CSE 405: Machine Learning

Supervised Learning

Dr Muhammad Abul Hasan

Department of Computer Science and Engineering
Green University of Bangladesh
muhammad.hasan@cse.green.edu.bd

Spring 2024

Outline

> *“I am always doing that which I cannot do, in order that I may learn how to do it.*
>
> *– Pablo Picasso”*

Supervised Learning

What is Supervised Learning?

Definition

Supervised learning is a type of machine learning where the algorithm learns from labeled training data to make predictions or decisions without being explicitly programmed.

- Labeled training data
- Predictions or decisions
- Example: Email spam classification

The Supervised Learning Workflow

Steps in Supervised Learning

1. Data collection
2. Data preprocessing
3. Model selection
4. Model training
5. Model evaluation
6. Prediction

Types of Supervised Learning Problems

Regression

Predicting a continuous output variable (e.g., price, temperature).

Classification

Assigning data to predefined classes or categories (e.g., spam or not spam, image recognition).

Learning a Class from Examples

- Class C of a "family car"
  - Prediction: Is car x a family car?
  - Knowledge extraction: What do people expect from a family car?
- Output: Positive (+) and negative (–) examples
- Input representation:
  - x_1: price
  - x_2: engine power

Training set X

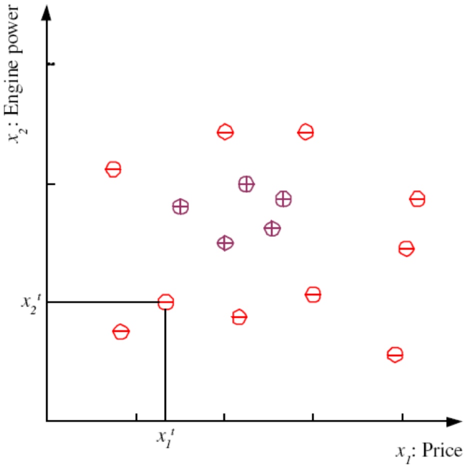
A training set is expressed as:

$$X = \{\mathbf{x}^t, r^t\}_{t=1}^{N}$$

where,

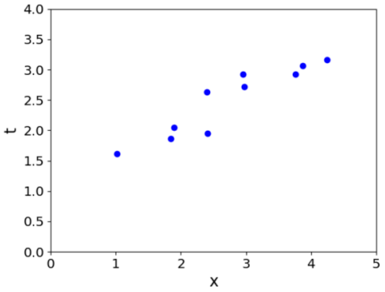
$$\mathbf{x}^t = \begin{bmatrix} x_1^t \\ x_2^t \end{bmatrix},$$

$$r^t = \begin{cases} 1 & \text{if } \mathbf{x}^t \text{ is positive} \\ 0 & \text{if } \mathbf{x}^t \text{ is negative} \end{cases}.$$



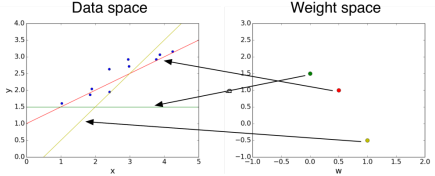
Linear Regression

Problem Setup



- Want to predict a scalar t as a function of a scalar x
- Given a dataset of pairs $(\mathbf{x}^{(i)}, t^{(i)})_{i=1}^{N}$
- The $\mathbf{x}^{(i)}$ are called inputs, and the $t(i)$ are called targets.

Problem Setup



- Model: y is a linear function of x:

$$y = wx + b$$

- y is the prediction
- w is the weight
- b is the bias
- w and b together are the parameters
- Settings of the parameters are called hypotheses

Problem Setup

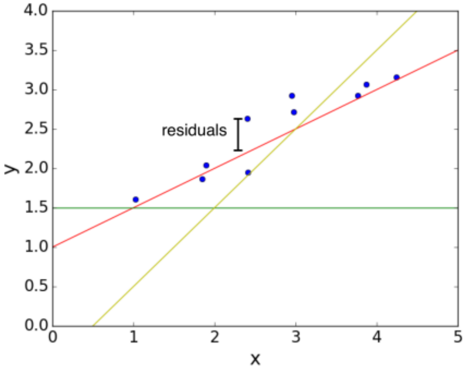
- Loss function: squared error (says how bad the fit is)

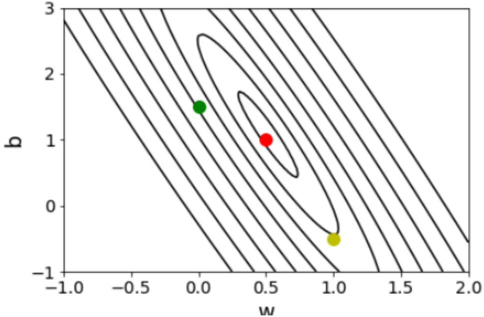
$$\mathcal{L}(y,t) = \frac{1}{2}(y-t)^2$$

- $y - t$ is the residual, and we want to make this small in magnitude
- The $\frac{1}{2}$ factor is just to make the calculations convenient.
- Cost function: loss function averaged over all training examples

$$\begin{aligned}\mathcal{J}(w,b) &= \frac{1}{2N}\sum_{i=1}^{N}\left(y^{(i)} - t^{(i)}\right)^2 \\ &= \frac{1}{2N}\sum_{i=1}^{N}\left(wx^{(i)} + b - t^{(i)}\right)^2\end{aligned}$$

Problem Setup





Problem Setup

- Suppose we have multiple inputs $x_1, \dots, x_D$. This is referred to as multivariable regression.
- This is no different than the single input case, just harder to visualize.
- Linear model:

$$y = \sum_j w_j x_j + b$$

How to decide the value of w_j and b?

Gradient descent. It is an optimization algorithm.

Solving the optimization problem

- We defined a cost function. This is what we'd like to minimize.
- Recall from calculus class: minimum of a smooth function (if it exists) occurs at a critical point, i.e. point where the derivative is zero.
- Multivariate generalization: set the partial derivatives to zero. We call this a direct solution.

Gradient Descent

- Gradient descent is an iterative algorithm, which means we apply an update repeatedly until some criterion is met.
- We initialize the weights to something reasonable (e.g. all zeros) and repeatedly adjust them in the direction of the steepest descent.

Gradient Descent

- Observe:
  - if $\frac{\partial \mathcal{J}}{\partial w_j} > 0$, then increasing w_j increases $\mathcal{J}$.
  - if $\frac{\partial \mathcal{J}}{\partial w_j} < 0$, then increasing w_j decreases $\mathcal{J}$.
- The following update decreases the cost function:

$$
\begin{aligned}
w_j &\leftarrow w_j - \alpha \frac{\partial \mathcal{J}}{\partial w_j} \\
&= w_j - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)}) x_j^{(i)}
\end{aligned}
$$

- α is a learning rate. The larger it is, the faster $\mathbf{w}$ changes.
  - We will see later how to tune the learning rate, but values are typically small, e.g. 0.01 or 0.0001.

Gradient Descent

- This gets its name from the gradient:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial \mathcal{J}}{\partial w_i} \\ \vdots \\ \frac{\partial \mathcal{J}}{\partial w_\mathcal{D}} \end{pmatrix}$$

  - This is the direction of fastest increase in $\mathcal{J}$
- Update rule in vector form:

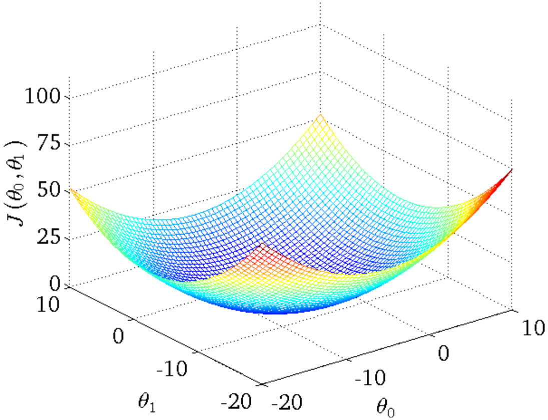
$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \alpha \frac{\partial \mathcal{J}}{\partial \mathbf{w}} \\ &= \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)}) \mathbf{x}^{(i)} \end{aligned}$$

- Hence, the gradient updates the weights in the direction of the fastest decrease.

Gradient Descent: Visualization

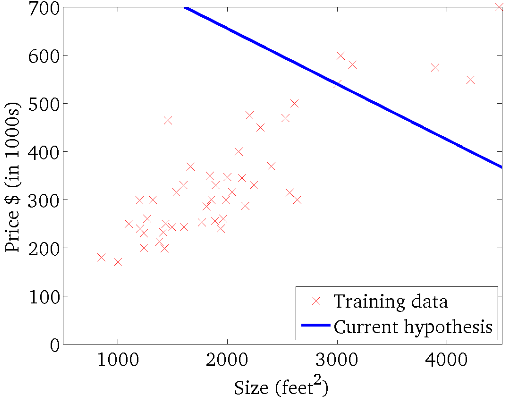
Gradient Descent: Visualization

For Linear Regression, J is bowl-shaped ("convex")



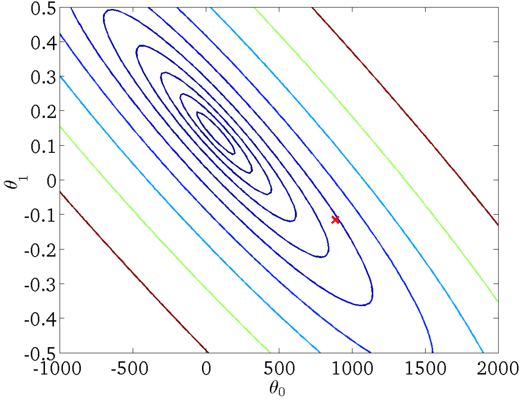
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



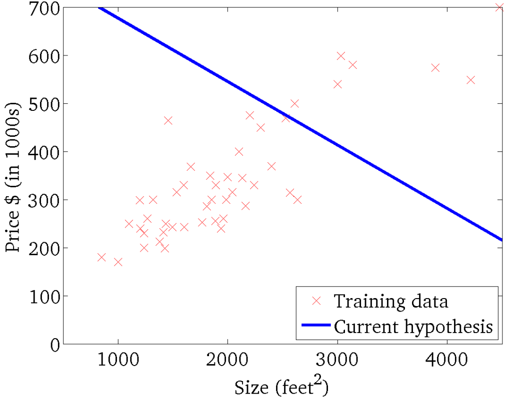
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



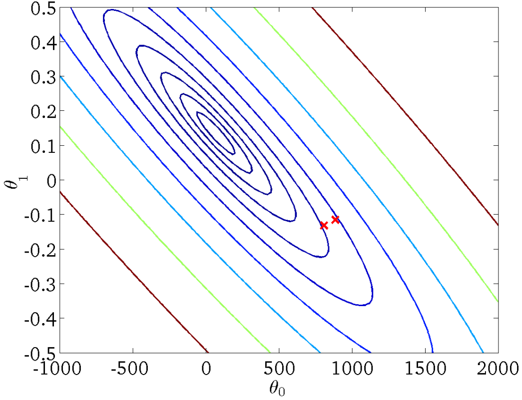
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



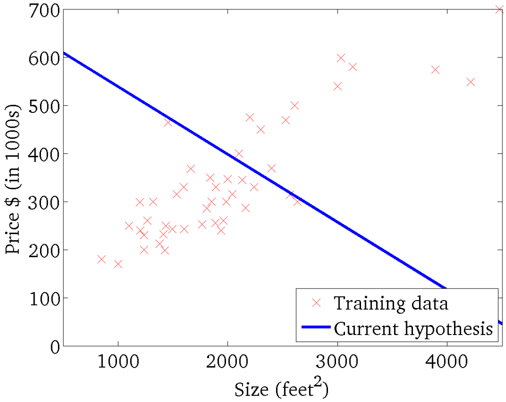
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



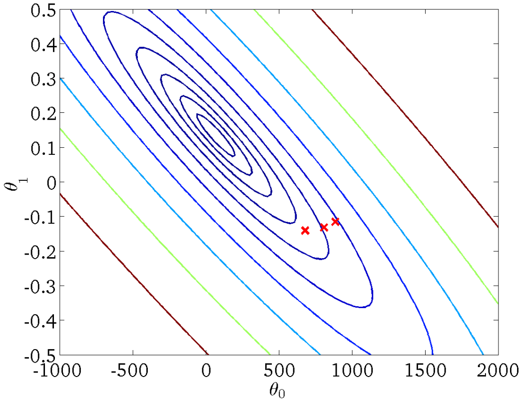
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



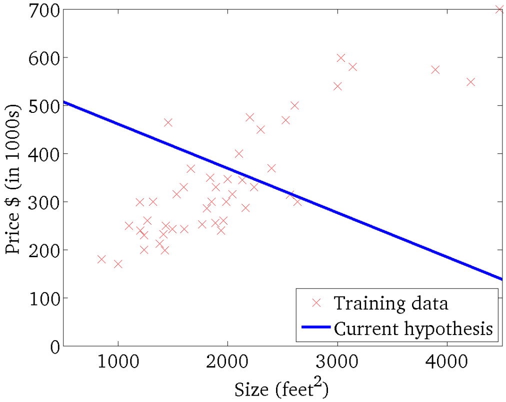
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



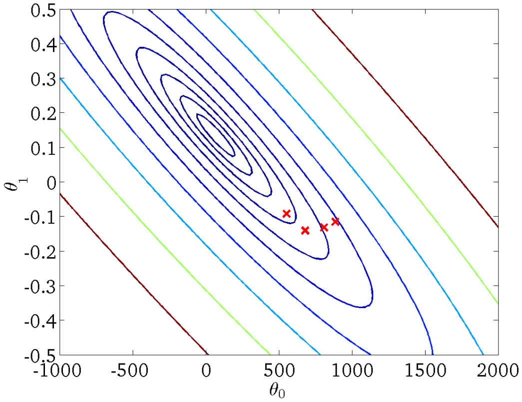
$$h_\theta(x)$$

(for fixed θ_0, θ_1, this is a function of x)



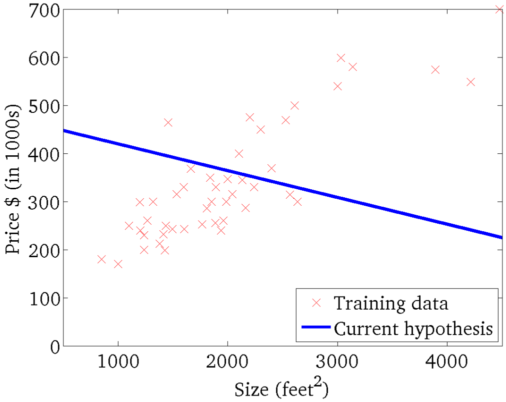
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



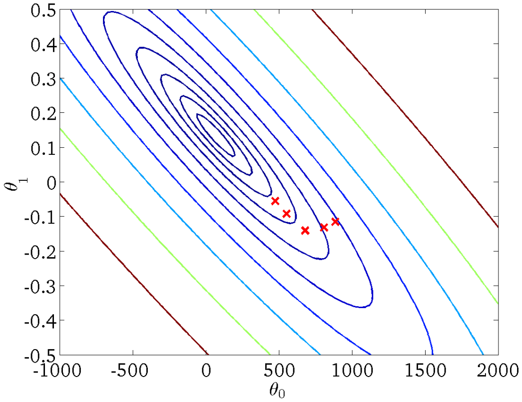
$$h_\theta(x)$$

(for fixed θ_0, θ_1, this is a function of x)



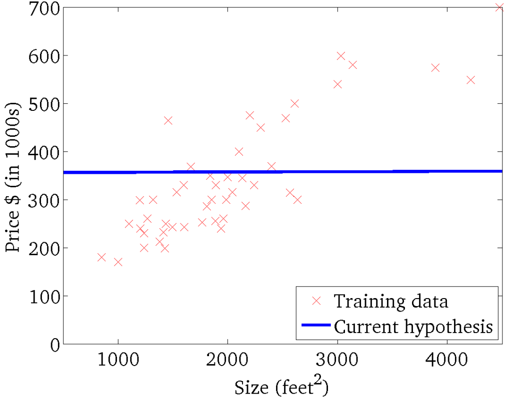
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



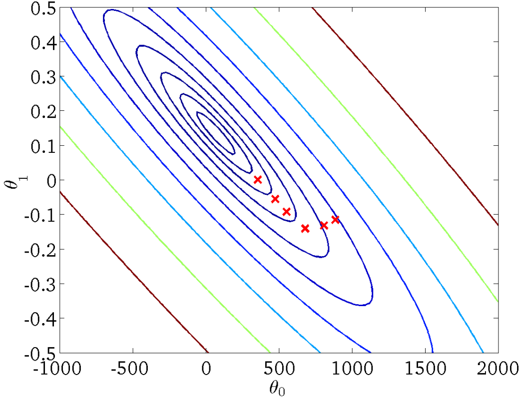
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



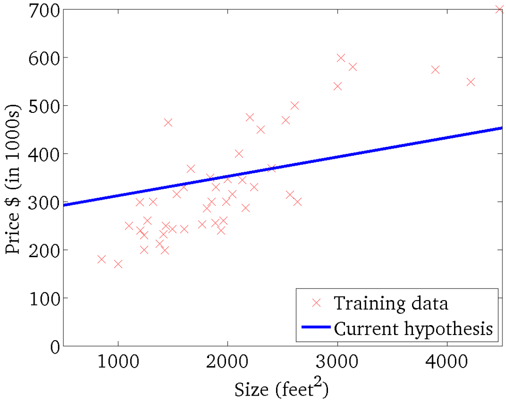
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



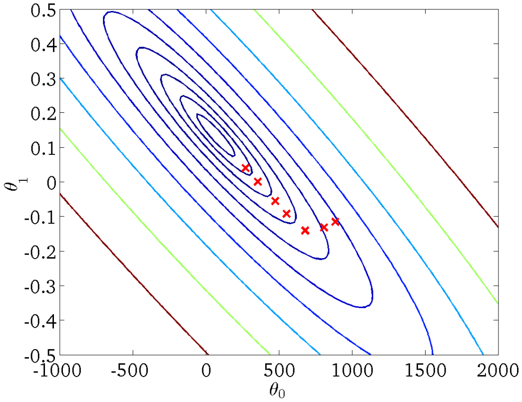
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



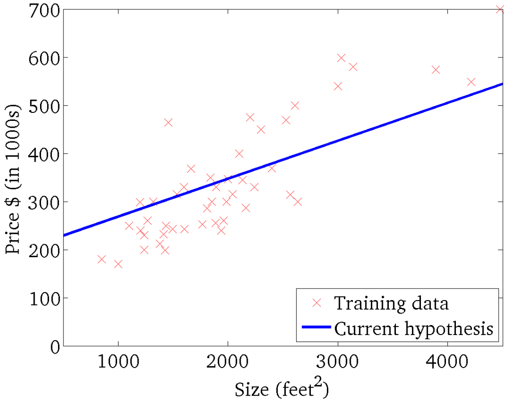
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



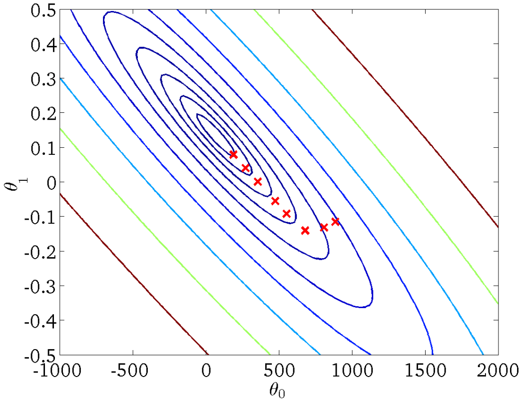
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



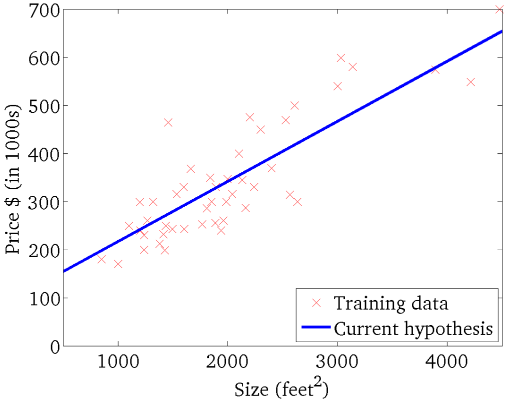
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



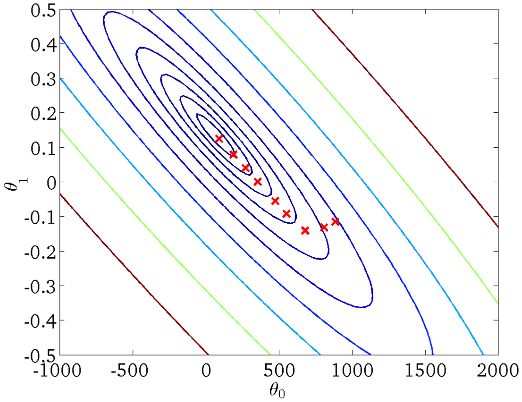
$$h_\theta(x)$$

(for fixed θ_0, θ_1, this is a function of x)



$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

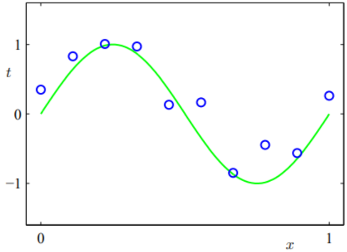


Gradient Descent

- Why gradient descent, if we can find the optimum directly?
  - GD can be applied to a much broader set of models
  - GD can be easier to implement than direct solutions, especially with automatic differentiation software
  - For regression in high-dimensional spaces, GD is more efficient than direct solution (matrix inversion is an $\mathcal{O}(D^3)$ algorithm).

Feature Mappings

- Suppose we want to model the following data



- One option: fit a low degree polynomial; this is is known as polynomial regression

$$y = w_3x^3 + w_2x^3 + w_1x + w_0$$

- Do we need to derive a whole new algorithm?

Feature Mappings

- We get polynomial regression for free!
- Define the feature map

$$\phi(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix}$$

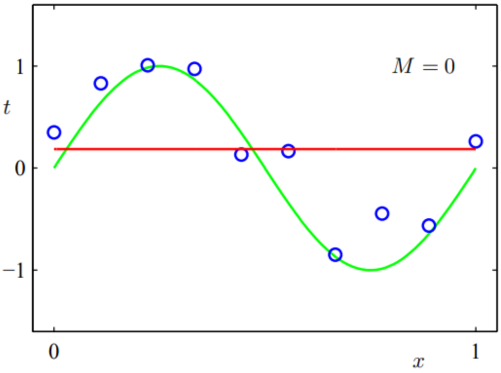
- Polynomial regression model:

$$y = \mathbf{w}^T \phi(x)$$

- All of the derivations and algorithms so far in the lecture remain the same!

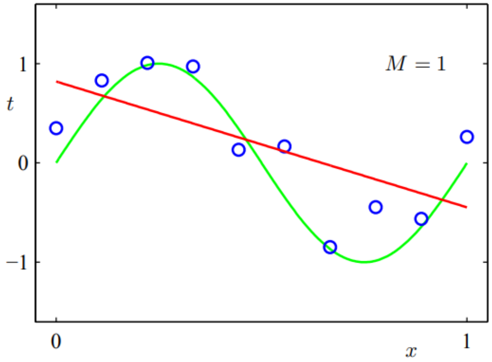
Fitting Polynomials

$$y = w_0$$



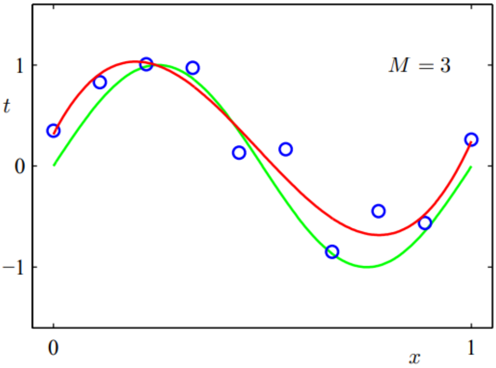
Fitting Polynomials

$$y = w_0 + w_1 x$$



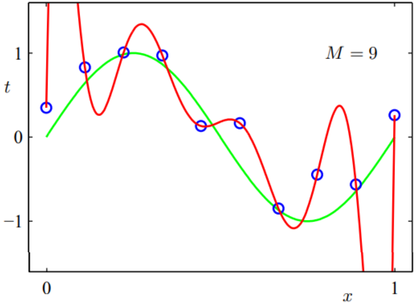
Fitting Polynomials

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$



Fitting Polynomials

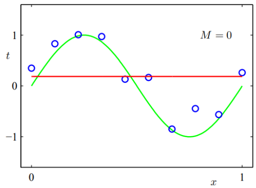
$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots + w_9 x^9$$



Generalization

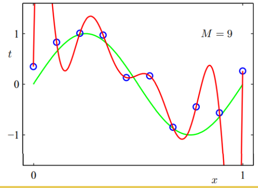
- Underfitting: model is too simple — does not fit the data.

$$y = w_0$$



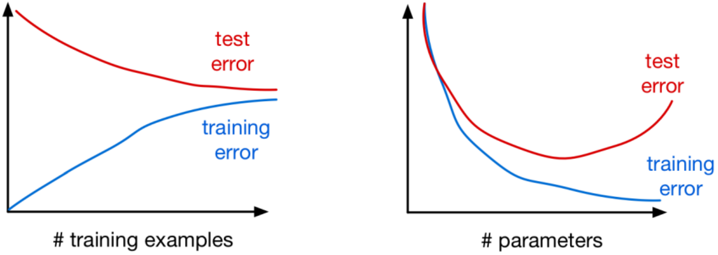
- Overfitting: model is too complex - fits perfectly, does not generalize.

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots + w_9 x^9$$



Generalizatoin

- Training and test error as a function of # training examples and # parameters:



Logistic Regression

Classification Problem

- The linear regression model assumes that the response variable y is quantitative (a value).
- In many situations, the response variable is instead qualitative (categorical).
  - Mango $\in \{\text{Lengra}, \text{Harivanga}, \text{Amropoli}\}$
  - Email $\in \{\text{Spam}, \text{Ham}\}$

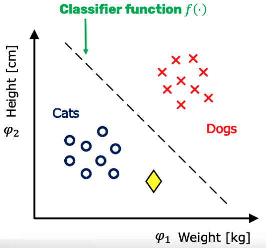
Classification Problem

- The process of estimating categorical outcomes using a set of features $\mathbf{x}$ is called classification.
- Estimating a categorical response for an observation $\mathbf{x}$ can be referred to as classifying that observation since it involves assigning the observation to a category, or class
- Often we are more interested in estimating the probabilities that $\mathbf{x}$ belongs to each category in $\mathcal{C}$
- The most probable category is then chosen as the class for the observation $\mathbf{x}$

Example: cat vs dog classification

Suppose that we measure the weight and height of some dogs and cats

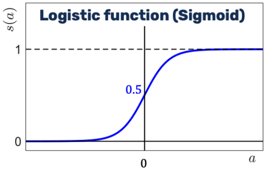
- We want to learn the function $f()$. that can tell us if a given input vector $\mathbf{x} = [x_1, x_2]^T$ is a dog or a cat
  - x_1: weight
  - x_2: height



Quiz: Which class would you classify the point marked as a yellow diamond?

Logistic Regression

- **Purpose**: Estimate the probability that a set of input features $\mathbf{x} \in \mathrm{R}^{d\times 1}$ belong to one of two classes $y \in \{0, 1\}$



Define the linear combination quantity $a = \sum_{i=0}^{d-1} x_i \cdot w_i = \mathbf{x}^T \cdot \mathbf{w}$

$s(a)$, Logistic Function Formula

$s(a) = \frac{1}{1+e^{-a}}$, if $a >> 0, s(a) = 1$ and if $a << 0, s(a) = 0$.

Logistic Regression

$$P(y = 1|\mathbf{x}) = s(a) = s(\mathbf{x}^T \cdot \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{x}^T \cdot \mathbf{w}}}$$

- The output of $s(\mathbf{x}^T \cdot \mathbf{w})$ is interpreted as a probability
  - $\mathbf{x}^T \cdot \mathbf{w} >> 0 \Rightarrow s(\mathbf{x}^T \cdot \mathbf{w}) >> 0.5 \Rightarrow P(y = 1|\mathbf{x}) \approx 1$
  - $\mathbf{x}^T \cdot \mathbf{w} << 0 \Rightarrow s(\mathbf{x}^T \cdot \mathbf{w}) << 0.5 \Rightarrow P(y = 1|\mathbf{x}) \approx 0$

Logistic Regression Cost Function

- Suppose we have a dataset $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$ where $x^{(i)} \in \mathrm{R}^{d \times 1}$ and $y^{(i)} \in \{0, 1\}, i = 1, \ldots, N$.
- Estimate a logistic regression model $P(y^{(i)} = 1|\mathbf{x}^{(i)}) = \frac{1}{1+e^{-\mathbf{x}^{(i)T}\mathbf{w}}} \equiv \pi^{(i)}$
- The logistic regression cost function $\mathcal{J}(\mathbf{w})$ is defined as:

$$\mathcal{J}(\mathbf{w}) = -\sum_{i=1}^{N}(y^{(i)}.\log \pi^{(i)} + (1 - y^{(i)}).\log[1 - \pi^{(i)}])$$

where:

- $\pi^{(i)}$ is the probability of $\mathbf{x}^{(i)}$ belonging to class 1.
- $\mathbf{x}^{(i)}$ is the input feature.
- $\mathbf{w}$ are the parameters to be learned.

Logistic Regression Cost Function

Quiz:

In the logistic regression cost function, where are the parameters w that we want to estimate?

$$\mathcal{J}(\mathbf{w}) = -\sum_{i=1}^{N}(y^{(i)}. \log \pi^{(i)} + (1 - y^{(i)}). \log[1 - \pi^{(i)}])$$

- In the $y^{(i)}$ terms
- In the $\log$ terms
- In the $\pi^{(i)}$ terms

Logistic Regression Cost Function

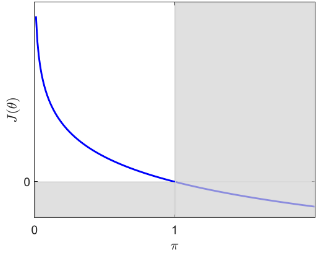
Cost function interpretation

Suppose there is only one datum $\mathcal{D} = \{(\mathbf{x}, y)\}$

$$\mathcal{J}(\mathbf{w}) = \begin{cases} -\log \pi, & \text{if } \; y = 1 \\ -\log[1-\pi], & \text{if } \; y = 0 \end{cases}$$

Case $y = 1$

$$\boxed{\mathcal{J}(\mathbf{w}) = -\log \pi \qquad \begin{array}{l} \mathcal{J}(\mathbf{w}) \approx 0 \text{ if } y = 1 \text{ and } \pi \approx 1 \\ \mathcal{J}(\mathbf{w}) \approx +\infty \text{ if } y = 1 \text{ and } \pi \approx 0 \end{array}}$$



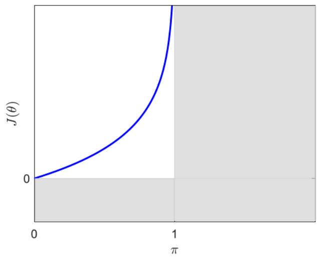
Logistic Regression Cost Function

Cost function interpretation
Suppose there is only one datum $\mathcal{D} = \{(\mathbf{x}, y)\}$

$$\mathcal{J}(\mathbf{w}) = \begin{cases} -\log \pi, & \text{if } \; y = 1 \\ -\log[1-\pi], & \text{if } \; y = 0 \end{cases}$$

Case $y = 0$

$$\boxed{\mathcal{J}(\mathbf{w}) = -\log[1-\pi] \quad \begin{array}{l} \mathcal{J}(\mathbf{w}) \approx 0 \text{ if } y = 0 \text{ and } \pi \approx 0 \\ \mathcal{J}(\mathbf{w}) \approx +\infty \text{ if } y = 0 \text{ and } \pi \approx 1 \end{array}}$$



Gradient Descent

Gradient Descent is used to find the parameters that minimize the cost function.

$$w_j = w_j - \alpha \frac{\partial \mathcal{J}(\mathbf{w})}{\partial w_j} \tag{1}$$

where:

- α is the learning rate.
- $\frac{\partial \mathcal{J}(\mathbf{w})}{\partial w_j}$ is the partial derivative of the cost function with respect to parameter w_j.

Summary

- Logistic Regression is used for binary classification.
- The cost function is the log-likelihood.
- Gradient Descent is used to update parameters.

Thank You!