Lecture 03: Gaussian Process Regression I

[AIX7026] Advanced Machine Learning

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

Contents

1. Linear regression

1.1 Linear regression

1.2 Gradient descent

1.3 Polynomial regression

1.4 Regularized linear model

2. Gaussian process

2.1 Bayesian method

2.2 Multivariate Gaussians

2.3 Bayesian linear regression

2.4 Gaussian process

2.5 Appendix

Contents

1. Linear regression

1.1 Linear regression

1.2 Gradient descent

1.3 Polynomial regression

1.4 Regularized linear model

2. Gaussian process

2.1 Bayesian method

2.2 Multivariate Gaussians

2.3 Bayesian linear regression

2.4 Gaussian process

2.5 Appendix

Linear regression

Linear model

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{1}$$

In this equation,

- $\hat{y}$ is the predicted value (for true y)
- n is the number of features
- x_i is the i-th feature value
- θ is the j-th model parameter (including the bias term θ_0 and the feature weights $\theta_1, \theta_2, ..., \theta_n$)

Linear regression

This can be written much more concisely using a vectorized form,

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{2}$$

$$= h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x} \tag{3}$$

In this equation,

- $\boldsymbol{\theta}$ is the model's *parameter vector*, containing the bias term θ_0 and the feature weights θ_1 to θ_n
- $\mathbf{x}$ is the instance's *feature vector*, containing x_0 to x_n always equal to 1
- $\boldsymbol{\theta} \cdot \mathbf{x}$ is the dot product of the vectors $\boldsymbol{\theta}$ and $\mathbf{x}$, which is of course equal to $\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$
- $h_{\boldsymbol{\theta}}$ is the hypothesis function, using the model parameter $\boldsymbol{\theta}$

Linear regression

That's the linear regression model – but how do we train it?

Recall that training a model means setting its parameters so that the model best fits the training set.

We first need a measure of how well (or poorly) the model fits the training data.

The most common performance measure of a regression model is the Root Mean Square Error (RMSE).

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h(\mathbf{x}^{(i)}) - y^{(i)}\right)^2} \tag{4}$$

Linear regression

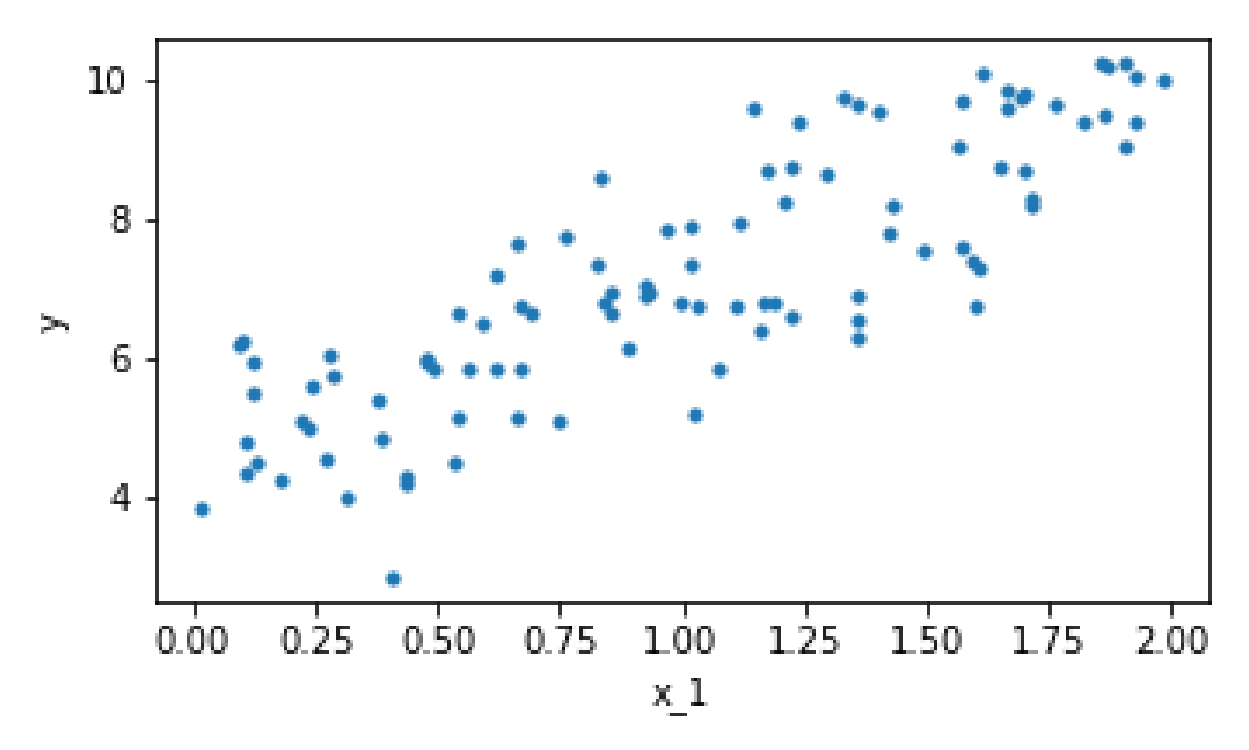


Figure 1: Linear regression: training dataset

Generating training dataset

$$y \approx \theta_0 + \theta_1 x \tag{5}$$

$$y = \theta_0 + \theta_1 x + \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{6}$$

Linear regression: design matrix

Design matrix (regressor matrix, model matrix, data matrix)

- 훈련 데이터(sample, example)이 m개
- feature vector $\mathbf{x}$의 차원이 n일 때,
- m개의 sample을 row vector로 한 design matrix $\mathbf{X}$로 표시할 수 있음

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \tag{7}$$

$$\mathbf{X} = \begin{bmatrix} \text{1st sample } \tilde{x}^{(1)} \\ \text{2nd sample } \tilde{x}^{(2)} \\ \cdots \\ \text{m-th sample } \tilde{x}^{(m)} \end{bmatrix} = \begin{bmatrix} \tilde{x}_1^{(1)} & \tilde{x}_2^{(1)} & \cdots & \tilde{x}_n^{(1)} \\ \tilde{x}_1^{(2)} & \tilde{x}_2^{(2)} & \cdots & \tilde{x}_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{x}_1^{(m)} & \tilde{x}_2^{(m)} & \cdots & \tilde{x}_n^{(m)} \end{bmatrix} \tag{8}$$

Linear regression: design matrix

그러면, $\hat{y} = \boldsymbol{\theta}^T \mathbf{x}$의 m개의 sample에 대해 다음과 같이 표현할 수 있음

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta} \tag{9}$$

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} \tilde{x}_1^{(1)} & \tilde{x}_2^{(1)} & \cdots & \tilde{x}_n^{(1)} \\ \tilde{x}_1^{(2)} & \tilde{x}_2^{(2)} & \cdots & \tilde{x}_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{x}_1^{(m)} & \tilde{x}_2^{(m)} & \cdots & \tilde{x}_n^{(m)} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{x}}^{(1)} \\ \tilde{\mathbf{x}}^{(2)} \\ \vdots \\ \tilde{\mathbf{x}}^{(m)} \end{bmatrix} \boldsymbol{\theta} \tag{11}$$

Normal equation: geometric approach

Design matrix $\mathbf{X}$의 각 열을 $\vec{\mathbf{x}}_j$라고 하면

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta} = \begin{bmatrix} \vec{\mathbf{x}}_1 & \vec{\mathbf{x}}_2 & \cdots & \vec{\mathbf{x}}_n \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \tag{12}$$

$$= \theta_1 \vec{\mathbf{x}}_1 + \theta_2 \vec{\mathbf{x}}_2 + \cdots + \theta_n \vec{\mathbf{x}}_n \tag{13}$$

즉, $\hat{\mathbf{y}}$는 $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \cdots, \vec{\mathbf{x}}_n$이 생성하는 hyperplane 상에 존재함

Residual 또는 error의 크기 $\|\mathbf{y} - \hat{\mathbf{y}}\|$를 최소화 하려면? 위의 hyperplane과 error $\mathbf{y} - \hat{\mathbf{y}}$가 서로 수직(orthogonal)해야함

Normal equation: geometric interpretation

Residual 또는 error의 크기 $\|\mathbf{y} - \hat{\mathbf{y}}\|$를 최소화 하려면? 위의 hyperplane과 error $\mathbf{y} - \hat{\mathbf{y}}$가 서로 수직(orthogonal)해야함

즉, 모든 column vector에 대해서

$$\vec{\mathbf{x}}_j(\mathbf{y} - \hat{\mathbf{y}}) = 0 \tag{14}$$

이 성립해야 함

전체 m개의 sample에 대해서

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = 0 \quad \Longrightarrow \quad \mathbf{X}^T\mathbf{X}\boldsymbol{\theta} = \mathbf{X}^T\mathbf{y} \tag{15}$$

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{16}$$

Normal equation: geometric interpretation

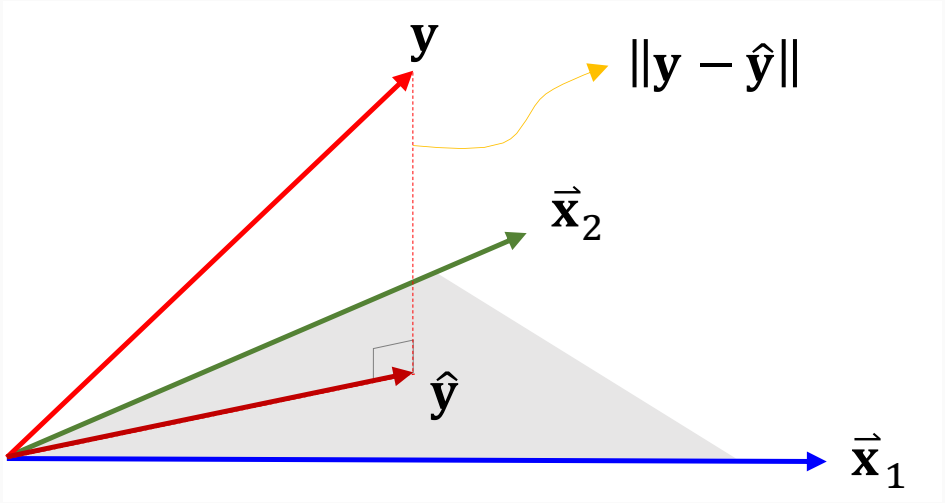


Figure 2: Normal equation: geometric interpretation

Normal equation

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{17}$$

Projection matrix

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\theta}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{18}$$

Normal equation: analytic approach

Sum of squared error (SSE)

$$\mathrm{SSE} = \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2 \tag{19}$$

$$= \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) \tag{20}$$

Using $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$

$$\mathrm{SSE}(\boldsymbol{\theta}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\boldsymbol{\theta} + (\mathbf{X}\boldsymbol{\theta})^T(\mathbf{X}\boldsymbol{\theta}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} \tag{21}$$

$$\frac{\partial \mathrm{SSE}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 2(\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - \mathbf{X}\mathbf{y}) = 0 \quad \Longrightarrow \quad \mathbf{X}^T\mathbf{X}\boldsymbol{\theta} = \mathbf{X}^T\mathbf{y} \tag{22}$$

Hence, we have

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{23}$$

$$\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{24}$$

Linear regression: cost function

We need to find the value of $\boldsymbol{\theta}$ that minimizes the RMSE. In practice, it is simpler to minimize the sum of squared error (SSE) than the MSE or the RMSE.

$$\hat{\boldsymbol{\theta}} = \arg\min \mathrm{RMSE}(\mathbf{X}, h_{\boldsymbol{\theta}}) = \arg\min \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2} \tag{25}$$

$$= \arg\min \mathrm{MSE}(\mathbf{X}, h_{\boldsymbol{\theta}}) \quad = \arg\min \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2 \tag{26}$$

$$= \arg\min \mathrm{SSE}(\mathbf{X}, h_{\boldsymbol{\theta}}) \quad = \arg\min \sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2 \tag{27}$$

Linear regression: closed from solution

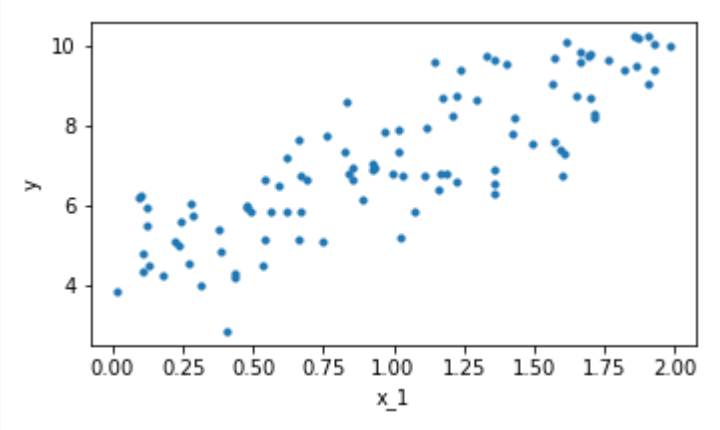


Figure 3: Linear regression: training dataset

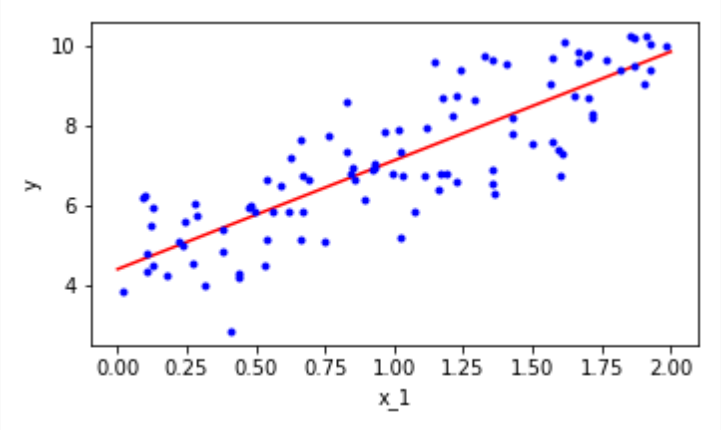


Figure 4: Linear regression: closed form solution

Linear regression: computational complexity

Normal equation의 계산

- Normal equation에 의한 예측치 $\hat{\mathbf{y}}$

$$\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{28}$$

- $\mathbf{X} \in \mathcal{R}^{m \times n}$
- $\mathbf{X}^T\mathbf{X} \in \mathcal{R}^{n \times n}$
- $(\mathbf{X}^T\mathbf{X})^{-1}$ 의 계산 복잡도 = $O(n^{2.4}) \sim O(n^3)$
- Feature 수의 약 세제곱으로 계산 시간이 증가

Contents

1. Linear regression

1.1 Linear regression

1.2 Gradient descent

1.3 Polynomial regression

1.4 Regularized linear model

2. Gaussian process

2.1 Bayesian method

2.2 Multivariate Gaussians

2.3 Bayesian linear regression

2.4 Gaussian process

2.5 Appendix

Batch gradient descent

Linear regression model $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$

$$J(\boldsymbol{\theta}) = \mathrm{SSE}(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 \tag{29}$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 2\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \tag{30}$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial}{\partial \theta_1} J(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} J(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} 2\sum_{i=1}^m x_1^{(i)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}\right) \\ 2\sum_{i=1}^m x_2^{(i)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}\right) \\ \vdots \\ 2\sum_{i=1}^m x_n^{(i)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}\right) \end{bmatrix} \tag{31}$$

Gradient descent step

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^t) \tag{32}$$

where iteration number t and $\boldsymbol{\theta}$ arbitrary initial value

Batch gradient descent

Batch gradient descent에서

$$\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 2\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \tag{33}$$

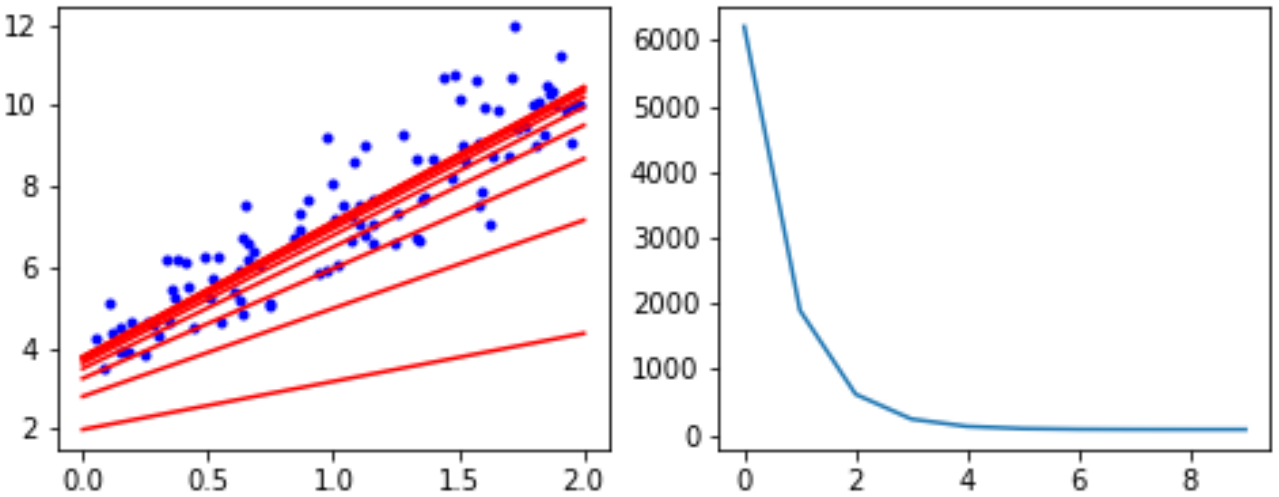
$$= 2 \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}2^{(m)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}^T\mathbf{x}^{(1)} - y^{(1)} \\ \boldsymbol{\theta}^T\mathbf{x}^{(2)} - y^{(2)} \\ \vdots \\ \boldsymbol{\theta}^T\mathbf{x}^{(m)} - y^{(m)} \end{bmatrix} \tag{34}$$

$$= 2\sum_{i=1}^{m} \mathbf{x}^{(i)} \left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)} \right) \tag{35}$$

그러므로 이 gradient vector의 j번째 component는

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = 2\sum_{i=1}^{m} x_j^{(i)} \left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)} \right) \tag{36}$$

Batch gradient descent



Batch gradient descent: computational complexity

Batch gradient descent algorithm

- 매 스텝마다 batch 전체에 대한 계산 필요
- 데이터셋이 커지면 속도가 느려짐
- Normal equation: feature 수에 따라 계산 속도가 지수적으로 느려짐
- Gradient descent: feature 수가 늘어도 크게 변하지 않음

Learning rate

- Hyperparameter인 학습률(learning rate) η가 너무 작은 경우 시간이 오래 걸림
- 너무 큰 경우 최적해를 지나쳐 해를 찾지 못할 수 있음

Learning schedule

- Constant learning rate
  - 보통 0.1, 0.01부터 시작하여 여러 가지 값으로 시험해보며 범위를 좁혀 나감
- Time-based decay

$$\eta = \frac{\eta_0}{(1 + kt)} \tag{37}$$

η_0 : 학습률 초기값, k : hyperparameter, t : iteration

- Step decay
  - 정해진 epoch마다 학습률을 줄이는 방법
  - 예: 5 epoch마다 반으로, 20 epoch마다 1/10로
  - Epoch: 훈련 데이터셋 전체를 모두 사용할 때 = 한 epoch
- Exponential decay

$$\eta = \eta_0 e^{-kt} \tag{38}$$

η_0 : 학습률 초기값, k : hyperparameter, t : iteration

Stochastic gradient descent

For our linear regression model $\hat{y} = \boldsymbol{\theta}^T \mathbf{x}$

$$J(\boldsymbol{\theta}) = \mathrm{SSE}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left(\mathbf{y}^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)} \right)^2 = \sum_{i=1}^{m} J_i(\boldsymbol{\theta}) \tag{39}$$

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - 2\eta \left(y^{(t)} - (\boldsymbol{\theta}^t)^T \mathbf{x}^{(t)} \right) \mathbf{x}^{(t)} \tag{40}$$

- 무작위로 선택한 한 개의 sample에 대해서만 gradient를 계산하여 parameter를 update
- sequential learning or online learning
- 대규모 데이터셋을 처리하는데 유리
- 선택하는 사례의 무작위성으로 움직임이 불규칙
- BGD에 비해 local optimum에서 쉽게 빠져나올 수 있음
- 최적해에 도달하지만 지속적으로 요동
- BGD와 마찬가지로 global optimum이라는 보장이 없음

Mini-batch gradient descent

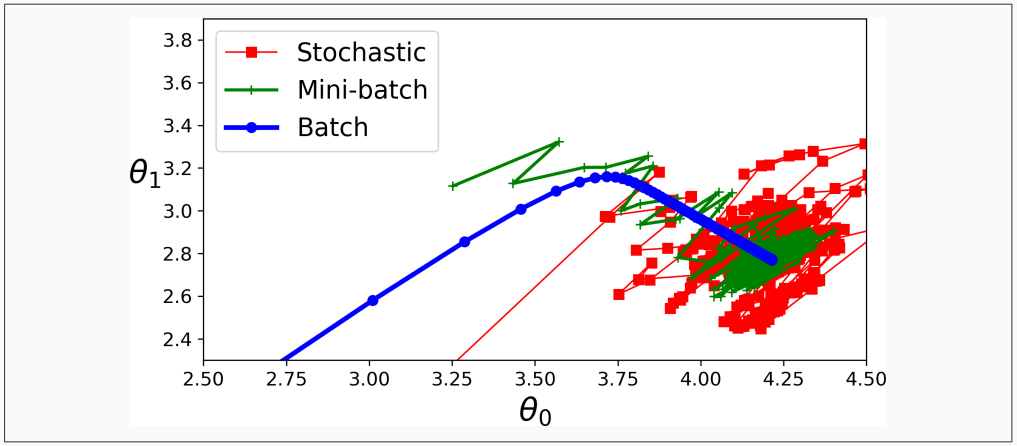


Figure 4-11. Gradient Descent paths in parameter space

- 훈련 데이터셋을 작은 크기의 무작위 부분 집합으로 나누어서 gradient를 구하는 방법
- 예 100,000개의 데이터 = (mini-batch size 100) $\times$ (1,000 mini-batches)
- Batch gradient descent와 stochastic gradient descent(SGD)의 절충
- SGD보다 불규칙한 움직임이 덜함
- SGD보다 local minimum에서 빠져나오기가 상대적으로 더 어려움
- GPU를 통한 매트릭스 연산의 속도를 높일 수 있음

Linear regression comparison

Table 4-1. Comparison of algorithms for Linear Regression

Algorithm	Large *m*	Out-of-core support	Large *n*	Hyperparams	Scaling required	Scikit-Learn
Normal Equation	Fast	No	Slow	0	No	n/a
SVD	Fast	No	Slow	0	No	`LinearRegression`
Batch GD	Slow	No	Fast	2	Yes	`SGDRegressor`
Stochastic GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`
Mini-batch GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`

There is almost no difference after training: all these algorithms end up with very similar models and make predictions in exactly the same way.

Contents

1. Linear regression

1.1 Linear regression

1.2 Gradient descent

1.3 Polynomial regression

1.4 Regularized linear model

2. Gaussian process

2.1 Bayesian method

2.2 Multivariate Gaussians

2.3 Bayesian linear regression

2.4 Gaussian process

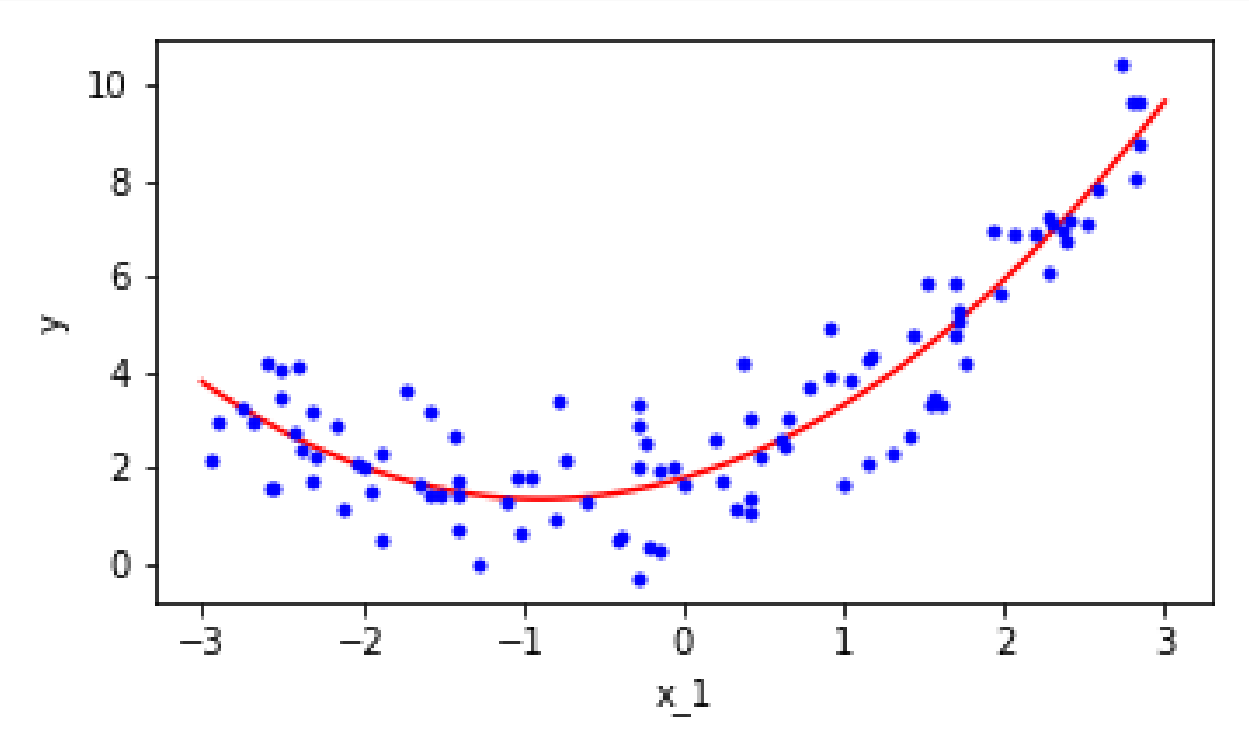
2.5 Appendix

Polynomial regression

Polynomial regression

- 데이터들이 비선형 관계를 이루고 있을 때
- polynomial의 각 항을 별개의 feature로 간주한 선형 모델과 유사
- 예 : x_1에 대한 2차 polynomial

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 \tag{41}$$



Polynomial regression: closed form solution

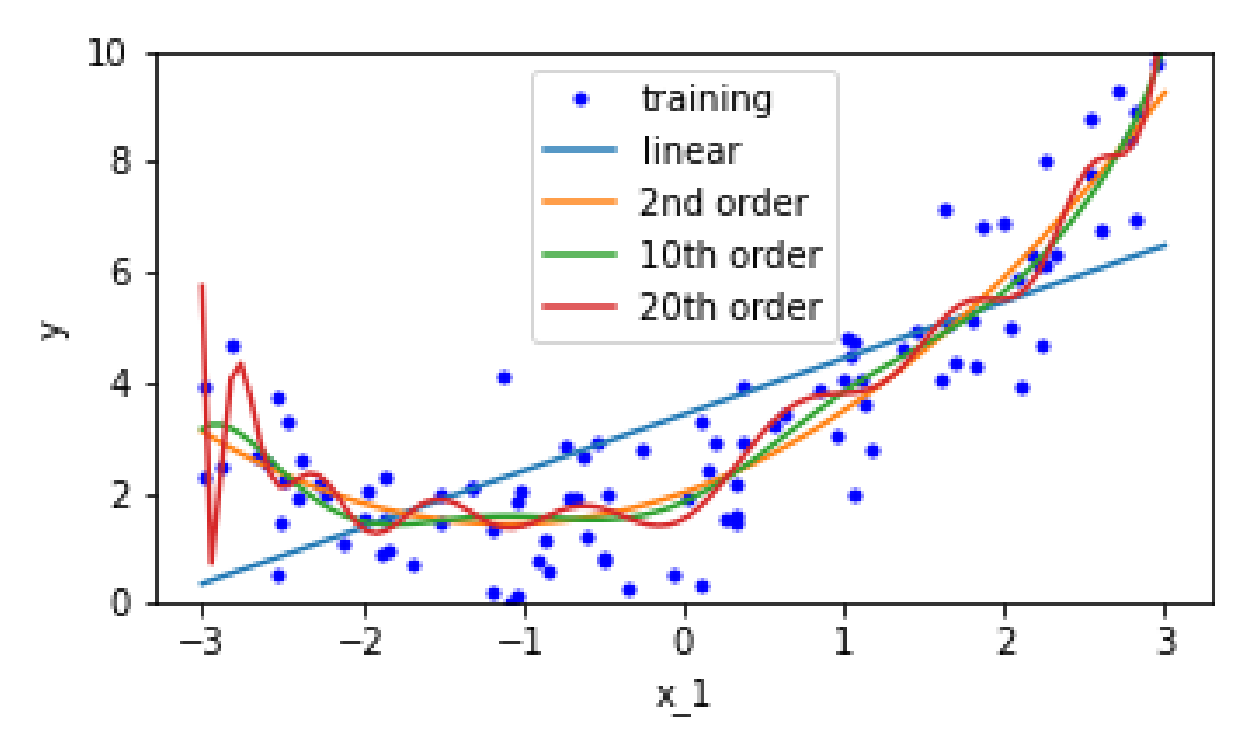


Figure 6: Polynomial regression: closed form solution

Learning curves: model selection

Learning curves

- 훈련 데이터셋의 크기 또는 반복되는 훈련의 횟수에 따른 training error와 validation error를 그린 것
- 각 error의 상대적인 변화를 보고 모델을 평가할 수 있음

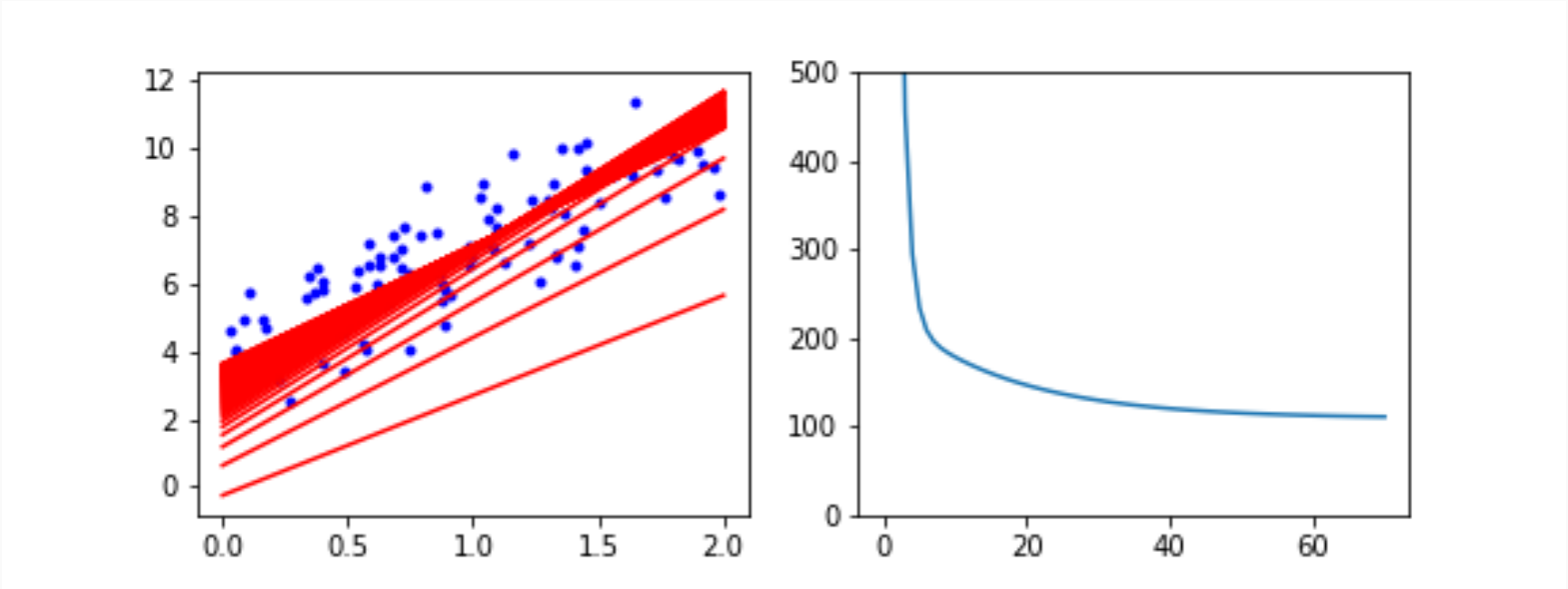


Figure 7: Linear regression with learning curve (SSE)

Learning curves: model selection

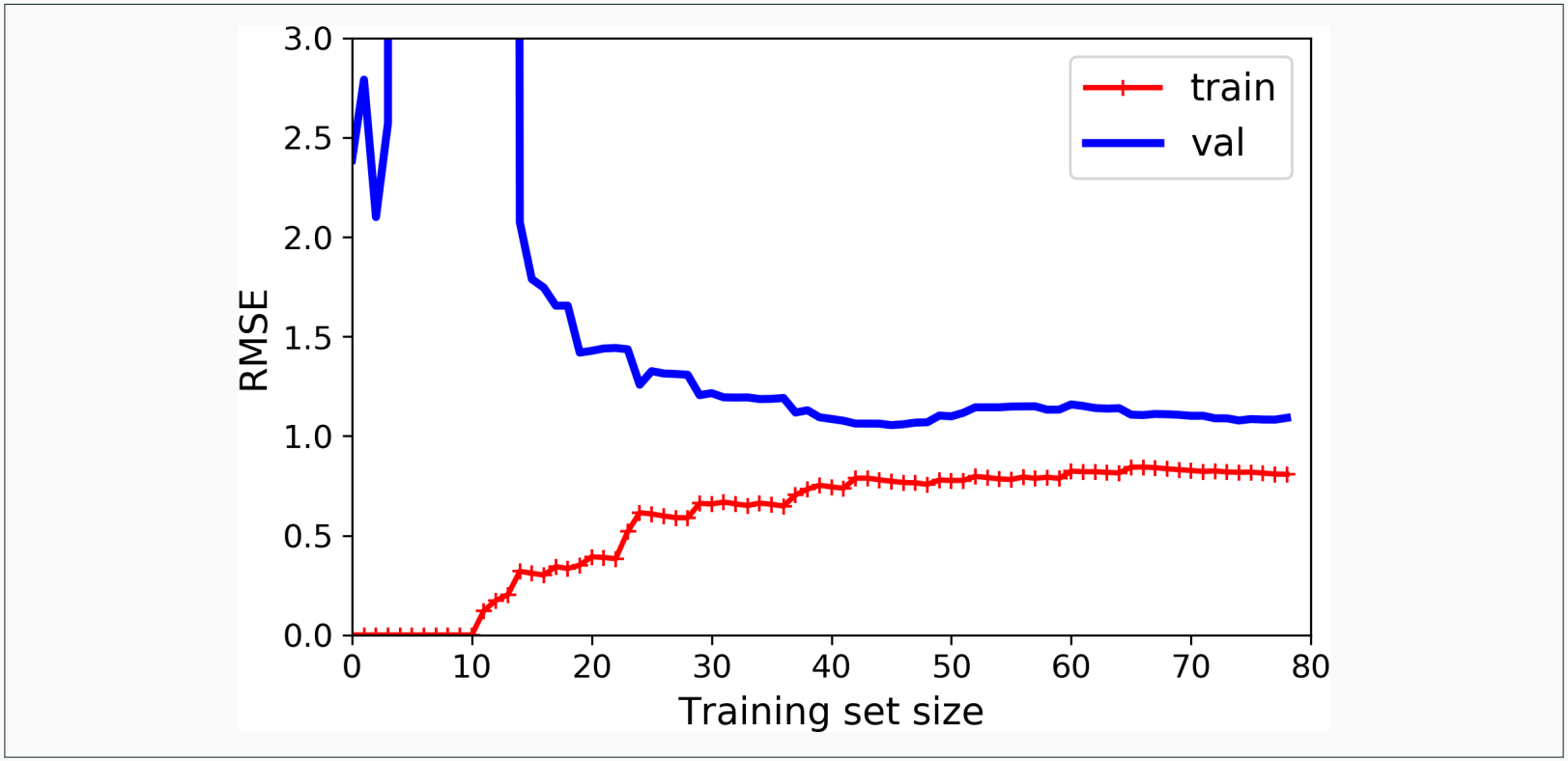


Figure 4-16. Learning curves for the polynomial model

Contents

1. Linear regression

1.1 Linear regression

1.2 Gradient descent

1.3 Polynomial regression

1.4 Regularized linear model

2. Gaussian process

2.1 Bayesian method

2.2 Multivariate Gaussians

2.3 Bayesian linear regression

2.4 Gaussian process

2.5 Appendix

Ridge regression

SSE 비용 함수에 model parameter에 대한 penalty를 추가

- 과적합을 방지하면서 model parameter를 가능한 작게 만들도록 함

$$J(\boldsymbol{\theta}) = \mathrm{SSE}(\boldsymbol{\theta}) + \frac{\alpha}{2}\|\boldsymbol{\theta}\|^2 \tag{42}$$

- Hyperparameter α로 cost의 두 항목 사이의 상대적인 비중을 조절할 수 있음
- Closed form solution $\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}_n)^{-1}\mathbf{X}^T\mathbf{y}$

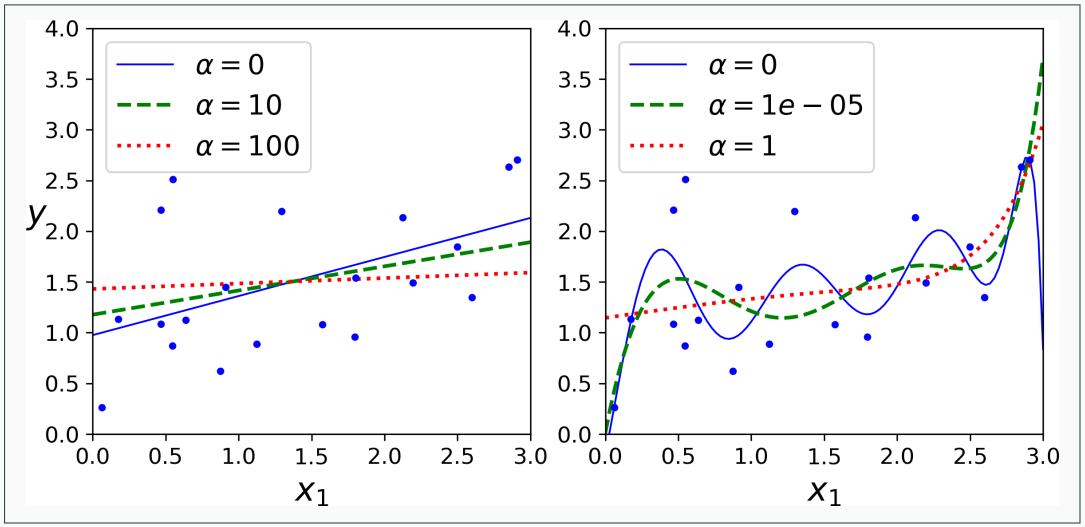


Figure 4-17. Ridge Regression

Lasso regression

SSE 비용 함수에 model parameter에 대한 penalty를 추가

- Least Absolute Shrinkage and Selection Operator Regression (LASSO)
- 중요하지 않은 feature들의 parameter를 완전히 0으로 만드는 경향

$$J(\boldsymbol{\theta}) = \mathrm{SSE}(\boldsymbol{\theta}) + \alpha \sum_{i=1}^{n} |\theta_i| \tag{43}$$

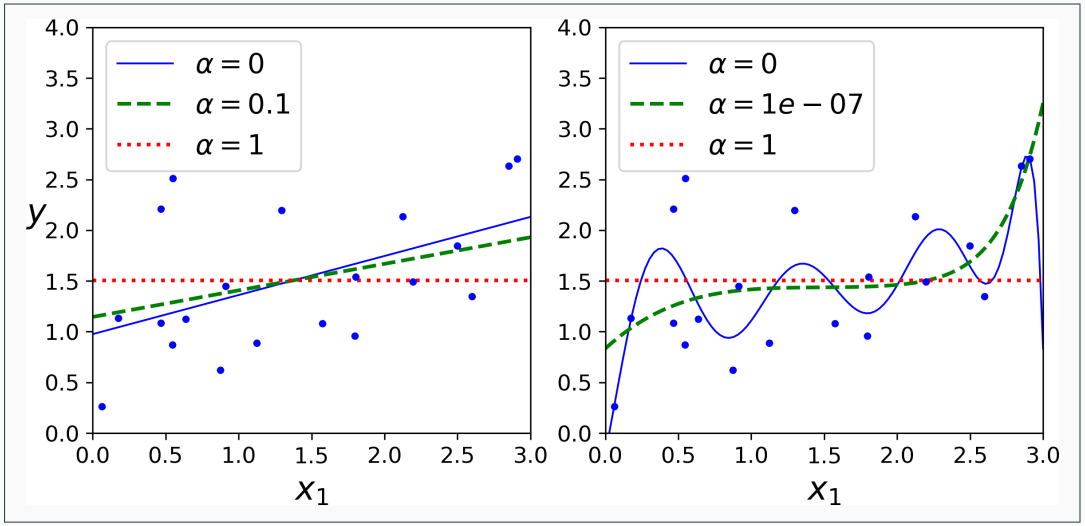


Figure 4-18. Lasso Regression

Contents

1. Linear regression

1.1 Linear regression

1.2 Gradient descent

1.3 Polynomial regression

1.4 Regularized linear model

2. Gaussian process

2.1 Bayesian method

2.2 Multivariate Gaussians

2.3 Bayesian linear regression

2.4 Gaussian process

2.5 Appendix

Bayesian method

- Unlike classical learning algorithm, Bayesian algorithms do not attempt to identify “best-fit” models of the data or similarly, make “best guess” predictions for new test inputs.
- Instead, they compute a **posterior distribution** over models, or similarly, compute **posterior predictive distributions** for new test inputs.
- These distributions provide a useful way to quantify our uncertainty in model estimates, and to exploit our knowledge of this uncertainty in order to make more robust predictions on new test points.

Regression problem

We focus on regression problems, where the goal is to learn a mapping from some input space $\mathcal{X} = \mathcal{R}^d$ of d-dimensional vectors to an output space $\mathcal{Y} = \mathcal{R}$ of real-valued targets. In particular, we will talk about a kernel-based fully Bayesian regression algorithm, known as Gaussian process regression.

Contents

1. Linear regression

1.1 Linear regression

1.2 Gradient descent

1.3 Polynomial regression

1.4 Regularized linear model

2. Gaussian process

2.1 Bayesian method

2.2 Multivariate Gaussians

2.3 Bayesian linear regression

2.4 Gaussian process

2.5 Appendix

Multivariate Gaussians

A vector-valued random variable $x \in \mathcal{R}^d$ is said to have a **multivariate normal (or Gaussian) distribution** with mean $\mu \in \mathcal{R}^d$ and covariance matrix $\Sigma \in S_{++}^d$ if

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right). \tag{44}$$

We write this as $x \sim \mathcal{N}(\mu, \Sigma)$. Here, recall from linear algebra that S_{++}^d refers to the space of symmetric positive definite $n \times d$ matrices.

Multivariate Gaussians

Consider a random vector $x \in \mathcal{R}^d$ with $x \sim \mathcal{N}(\mu, \Sigma)$. Suppose also that the variables in x have been partitioned into two sets $x_A = \begin{bmatrix} x_1 & \cdots & x_r \end{bmatrix}^T \in \mathcal{R}^r$ and $x_B = \begin{bmatrix} x_{r+1} & \cdots & x_d \end{bmatrix}^T \in \mathcal{R}^{d-r}$ (and similarly for μ and Σ), such that

$$x = \begin{bmatrix} x_A \\ x_B \end{bmatrix} \qquad \mu = \begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix}. \tag{45}$$

The following properties hold:

Multivariate Gaussians: properties

1. **Normalization**: The density function normalizes, i.e.,

$$\int_x p(x; \mu, \Sigma) dx = 1. \tag{46}$$

2. **Marginalization**: The marginal densities,

$$p(x_A) = \int_{x_B} p(x_A, x_B; \mu, \Sigma) dx_B \tag{47}$$

$$p(x_B) = \int_{x_A} p(x_A, x_B; \mu, \Sigma) dx_A \tag{48}$$

   are Gaussian:

$$x_A \sim \mathcal{N}(\mu_A, \Sigma_{AA}) \tag{49}$$

$$x_B \sim \mathcal{N}(\mu_B, \Sigma_{BB}) \tag{50}$$

Multivariate Gaussians: properties

3. **Conditioning**: The conditional densities

$$p(x_A \mid x_B) = \frac{p(x_A, x_B; \mu, \Sigma)}{\int_{x_A} p(x_A, x_B; \mu, \Sigma) dx_A} \tag{51}$$

$$p(x_B \mid x_A) = \frac{p(x_A, x_B; \mu, \Sigma)}{\int_{x_B} p(x_A, x_B; \mu, \Sigma) dx_B} \tag{52}$$

are also Gaussian:

$$x_A \mid x_B \sim \mathcal{N}(\mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B), \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}) \tag{53}$$

$$x_B \mid x_A \sim \mathcal{N}(\mu_B + \Sigma_{BA}\Sigma_{AA}^{-1}(x_A - \mu_A), \Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB}) \tag{54}$$

4. **Summation**: The sum of independent Gaussian random variables (with the same dimensionality), $y \sim \mathcal{N}(\mu, \Sigma)$ and $z \sim \mathcal{N}(\mu', \Sigma')$, is also Gaussian:

$$y + z \sim \mathcal{N}(\mu + \mu', \Sigma + \Sigma') \tag{55}$$

Contents

1. Linear regression

1.1 Linear regression

1.2 Gradient descent

1.3 Polynomial regression

1.4 Regularized linear model

2. Gaussian process

2.1 Bayesian method

2.2 Multivariate Gaussians

2.3 Bayesian linear regression

2.4 Gaussian process

2.5 Appendix

Bayesian linear regression

Let $\mathcal{S} = \{(x^{(i)}, y^{(i)}\}_{i=1}^{n}$ be a training set of i.i.d. examples from some unknown distribution. The standard probabilistic interpretation of linear regression states that

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}, \quad i = 1, ..., n \tag{56}$$

where $\epsilon^{(i)}$ are i.i.d. white noise variables with independent $\mathcal{N}(0, \sigma^2)$ distributions. It follows that $y^{(i)} - \theta^T x^{(i)} \sim \mathcal{N}(0, \sigma^2)$, or equivalently,

$$P(y^{(i)} \mid x^{(i)}, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right). \tag{57}$$

For notational convenience, we define

$$X = \begin{bmatrix} — \ (x^{(1)})^T \ — \\ — \ (x^{(2)})^T \ — \\ \vdots \\ — \ (x^{(n)})^T \ — \end{bmatrix} \in \mathcal{R}^{n \times d} \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} \in \mathcal{R}^n \quad \vec{\epsilon} = \begin{bmatrix} \epsilon^{(1)} \\ \epsilon^{(2)} \\ \vdots \\ \epsilon^{(n)} \end{bmatrix} \in \mathcal{R}^n \tag{58}$$

Bayesian linear regression

In Bayesian linear regression, we assume that a **prior distribution** over parameters is also given; a typical choice, for instance, is $\theta \sim \mathcal{N}(0, \tau^2 I)$. Using Bayes' rule, we obtain the **parameter posterior**,

$$p(\theta \mid \mathcal{S}) = \frac{p(\theta)p(\mathcal{S} \mid \theta)}{\int_{\theta'} p(\theta')p(\mathcal{S} \mid \theta')d\theta'} = \frac{p(\theta)\prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}, \theta)}{\int_{\theta'} p(\theta')\prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}, \theta')d\theta'}. \tag{59}$$

Assuming the same noise model on testing points as on our training points, the "output" of Bayesian linear regression on a new test point x_* is not just a single guess "y_*", but rather an entire probability distribution over possible outputs, known as the **posterior predictive distribution**:

$$p(y_* \mid x_*, \mathcal{S}) = \int_{\theta} p(y_* \mid x_*, \theta)p(\theta \mid \mathcal{S})d\theta. \tag{60}$$

For many types of models, the integrals are difficult to compute, and hence, we often resort to approximations, such as MAP estimation.

Bayesian linear regression

In the case of Bayesian linear regression, however, the integrals actually are *tractable*! In particular, for Bayesian linear regression, one can show that

$$\theta \mid \mathcal{S} \sim \mathcal{N}\left(\frac{1}{\sigma^2} A^{-1} X^T \vec{y}, A^{-1}\right) \tag{61}$$

$$y_* \mid x_*, \mathcal{S} \sim \mathcal{N}\left(\frac{1}{\sigma^2} x_*^T A^{-1} X^T \vec{y}, x_*^T A^{-1} x_* + \sigma^2\right) \tag{62}$$

where $A = \frac{1}{\sigma^2} X^T X + \frac{1}{\tau^2} I$.

Bayesian linear regression

The derivation of these formulas is somewhat involved. Nonetheless, from these equations, we get at least a flavor of what Bayesian methods are all about: the posterior distribution over the test output y_* for a test input x_* is a Gaussian distribution–this distribution reflects the uncertainty in our predictions $y_* = \theta^T x_* + \epsilon_*$ arising from both the randomness in ϵ_* and the uncertainty in our choice of parameters θ. In contrast, classical probabilistic linear regression models estimate parameters θ directly from the training data but provide no estimate of how reliable these learned parameters may be (see Figure 1).

Bayesian linear regression

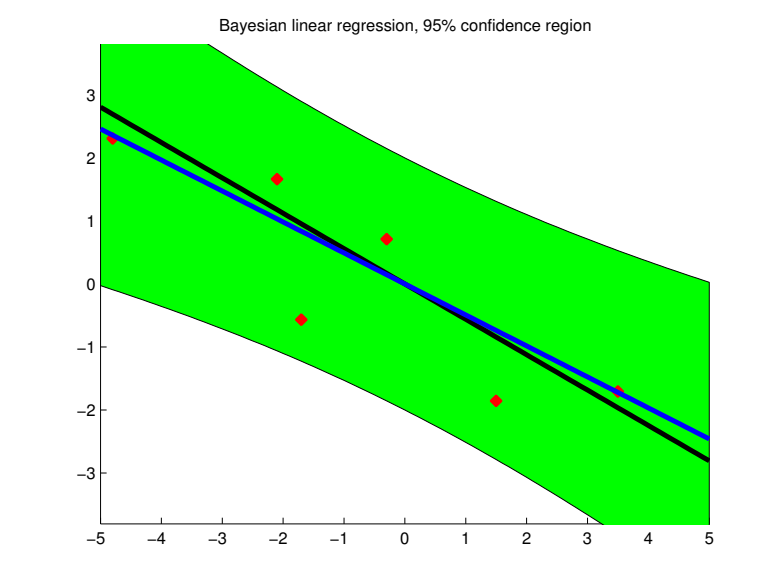


Figure 1: Bayesian linear regression for a one-dimensional linear regression problem, $y^{(i)} = \theta x^{(i)} + \epsilon^{(i)}$, with $\epsilon^{(i)} \sim \mathcal{N}(0, 1)$ i.i.d. noise. The green region denotes the 95% confidence region for predictions of the model. Note that the (vertical) width of the green region is largest at the ends but narrowest in the middle. This region reflects the uncertain in the estimates for the parameter θ. In contrast, a classical linear regression model would display a confidence region of constant width, reflecting only the $\mathcal{N}(0, \sigma^2)$ noise in the outputs.

Contents

1. Linear regression

1.1 Linear regression

1.2 Gradient descent

1.3 Polynomial regression

1.4 Regularized linear model

2. Gaussian process

2.1 Bayesian method

2.2 Multivariate Gaussians

2.3 Bayesian linear regression

2.4 Gaussian process

2.5 Appendix

Gaussian process

Multivariate Gaussian distributions are useful for modeling finite collections of real-valued variables because of their nice analytical properties. Gaussian processes are the extension of multivariate Gaussians to infinite-sized collections of real-valued variables. In particular, this extension will allow us to think of **Gaussian processes** as distributions not just over random vectors but in fact distributions over **random functions**.

Probability distributions over function with finite domains

To understand how one might parameterize probability distribution over functions, consider the following simple example. Let $\mathcal{X} = \{x_1, ..., x_n\}$ be any finite set of elements. Now, consider the set $\mathcal{H}$ of all possible functions mapping from $\mathcal{X}$ to $\mathcal{R}$. For instance, one example of a function $f_0(\cdot) \in \mathcal{H}$ is given by

$$f_0(x_1) = 5, \quad f_0(x_2) = 2.3, \quad f_0(x_3) = -7, \tag{63}$$

$$..., \quad f_0(x_{n-1}) = -\pi, \quad f_0(x_n) = 8. \tag{64}$$

Since the domain of any $f(\cdot) \in \mathcal{H}$ has only n elements, we can always represent $f(\cdot)$ compactly as an n-dimensional vector, $\vec{f} = [\ f(x_1) \quad f(x_2) \quad \cdots \quad f(x_n)\ ]^T$. In order to specify a probability distribution over functions $f(\cdot) \in \mathcal{H}$, we must associate some "probability density" with each function in $\mathcal{H}$. One natural way to do this is to exploit the one-to-one correspondence between functions $f(\cdot) \in \mathcal{H}$ and their vector representations, $\vec{f}$.

Probability distributions over function with finite domains

In particular, if we specify that $\vec{f} \sim \mathcal{N}(\vec{\mu}, \Sigma^2 I)$, then this in turn implies a probability distribution over functions $f(\cdot)$, whose probability density function is given by

$$p(\vec{f}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(f(x_i) - \mu_i)^2\right). \tag{65}$$

In the example above, we showed that probability distributions over functions with finite domains can be represented using a finite-dimensional multivariate Gaussian distribution over function outputs $f(x_1), ..., f(x_n)$ at a finite number of input points $x_1, ..., x_n$. How can we specify probability distributions over functions when the domain size may be infinite? For this, we turn to a fancier type of probability distribution known as a Gaussian process.

Probability distributions over function with infinite domains

A stochastic process is a collection of random variables, $\{f(x) : x \in \mathcal{X}\}$, indexed by elements from some set $\mathcal{X}$, known as the index set. A **Gaussian process** is a stochastic process such that any finite subcollection of random variables has a multivariate Gaussian distribution.

In particular, a collection of random variables $\{f(x) : x \in \mathcal{X}\}$ is said to be drawn from Gausian process with **mean function** $m(\cdot)$ and **covariance function** $k(\cdot, \cdot)$ if for any finite set of elements $x_1, ..., x_n \in \mathcal{X}$, the associated finite set of random variables $f(x_1), ..., f(x_n)$ have distribution,

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(x_1) \\ \vdots \\ m(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}\right). \quad (66)$$

Probability distributions over function with infinite domains

We denote this using the notation,

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)). \tag{67}$$

Observe that the mean function and covariance function are aptly named since the above properties imply that

$$m(x) = \mathbb{E}[f(x)] \tag{68}$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \tag{69}$$

for any $x, x' \in \mathcal{X}$.

Probability distributions over function with infinite domains

Intuitively, one can think of a function $f(\cdot)$ drawn from a Gaussian process prior as an extremely high-dimensional vector drawn from an extremely high-dimensional multivariate Gaussian. Here, each dimension of the Gaussian corresponds to an element x from the index set $\mathcal{X}$, and the corresponding component of the random vector represents the value of $f(x)$. Using the marginalization property for multivariate Gaussians, we can obtain the marginal multivariate Gaussian density corresponding to any finite subcollection of variables.

Probability distributions over function with infinite domains

What sort of functions $m(\cdot)$ and $k(\cdot, \cdot)$ give rist to valid Gaussian processes? In general, any real-valued function $m(\cdot)$ is acceptable, but for $k(\cdot, \cdot)$, it must be the case that for any set of elements $x_1, ..., x_n \in \mathcal{X}$, the resulting matrix

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \tag{70}$$

is a valid covariance matrix corresponding to some multivariate Gaussian distribution. A standard result in probability theory states that this is true provided that K is positive semidefinite. Sound familiar?

Probability distributions over function with infinite domains

The positive semidefiniteness requirement for covariance matrices computed based on arbitrary input points is, in fact, identical to Mercer's condition for kernels! A function $k(\cdot, \cdot)$ is a valid kernel provided the resulting kernel matrix K defined as above is always positive semidefinite for any set of input points $x_1, ..., x_n \in \mathcal{X}$. Gaussian processes, therefore, are kernel-based probability distributions in the sense that any valid kernel function can be used as a covariance function!

The squared exponential kernel

In order to get an intuition for how Gaussian processes work, consider a simple zero-mean Gaussian process,

$$f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot)) \tag{71}$$

defined for functions $h : \mathcal{X} \longrightarrow \mathcal{R}$ where we take $\mathcal{X} = \mathcal{R}$. Here, we choose the kernel function $k(\cdot, \cdot)$ to be the **squared exponential** kernel function, defined as

$$k_{\mathrm{SE}}(x, x') = \exp\left(-\frac{1}{2\tau^2}\|x - x'\|^2\right) \tag{72}$$

for some $\tau > 0$. What do random functions sampled from this Gaussian process look like?

The squared exponential kernel

In our example, since we use a zero-mean Gaussian process, we would expect that for the function values from our Gaussian process will tend to be distributed around zero. Furthermore, for any pair of elements, $x, x' \in \mathcal{X}$.

- $f(x)$ and $f(x')$ will tend to have high covariance x and x' are "nearby" in the input space, i.e., $\|x - x'\| = |x - x'| \approx 0$, so $\exp\left(-\frac{1}{2\tau^2}\|x - x'\|^2\right) \approx 1$.
- $f(x)$ and $f(x')$ will tend to have low covariance when x and x' are "far apart", i.e., $\|x - x'\| >> 0$, so $\exp\left(-\frac{1}{2\tau^2}\|x - x'\|^2\right) \approx 0$.

More simply stated, functions drawn from a zero-mean Gaussian process prior with the squared exponential kernel will tend to be "locally smooth" with high probability; i.e., nearby function values are highly correlated, and the correlation drops off as a function of distance in the input space (see Figure 2).

The squared exponential kernel

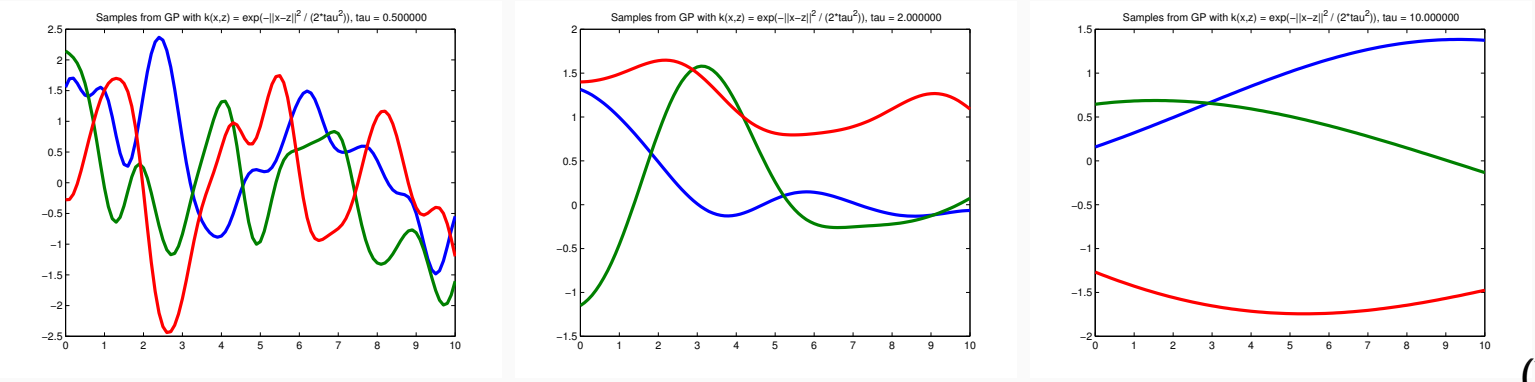


Figure 2: Samples from a zero-mean Gaussian process prior with $k_{SE}(\cdot, \cdot)$ covariance function, using (a) $\tau = 0.5$, (b) $\tau = 2$, and (c) $\tau = 10$. Note that as the bandwidth parameter τ increases, then points which are farther away will have higher correlations than before, and hence the sampled functions tend to be smoother overall.

Contents

1. Linear regression

1.1 Linear regression

1.2 Gradient descent

1.3 Polynomial regression

1.4 Regularized linear model

2. Gaussian process

2.1 Bayesian method

2.2 Multivariate Gaussians

2.3 Bayesian linear regression

2.4 Gaussian process

2.5 Appendix

Reference and further reading

- “Chap 6.4 | Gaussian Processes” of C. Bishop, Pattern Recognition and Machine Learning
- “Gaussian Processes” of Chuong B. Do (updated by Honglak Lee)
- “Chap 4 | Training Models” of A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow