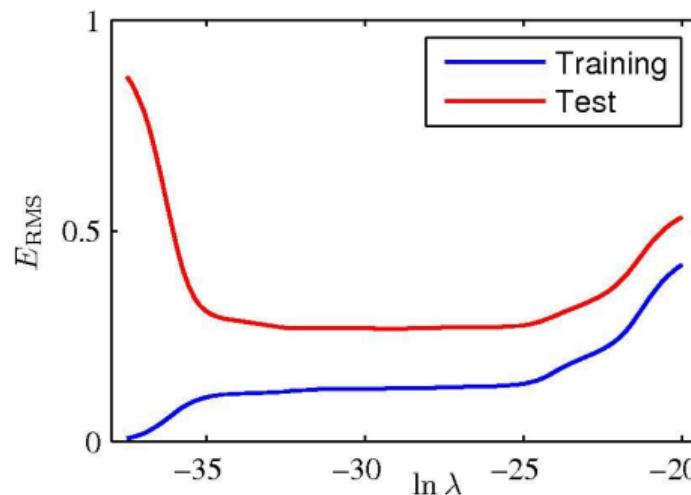
$$J_\lambda(\mathbf{w}) = \sum_{i=1}^n (t^{(n)} - f(\mathbf{x}^{(n)}; \mathbf{w}))^2 + \lambda \mathbf{w}^T \mathbf{w}$$

$$f(\mathbf{x}^{(n)}; \mathbf{w}) = w_0 + w_1 x + \cdots + w_9 x^9$$

Effect of regularization on weights

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
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

Regularization parameter



- λ controls the effective complexity of the model
- hence the degree of overfitting

1 Generalization

2 Probabilistic regression

3 References

Introduction to Regression (Probabilistic Perspective)

- **Objective:** Model the relationship between input \mathbf{x} and output y .
- **Uncertainty:** Output y has an associated uncertainty modeled by a probability distribution.
- **Example:**

$$y = f(\mathbf{x}; \mathbf{w}) + \epsilon \quad , \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

- The goal is to learn $f(\mathbf{x}; \mathbf{w})$ to predict y .

Curve Fitting with Noise

- In real-world scenarios, observed output y is noisy.
- **Model: True output plus noise**

$$y = f(\mathbf{x}; \mathbf{w}) + \epsilon$$

- Noise represents unknown or unmodeled factors.
- **Example:** Predicting house prices based on features with inherent unpredictability.

Expected Value of Output

- Best Estimate: The conditional expectation of y given \mathbf{x} .

$$\mathbb{E}[y|\mathbf{x}] = f(\mathbf{x}; \mathbf{w})$$

- Goal: Learn a function $f(\mathbf{x}; \mathbf{w})$ that represents the average behavior of the data.
- Key Point: The model captures the mean of the target variable given input \mathbf{x} .

Maximum Likelihood Estimation (MLE)

- **MLE:** A method to estimate parameters that maximize the likelihood of the data.
- Given data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, MLE maximizes:

$$L(\mathcal{D}; \mathbf{w}, \sigma^2) = \prod_{i=1}^n p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma^2)$$

- MLE finds parameters \mathbf{w} and σ^2 that best explain the data.

Maximum Likelihood Estimation (cont.)

- Instead of maximizing the likelihood, it is often easier to maximize the log-likelihood:

$$\log L(\mathcal{D}; \mathbf{w}, \sigma^2) = \sum_{i=1}^n \log p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma^2)$$

- It is because $\log f(x)$ preserves the behaviour of $f(x)$.
- It is also easier to find derivative on summation of terms.

Univariate Linear Function Example

- Assuming Gaussian noise with parameters $(0, \sigma^2)$, probability of observing real output value y is:

$$p(y|\mathbf{x}, \mathbf{w}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - f(\mathbf{x}; \mathbf{w}))^2}{2\sigma^2}\right)$$

- For a simple linear model $f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x$ we have:

$$p(y|x, \mathbf{w}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - w_0 - w_1 x)^2}{2\sigma^2}\right)$$

- Key Observation:** Points far from the fitted line will have a low likelihood value.

Log-Likelihood and Sum of Squares

- Using log-likelihood we have:

$$\log L(\mathcal{D}; \mathbf{w}, \sigma^2) = -n \log \sigma - \frac{n}{2} \log(2\pi) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y^{(i)} - f(\mathbf{x}^{(i)}; \mathbf{w}))^2$$

- Since the objective of MLE is to optimize with regards to random variables, we can rule out the constants:

$$\log L(\mathcal{D}; \mathbf{w}, \sigma^2) \sim - \sum_{i=1}^n (y^{(i)} - f(\mathbf{x}^{(i)}; \mathbf{w}))^2$$

- Equivalence:** Maximizing the log-likelihood is equivalent to minimizing the Sum of Squared Errors (SSE):

$$J(\mathbf{w}) = \sum_{i=1}^n (y^{(i)} - f(\mathbf{x}^{(i)}; \mathbf{w}))^2$$

Estimating σ^2

- The maximum likelihood estimate of the noise variance σ^2 :

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n \left(y^{(i)} - f(\mathbf{x}^{(i)}; \hat{\mathbf{w}}) \right)^2$$

- Interpretation: Mean squared error of the predictions.
- Note: σ^2 reflects the noise level in the observations.

1 Generalization

2 Probabilistic regression

3 References

Contributions

- **These slides are authored by:**
 - Arshia Gharooni
 - Mahan Bayaghī

- [1] C. M., *Pattern Recognition and Machine Learning*.
Information Science and Statistics, New York, NY: Springer, 1 ed., Aug. 2006.
- [2] M. Soleymani Baghshah, “Machine learning.” Lecture slides.
- [3] A. Ng and T. Ma, *CS229 Lecture Notes*.
- [4] T. Mitchell, *Machine Learning*.
McGraw-Hill series in computer science, New York, NY: McGraw-Hill Professional, Mar. 1997.
- [5] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data: A Short Course*.
New York, NY: AMLBook, 2012.
- [6] S. Goel, H. Bansal, S. Bhatia, R. A. Rossi, V. Vinay, and A. Grover, “CyCLIP: Cyclic Contrastive Language-Image Pretraining,” *ArXiv*, vol. abs/2205.14459, May 2022.