Machine learning 02

Byung Chang Chung

Gyeongsang National University

bcchung@gnu.ac.kr

Contents

- Model training
- Support vector machine

Linear regression

- Linearity
  - graphically represented as a straight line
  - closely related to proportionality

  - additivity: $f(x+y) = f(x) + f(y)$
  - homogeneity: $f(\alpha x) = \alpha f(x), \forall \alpha$

Gyeongsang National University 경상국립대학교

Linear regression

- Regression
  - statistical technique for estimating the relationships among variables

- What is linear regression?
  - regression with linear model

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

GNU
Gyeongsang National University
경상국립대학교

Linear regression

- Form of linear regression

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

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

- Objective
  - Find θ which minimize the MSE

Linear regression

- Objective
  - Find θ which minimize the MSE
- Solution of linear regression
  - From linear algebra, normal equation minimizes MSE

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

GNU
Gyeongsang
National
University
경상국립대학교

Linear regression

- Proof of solution (“정규방정식” in Korean)
  - properties of transpose matrix
  - matrix differentiation

https://en.wikipedia.org/wiki/Proofs_involving_ordinary_least_squares#Derivation_of_the_normal_equations

Linear regression

- Proof of solution
  - $\frac{1}{m}\sum_{i=1}^{m}\left(\theta^T x^{(i)} - y^{(i)}\right)^2$
  - $= \frac{1}{m}\left(\theta^T x^{(i)} - y^{(i)}\right)^T\left(\theta^T x^{(i)} - y^{(i)}\right)$
  - $= \frac{1}{m}\left((\theta^T x^{(i)})^T - (y^{(i)})^T\right)\left(\theta^T x^{(i)} - y^{(i)}\right)$
  - $= \frac{1}{m}(\theta^T x^{(i)})^T\theta^T x^{(i)} - (\theta^T x^{(i)})^T y^{(i)} - (y^{(i)})^T\theta^T x^{(i)} + (y^{(i)})^T y^{(i)}$

Gyeongsang National University 경상국립대학교

Linear regression

- Proof of solution
  - $= \frac{1}{m}(\theta^T x^{(i)})^T \theta^T x^{(i)} - (\theta^T x^{(i)})^T y^{(i)} - (y^{(i)})^T \theta^T x^{(i)} + (y^{(i)})^T y^{(i)}$
  - $= \frac{1}{m}\left(x^{(i)}\right)^T \theta\theta^T x^{(i)} - 2(\theta^T x^{(i)})^T y^{(i)} + (y^{(i)})^T y^{(i)}$
  - $= \frac{1}{m}\left(x^{(i)}\right)^T x^{(i)}(\theta^T)^2 - 2x^{(i)^T} y^{(i)}\theta + (y^{(i)})^T y^{(i)}$

GNU Gyeongsang National University 경상국립대학교

Linear regression

- Proof of solution
  - $MSE(X, h_\theta) = \frac{1}{m}\left(x^{(i)}\right)^T x^{(i)}(\theta^T)^2 - 2x^{(i)^T} y^{(i)}\theta + (y^{(i)})^T y^{(i)}$
  - $\frac{dMSE(X,h_\theta)}{d\theta} = \frac{1}{m}\left(2\left(x^{(i)}\right)^T x^{(i)}\theta^T - 2\left(x^{(i)}\right)^T y^{(i)}\right) = 0$
  - $\left(2\left(x^{(i)}\right)^T x^{(i)}\theta^T - 2\left(x^{(i)}\right)^T y^{(i)}\right) = 0$

Linear regression

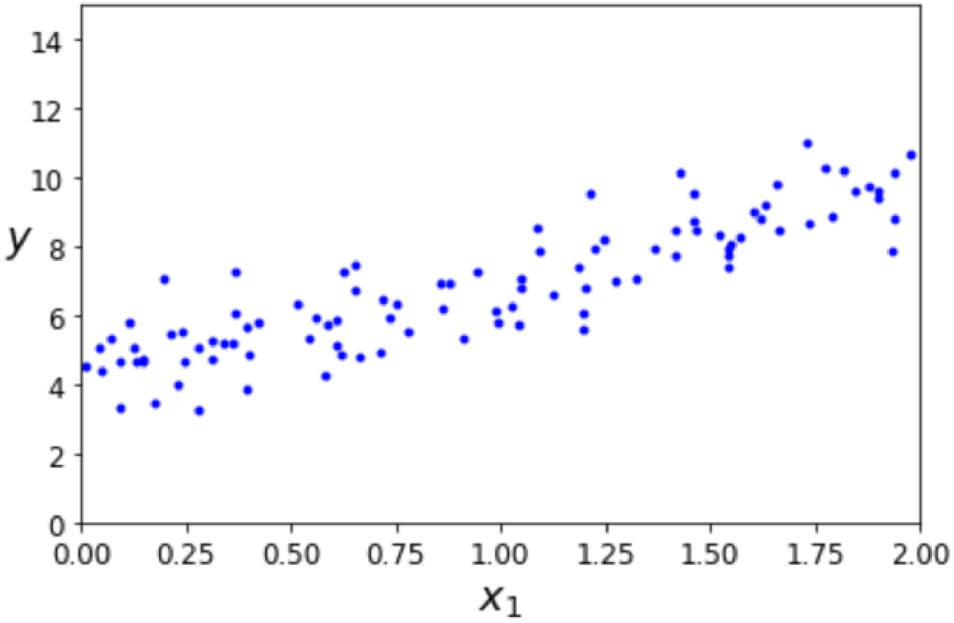
- Proof of solution
  - $2\left(x^{(i)}\right)^T x^{(i)} \theta^T = 2\left(x^{(i)}\right)^T y^{(i)}$
  - $\left(x^{(i)}\right)^T x^{(i)} \theta^T = \left(x^{(i)}\right)^T y^{(i)}$
  - $\theta^T = \left(\left(x^{(i)}\right)^T x^{(i)}\right)^{-1} \left(x^{(i)}\right)^T y^{(i)}$

Linear regression

- Test of proof using python

```
import numpy as np

X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)
```

```
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([0, 2, 0, 15])
save_fig("generated_data_plot")
plt.show()
```



Linear regression

- Test of proof using python

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

```
X_b = np.c_[np.ones((100, 1)), X]  # 모든 샘플에 x0 = 1을 추가합니다.
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

```
theta_best
```

```
array([[4.21509616],
       [2.77011339]])
```

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

```
X_new = np.array([[0], [2]])
X_new_b = np.c_[np.ones((2, 1)), X_new]  # 모든 샘플에 x0 = 1을 추가합니다.
y_predict = X_new_b.dot(theta_best)
y_predict
```

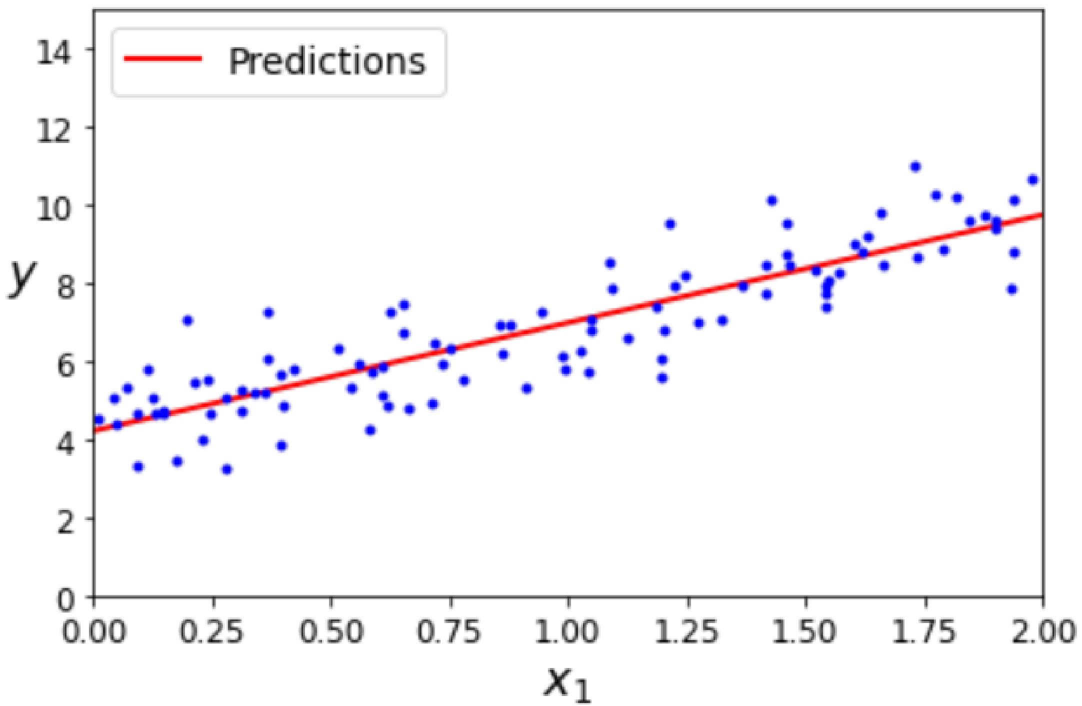
```
array([[4.21509616],
       [9.75532293]])
```

```
plt.plot(X_new, y_predict, "r-")
plt.plot(X, y, "b.")
plt.axis([0, 2, 0, 15])
plt.show()
```

Gyeongsang National University 경상국립대학교

Linear regression

- Test of proof using python



Gradient descent

- Gradient
  - In vector calculus, the gradient of a scalar-valued differentiable function f of several variables is the vector field (or vector-valued function) ∇f whose value at a point p is the vector whose components are the partial derivatives of f at p

Gyeongsang National University 경상국립대학교

Gradient descent

- Illustration of gradient descent

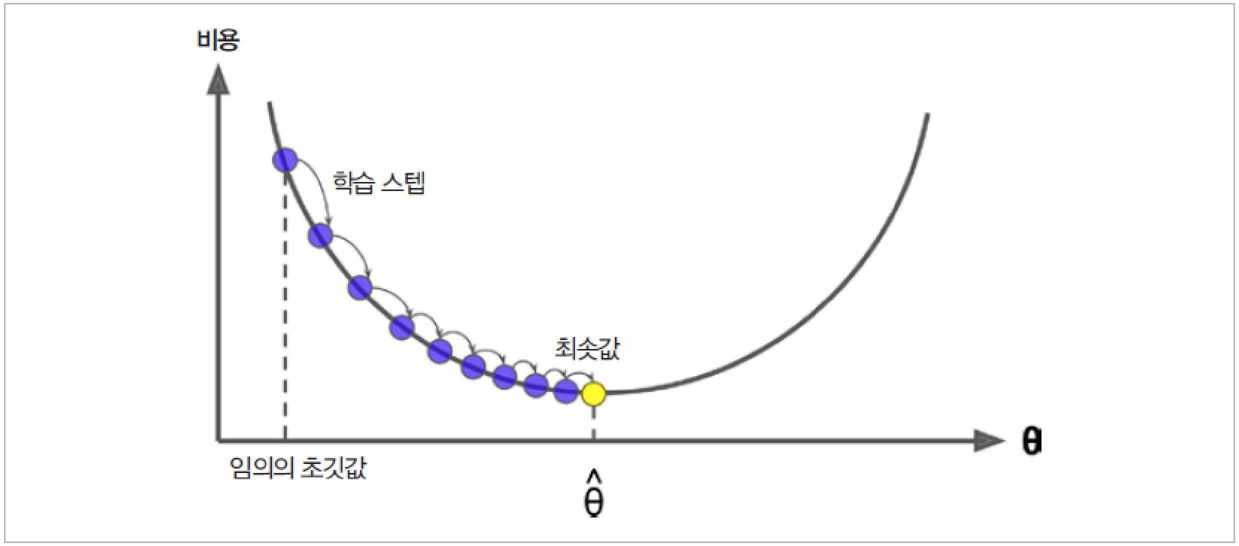
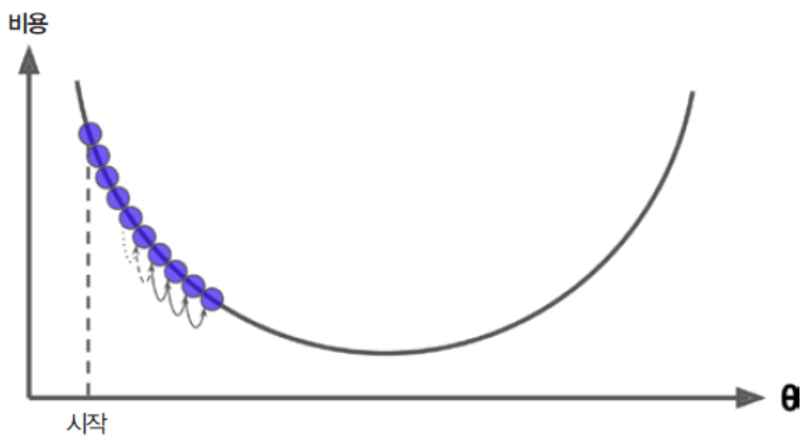


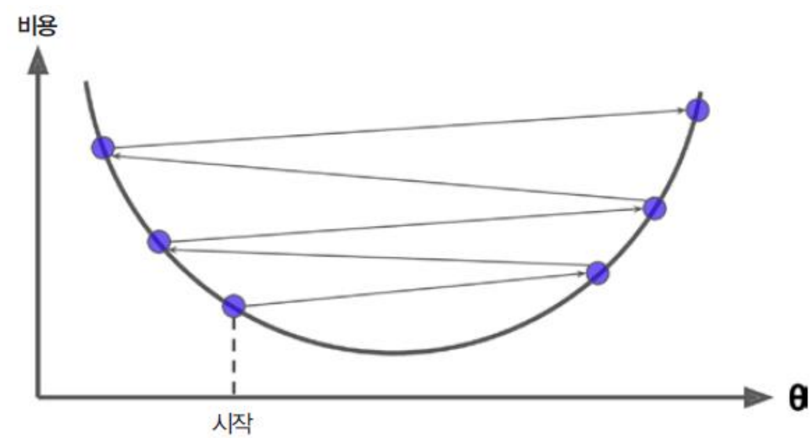
그림 4-3 이 경사 하강법 그림에서 모델 파라미터가 무작위하게 초기화된 후 반복적으로 수정되어 비용 함수를 최소화합니다. 학습 스텝 크기는 비용 함수의 기울기에 비례합니다. 따라서 파라미터가 최솟값에 가까워질수록 스텝 크기가 점진적으로 줄어듭니다.

Gyeongsang National University 경상국립대학교

Gradient descent

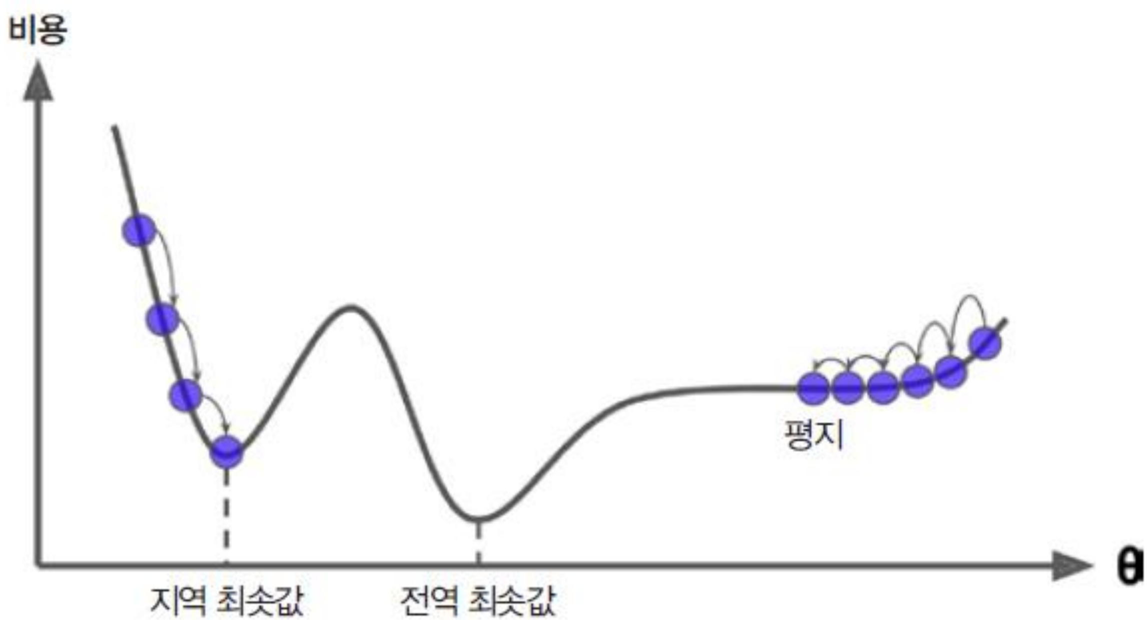
- Learning rate in gradient descent
  - when learning rate is relatively small
  - when learning rate is relatively large





Gradient descent

- Problem of gradient descent



Gyeongsang National University 경상국립대학교

Gradient descent

- In linear regression,
  - the form of MSE is convex
    - no local optima
    - unique global optimum
  - guarantees that gradient descent algorithm could converge to global optimum point

Gradient descent

- Batch gradient descent
  - partial derivative of MSE

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

  - generalization on matrix

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta}) = \frac{2}{m}\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

GNU Gyeongsang National University 경상국립대학교

Gradient descent

- Batch gradient descent
  - step algorithm

$$\boldsymbol{\theta}^{(\text{next step})} = \boldsymbol{\theta} - \eta \frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta})$$

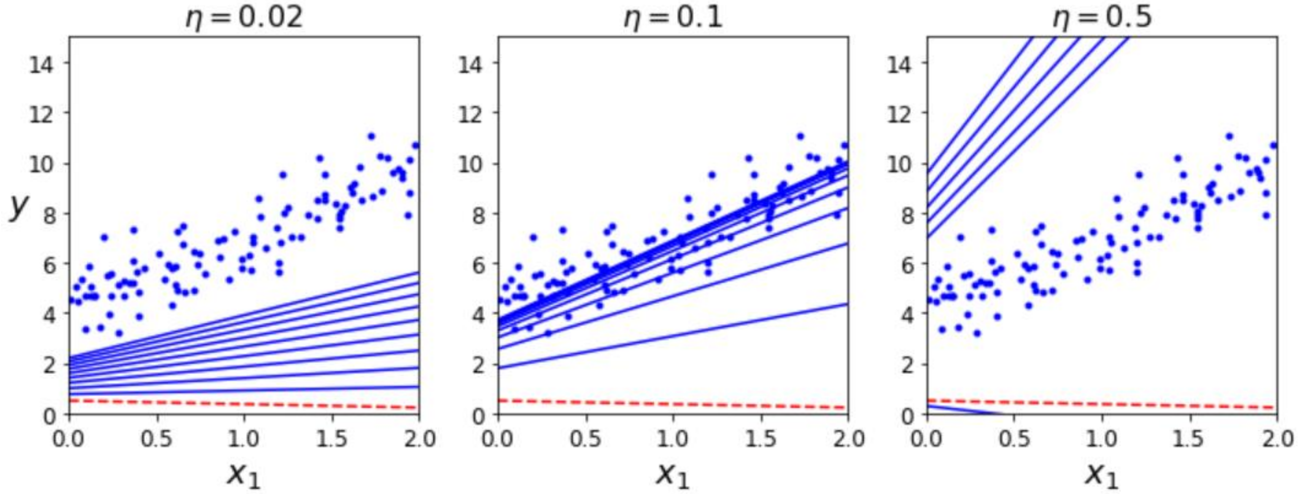
  - code-level implementation

```python
eta = 0.1  # 학습률
n_iterations = 1000
m = 100

theta = np.random.randn(2,1)  # 랜덤 초기화

for iteration in range(n_iterations):
    gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
    theta = theta - eta * gradients
```

Gyeongsang National University 경상국립대학교

Gradient descent

- Batch gradient descent
  - visualize example



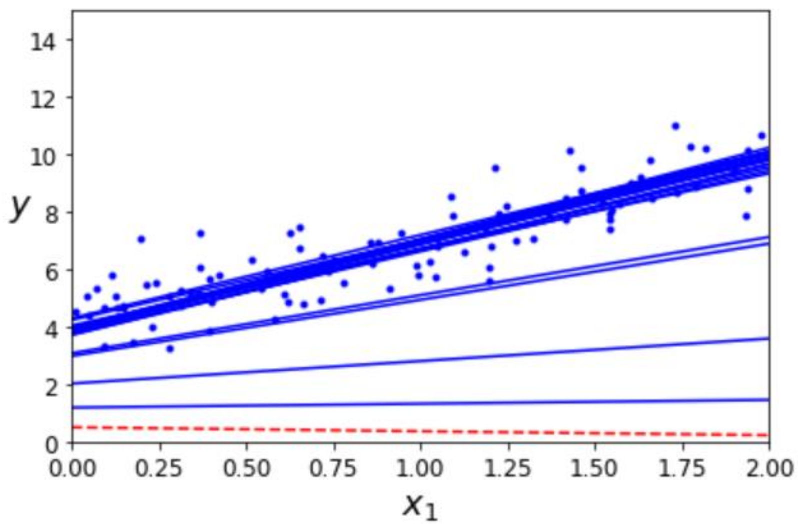
Gradient descent

- Stochastic gradient descent
  - iterative method for optimizing an objective function with suitable smoothness properties
  - regarded as a stochastic approximation of gradient descent optimization

Gyeongsang National University
경상국립대학교

Gradient descent

- Stochastic gradient descent
  - Implementation

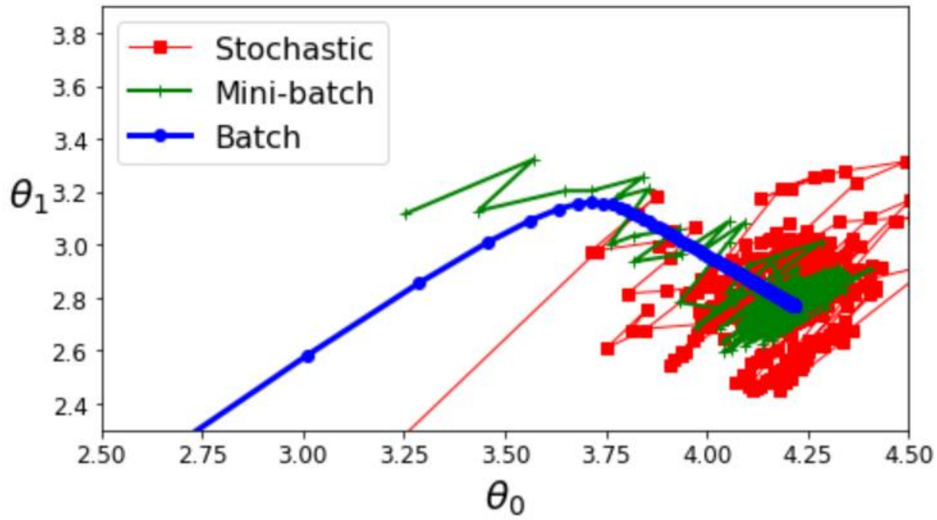



```
n_epochs = 50
t0, t1 = 5, 50  # 학습 스케줄 하이퍼파라미터

def learning_schedule(t):
    return t0 / (t + t1)

theta = np.random.randn(2,1)  # 랜덤 초기화

for epoch in range(n_epochs):
    for i in range(m):
        if epoch == 0 and i < 20:
            y_predict = X_new_b.dot(theta)
            style = "b-" if i > 0 else "r--"
            plt.plot(X_new, y_predict, style)
        random_index = np.random.randint(m)
        xi = X_b[random_index:random_index+1]
        yi = y[random_index:random_index+1]
        gradients = 2 * xi.T.dot(xi.dot(theta) - yi)
        eta = learning_schedule(epoch * m + i)
        theta = theta - eta * gradients
        theta_path_sgd.append(theta)

plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([0, 2, 0, 15])
save_fig("sgd_plot")
plt.show()
```

Gyeongsang National University 경상국립대학교

Gradient descent

- Minibatch gradient descent
  - Gradient descent using minibatch
  - Minibatch: small amount of data from batch



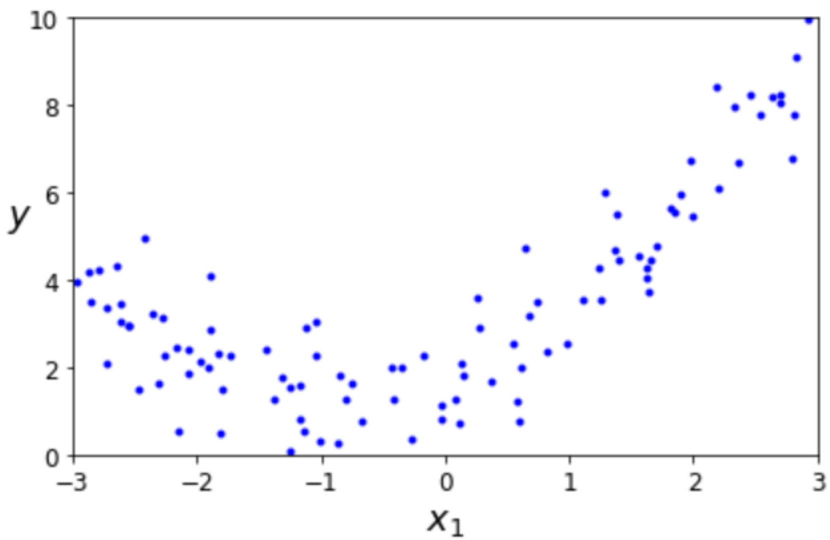
Polynomial regression

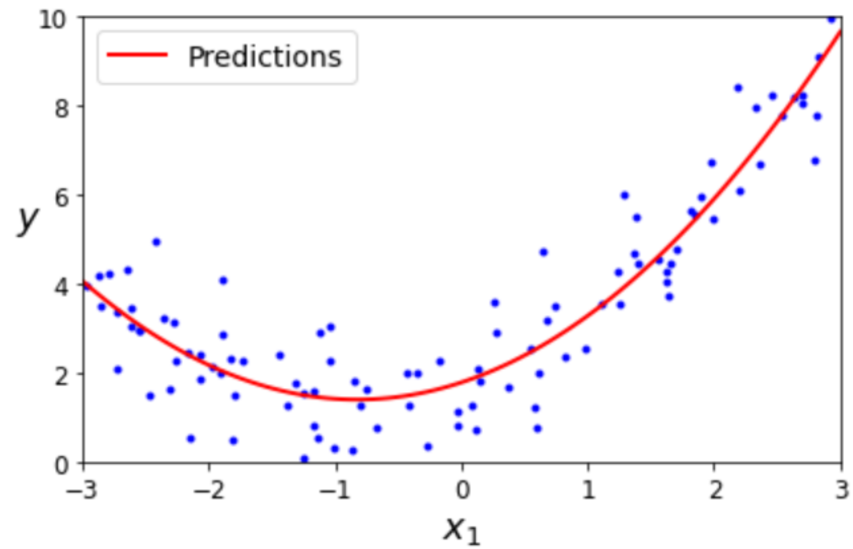
- Polynomial
  - an expression consisting of indeterminates (also called variables) and coefficients, that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponentiation of variables
    - $x^3 + 2x^2 - 3x + 4$
    - $5xy - 6x^2 + 7y - 8$

Gyeongsang National University
경상국립대학교

Polynomial regression

- Example of polynomial regression

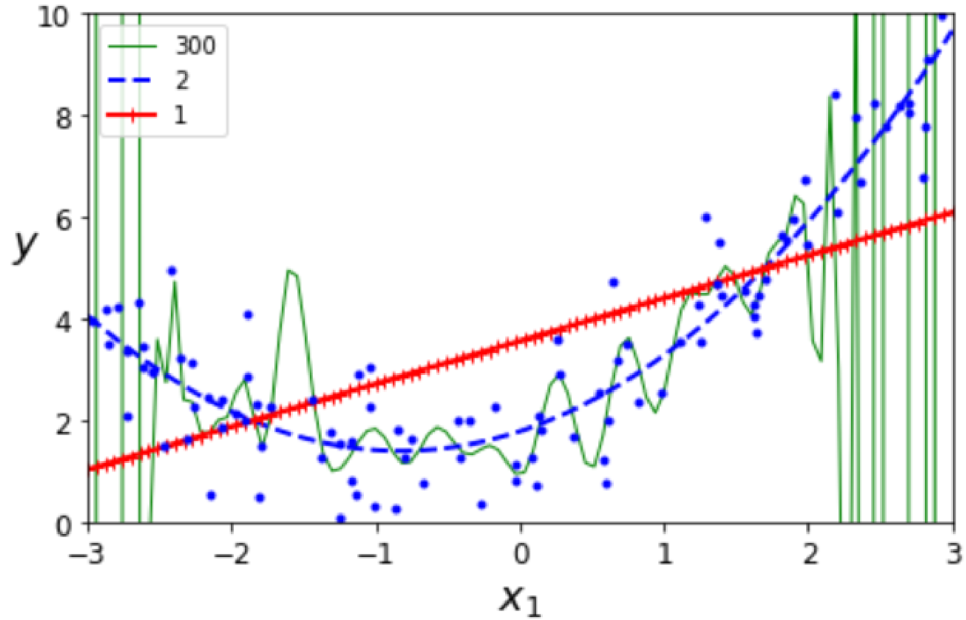




Gyeongsang National University 경상국립대학교

Learning curves

- Regression according to polynomial order
  - overfit or underfit

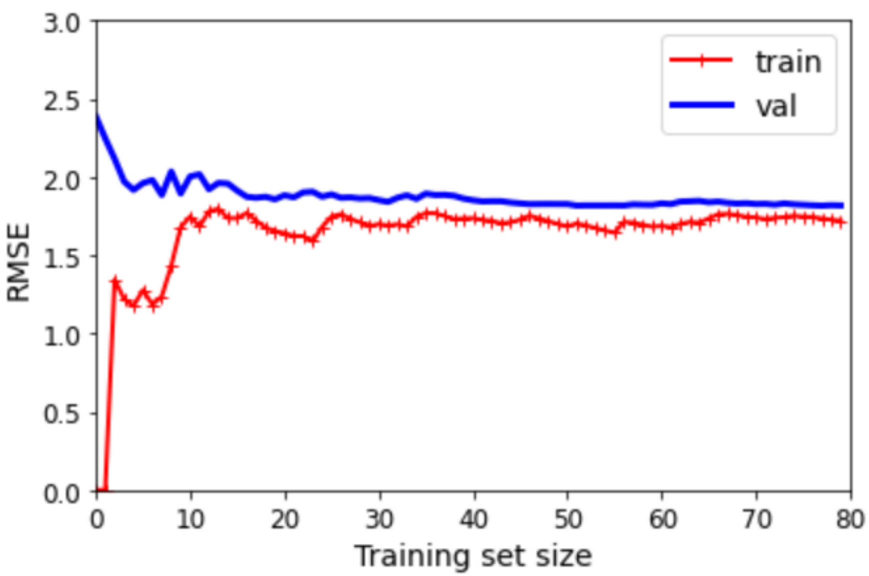


Learning curves

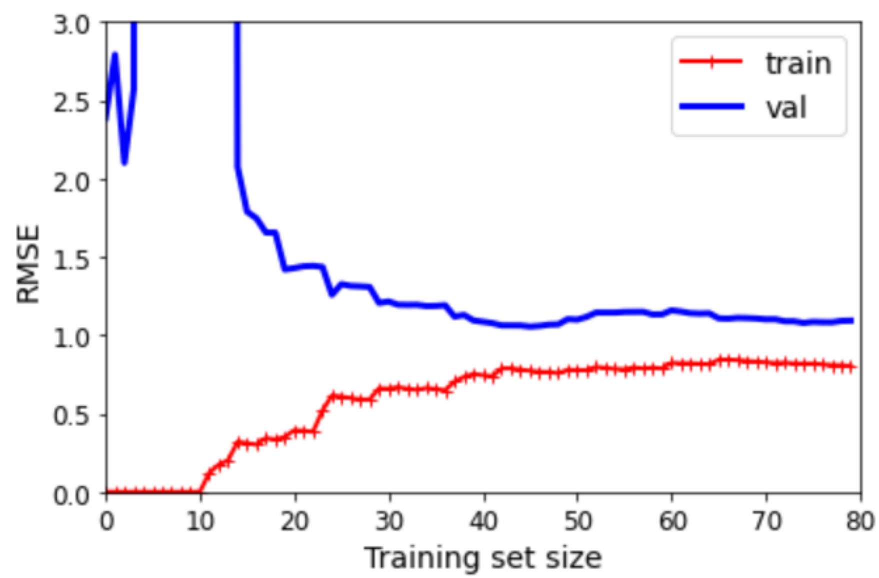
- Checking overfitting or underfitting
  - cross validation
  - learning curves
    - plotting the relation of RMSE and training set size
    - plot for 2 cases: training and test dataset
    - check the gap between training and test dataset

Learning curves

- Example of learning curve



linear regression



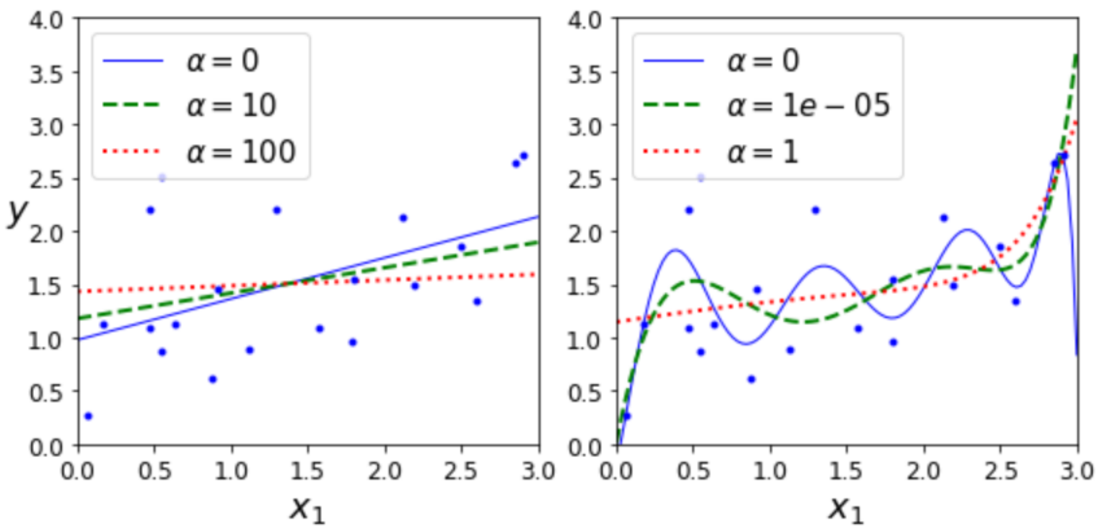
polynomial regression (degree=10)

Linear model with regulation

- Way of reducing overfitting
  - reduce the degree of polynomial equation
  - in linear model, cost function is changed
    - ridge regression
    - lasso regression
    - elastic net
  - early stopping

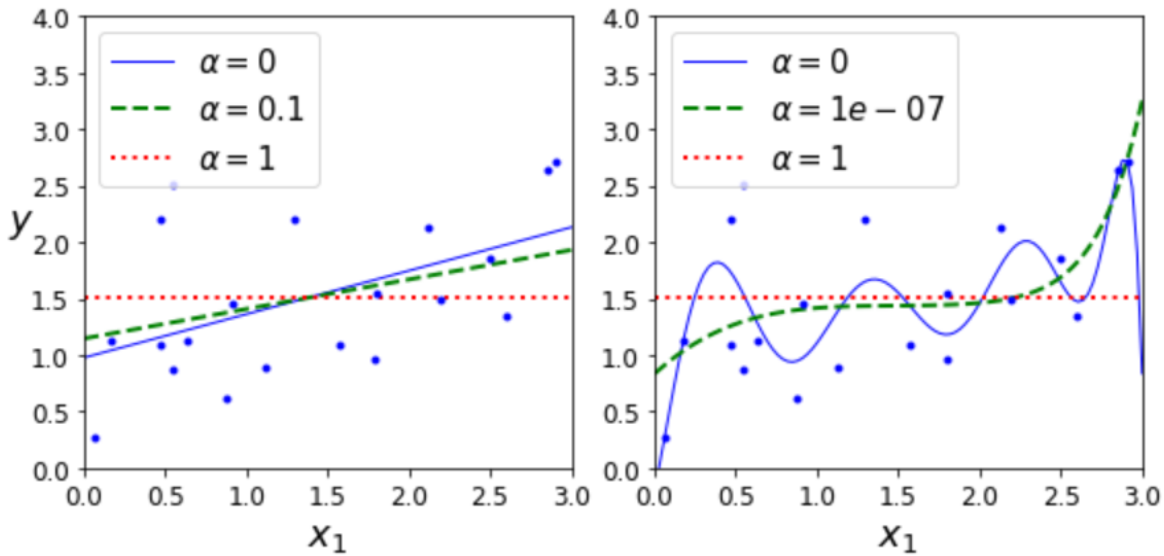
Linear model with regulation

- Ridge regression
  - cost function $$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \frac{1}{2} \sum_{i=1}^{n} {\theta_i}^2$$



Linear model with regulation

- Lasso regression
  - cost function $$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \sum_{i=1}^{n} |\theta_i|$$



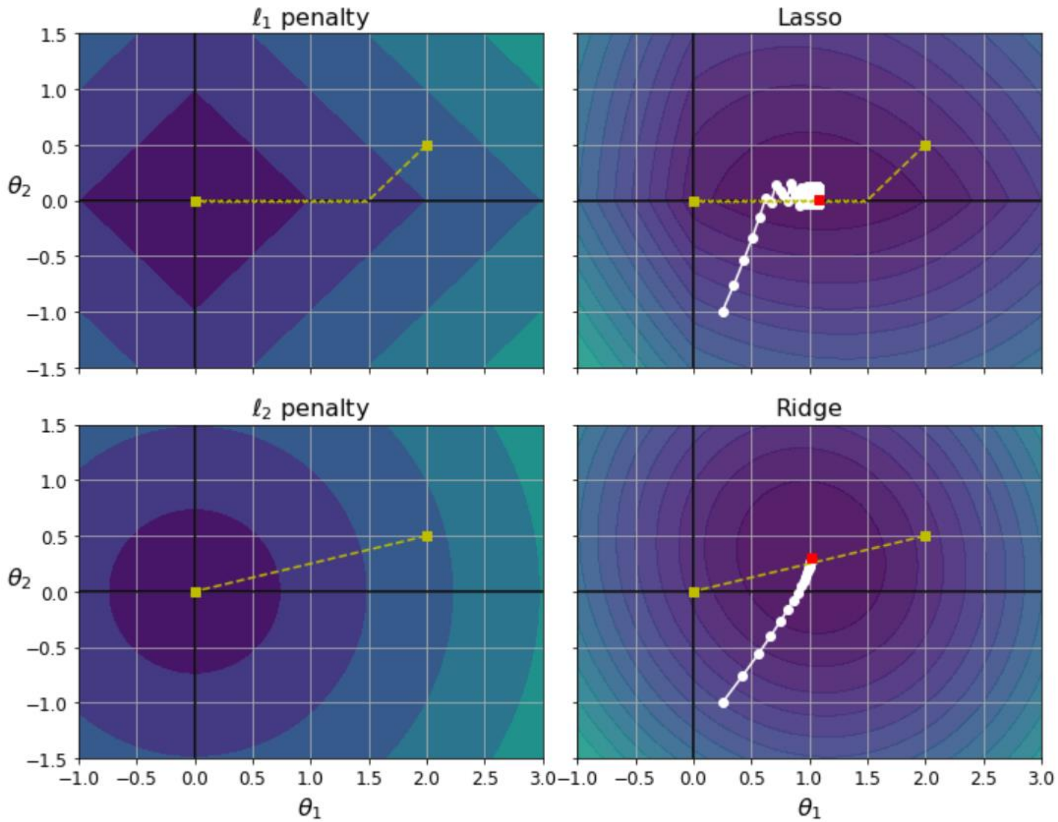
Linear model with regulation

- Elastic net
  - cost function $$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^{n} {\theta_i}^2$$
  - hybrid solution of ridge and lasso regression

Gyeongsang National University 경상국립대학교

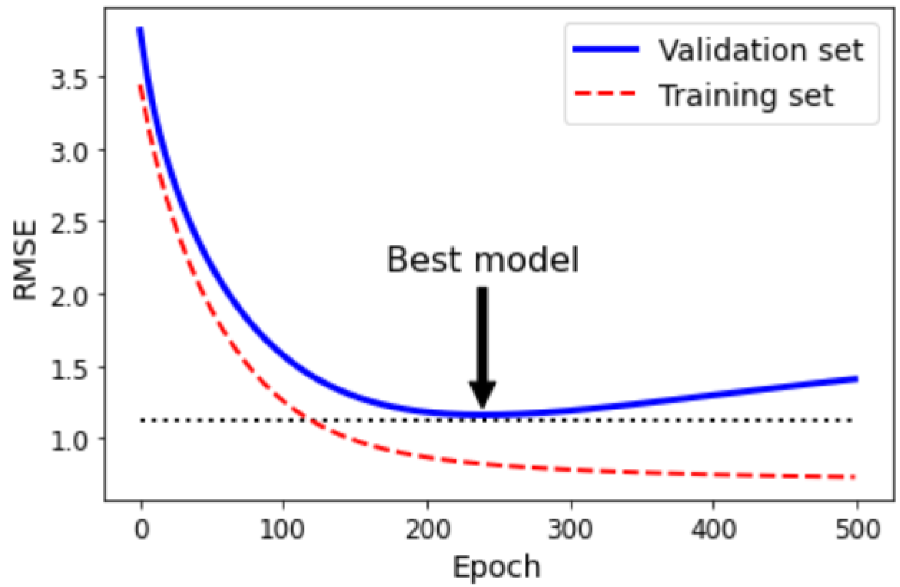
Linear model with regulation

- Ridge versus lasso regression



Linear model with regulation

- Early stopping
  - stop when validation error is minimized



Gyeongsang National University 경상국립대학교

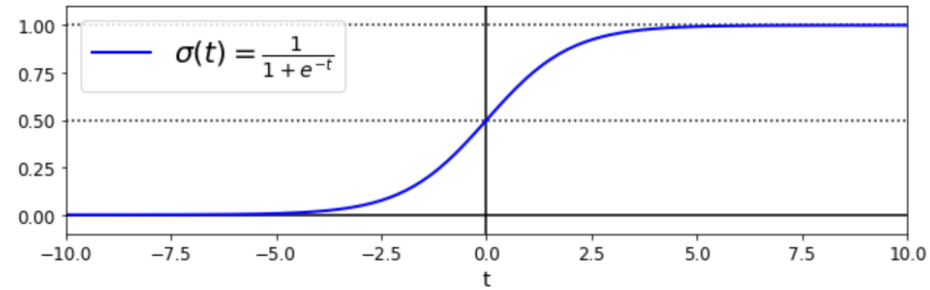
Logistic regression

- Regression for classification
  - expect probability
    - whether probability is greater than 0.5 or not
    - not immediately derive result value, but the logistic of value

$$\hat{p} = h_\theta(\mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

  - logistic
    - sigmoid function

$$\sigma(t) = \frac{1}{1+\exp(-t)}$$



Logistic regression

- Objective of logistic regression
  - to find a parameter vector θ
  - which expects high value for positive sample
  - and expects low value for negative sample

$$\hat{y} = \begin{cases} 0 \text{ when } \hat{p} < 0.5 \\ 1 \text{ when } \hat{p} \geq 0.5 \end{cases}$$

Gyeongsang National University 경상국립대학교

Logistic regression

- Cost function design

$$c(\boldsymbol{\theta}) = \begin{cases} -\log(\hat{p}) & y = 1\text{일 때} \\ -\log(1-\hat{p}) & y = 0\text{일 때} \end{cases}$$

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(\hat{p}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-\hat{p}^{(i)}\right)\right]$$

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

Logistic regression

- Implementation example

그림 4-22 세 종류의 붓꽃[40]

Logistic regression

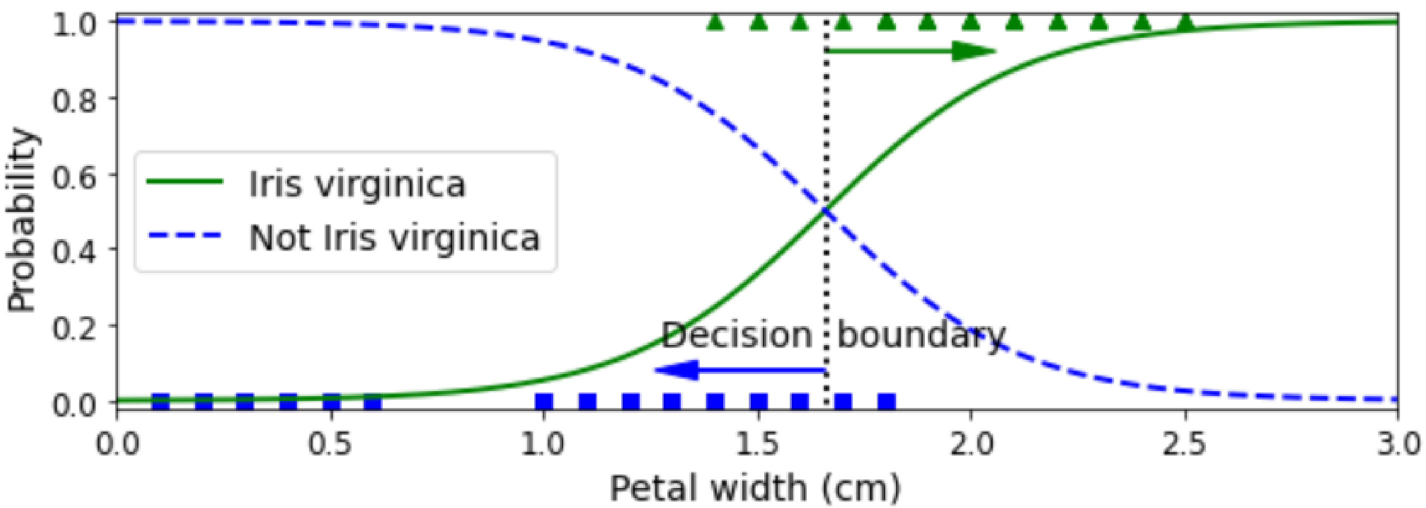
- Implementation example

```
Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================
```

Gyeongsang National University 경상국립대학교

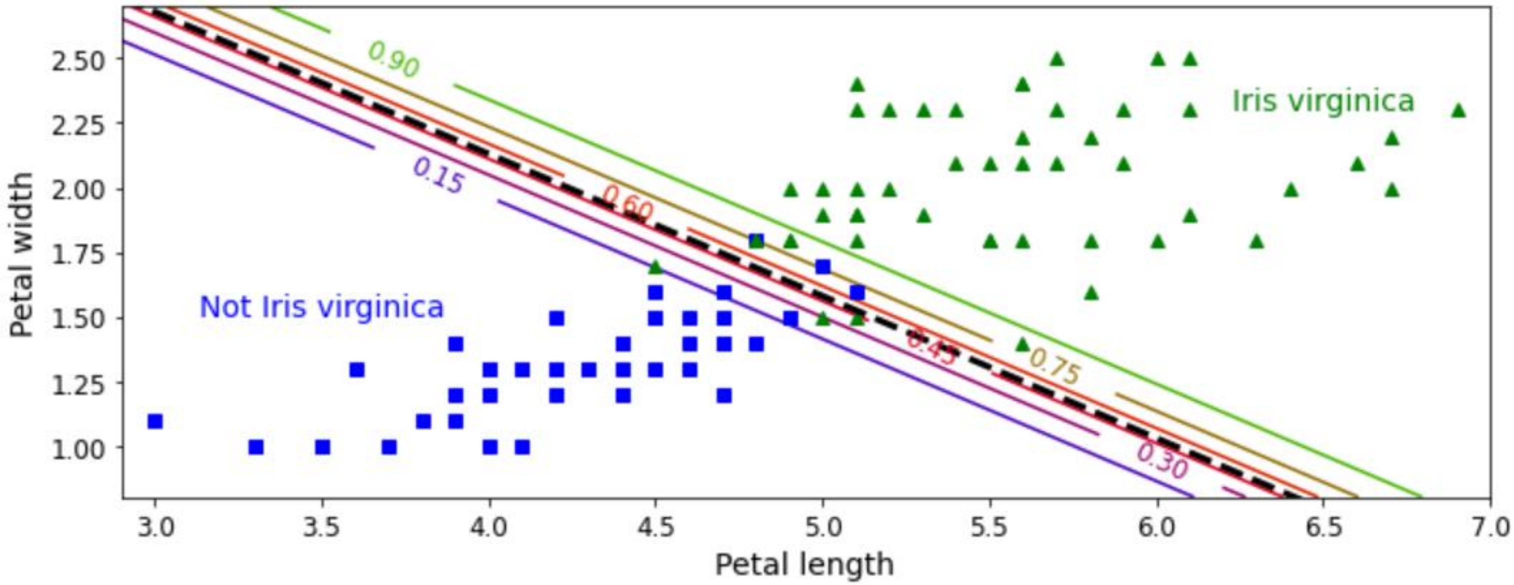
Logistic regression

- Implementation example



Logistic regression

- Implementation example



GNU Gyeongsang National University 경상국립대학교

Softmax regression

- Multinomial logistic regression
  - logistic regression for multiclass problem
  - softmax score for class k

$$s_k(\mathbf{x}) = \left(\boldsymbol{\theta}^{(k)}\right)^T \mathbf{x}$$

  - softmax function

$$\hat{p}_k = \sigma\left(\mathbf{s}(\mathbf{x})\right)_k = \frac{\exp\left(s_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(s_j(\mathbf{x})\right)}$$

GNU Gyeongsang National University 경상국립대학교

Softmax regression

- Multinomial logistic regression
  - class expectation

$$\hat{y} = \underset{k}{\operatorname{argmax}}\, \sigma\left(\mathbf{s}(\mathbf{x})\right)_k = \underset{k}{\operatorname{argmax}}\, s_k(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\left(\left(\boldsymbol{\theta}^{(k)}\right)^T \mathbf{x}\right)$$

  - cost function

$$J(\boldsymbol{\Theta}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$$

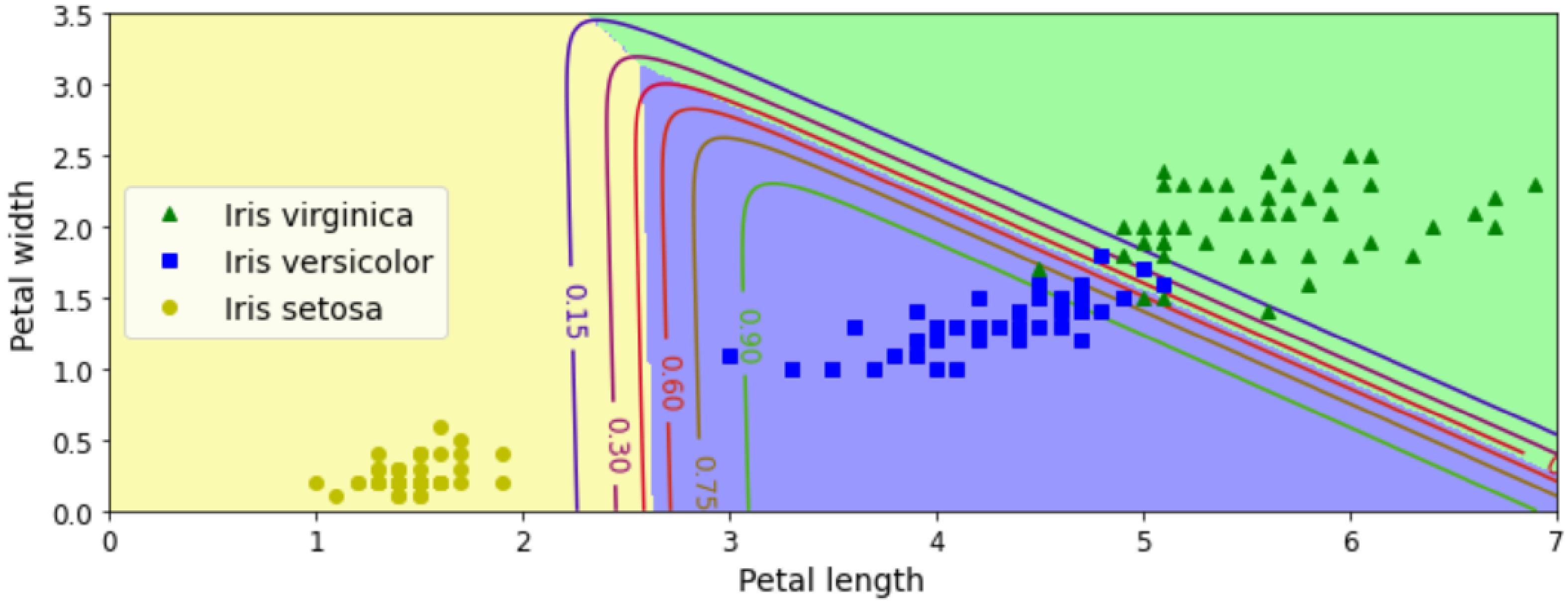
  - gradient vector

$$\nabla_{\boldsymbol{\theta}^{(k)}} J(\boldsymbol{\Theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{p}_k^{(i)} - y_k^{(i)}\right)\mathbf{x}^{(i)}$$

GNU Gyeongsang National University 경상국립대학교

Softmax regression

- Implementation example

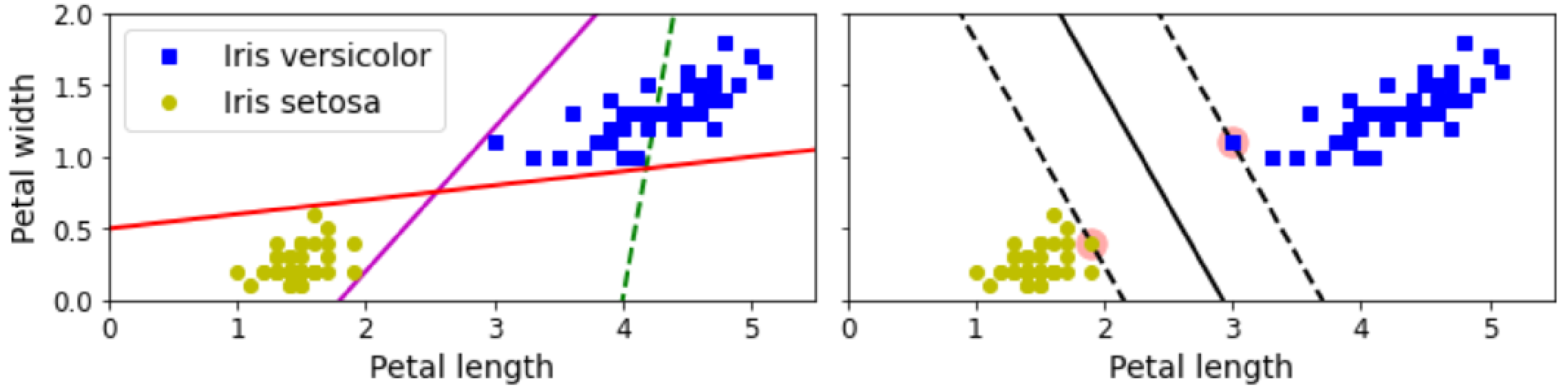


Support vector machine

- What is SVM?
  - supervised learning models with associated learning algorithms that analyze data for classification and regression analysis

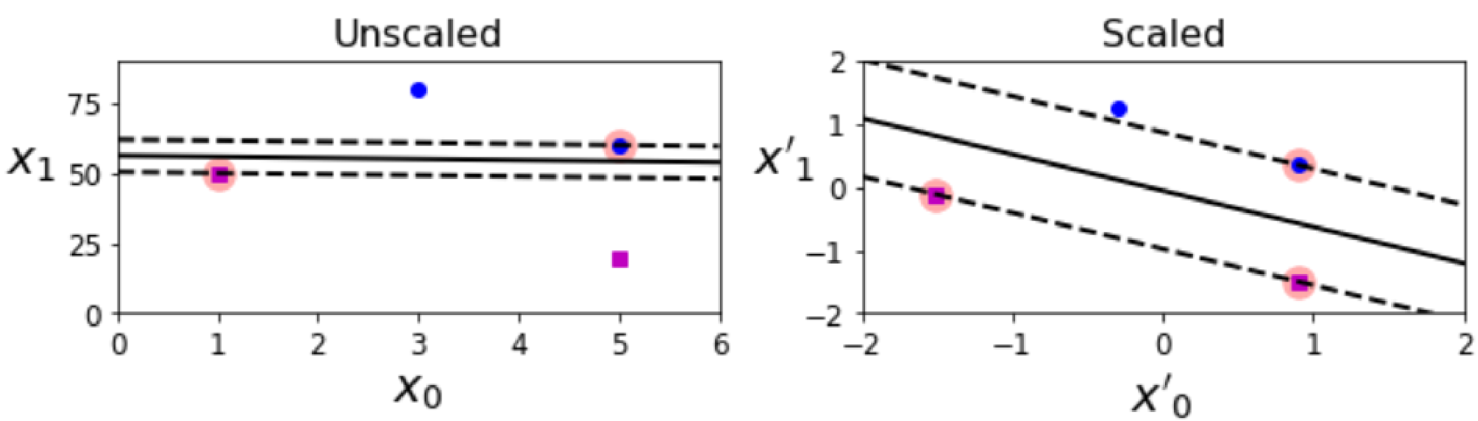
Linear SVM classification

- Basic idea
  - find the border which maximize the margin
  - large margin classification



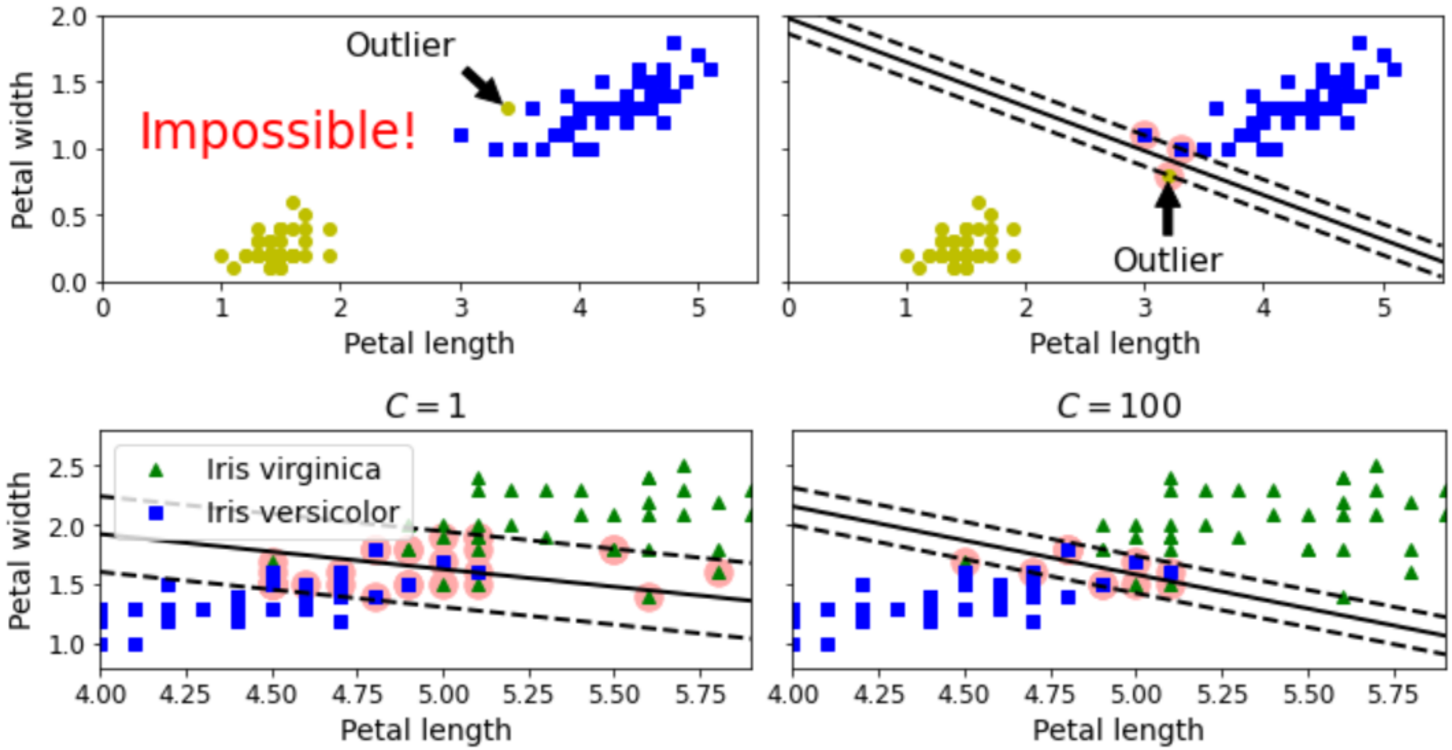
Linear SVM classification

- Importance of scaling
  - vulnerable when data is unscaled



Linear SVM classification

- Hard margin vs soft margin



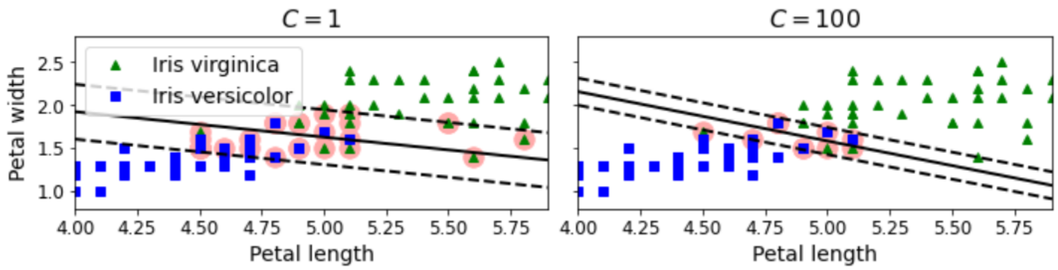
Gyeongsang National University 경상국립대학교

Linear SVM classification

- Implementation

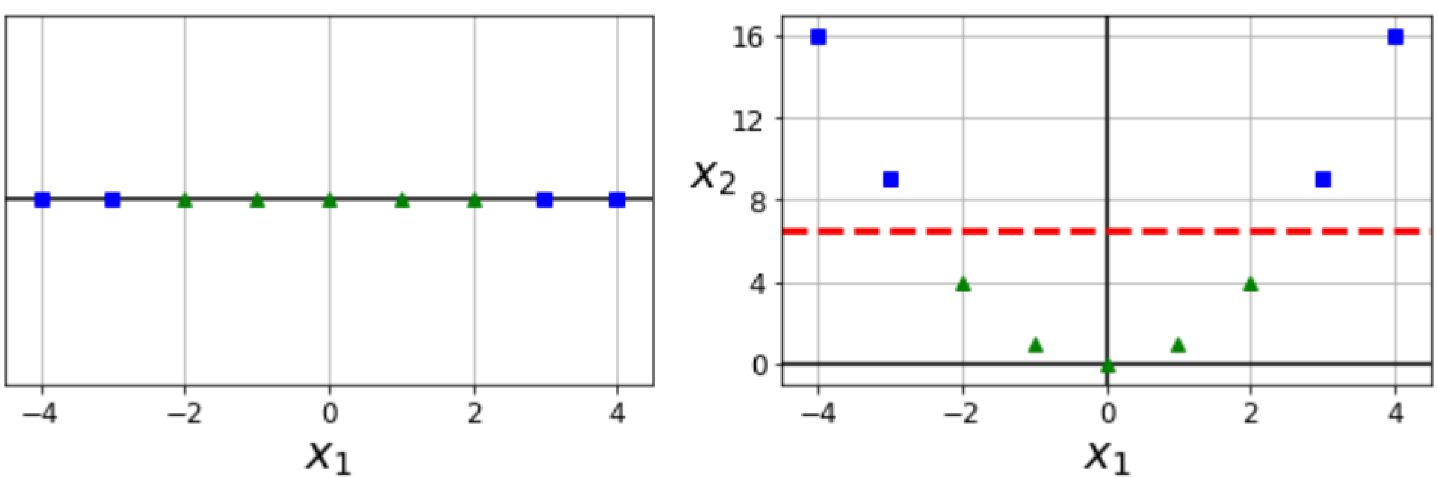
```
import numpy as np
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
```

```
scaler = StandardScaler()
svm_clf1 = LinearSVC(C=1, loss="hinge", random_state=42)
svm_clf2 = LinearSVC(C=100, loss="hinge", random_state=42)

scaled_svm_clf1 = Pipeline([
        ("scaler", scaler),
        ("linear_svc", svm_clf1),
    ])
scaled_svm_clf2 = Pipeline([
        ("scaler", scaler),
        ("linear_svc", svm_clf2),
    ])

scaled_svm_clf1.fit(X, y)
scaled_svm_clf2.fit(X, y)
```



Nonlinear SVM classification

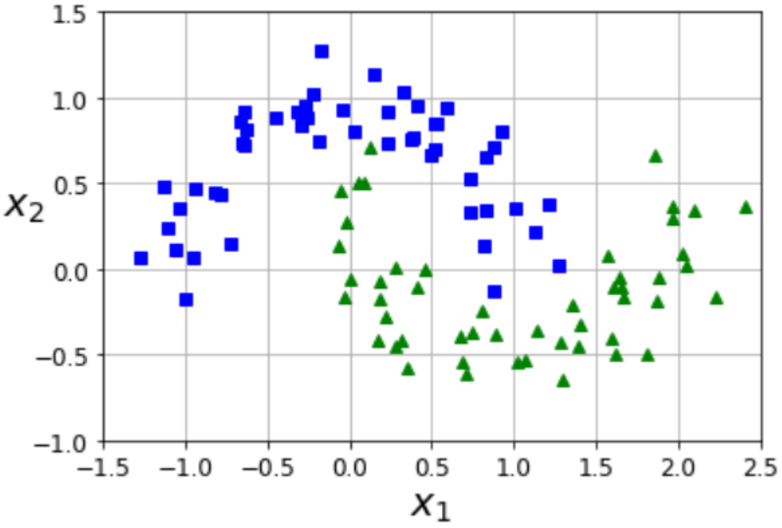
- Why we need nonlinearity?
  - new dimension can expand the possibility of classification



Nonlinear SVM classification

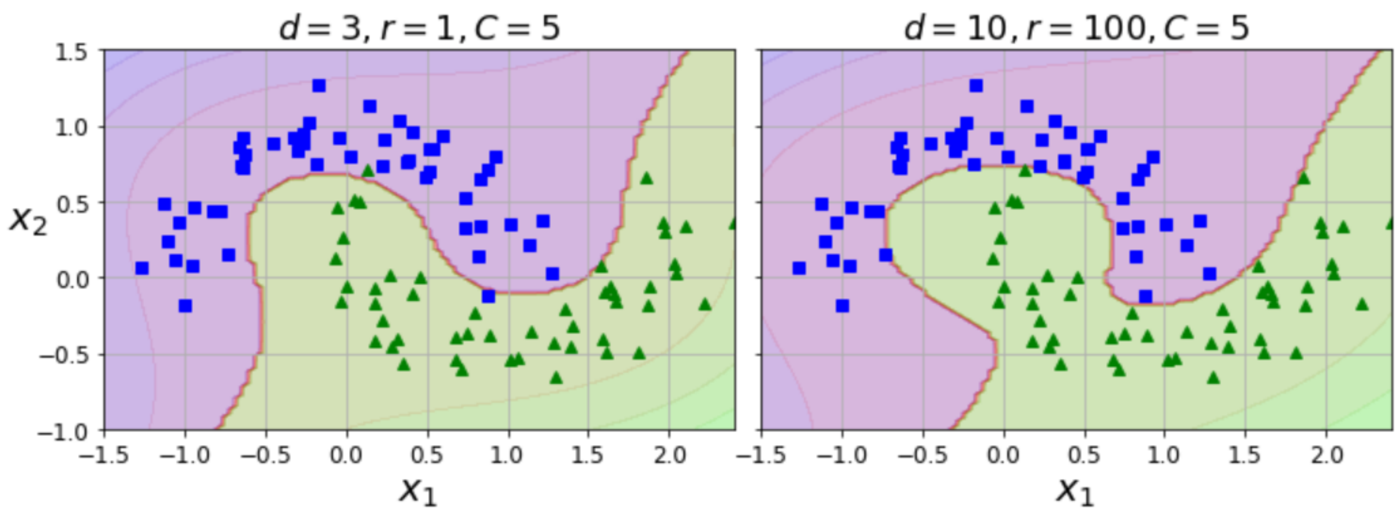
- Nonlinear data example

```
from sklearn.datasets import make_moons
X, y = make_moons(n_samples=100, noise=0.15, random_state=42)

def plot_dataset(X, y, axes):
    plt.plot(X[:, 0][y==0], X[:, 1][y==0], "bs")
    plt.plot(X[:, 0][y==1], X[:, 1][y==1], "g^")
    plt.axis(axes)
    plt.grid(True, which='both')
    plt.xlabel(r"$x_1$", fontsize=20)
    plt.ylabel(r"$x_2$", fontsize=20, rotation=0)

plot_dataset(X, y, [-1.5, 2.5, -1, 1.5])
plt.show()
```



Nonlinear SVM classification

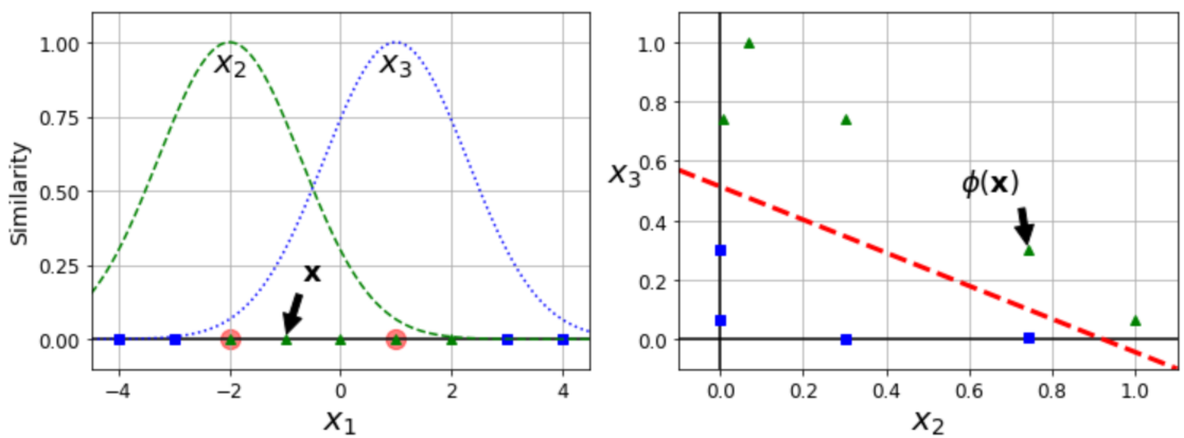
- Polynomial kernel
  - requires high computation complexity
  - makes model slower to converge



Gyeongsang National University 경상국립대학교

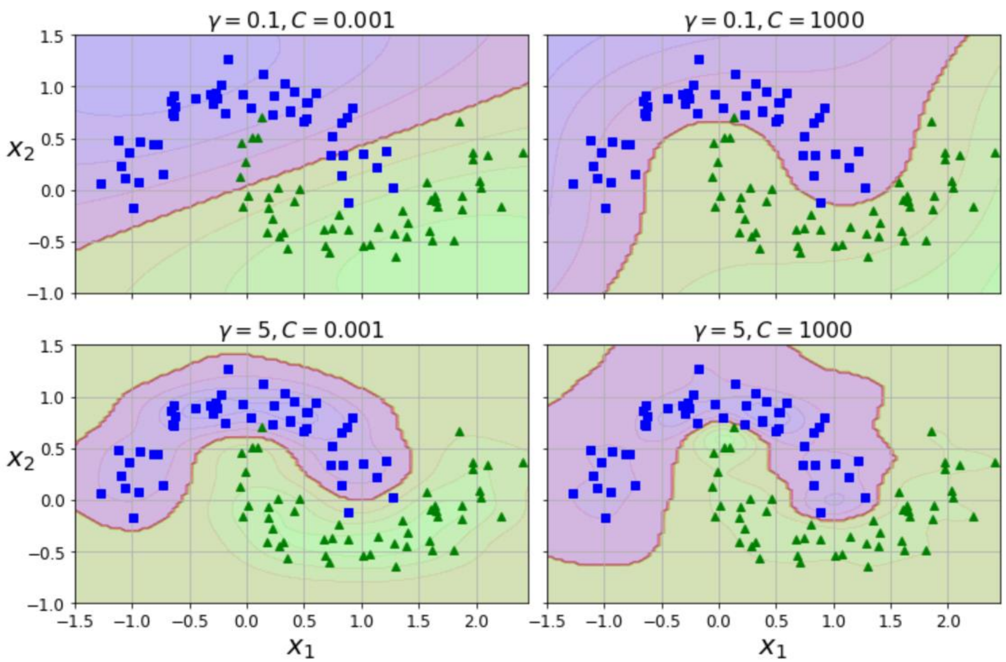
Nonlinear SVM classification

- Similarity analysis
  - take landmarks and make radial basis function (RBF) as a similarity function $\phi_\gamma(\mathbf{x}, \boldsymbol{\ell}) = \exp(-\gamma \|\mathbf{x} - \boldsymbol{\ell}\|^2)$
  - transform original data into RBF



Nonlinear SVM classification

- SVC with Gaussian RBF
  - same ways as polynomial kernel



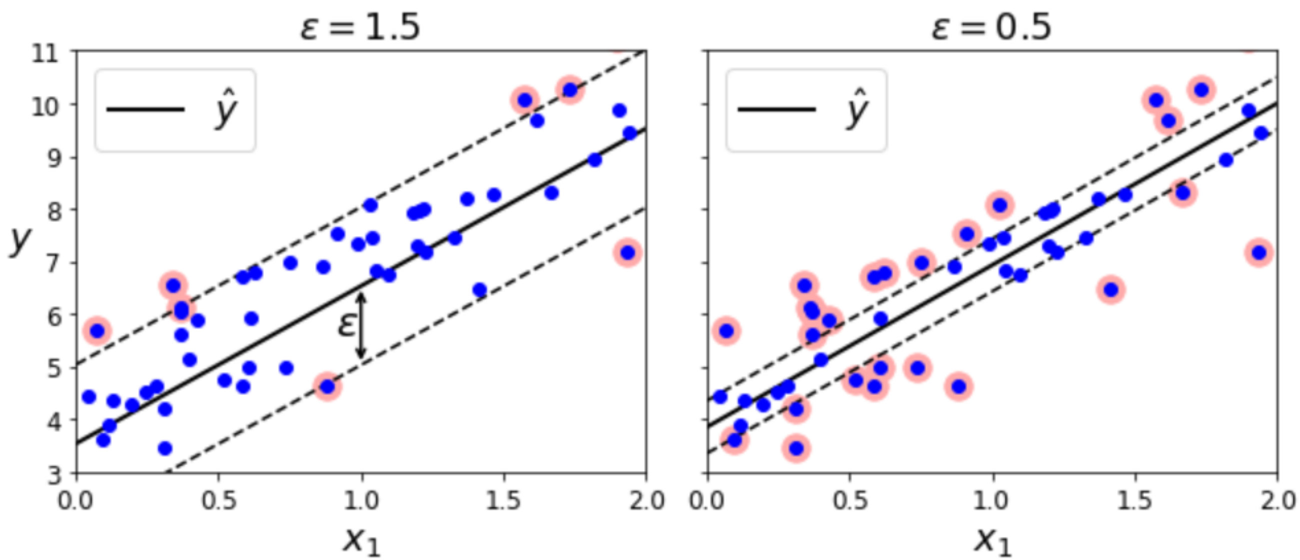
Nonlinear SVM classification

- Complexity analysis
  - Complexity in python class

Python class	Time complexity	Training from external memory	Needs for scaling	Kernel trick
LinearSVC	$O(m \times n)$	no	yes	no
SGDClassifier	$O(m \times n)$	yes	yes	no
SVC	$O(m^2 \times n) \sim O(m^3 \times n)$	no	yes	yes

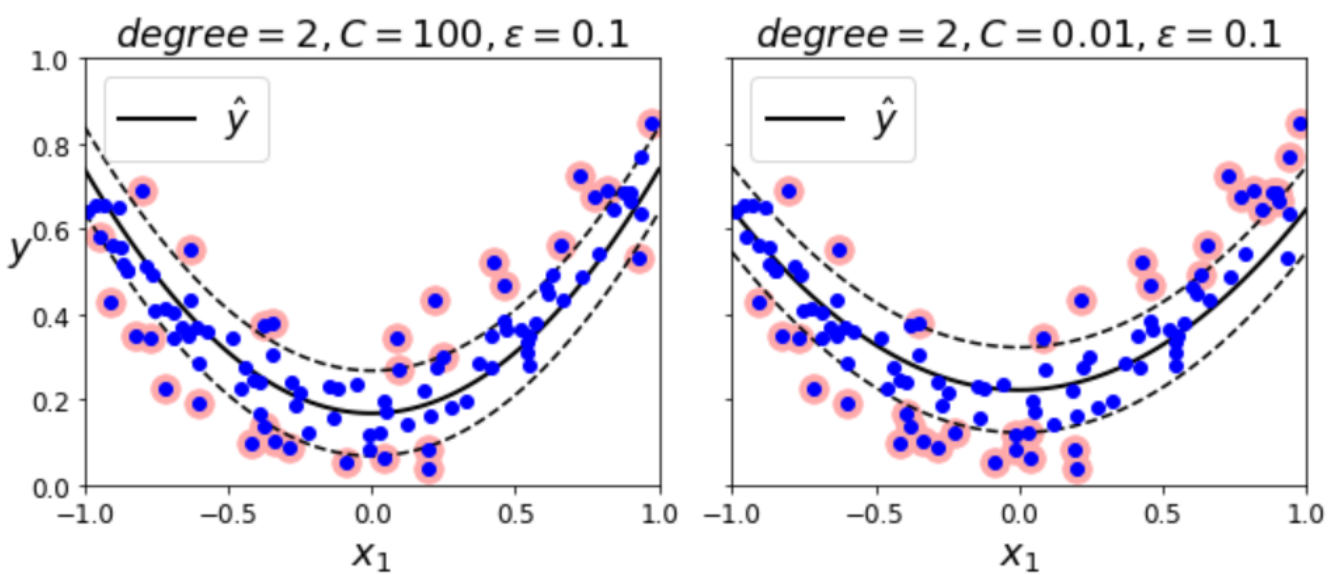
SVM regression

- Concept could be used in regression
  - find the range that contains lots of samples



SVM regression

- Nonlinear SVM regression
  - C: parameter for regulation
  - ε: parameter for margin



SVM theory

- Notation
  - linear regression: bias and weight $\theta_0, \theta_1, \dots, \theta_n \rightarrow \boldsymbol{\theta}$
  - input for bias $x_0 = 1$
  - SVM: bias b and weight $\boldsymbol{w}$
  - no input for bias in SVM

SVM theory

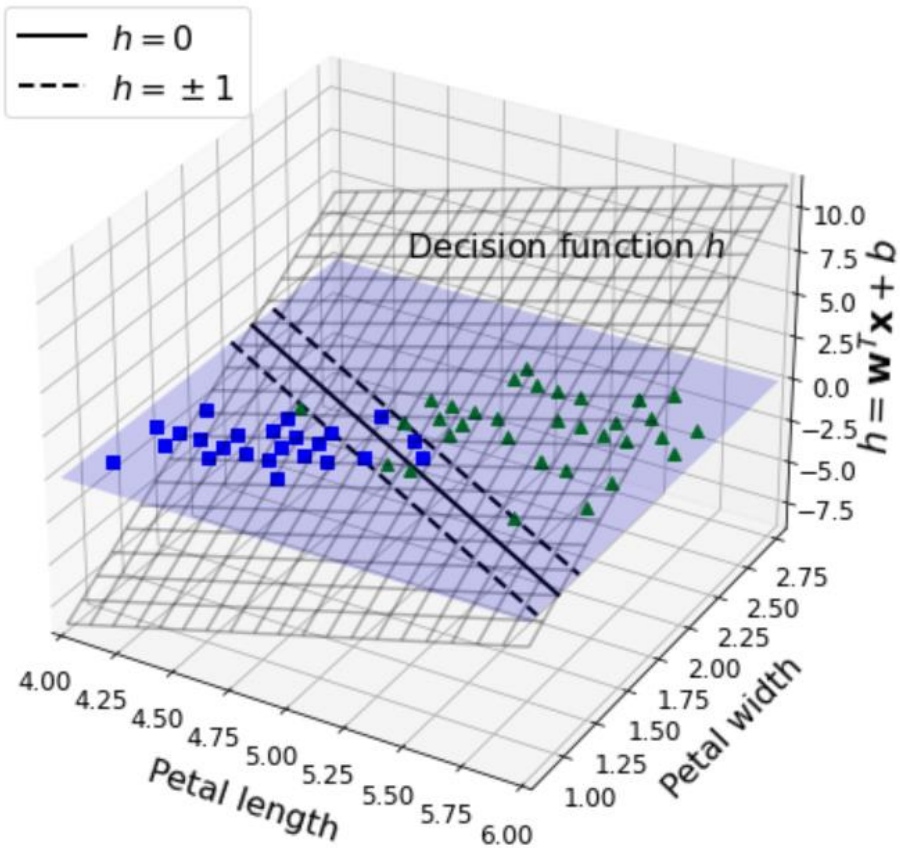
- Expectation in linear SVM

$$\hat{y} = \begin{cases} 0 & \mathbf{w}^T\mathbf{x} + b < 0 \\ 1 & \mathbf{w}^T\mathbf{x} + b \geq 0 \end{cases}$$

  - for example, in iris dataset
    - input: 2-dimension(petal length, petal width)
    - output: result of calculation $\mathbf{w}^T\mathbf{x} + b$

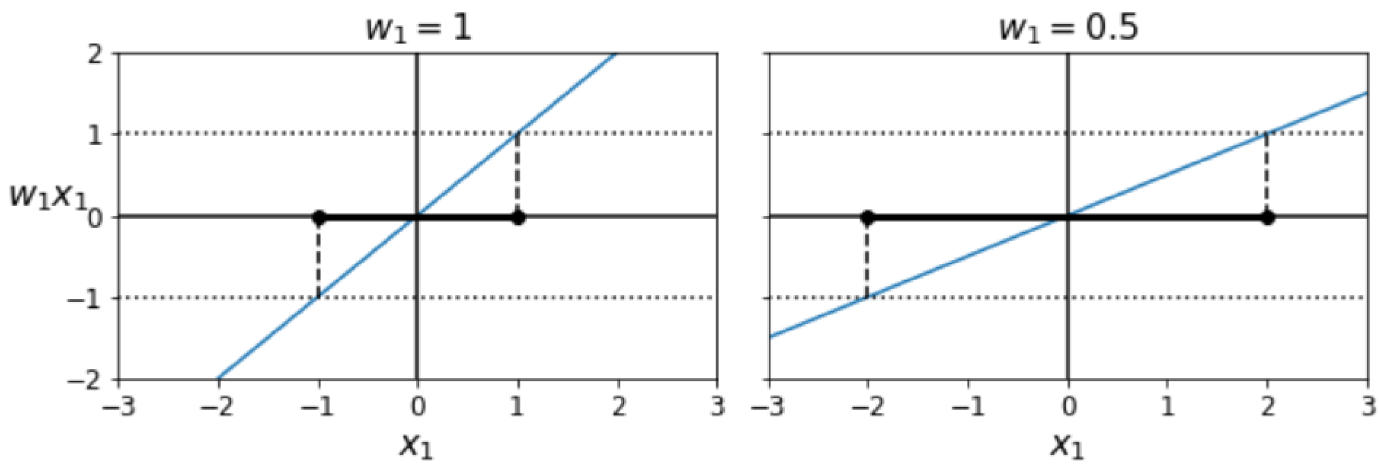
SVM theory

- Visualization in linear SVM



SVM theory

- Relation of weight and margin
  - smaller weight results larger margin



SVM theory

- Problem formulation in hard margin case

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 \quad \text{for } i = 1, 2, \ldots, m \end{aligned}$$

  - negative sample: $y^{(i)} = 0 \rightarrow t^{(i)} = -1$
  - positive sample: $y^{(i)} = 1 \rightarrow t^{(i)} = +1$

GNU Gyeongsang National University 경상국립대학교

SVM theory

- Problem formulation in soft margin case

$$\underset{\mathbf{w},b,\zeta}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\zeta^{(i)}$$

$$\text{subject to} \quad t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \zeta^{(i)} \quad \text{and} \quad \zeta^{(i)} \geq 0 \quad \text{for } i = 1, 2, \ldots, m$$

  - slack variable $\zeta^{(i)}$
    - how much the solution violates the margin

Gyeongsang National University 경상국립대학교

SVM theory

- Second-order optimization problem
  - quadratic programming (QP)
  - out of scope in this lecture
    - one solution: duality
    - formulating dual problem

SVM theory

- Kernel trick
  - how to make non-linearity?

$$\phi(\mathbf{x}) = \phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} {x_1}^2 \\ \sqrt{2}\, x_1 x_2 \\ {x_2}^2 \end{pmatrix}$$

$$\begin{aligned} \phi(\mathbf{a})^T \phi(\mathbf{b}) &= \begin{pmatrix} {a_1}^2 \\ \sqrt{2}\, a_1 a_2 \\ {a_2}^2 \end{pmatrix}^T \begin{pmatrix} {b_1}^2 \\ \sqrt{2}\, b_1 b_2 \\ {b_2}^2 \end{pmatrix} = {a_1}^2 {b_1}^2 + 2 a_1 b_1 a_2 b_2 + {a_2}^2 {b_2}^2 \\ &= (a_1 b_1 + a_2 b_2)^2 = \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}\right)^2 = (\mathbf{a}^T \mathbf{b})^2 \end{aligned}$$

SVM theory

- Online SVM
  - loss function

$$J(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m} max\left(0, t^{(i)} - (\mathbf{w}^T\mathbf{x}^{(i)} + b)\right)$$

  - first term: SVM original
  - second term: margin error
  - gradient descent

GNU Gyeongsang National University 경상국립대학교

Feel free to question

Through e-mail & LMS

본 자료의 연습문제는 수업의 본교재인
한빛미디어, Hands on Machine Learning(2판)에서 주로 발췌함

GNU Gyeongsang National University 경상국립대학교