4. Linear Regression

Dongwoo Kim

dongwookim.ac.kr

CSED515 - 2023 Spring

Regression

Regression aims at modeling the dependence f of output y on input x, e.g., given height and weight (x), how long is s/he going to live (y)?:

$$y = f(x) + \varepsilon \ ,$$

where we also use:

- x: input, independent variable, predictor, regressor, covariate
- y: output, dependent variable, response
- ϵ: noise or some unobserved factors

The dependence of output on input is captured via a conditional distribution $p(y \mid x)$.

Regression Function: Conditional Mean

Consider the minimization of mean squared error (MSE):

$$\begin{aligned}\mathcal{E}(f) &= \mathbb{E}[\|y - f(x)\|^2] \\ &= \int\int \|y - f(x)\|^2 p(x,y)dxdy \\ &= \int\int \|y - f(x)\|^2 p(x)p(y \mid x)dxdy \\ &= \int p(x) \underbrace{\int \|y - f(x)\|^2 p(y \mid x)dy}_{\text{to be minimized}} dx \ .\end{aligned}$$

Then, by taking derivative w.r.t. $f(x)$ and setting that to 0, i.e., $\frac{\partial}{\partial f(x)} \left(\int \|y - f(x)\|^2 p(y \mid x)dy \right) = 0$, the minimum MSE (MMSE) estimate is

$$f(x) = \int y p(y \mid x)dy = \mathbb{E}[y \mid x] \ .$$

Function Approximation and Curve Fitting

Regression can be seen as function approximation or curve fitting of $\mathbb{E}[y \mid x]$ using a class of functions f with few parameters:

- Linear function: $y = \boldsymbol{w}^\top \boldsymbol{x}$
- Neural networks: $f(x) = G_w(x)$, e.g., a fully connected linear network of L layers has $G_w(x) = w_L w_{L-1} \dots w_1 x$ where w_ℓ is a $d_\ell \times d_{\ell-1}$ matrix.

Table of Contents

Linear Models

- Linear models tackle the regression problem with the following assumption: there exists a linear relation between input $\boldsymbol{x} \in \mathbb{R}^d$ and output $y \in R$

$$y = \boldsymbol{w}^\top \boldsymbol{x} = \boldsymbol{x}^\top \boldsymbol{w}$$

- Given a collection of $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_N, y_N)\}$, the relation can be

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}$$

  where $\boldsymbol{y} \in \mathbb{R}^N, \boldsymbol{X} \in \mathbb{R}^{N \times d}$.

- Are linear model too simple to capture complex relations?

Basis (Feature) Functions

- Basis function ϕ extracts useful information from observation $\boldsymbol{x}$.
- It needs to be manually crafted according to domain.
- For example, given basis function ϕ with input $x \in \mathbb{R}$:

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

basis function replaces the input in linear model as

$$\boldsymbol{w}^\top \phi(x) = w_1 + w_2 x + w_3 x^2 + w_4 x^3$$

- Therefore, with basis function, linear model can models non-linear relation between inputs and outputs.

Linear Regression

Let $\boldsymbol{x} \in \mathbb{R}^D$. Linear regression refers to a model of which f is a linear combination of basis functions $\{\phi_\ell : \mathbb{R}^D \to \mathbb{R}\}_{\ell \in [L]}$:

$$f(\boldsymbol{x}) = \sum_{\ell=1}^{L} w_\ell \phi_\ell(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) \,,$$

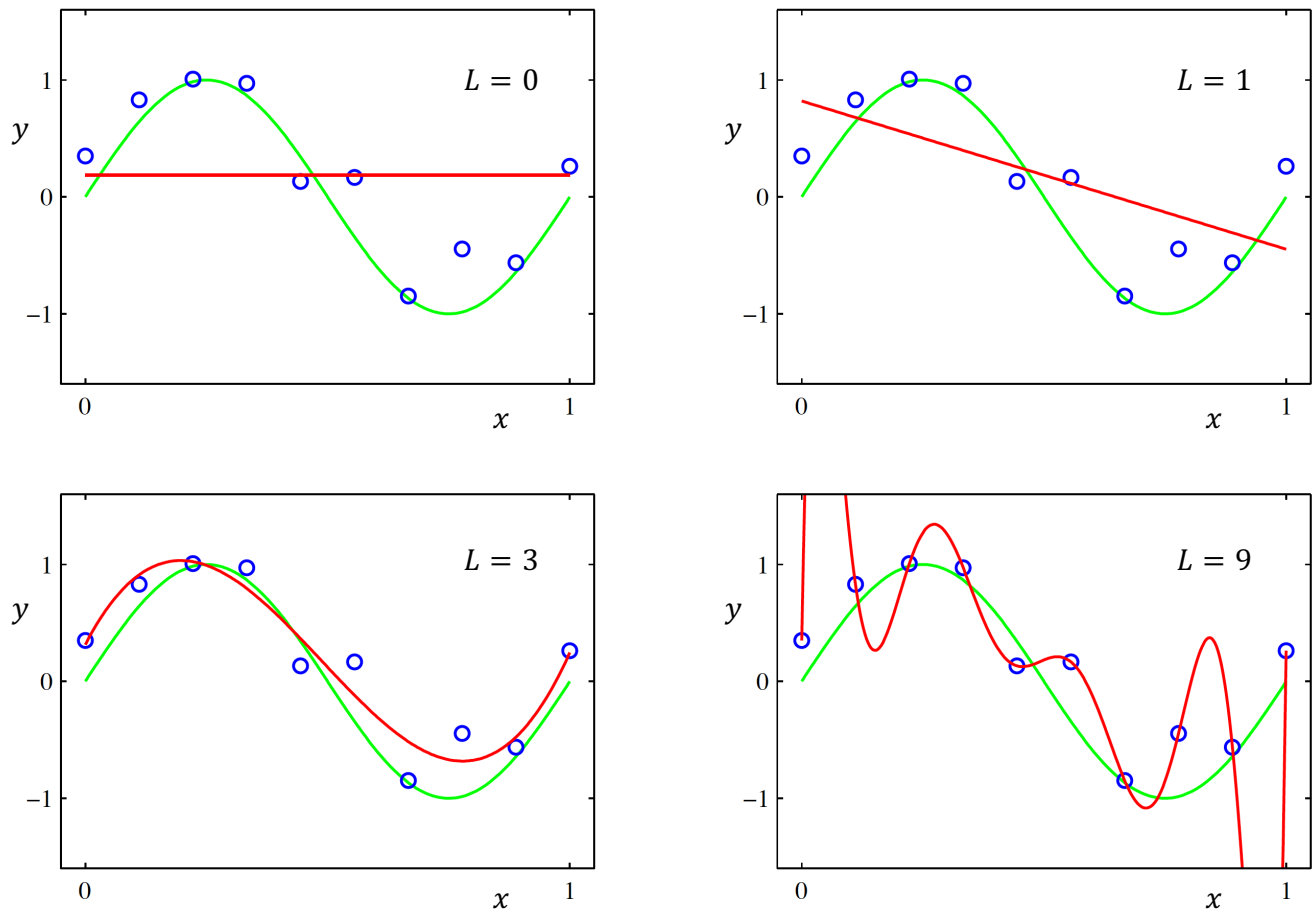
where

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_L \end{bmatrix} \,, \quad \text{and} \quad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_1(\boldsymbol{x}) \\ \phi_2(\boldsymbol{x}) \\ \vdots \\ \phi_L(\boldsymbol{x}) \end{bmatrix} \,.$$

Note that using nonlinear basis functions, we allow the function $f(\boldsymbol{x})$ to be nonlinear w.r.t. $\boldsymbol{x}$, while $f(\boldsymbol{x})$ is linear w.r.t. $\boldsymbol{w}$.

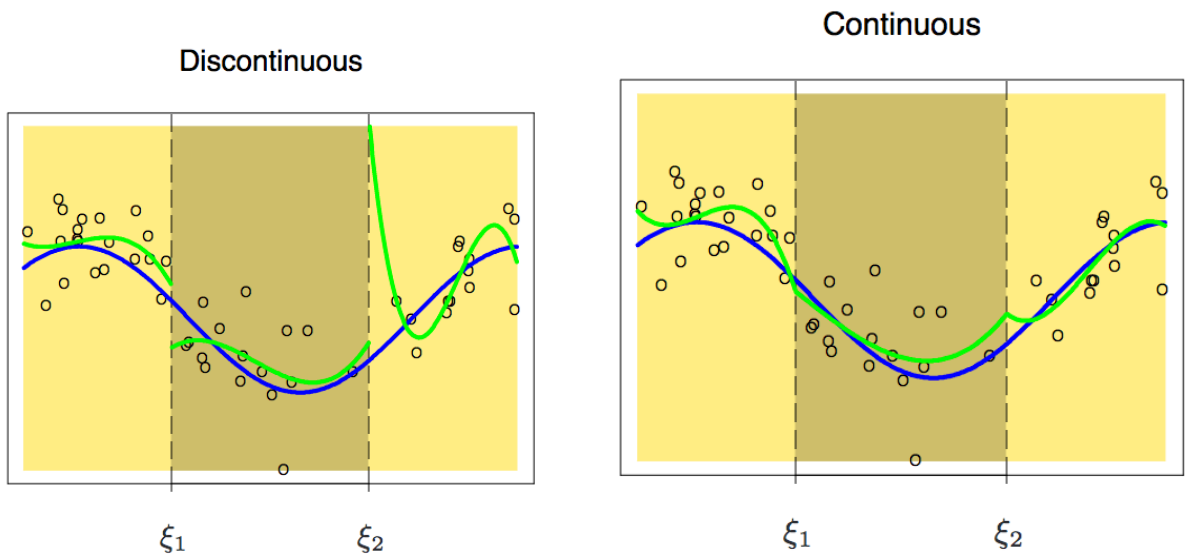
Polynomial Regression

$$f(x) = \sum_{\ell=1}^{L} w_\ell \phi_\ell(x) = \sum_{\ell=1}^{L} w_\ell x^{\ell-1} \ .$$



Basis Functions

- Polynomial basis: $\phi_\ell(x) = x^{\ell-1}$
- Gaussian basis: $\phi_\ell(x) = \exp\left(-\frac{\|x-\mu_\ell\|^2}{2\sigma^2}\right)$
- Spline basis: Piecewise polynomials, i.e., we divide the input space into several regions and fit a different polynomial in each region



- Fourier basis, hyperbolic tangent basis, sigmoidal basis, wavelet basis, ..., etc.

Least Square Method

Given a set of training data $\{\boldsymbol{x}_n \in \mathbb{R}^D, y_n \in \mathbb{R}\}_{n\in[N]}$, we determine the weight vector $\boldsymbol{w} = [w_1, \ldots, w_L]^\top$ to minimize

$$\mathcal{E}_{\mathrm{LS}}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(y_n - \boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2 = \frac{1}{2}\|\boldsymbol{y} - \Phi\boldsymbol{w}\|^2$$

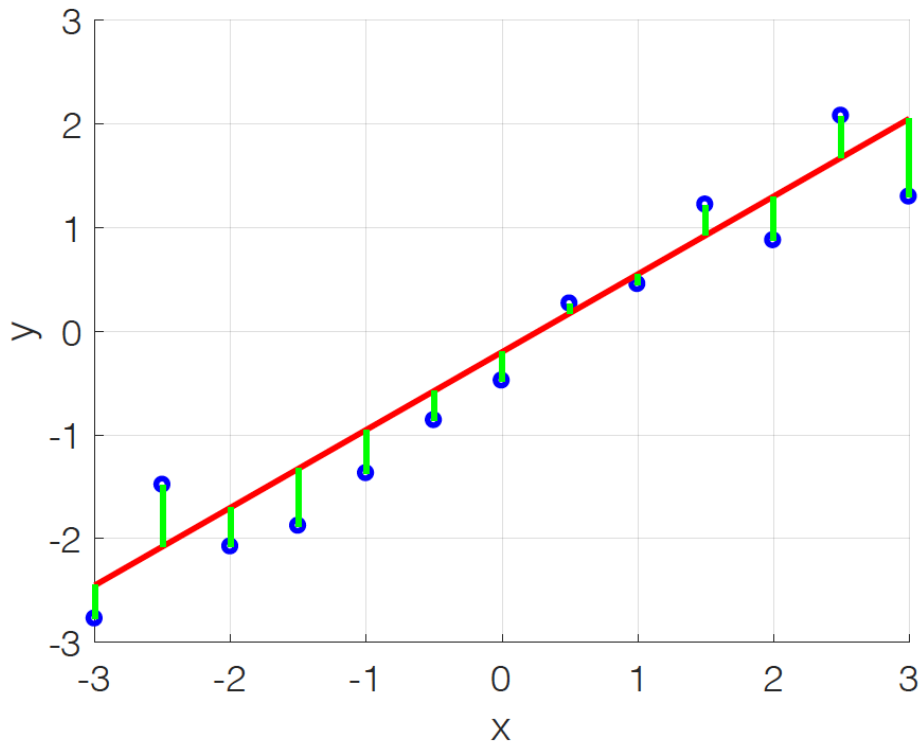
where $\boldsymbol{y} = [y_1, ..., y_N]^\top$ and design matrix Φ

$$\Phi = \begin{bmatrix} \phi_1(\boldsymbol{x}_1) & \phi_2(\boldsymbol{x}_1) & \ldots & \phi_L(\boldsymbol{x}_1) \\ \phi_1(\boldsymbol{x}_2) & \phi_2(\boldsymbol{x}_2) & \ldots & \phi_L(\boldsymbol{x}_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_N) & \phi_2(\boldsymbol{x}_N) & \ldots & \phi_L(\boldsymbol{x}_N)\ . \end{bmatrix} \in \mathbb{R}^{N\times L}$$

Least Square Error

Assuming model $y = w_1 x + w_2$. Least square method minimizes

$$\arg\min_{w_1, w_2} \frac{1}{2} \sum_{n=1}^{N} \left(y_n - (w_1 x_n + w_2)\right)^2 \ .$$



Least Square Estimate

The stability condition $\frac{\partial}{\partial \boldsymbol{w}} \mathcal{E}_{\mathsf{LS}}(\boldsymbol{w}) = \frac{\partial}{\partial \boldsymbol{w}} \frac{1}{2} \|\boldsymbol{y} - \Phi \boldsymbol{w}\|^2 = 0$ gives:

$$\Phi^\top \Phi \boldsymbol{w} = \Phi^\top \boldsymbol{y} \ .$$

Hence, the least square (LS) estimate $\boldsymbol{w}_{\mathsf{LS}}$ is given by

$$\boldsymbol{w}_{\mathsf{LS}} = \left(\Phi^\top \Phi\right)^{-1} \Phi^\top \boldsymbol{y} = \Phi^\dagger \boldsymbol{y} \ ,$$

where $\Phi^\dagger$ is known as the Moore-Penrose pseudo-inverse.

Table of Contents

An Understanding of LS: MLE

We assume that output y_n is given by a deterministic function $f(\boldsymbol{x}_n) = \boldsymbol{w}^\top \phi(\boldsymbol{x}_n)$ with additive Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$:

$$\boldsymbol{y} = \Phi \boldsymbol{w} + \varepsilon \ .$$

The log-likelihood $\mathcal{L}$ is given as:

$$\begin{aligned}\mathcal{L} &= \log p(\boldsymbol{y} \mid \Phi, \boldsymbol{w}) = \sum_{n=1}^{N} \log p(y_n \mid \phi(\boldsymbol{x}_n), \boldsymbol{w}) \\ &= -\frac{N}{2} \log \sigma^2 - \frac{N}{2} \log 2\pi - \frac{1}{\sigma^2} \underbrace{\left(\frac{1}{2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \phi(\boldsymbol{x}_n))^2 \right)}_{=\mathcal{E}_{\mathrm{LS}}} \ .\end{aligned}$$

Therefore, under the assumption of additive Gaussian noise,

$$\boldsymbol{w}_{\mathrm{LS}} = \boldsymbol{w}_{\mathrm{MLE}} \ .$$

An Understanding of LS: MLE

We assume that output y_n is given by a deterministic function $f(\boldsymbol{x}_n) = \boldsymbol{w}^\top \phi(\boldsymbol{x}_n)$ with additive Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$:

$$\boldsymbol{y} = \Phi \boldsymbol{w} + \varepsilon \ .$$

e.g., assuming $y = w_1 x + w_2 + \varepsilon$,

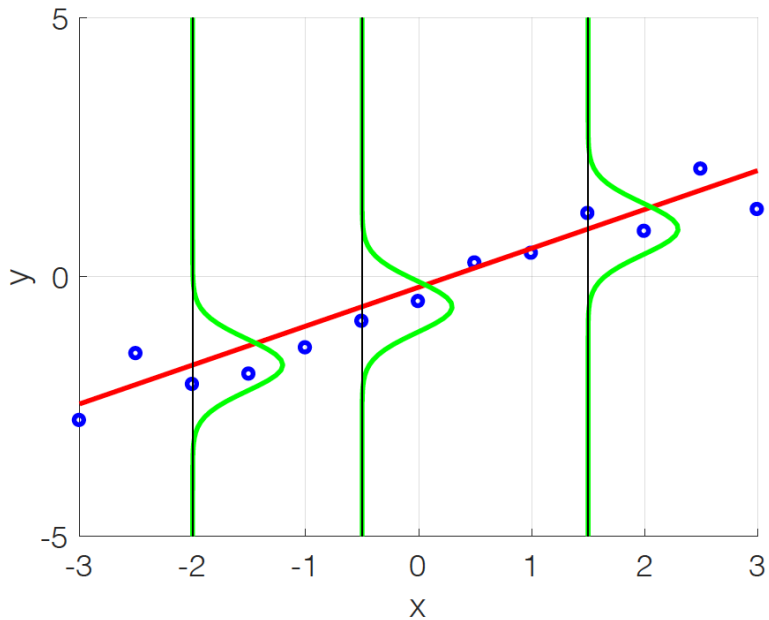
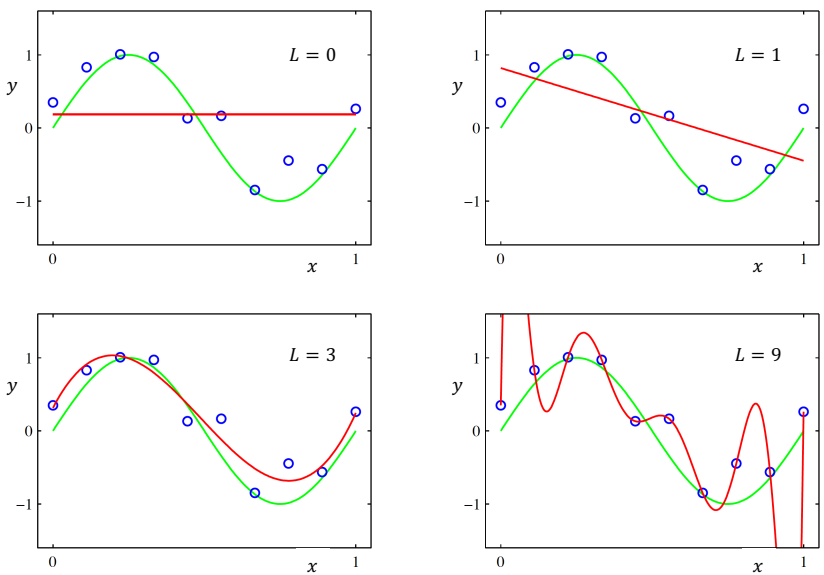


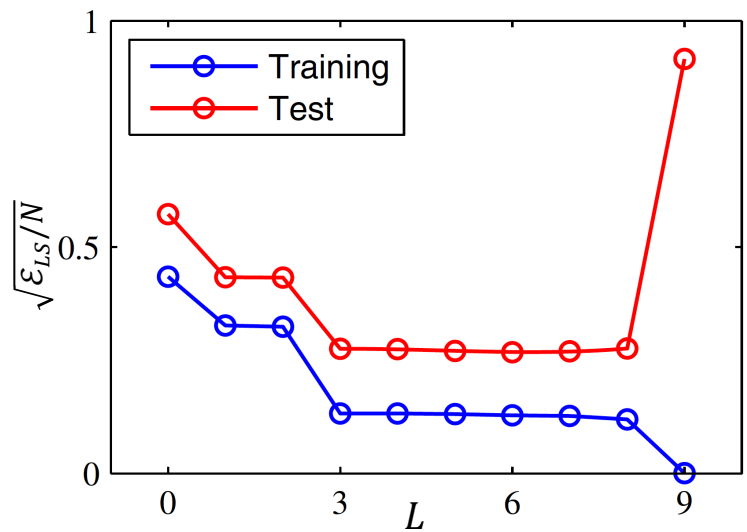
Table of Contents

Overfitting

When the number of parameters is large, an overfitting issue can occur:

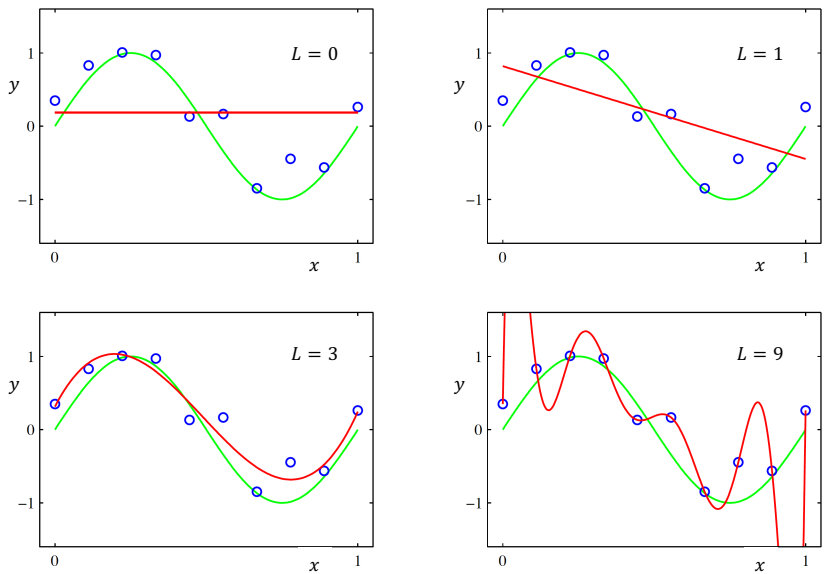
- The function approximation focuses on memorizing training data rather than extracting patterns
- Trade-off between training and test errors, e.g., polynomial curve fitting $f(x) = \sum_{\ell=1}^{L} w_\ell \phi_\ell(x) = \sum_{\ell=1}^{L} w_\ell x^{\ell-1}$





Regularization

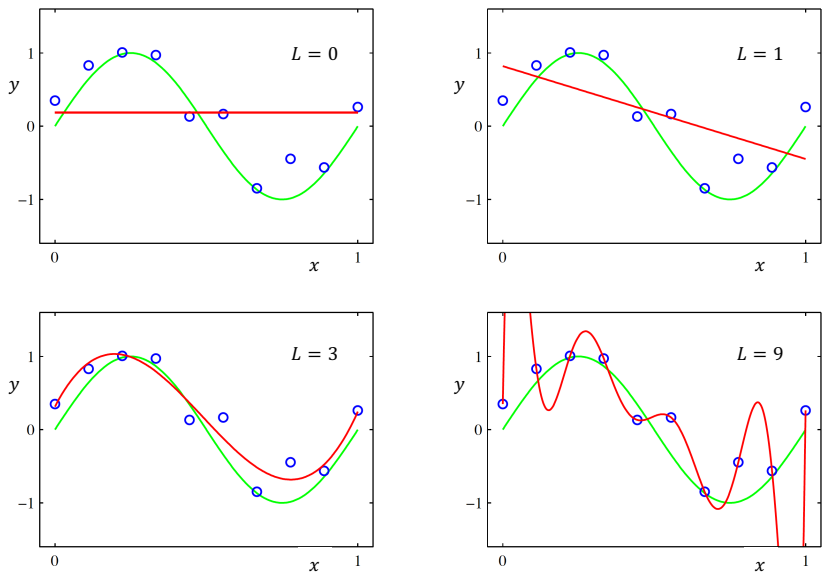
We often observe overfitting issues when the magnitude of parameters is large (or the function complexity is high):



	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Regularization

We often observe overfitting issues when the magnitude of parameters is large (or the function complexity is high):



	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Ridge regression minimizes the fitting error with a regularization term:

$$\mathcal{E} = \underbrace{\frac{1}{2}\|\boldsymbol{y} - \Phi\boldsymbol{w}\|^2}_{\text{Fitting error}} + \underbrace{\frac{\lambda}{2}\|\boldsymbol{w}\|^2}_{\text{Regularizer}},$$

where λ controls the trade-off.

Ridge Regression

Ridge regression minimizes

$$\mathcal{E} = \underbrace{\frac{1}{2}\|\boldsymbol{y} - \Phi\boldsymbol{w}\|^2}_{\text{Fitting error}} + \underbrace{\frac{\lambda}{2}\|\boldsymbol{w}\|^2}_{\text{Regularizer}} ,$$

where λ controls the trade-off.

Solving $\frac{\partial \mathcal{E}}{\partial \boldsymbol{w}} = 0$ for $\boldsymbol{w}$ leads to

$$\boldsymbol{w}_{\text{ridge}} = \left(\lambda I + \Phi^\top \Phi\right)^{-1} \Phi^\top \boldsymbol{y} .$$

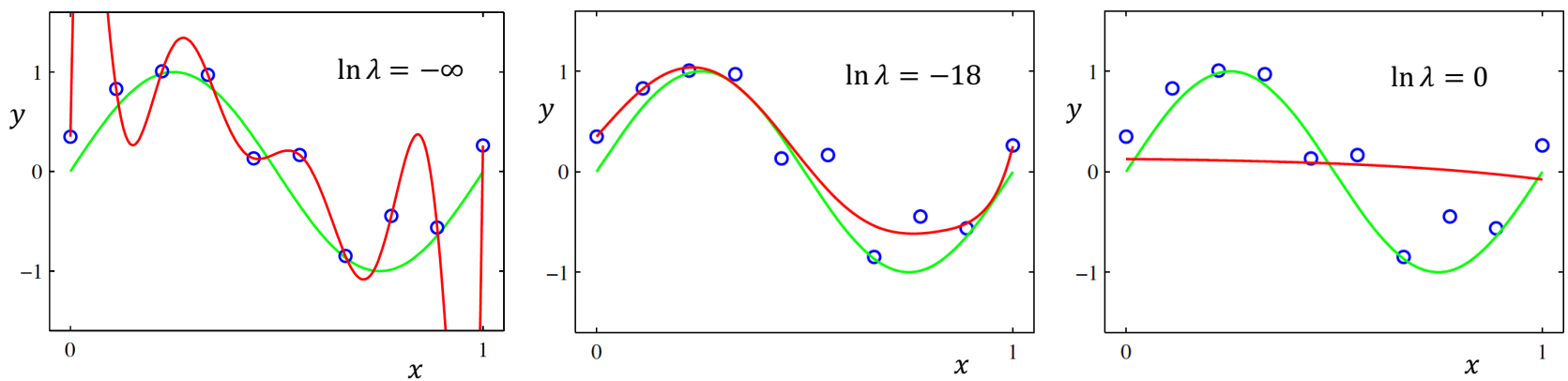


Table of Contents

Ridge Regression: MAP Perspective

From the observation, we believe that a good choice of parameter $\boldsymbol{w}$ may have a small magnitude, i.e., we assume a zero-mean Gaussian prior with covariance Σ for parameters $\boldsymbol{w}$:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|0, \Sigma) \ .$$

Recall that $\boldsymbol{y} = \Phi\boldsymbol{w} + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$, i.e.,

$$p(\boldsymbol{y}|\Phi, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{y} \mid \Phi\boldsymbol{w}, \sigma^2 I) \ .$$

Then the posterior over $\boldsymbol{w}$ is given as:

$$p(\boldsymbol{w}|\boldsymbol{y}, \Phi) = \frac{p(\boldsymbol{y}|\Phi, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\Phi)} \ ,$$

From the Gaussian identities, it can be seen that the posterior is still Gaussian with mean and mode at

$$\boldsymbol{w}_{MAP} = \left(\sigma^2\Sigma^{-1} + \Phi^\top\Phi\right)^{-1}\Phi^\top\boldsymbol{y} \ .$$

Lemma (Gaussian identities[1])

Consider an augmented random vector $z = [x^\top, y^\top]^\top$ *of which joint distribution is*

$$z = \begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} a \\ b \end{bmatrix}, \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix} \right) \quad \text{with cross-covariance } C = \mathbb{E}[xy^\top] \,,$$

so that $x \sim \mathcal{N}(a, A)$ *and* $y \sim \mathcal{N}(b, B)$*. Then, the conditional distributions are given as:*

$$p(x \mid y) = \mathcal{N}(x \mid a + CB^{-1}(y - b), A - CB^{-1}C^\top) \,,$$
$$p(y \mid x) = \mathcal{N}(y \mid b + C^\top A^{-1}(x - a), B - C^\top A^{-1}C) \,.$$

[1] c.f. Pattern Recognition and Machine Learning, Sec 2.3.1

Ridge Regression: MAP Perspective

Based on the belief that good parameter has small magnitude, i.e., prior $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|0, \Sigma)$, the posterior over $\boldsymbol{w}$ is given as:

$$p(\boldsymbol{w}|\boldsymbol{y}, \Phi) = \frac{p(\boldsymbol{y}|\Phi, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\Phi)} ,$$

which is maximized at

$$\boldsymbol{w}_{\text{MAP}} = \left(\sigma^2\Sigma^{-1} + \Phi^\top\Phi\right)^{-1}\Phi^\top\boldsymbol{y} .$$

Recall

$$\boldsymbol{w}_{\text{ridge}} = \left(\lambda I + \Phi^\top\Phi\right)^{-1}\Phi^\top\boldsymbol{y} .$$

When $\Sigma = \frac{\sigma^2}{\lambda}I$, the MAP becomes equivalent to ridge regression.

Table of Contents

Lasso Regression

- In ridge regression, we use ℓ_2 norm to regularize the weight of parameter $\boldsymbol{w}$.
- What if we use ℓ_1 norm as a regularization?
- Lasso regression minimizes

$$\mathcal{E}(\boldsymbol{w}) = \underbrace{\frac{1}{2}\|\boldsymbol{y} - \Phi\boldsymbol{w}\|^2}_{\text{Fitting error}} + \underbrace{\lambda|\boldsymbol{w}|}_{\text{Regularizer}} \quad ,$$

  where λ controls the trade-off.

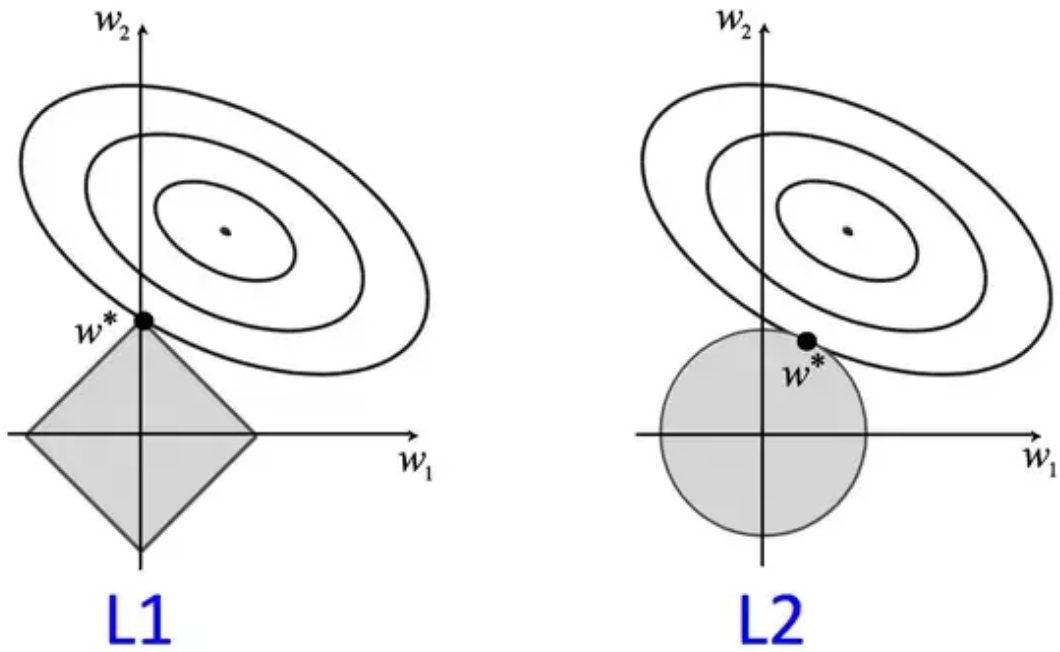
Lasso Regression

- In ridge regression, we use ℓ_2 norm to regularize the weight of parameter $\boldsymbol{w}$.
- What if we use ℓ_1 norm as a regularization?
- Lasso regression minimizes

$$\mathcal{E}(\boldsymbol{w}) = \underbrace{\frac{1}{2}\|\boldsymbol{y} - \Phi\boldsymbol{w}\|^2}_{\text{Fitting error}} + \underbrace{\lambda|\boldsymbol{w}|}_{\text{Regularizer}} \quad ,$$

  where λ controls the trade-off.
- Note that the objective is not differentiable.
  - You can use subgradient method instead.

Comparison between ℓ_2 and ℓ_1 Regularization



$$\mathcal{E}_{\text{Ridge}}(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \Phi\boldsymbol{w}\|^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

$$\mathcal{E}_{\text{Lasso}}(\boldsymbol{w}) = \underbrace{\frac{1}{2}\|\boldsymbol{y} - \Phi\boldsymbol{w}\|^2}_{\text{Quadratic}} + \lambda|\boldsymbol{w}|$$

Example: Lasso as Feature Selection

Term	OLS	Best Subset	Ridge	Lasso
intercept	2.465	2.477	2.467	2.465
lcalvol	0.676	0.736	0.522	0.548
lweight	0.262	0.315	0.255	0.224
age	-0.141	0.000	-0.089	0.000
lbph	0.209	0.000	0.186	0.129
svi	0.304	0.000	0.259	0.186
lcp	-0.287	0.000	-0.095	0.000
gleason	-0.021	0.000	0.025	0.000
pgg45	0.266	0.000	0.169	0.083
Test error	0.521	0.492	0.487	0.457
Std error	0.176	0.141	0.157	0.146

Figure: Results of different methods on the prostate cancer data, which has 8 features and 67 training cases. Methods are: OLS = ordinary least squares, Subset = best subset regression, Ridge, Lasso. Rows represent the coefficients; we see that subset regression and lasso give sparse solutions. Bottom row is the mean squared error on the test set (30 cases). (source: pp. 385,textbook)

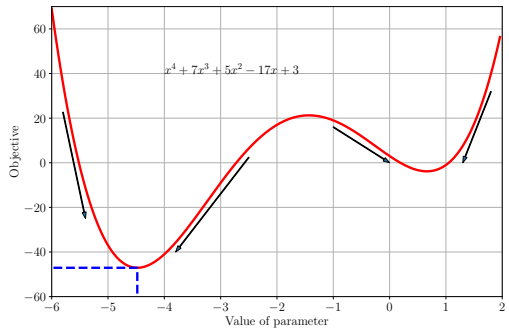
Table of Contents

Learning as Continuous Optimization

- We have defined a model with unknown parameters
- Many parameter estimation problem can be formulated as a continuous optimization problem.
- To find a maximum or minimum of a loss (or risk) function.

$$\arg\min_{\theta} L(\theta) = \arg\min_{\theta} \sum_{i=1}^{N} \ell(y_i, f(x_i, \theta))$$

Simple Example



- Say some cost function parameterized by θ with training set is:

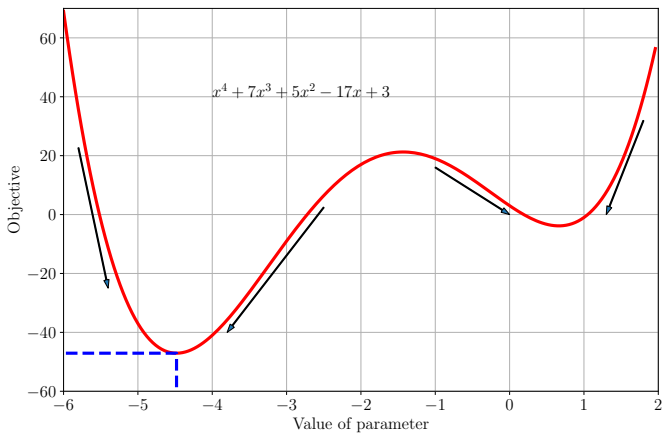
$$L(\theta) = \sum_{i=1}^{N} \ell(y_i, f(x_i, \theta)) = \theta^4 + 7\theta^3 + 5\theta^2 - 17\theta + 3$$

- To find the parameter which minimizes the empirical risk, we need to find a point where the gradient is zero.

$$\frac{dL(\theta)}{d\theta} = 0$$

- Use the second order derivative to check minimum or maximum.

Limitations



- Closed-form (or analytic) expression [2] of derivative is not available for some models with loss, i.e.:

$$\frac{dL(\theta)}{d\theta} = 0$$

  is not tractable.

- Abel-Ruffini theorem says there's no algebraic solution for polynomials of degree 5 or more.

[2] https://en.wikipedia.org/wiki/Closed-form_expression#Analytic_expression

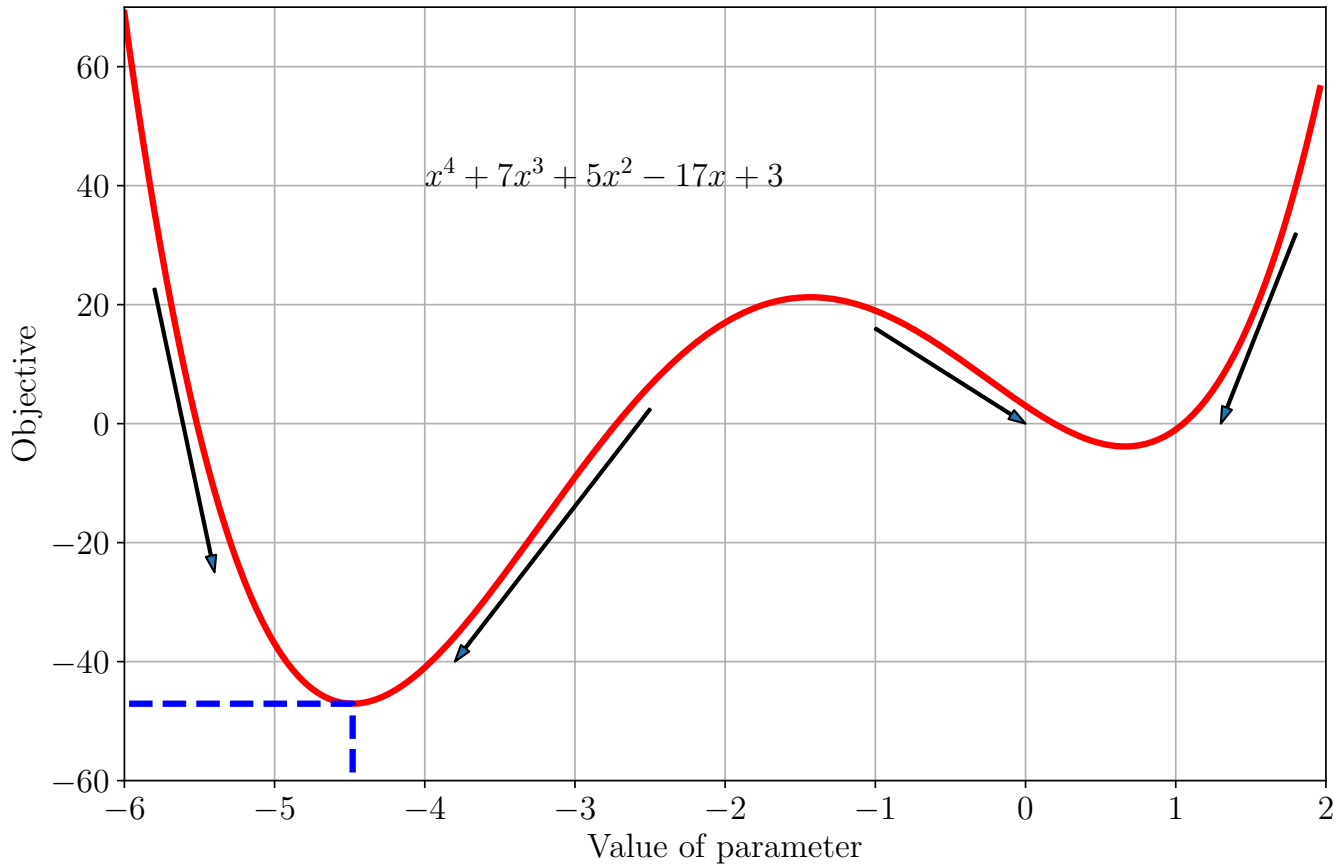
Gradient Descent (GD)

- Given objective function $L : \mathbb{R}^n \to \mathbb{R}$
- We will assume that L is *differentiable*, but unable to find a solution in analytic form.
  - Instead, we can evaluate the derivative at a given input point.
- To minimize the objective function, we start from initial point θ_0 and follow the gradient path as

$$\theta_1 = \theta_0 - \gamma(\underbrace{(\nabla L)(\theta_0)}_{\text{Derivative at } \theta_0})^\top,$$

  where $\gamma \geq 0$ is a small step-size.
- A simple gradient descent algorithm is:

$$\theta_{t+1} = \theta_t - \gamma_t((\nabla L)(\theta_t))^\top,$$


$x^4 + 7x^3 + 5x^2 - 17x + 3$
Objective
Value of parameter
60
40
20
0
−20
−40
−60
−6
−5
−4
−3
−2
−1
0
1
2

Example with Linear Regression

From the least square objective

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\boldsymbol{\phi}_n^\top \boldsymbol{w} - y_n\right)^2 = \frac{1}{2}\|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2$$

The gradient is

$$\boldsymbol{g}_t = \sum_{n=1}^{N}\left(\boldsymbol{w}_t^\top \boldsymbol{\phi}_n - y_n\right)\boldsymbol{\phi}_n$$

The update becomes

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma_t \sum_{n=1}^{N}\left(\boldsymbol{w}_t^\top \boldsymbol{\phi}_n - y_n\right)\boldsymbol{\phi}_n$$

Example with Quadratic Function

$$L\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{2}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- With $\boldsymbol{x}_0 = [-3, -1]^\top$ and $\gamma = 0.085$, GD converges to the minimum.
- GD converges slowly (ill-conditioned).

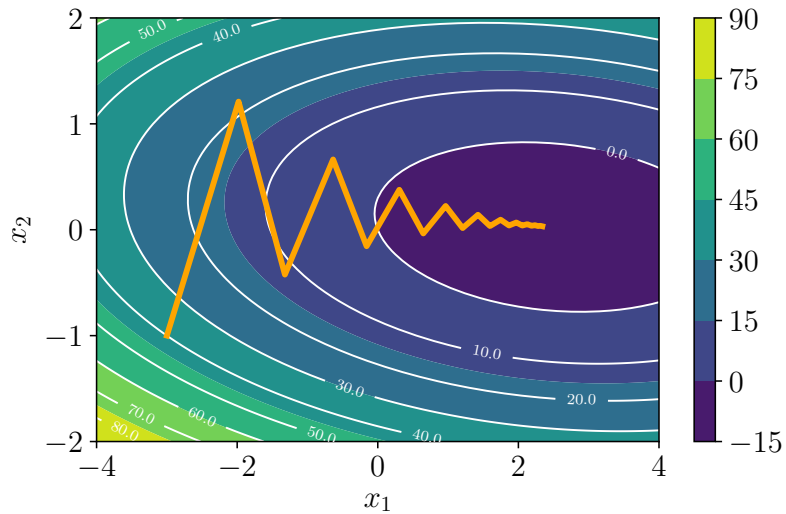


Figure: GD converges to the minimum

Stepsize γ_i

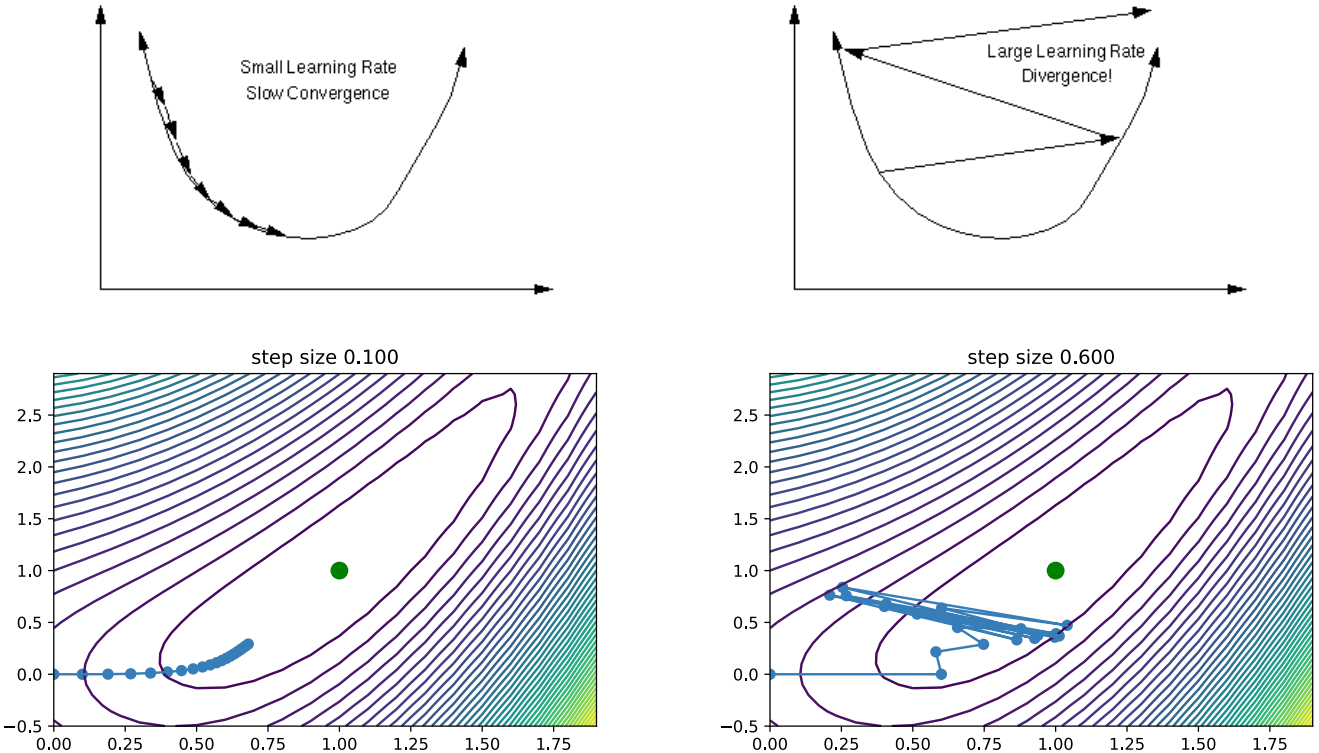


Figure: GD with different learning rates.

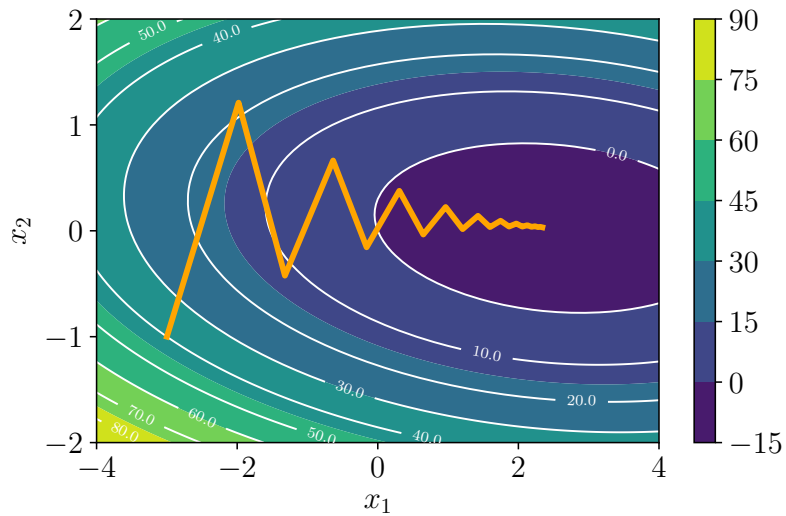
- Good step-size is important in gradient descent.
- If it is too small, GD can be slow.
- If it is too large, GD can overshoot, fail to converge, or even diverge.

Simple Heuristics to Find Good Stepsize

- When the function value increases after a gradient step, the step-size was too large.
  - Undo the step and decrease the step-size
- When the function value decreases, the step could have been larger
  - Try to increase the step-size.
- Is this a good solution?
  - What if it takes too much time to evaluate the objective function?
- Let's talk about some techniques to improve.

Gradient Descent with Momentum

- The GD update oscillates along the second axis.
- Give gradient descent some memory to avoid this behavior.
- Momentum is an additional term to remember what happened in the previous iteration.



Momentum Methods

$$\theta_{i+1} = \overbrace{\theta_i - \gamma_i((\nabla f)(\theta_i))^\top}^{\text{GD}} + \overbrace{\alpha \Delta\theta_i}^{\text{momentum}} \qquad \alpha \in [0, 1]$$
$$\Delta\theta_i = \theta_i - \theta_{i-1} = \alpha\Delta\theta_{i-1} - \gamma_{i-1}((\nabla f)(\theta_{i-1}))^\top$$

- The momentum-based method remembers the previous update.
- Determines the next update as a linear combination of the current and previous gradients.
- The memory dampens oscillations and smoothes out the gradient updates.
  - It also accelerates some updates along the way[3].

[3] Illustrative example: `https://distill.pub/2017/momentum/`

Stochastic Gradient Descent (SGD)

- Note that the *(batch)* gradient given a model is a sum of gradient with respect to individual data points

$$L(\theta) = \sum_{i=1}^{N} L_i(\theta) = \sum_{i=1}^{N} \ell(y_i, f_\theta(x_i))$$

$$\frac{dL(\theta)}{d\theta} = \sum_{i=1}^{N} \frac{dL_i(\theta)}{d\theta}$$

- If the dataset is too large, computing a single estimate of gradient take too much time.
- We may approximate the gradient with only a few number of data points *(mini-batch)*.

Noisy Gradient

- If we take a few data points and computes the gradient:

$$\frac{dL_{\mathcal{B}}(\theta)}{d\theta} = \sum_{i:x_i \in \mathcal{B}} \frac{dL_i(\theta)}{d\theta}$$

  where $\mathcal{B}$ is a subset of data $\mathcal{D} = \{(x_1, y_1), ..., (x_N, y_N)\}$.

- This makes sense because the batch gradient is expectation of gradient w.r.t data distribution $p(x, y)$.

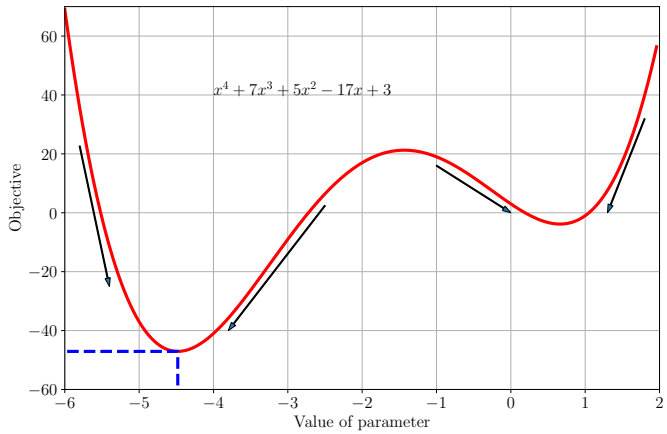
- Likewise, the SGD is an expectation of gradient w.r.t data distribution

$$\mathbb{E}\left[\frac{dL(\theta)}{d\theta}\right] = \mathbb{E}\left[\sum_{i=1}^{N} \frac{dL_i(\theta)}{d\theta}\right] = N\,\mathbb{E}\left[\frac{dL_i(\theta)}{d\theta}\right]$$

$$= N\,\mathbb{E}\left[\frac{1}{|\mathcal{B}|}\sum_{i:x_i \in \mathcal{B}} \frac{dL_i(\theta)}{d\theta}\right] = \frac{N}{|\mathcal{B}|}\,\mathbb{E}\left[\sum_{i:x_i \in \mathcal{B}} \frac{dL_i(\theta)}{d\theta}\right]$$

Why Stochastic Gradient?

- Stochastic gradient descent often results a better solution than the batch gradient descent
- The noise in the gradient allow us to escape some bad local optima (sometimes!).
- The batch gradient cannot escape from the local optima.



Remark

SGD converges when the learning rate decreases at an appropriate rate.

Summary

- Linear regression and least square (LS) method
- Interpretation of LS method: maximum likelihood estimate (MLE)
- Overfitting issue and regularization (ridge regression)
- Interpretation of ridge regression: maximum a posteriori (MAP)
- Non-linear regression and gradient method

Further Readings

- Chapter 11 (Linear Regression) of Textbook
  - Chapter 11.2 Least squares linear regression
  - Chapter 11.3 Ridge regression
  - Chapter 11.4 Lasso regression
- Chapter 8 (Optimization) of Textbook
  - Chapter 8.2 First-order methods
  - Chapter 8.4 Stochastic gradient descent