TEXAS A&M UNIVERSITY

CSCE 633: Machine Learning

Lecture 4

Overview

Linear Regression

- Example
- Representation, Evaluation
- Optimization: Closed form solution via Ordinary Least Squares
- Optimization: Numerical solution via Gradient Descent
    - General gradient descent
    - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression & Overfitting

[Parts of these slides have been adapted from K. Murphy (Machine Learning: A probabilistic perspective), Dr. Andrew Ng's Machine Learning course at Coursera, and CSCI567 Machine Learning (USC, Drs. Sha & Liu)]

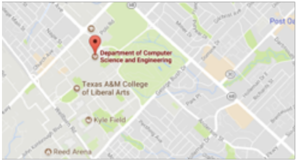
Overview

Linear Regression

- Example
- Representation, Evaluation
- Optimization: Closed form solution via Ordinary Least Squares
- Optimization: Numerical solution via Gradient Descent
  - General gradient descent
  - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression & Overfitting

[Parts of these slides have been adapted from K. Murphy (Machine Learning: A probabilistic perspective), Dr. Andrew Ng's Machine Learning course at Coursera, and CSCI567 Machine Learning (USC, Drs. Sha & Liu)]

Linear Regression: Example




Apartment Amenities

Pet Policy
Dogs Allowed: Breed Restrictions May Apply.
- $250 Fee
- 2 Pet Limit

Cats Allowed
- $250 Fee
- 2 Pet Limit

Fitness & Recreation
- Fitness Center
- Pool
- Volleyball Court

Living Space
- Carpet
- Attic
- Crown Molding
- Views
- Walk-In Closets

Outdoor Space
- Balcony
- Patio
- Porch
- Yard
- Barbecue Area
- Barbecue/Grill

Parking
Surface Lot
6 spaces; Assigned Parking.

Services
- Maintenance on site
- Property Manager on Site
- Bilingual
- Courtesy Patrol
- Trash Pickup - Curbside
- Recycling
- Planned Social Activities
- Pet Play Area

Kitchen
- Dishwasher
- Disposal
- Ice Maker
- Granite Countertops
- Kitchen
- Microwave
- Oven
- Refrigerator
- Freezer
- Instant Hot Water

Property Information
- Built in 2015
- 442 Units/2 Stories

Lease Length
August 1 - July 22

Outdoor Space
- Grill
- Waterfront
- Pond

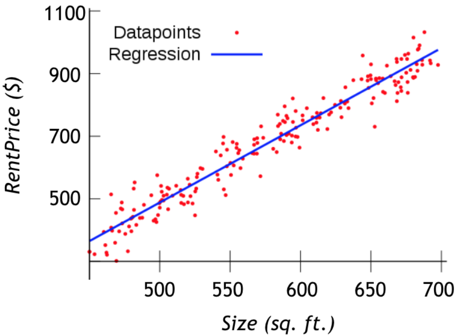
Features
- High Speed Internet Access
- Washer/Dryer
- Air Conditioning
- Heating
- Ceiling Fans
- Smoke Free
- Cable Ready
- Tub/Shower
- Framed Mirrors

Source: apartments.com

$$RentPrice = w_0 + w_1 \times Size + w_2 \times DistanceFromCS + \ldots$$

Linear Regression: Example

$$RentPrice = w_0 + w_1 \times Size + w_2 \times DistanceFromCS + \ldots$$



Linear Regression: Example

$$RentPrice = w_0 + w_1 \times Size + w_2 \times DistanceFromCS + \ldots$$

$$\mathrm{RentPrice} = w_0 + w_1 \times \mathrm{Size} + w_2 \times \mathrm{DistanceFromCS} + \ldots$$

How do we find the unknown model parameters $\{w_0, w_1, w_2, \ldots\}$?

We use training data!

Training Sample	Size (sq.ft.)	DistanceFromCS (miles)	RentPrice ($)
1	498	11.9	675
2	513	8.6	750
3	621	8.3	800
4	710	3.4	965
...	...	...	...

Overview

Linear Regression

- Example
- Representation, Evaluation
- Optimization: Closed form solution via Ordinary Least Squares
- Optimization: Numerical solution via Gradient Descent
  - General gradient descent
  - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression & Overfitting

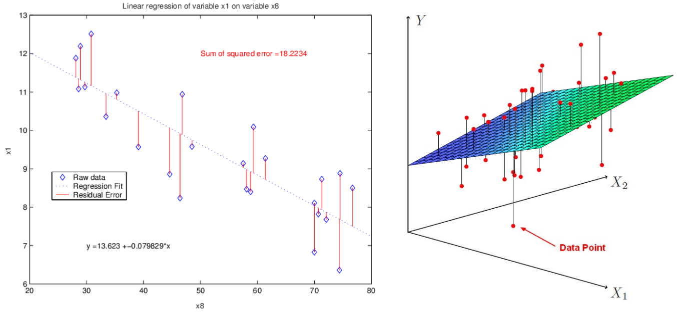
[Parts of these slides have been adapted from K. Murphy (Machine Learning: A probabilistic perspective), Dr. Andrew Ng's Machine Learning course at Coursera, and CSCI567 Machine Learning (USC, Drs. Sha & Liu)]

Linear Regression: Representation

- **Input:** $\mathbf{x} \in \mathbb{R}^{D+1}$ (D covariates/predictors/features, 1 extra term in the first position that corresponds to the bias term)
- **Output:** $y \in \mathbb{R}$ (responses, targets, outcomes, etc.)
- **Training Data:** $\mathcal{D} = \{(\mathbf{x_1}, y_1), \dots, (\mathbf{x_N}, y_N)\}$
- **Model:** $f : \mathbf{x} \rightarrow y, f(\mathbf{x}) = w_0 + \sum_{n=1}^{D} w_d x_d = \mathbf{w}^T \mathbf{x}$
  w_0: bias term
  $\mathbf{w} = [w_0, w_1, \dots, w_D]^T \in \mathbb{R}^{D+1}$: parameters/weights

Linear Regression: Evaluation

Minimizing the difference between predicted and actual labels (i.e., prediction error)



1-dimentional input (left), 2-dimensional input (right)

Linear Regression: Evaluation

- A reasonable thing would be to **minimize the prediction error** (also called residual sum of squares)

$$RSS(\mathbf{w}) = \sum_{n=1}^{N} (y_n - f(\mathbf{x_n}))^2 = \sum_{n=1}^{N} \left[y_n - \left(w_0 + \sum_{n=1}^{D} w_d x_{nd} \right) \right]^2$$

x_{nd}: d^{th} feature on the n^{th} training sample, N samples, D features

- An equivalent expression is: $RSS(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$

$$\mathbf{X} = \underbrace{\begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1D} \\ 1 & x_{21} & x_{22} & \dots & x_{2D} \\ & & \vdots & & \\ 1 & x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix}}_{\in \mathbb{R}^{N \times (D+1)}} = \begin{bmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_N}^T \end{bmatrix}$$

$$\mathbf{y} = [y_1, \dots, y_N]^T, \mathbf{x_n} = [1, x_{n1}, \dots, x_{nD}]^T, \mathbf{w} = [w_0, w_1, \dots, w_D]^T$$

Overview

Linear Regression

- Example
- Representation, Evaluation
- Optimization: Closed form solution via Ordinary Least Squares
- Optimization: Numerical solution via Gradient Descent
  - General gradient descent
  - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression & Overfitting

[Parts of these slides have been adapted from K. Murphy (Machine Learning: A probabilistic perspective), Dr. Andrew Ng's Machine Learning course at Coursera, and CSCI567 Machine Learning (USC, Drs. Sha & Liu)]

Linear Regression: Optimization

- Our goal is to find the solution $\mathbf{w}^*$ to minimize the prediction error: $\mathbf{w}^* = \mathit{arg}\min_{\mathbf{w}} RSS(\mathbf{w})$
- The cost function has a 1st order derivative, which if we set to zero, we can find a **closed-form solution**

We will first expand the vector/matrix expression of $RSS(\mathbf{w})$:

$$\begin{aligned} RSS(\mathbf{w}) &= (\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w}) = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T(\mathbf{X}\mathbf{w}) - (\mathbf{X}\mathbf{w})^T\mathbf{y} + (\mathbf{X}\mathbf{w})^T(\mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T\mathbf{y} - 2(\mathbf{X}\mathbf{w})^T\mathbf{y} + (\mathbf{X}\mathbf{w})^T(\mathbf{X}\mathbf{w}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T(\mathbf{X}^T\mathbf{y}) + \mathbf{w}^T(\mathbf{X}^T\mathbf{X})\mathbf{w} \end{aligned}$$

We then compute the first-order derivative $\frac{\vartheta RSS(\mathbf{w})}{\vartheta \mathbf{w}}$:

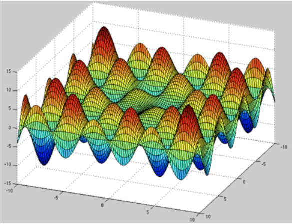
$$\frac{\vartheta RSS(\mathbf{w})}{\vartheta \mathbf{w}} = -2(\mathbf{X}^T\mathbf{y}) + 2(\mathbf{X}^T\mathbf{X})\mathbf{w}$$

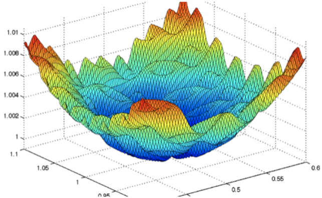
We set the first-order derivative to 0 and solve with respect to $\mathbf{w}$:

$$\frac{\vartheta RSS(\mathbf{w})}{\vartheta \mathbf{w}} = 0 \Rightarrow \quad -2(\mathbf{X}^T\mathbf{y}) + 2(\mathbf{X}^T\mathbf{X})\mathbf{w} = 0 \Rightarrow (\mathbf{X}^T\mathbf{X})\mathbf{w} = (\mathbf{X}^T\mathbf{y}) \Rightarrow \quad \mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Linear Regression: Optimization

Why should convexity be a problem in optimization?



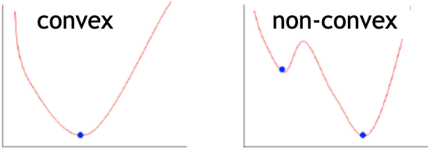


Loss functions might have more than one local optima (minima or maxima)

Linear Regression: Optimization

Theorem

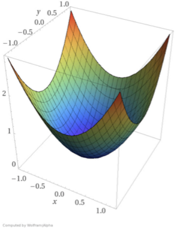
Consider an optimization problem:
$\min f(\mathbf{x})$ *s.t.* $\mathbf{x} \in \mathbf{\Omega}$ *where* f *is a convex function and* $\mathbf{\Omega}$ *is a convex set.*
Then any local minimum is also a global minimum



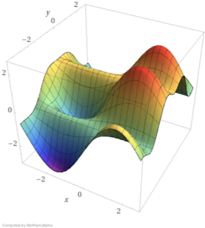
Linear Regression: Optimization

The second order derivative test

If the Hessian matrix $\mathbf{H_f}$ of function $f(\mathbf{x})$ is positive semi-definite, then f is convex, i.e., $\mathbf{u}^T\mathbf{H_f}\mathbf{u} \geq 0$ for every $\mathbf{u}$



convex



non-convex

Linear Regression: Optimization

Convexity of optimization criterion

$RSS(\mathbf{w}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T(\mathbf{X}^T\mathbf{y}) + \mathbf{w}^T(\mathbf{X}^T\mathbf{X})\mathbf{w}$

$\frac{\vartheta RSS(\mathbf{w})}{\vartheta \mathbf{w}} = -2(\mathbf{X}^T\mathbf{y}) + 2(\mathbf{X}^T\mathbf{X})\mathbf{w}$

$\mathbf{H}_{RSS(\mathbf{w})} = \frac{\theta^2 RSS(\mathbf{w})}{\theta \mathbf{w}^2} = \frac{\theta}{\theta \mathbf{w}}\left(\frac{\theta RSS(\mathbf{w})}{\theta \mathbf{w}}\right) = \frac{\theta}{\theta \mathbf{w}}\left(-2(\mathbf{X}^T\mathbf{y}) + 2(\mathbf{X}^T\mathbf{X})\mathbf{w}\right) = 2(\mathbf{X}^T\mathbf{X})$

For every $\mathbf{u} \in \mathbb{R}^D$ we have (by applying the transpose product rule and the definition of $l2$-norm):
$\mathbf{u}^T\mathbf{H}_{RSS(\mathbf{w})}\mathbf{u} = 2\mathbf{u}^T(\mathbf{X}^T\mathbf{X})\mathbf{u} = 2\mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u} = 2(\mathbf{X}\mathbf{u})^T\mathbf{X}\mathbf{u} = 2\|\mathbf{X}\mathbf{u}\|_2^2 \geq 0$

Therefore the Hessian $\mathbf{H}_{RSS(\mathbf{w})}$ of the RSS error is positive semi-definite, thus $RSS(\mathbf{w})$ is convex and any local optima is a global minimum. Therefore the solution $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \in \mathbb{R}^D$ is a global minimum of the RSS error of the linear regression problem.

Linear Regression: Optimization

Question: Assume the following non-linear regression model. Which if the following is true?

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

$\mathcal{D}^{train} = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_N}, y_N)\}$

$RSS(w_0, w_1, w_2) = \sum_{n=1}^{N} \left(y_n - \left(w_0 + w_1 x_n + w_2 x_n^2\right)\right)^2$

(A) We don't know if RSS has a global minimum with respect to $[w0, w1, w2]^T$
(B) RSS has a single local minimum w.r.t. $[w0, w1, w2]^T$, which is also global
(C) It depends on the training data whether RSS has a minimum

Linear Regression: Optimization

Question: Assume the following non-linear regression model. Which if the following is true?

$$y = w_0 + w_1 x_1 + w_2 x^2$$

$$\mathcal{D}^{train} = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_N}, y_N)\}$$

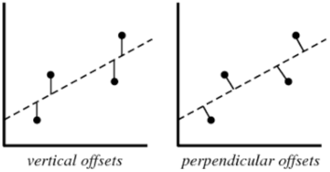
$$RSS(w_0, w_1, w_2) = \sum_{n=1}^{N} \left(y_n - \left(w_0 + w_1 x_n + w_2 x_n^2\right)\right)^2$$

(A) We don't know if RSS has a global minimum with respect to $[w0, w1, w2]^T$
(B) RSS has a single local minimum w.r.t. $[w0, w1, w2]^T$, which is also global
(C) It depends on the training data whether RSS has a minimum

The correct answer is B. RSS is a convex function w.r.t. $\mathbf{w}$, because the only thing that has changed in the loss function is the data matrix, rather than the weight vector.

Linear Regression: Optimization

Question: In a linear regression problem with one input variable, which of the following distances (offsets) do we use when we compute the residual sum of squares (RSS) error?

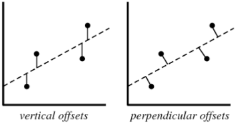


x-axis: input feature; y-axis: output

(A) Vertical offset
(B) Perpendicular offset
(C) Either, depending on the situation

Linear Regression: Optimization

Question: In a linear regression problem with one input variable, which of the following distances (offsets) do we use when we compute the residual sum of squares (RSS) error?



x-axis: input feature; y-axis: output

(A) Vertical offset
(B) Perpendicular offset
(C) Either, depending on the situation

The correct answer is A. The RSS error measures the distance between ground truth and predicted values with respect to the output space (y-axis).

Overview

Linear Regression

- Example
- Representation, Evaluation
- Optimization: Closed form solution via Ordinary Least Squares
- Optimization: Numerical solution via Gradient Descent
  - General gradient descent
  - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression & Overfitting

[Parts of these slides have been adapted from K. Murphy (Machine Learning: A probabilistic perspective), Dr. Andrew Ng's Machine Learning course at Coursera, and CSCI567 Machine Learning (USC, Drs. Sha & Liu)]

Linear Regression: Computational Complexity

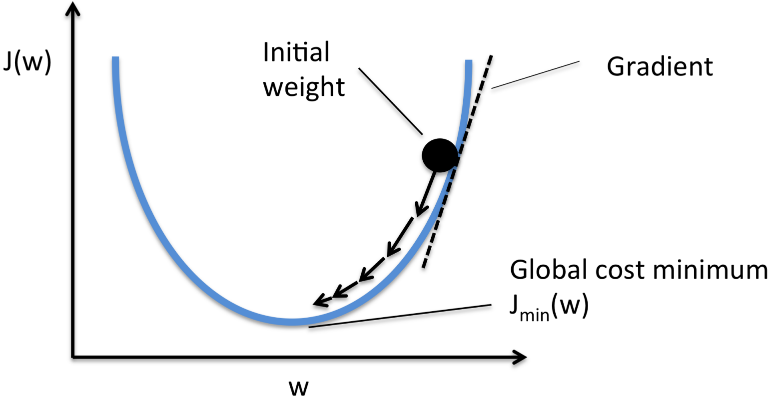
- Bottleneck for computing the solution $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ is to invert the matrix $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{D \times D}$
- Computational complexity is $O((D+1)^3)$ using Gauss-Jordan elimination
  - Impractical for large D
- Alternative
  - Find approximate solution through an iterative optimization algorithm
  - e.g. Gradient Descent

Gradient Descent

- Iterative algorithm finding a function's minimum via local search
- Standard optimization algorithm in many ML applications
    - e.g. linear regression, logistic regression
    - scales well to large datasets (e.g. no matrix multiplication)
    - proof that it solves many convex problems
    - good heuristic to non-convex problems as well
- **Input:** Function $J(\mathbf{w}) \in \mathbb{R}$
- **Output:** Local minimum $\mathbf{w}^*$
- **Goal:** Minimize $J(\mathbf{w})$ via greedy local search

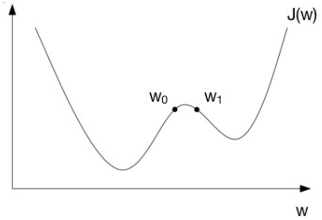
Gradient Descent

1-dimensional example: $J(w) = w^2$



Gradient Descent: 1-dimensional case

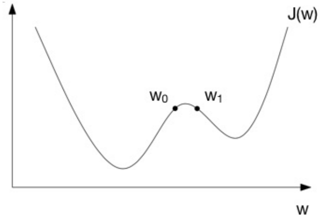
What will happen if we try to minimize $J(w)$ via a local search?



- Starting from w_0
  - We look to the right ($J(w) \uparrow$) and to the left ($J(w) \downarrow$)
  - We take a small step to the left
  - We repeat the above until we reach the left basin
- Starting from w_1
  - We similarly reach the right basin
- It is clear that the outcome depends on the starting point

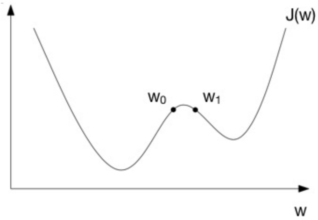
Gradient Descent: 1-dimensional case

More formally, we do the following



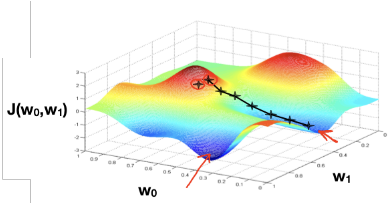
- $J'(w) = \frac{dj(w)}{dw} \approx \frac{J(w+\epsilon)-J(w)}{\epsilon}$, for $\epsilon \to 0$ (def. 1^{st} order derivative)
- While $J'(w) \neq 0$
  - If $J'(w) > 0$ (i.e. $J(w+\epsilon) > J(w)$ and $J(w) \uparrow$), move w a little bit to the left
  - If $J'(w) < 0$ (i.e. $J(w+\epsilon) < J(w)$ and $J(w) \downarrow$), move w a little bit to the right
- The derivative $J'(w)$ is used to decide which direction to move
- In other words, move w towards the direction of $-J'(w)$

Gradient Descent: Algorithm Outline



1-dimensional

1. Initialize w, ϵ, $\alpha(\cdot)$, $k := 0$
2. While $|\frac{dJ(w)}{dw}| > \epsilon$
   - 2a $k := k + 1$
   - 2b $w := w - \alpha(k) \cdot \frac{dJ(w)}{dw}$

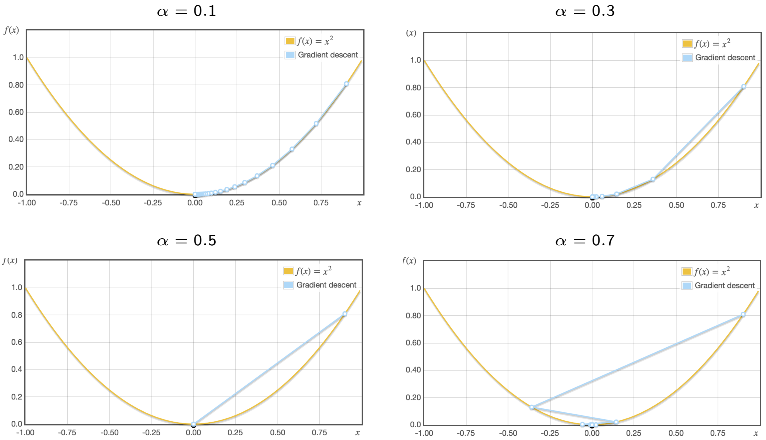


[Source: Machine Learning, Coursera, Andrew Ng]

d-dimensional

1. Initialize $\mathbf{w}$, ϵ, $\alpha(\cdot)$, $k := 0$
2. While $\|\nabla J(\mathbf{w})\|_2 > \epsilon$
   - 2a $k := k + 1$
   - 2b $\mathbf{w} := \mathbf{w} - \alpha(k) \cdot \nabla J(\mathbf{w})$

Gradient Descent: Step size (or learning rate) α



- If $\alpha(k)$ too small, convergence is unnecessarily slow
- If $\alpha(k)$ too large, correction process will overshoot and can diverge

Source: `http://www.onmyphd.com/?p=gradient.descent`

Gradient Descent: Step size (or learning rate) α

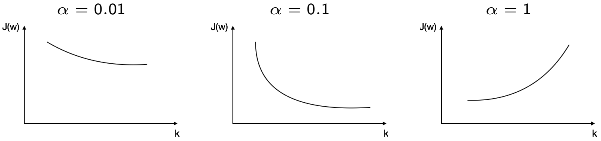
How to chose α?

- In practice, through experimentation
  - Check how $J(\mathbf{w})$ behaves over iterations for multiple α
  - α is a hyper-parameter
  - Therefore it can be tuned using a dev-set or a cross-validation framework

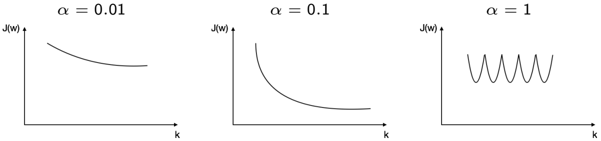
Gradient Descent: Step size (or learning rate) α

Question: A cost function $J(\mathbf{w})$ is optimized with Gradient Descent (GD) using different step size values α. We plot $J(\mathbf{w})$ w.r.t. the number of GD iterations k. Which of the following graph triplets can hold?

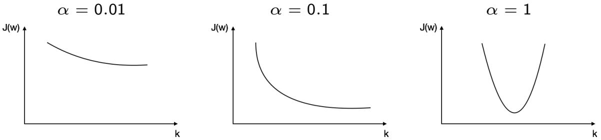
A



B



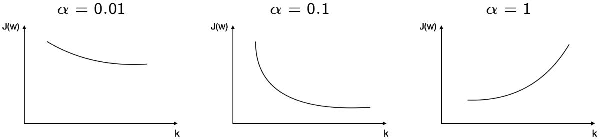
C



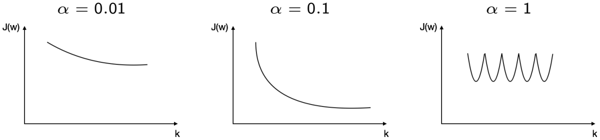
Gradient Descent: Step size (or learning rate) α

Question: A cost function $J(\mathbf{w})$ is optimized with Gradient Descent (GD) using different step size values α. We plot $J(\mathbf{w})$ w.r.t. the number of GD iterations k. Which of the following graph triplets can hold?

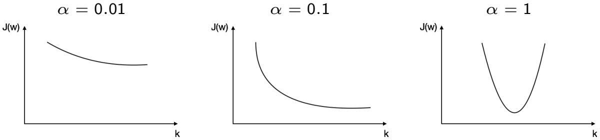
A



B



C



All answers can occur.

Gradient Descent: Stopping rule

- Hyper-parameter ϵ (i.e. $\|\nabla J(\mathbf{w})\|_2 > \epsilon$) determines when to stop
- Small ϵ: many iterations but higher quality solution
- Large ϵ: less iterations with the cost of more approximate solution
- How to chose ϵ in practice?
    - Try various values to achieve balance between cost and precision
    - Again use some type of cross-validation framework

- Hyperparameters: Parameters set before the beginning of the learning process (e.g. α, ϵ in gradient descent)
- Hyperparameter tuning: The process of choosing a set of optimal hyperparameters for the learning process
- Model parameters: The parameters learned during the learning process (e.g. weights $\mathbf{w}$ in linear regression)

Linear Regression

- Example
- Representation, Evaluation
- Optimization: Closed form solution via Ordinary Least Squares
- Optimization: Numerical solution via Gradient Descent
  - General gradient descent
  - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression & Overfitting

[Parts of these slides have been adapted from K. Murphy (Machine Learning: A probabilistic perspective), Dr. Andrew Ng's Machine Learning course at Coursera, and CSCI567 Machine Learning (USC, Drs. Sha & Liu)]

Gradient Descent in Linear Regression

We can now derive the algorithm outline for minimizing the residual square sum (RSS) error of linear regression with gradient descent

- The residual sum of squares is the cost function:

$$\begin{aligned} J(\mathbf{w}) = RSS(\mathbf{w}) &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T\mathbf{y} - 2(\mathbf{X}\mathbf{w})^T\mathbf{y} + (\mathbf{X}\mathbf{w})^T(\mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T(\mathbf{X}^T\mathbf{y}) + \mathbf{w}^T(\mathbf{X}^T\mathbf{X})\mathbf{w} \end{aligned}$$

- Gradient Descent optimization expression:

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

$$\nabla J(\mathbf{w}) = \frac{\vartheta RSS(\mathbf{w})}{\vartheta \mathbf{w}} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{w}$$

Gradient Descent in Linear Regression

Question: Derive the algorithm outline for minimizing the residual square sum (RSS) error of linear regression with gradient descent

(Batch) Gradient Descent for Linear Regression

1 Initialize $\mathbf{w}$, ϵ, $\alpha(\cdot)$, $k := 0$

2 While $\|\nabla RSS(\mathbf{w})\|_2 > \epsilon$

  2a $k := k + 1$

  2b $\mathbf{w} := \mathbf{w} - \alpha(k) \cdot \left(\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y}\right)$

Gradient Descent in Linear Regression

Stochastic Gradient Descent for Linear Regression

Update weights using one sample at a time

1 Initialize $\mathbf{w}$, ϵ, $\alpha(\cdot)$, $k := 0$

2 Loop until convergence

  2a $k := k + 1$

  2b Randomly choose a sample $(\mathbf{x_i}, y_i)$

  2c Compute its contribution to the gradient $\mathbf{g_i} = (\mathbf{x_i}^T \mathbf{w} - y_i) \cdot \mathbf{x_i}$

  2d Update the weights $\mathbf{w} := \mathbf{w} - \alpha(k) \cdot \mathbf{g_i}$

Gradient Descent in Linear Regression

Mini-Batch Gradient Descent for Linear Regression

Update weights using subset of samples at a time

1. Initialize $\mathbf{w}$, ϵ, $\alpha(\cdot)$, $k := 0$
2. Loop until convergence
   - 2a $k := k + 1$
   - 2b Randomly choose a subset of samples $S = \{(\mathbf{x_i}, y_i), \ldots, (\mathbf{x_{i+M}}, y_{i+M})\}$
   - 2c Form the mini-batch data matrix $\mathbf{X_S} = \begin{bmatrix} \mathbf{x_i}^T \\ \vdots \\ \mathbf{x}_{\mathbf{i+M}}^T \end{bmatrix}$
   - 2d Update the weights $\mathbf{w} := \mathbf{w} - \alpha(k) \cdot \left(\mathbf{X_S}^T \mathbf{X_S} \mathbf{w} - \mathbf{X_S}^T \mathbf{y} \right)$

- Good compromise between batch and stochastic gradient descent
- Common mini-batch sizes range between M=50-250 samples

Gradient Descent in Linear Regression

- Batch gradient descent computes exact gradient
- Stochastic gradient descent
  - Computes approximate gradient using one sample per iteration
  - Its expectation equals the true gradient
- Mini-batch gradient descent
  - Computes gradient based on subset of samples
- For large-scale problems stochastic or mini-batch descent often work well

Overview

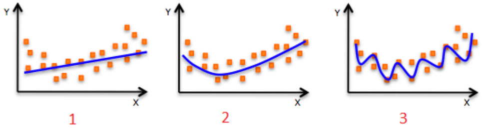
Linear Regression

- Example
- Representation, Evaluation
- Optimization: Closed form solution via Ordinary Least Squares
- Optimization: Numerical solution via Gradient Descent
  - General gradient descent
  - Gradient descent for linear regression (batch, stochastic, mini-batch)
- Non-linear basis function for regression & Overfitting

[Parts of these slides have been adapted from K. Murphy (Machine Learning: A probabilistic perspective), Dr. Andrew Ng's Machine Learning course at Coursera, and CSCI567 Machine Learning (USC, Drs. Sha & Liu)]

Non-Linear Regression

Question: Given a set of training data (red square points), which of the following regression models (blue line) would you choose to fit the data (and why)?



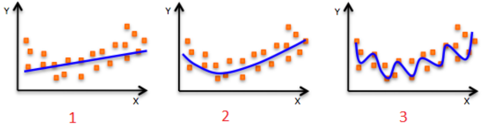
A) Model 1, since the data depicts and increasing trend

B) Model 2, since the line best captures the overall trend in the data

C) Model 3, since the line provides the smallest RSS error

Non-Linear Regression

Question: Given a set of training data (red square points), which of the following regression models (blue line) would you choose to fit the data (and why)?



A) Model 1, since the data depicts and increasing trend

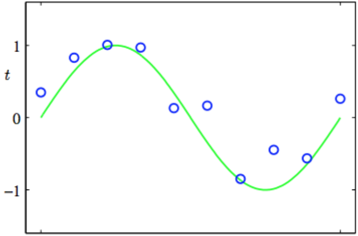
B) Model 2, since the line best captures the overall trend in the data

C) Model 3, since the line provides the smallest RSS error

The correct answer is B. Model 1 is too simple for this data. Model 3 is too complex, and likely more difficult to generalize on new unseen data.

What if the data does not fit a line?

Example: Samples from a sine function



We can use a non-linear basis function

$\phi(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^D \to \mathbf{z} \in \mathbb{R}^M$

We can apply our linear regression model to the new features

$y_i = \mathbf{w}^T \mathbf{z_i} = \mathbf{w}^T \phi(\mathbf{x_i})$

$RSS(\mathbf{w}) = \sum_{n=1}^{N} \left(y_i - \mathbf{w}^T \phi(\mathbf{x_i})\right)^2, \ \mathbf{w} \in \mathbb{R}^M$

Example: $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^2$, $\phi(\mathbf{x}) = [x_1, x_2^2, x_1^3 + x_2]^T \in \mathbb{R}^3$

Non-Linear Basis Function

Residual sum of squares

$$RSS(\mathbf{w}) = \sum_{n=1}^{N} \left(y_i - \mathbf{w}^T \phi(\mathbf{x_i})\right)^2 = (\mathbf{y} - \boldsymbol{\Phi}\mathbf{w})^T(\mathbf{y} - \boldsymbol{\Phi}\mathbf{w})$$

Non-linear design matrix

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi(\mathbf{x_1})^T \\ \phi(\mathbf{x_2})^T \\ \vdots \\ \phi(\mathbf{x_N})^T \end{bmatrix} \in \mathbb{R}^{N \times M}$$

LMS solution with the non-linear design matrix

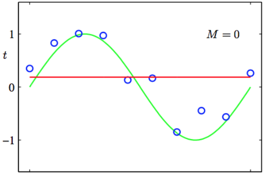
$$\mathbf{w}^{LMS} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{y}$$

Non-Linear Basis Function

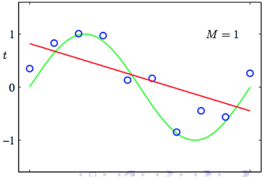
Example: Samples from a sine function

Polynomial basis function $\phi(\mathbf{x}) = [1 \ x \ \ldots \ x^M]^T$

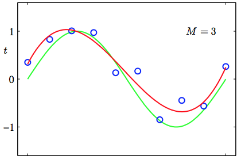
$M = 0$ underfitting



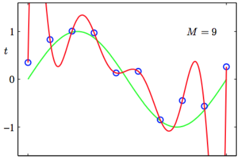
$M = 1$ underfitting



$M = 3$



$M = 9$ overfitting



Overfitting

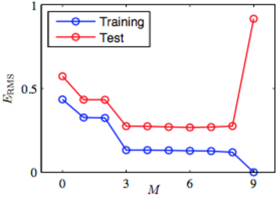
Weights of high order polynomials are very large

$$y_i = \mathbf{w}^T \mathbf{z_i} = \mathbf{w}^T \phi(\mathbf{x_i}), \quad \mathbf{z_i} = \phi(\mathbf{x_i}) \in \mathbb{R}^M$$

	$M=0$	$M=1$	$M=3$	$M=9$
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

Overfitting

- The risk of using highly flexible (complicated) models without enough data
- Leads to poor generalization
- How to detect overfitting?
  - Plot model complexity (e.g. polynomial order) versus objective function
  - As complexity increases, performance on training improves, while on testing first improves and then deteriorates
- How to avoid overfitting?
  - More data or regularization

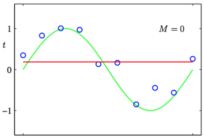


Overfitting

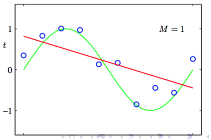
Example: Non-linear regression $y = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$

Samples from a sine function $x_i = \sin(t_i)$, $t_i \sim \mathrm{Uniform}(0, 2\pi)$

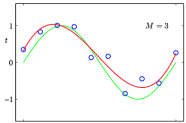
$M = 0$ underfitting



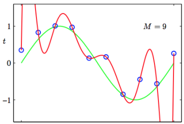
$M = 1$ underfitting



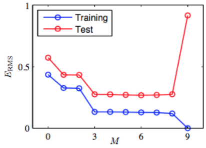
$M = 3$



$M = 9$ overfitting



	$M = 0$	$M = 1$	$M = 3$	$M = 9$
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

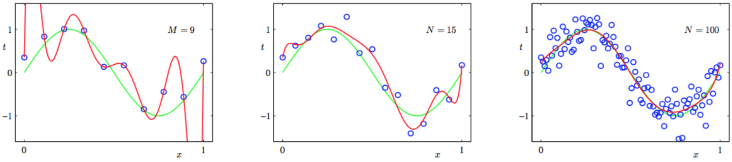


As model becomes more complex, performance on training keeps improving while on test data improve first and deteriorate later.

The larger a coefficient w_i, the easier for the model to "swing" in that dimension, increasing chance to fit more noise.

How can we avoid overfitting?

One solution: Use more training data



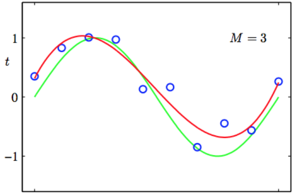
What if we don't have a lot of data?

Another solution: Use less features (e.g. feature selection algorithms)
Intuitively, this will reduce the complexity of the model, therefore it is likely to result in less overfitting.

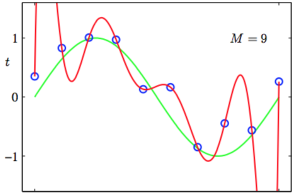
How can we avoid overfitting?

A more general solution: Regularization

$M = 3$



$M = 9$



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

How about penalizing and making small $w_3, \ldots, w_9$?

The cost function to be minimized would become:

$J(\mathbf{w}) = RSS(\mathbf{w}) + w_3^2 + \ldots w_9^2$

But we may not know in advance which parameters we want to penalize

$\rightarrow$ So we can penalize them all

How can we avoid overfitting?

A more general solution: Regularization

Suppose we have a learning model whose evaluation criterion $EC(\mathbf{w})$ we want to optimize with respect to weights $\mathbf{w} = [w_1, \ldots, w_D]^T$

- $J(\mathbf{w}) = EC(\mathbf{w}) + \lambda \sum_{d=1}^{D} w_d^2 = EC(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$
  $\rightarrow$ l2-norm regularization
- $J(\mathbf{w}) = EC(\mathbf{w}) + \frac{\lambda}{N} \sum_{d=1}^{D} w_d^2$
  (as #data N increases, we need to worry less about overfitting)
- $J(\mathbf{w}) = EC(\mathbf{w}) + \lambda \sum_{d=1}^{D} \|w_d\| = EC(\mathbf{w}) + \lambda \|\mathbf{w}\|$
  $\rightarrow$ l1-norm regularization

Evaluation criterion $EC(\mathbf{w})$ can be RSS or log-likelihood for linear regression, negative cross-entropy for logistic regression, etc.

$\lambda \geq 0$ is the model complexity penalty

Regularization for Non-Linear Regression

l2-norm regularization

Linear: $J(\mathbf{w}) = RSS(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2$

Non-linear: $J(\mathbf{w}) = RSS(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2 = (\mathbf{y} - \boldsymbol{\Phi}\mathbf{w})^T(\mathbf{y} - \boldsymbol{\Phi}\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2$

Closed-form solution:

Linear: $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_{D\times D})^{-1}\mathbf{X}^T\mathbf{y}$

Non-linear: $\mathbf{w}^* = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\mathbf{I}_{D\times D})^{-1}\boldsymbol{\Phi}^T\mathbf{y}$

The above reduces to ordinary least squares (OLS) solution when $\lambda = 0$ (see handout for derivation)

Overfitting

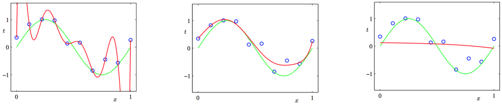
Why would we need regularization in linear regression?

- Multicollinearity (or collinearity): the existence of near-linear relationships among the features
  - e.g., same variable represented in two different units
  - e.g., three ingredients of a mixture summing to 100% ($x_1 + x_2 + x_3 = 100$)
- Multicollinear variables result in data matrices close to non-invertible, therefore causing inaccurate estimates of the regression coefficients
- Regularization would cause some of the coefficients (potentially the ones corresponding to one of the multicollinear variables) to be close to zero

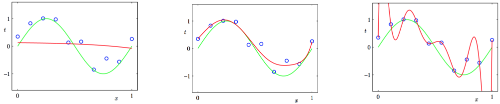
Regularization for Non-Linear Regression

Question: Assume a set of samples generated from a sine function $x_i = \sin(t_i)$ (green line), modeled with **regularized** non-linear regression $y = w_0 + w_1 x + \ldots + w_9 x^9$. How does the resulting model (red line) look as we increase the amount of regularization λ?

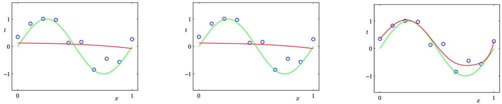
A) $\lambda = 0$ $\lambda = e^{-10}$ $\lambda = 1$



B) $\lambda = 0$ $\lambda = e^{-10}$ $\lambda = 1$



C) $\lambda = 0$ $\lambda = e^{-10}$ $\lambda = 1$

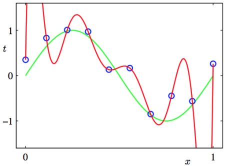


Regularization for Non-Linear Regression

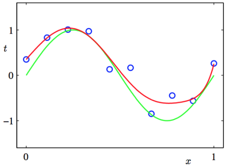
Question: Assume a set of samples generated from a sine function $x_i = \sin(t_i)$ (green line), modeled with **regularized** non-linear regression $y = w_0 + w_1 x + \ldots + w_9 x^9$. How does the resulting model (red line) look as we increase the amount of regularization λ?

The correct answer is A

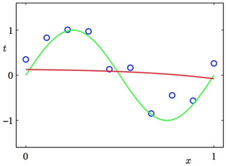
$\lambda = 0:$ no regularization



$\lambda = e^{-18}$



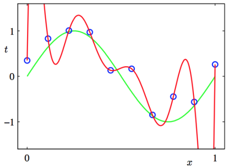
$\lambda = 1:$ underfitting



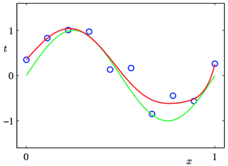
Overfitting is reduced with the help of increasing regularizers

Regularization for Non-Linear Regression

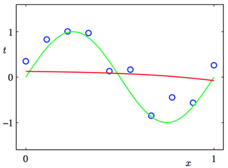
$\lambda = 0$: no regularization

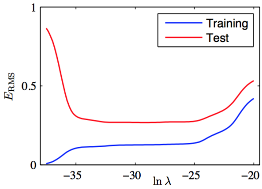


$\lambda = e^{-18}$



$lambda = 1$





	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
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	-0.06
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

For a complex model ($M = 9$), training error increases with increasing regularization.

Linear Regression: To summarize

- **Representation:** linear and non-linear basis

  $f : \mathbf{x} \rightarrow y, \quad f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$

  $f : \mathbf{z} \rightarrow y, \quad f(\mathbf{x}) = \mathbf{w}^T \mathbf{z} = \mathbf{w}^T \phi(\mathbf{x}), \quad \phi : \mathbf{x} \in \mathbb{R}^D \rightarrow \mathbf{z} \in \mathbb{R}^M$

- **Evaluation**: Minimizing residual sum of squares

  $\min_{\mathbf{w}} RSS(\mathbf{w}), \quad RSS(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$

  $\min_{\mathbf{w}} RSS(\mathbf{w}), \quad RSS(\mathbf{w}) = (\mathbf{y} - \boldsymbol{\Phi}\mathbf{w})^T(\mathbf{y} - \boldsymbol{\Phi}\mathbf{w})$

- **Analytic Optimization**: Ordinary least squares (OLS) solution

  $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \quad \mathbf{w}^* = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{y}$

- **Approximate Optimization**: Gradient descent (batch, stochastic, mini-batch)
- **Readings:** Alpaydin Ch 2, Abu-Mostafa Ch 3.2