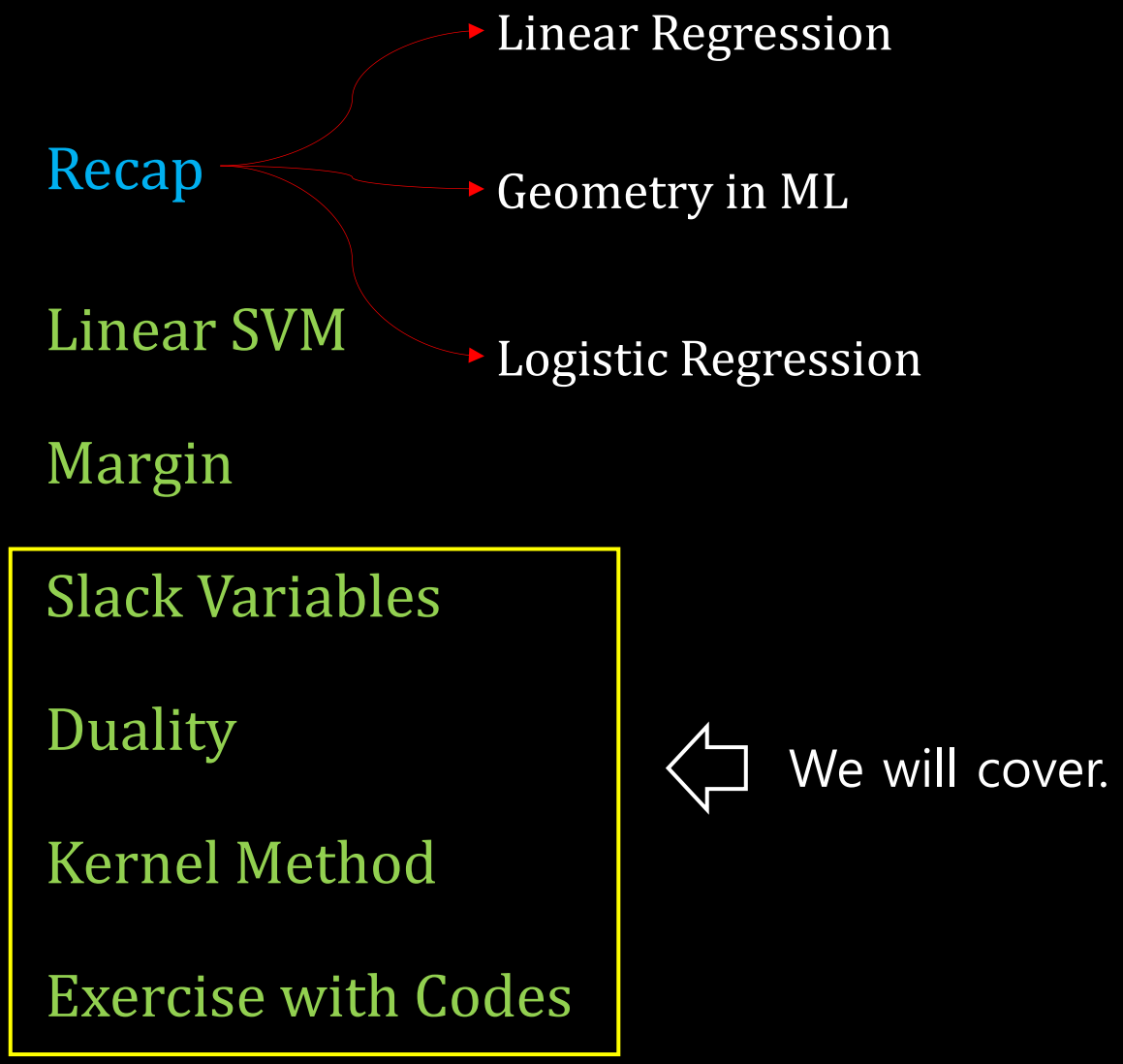
Machine Learning 2

Support Vector Machine (SVM)

Dept. SW and Communication Engineering

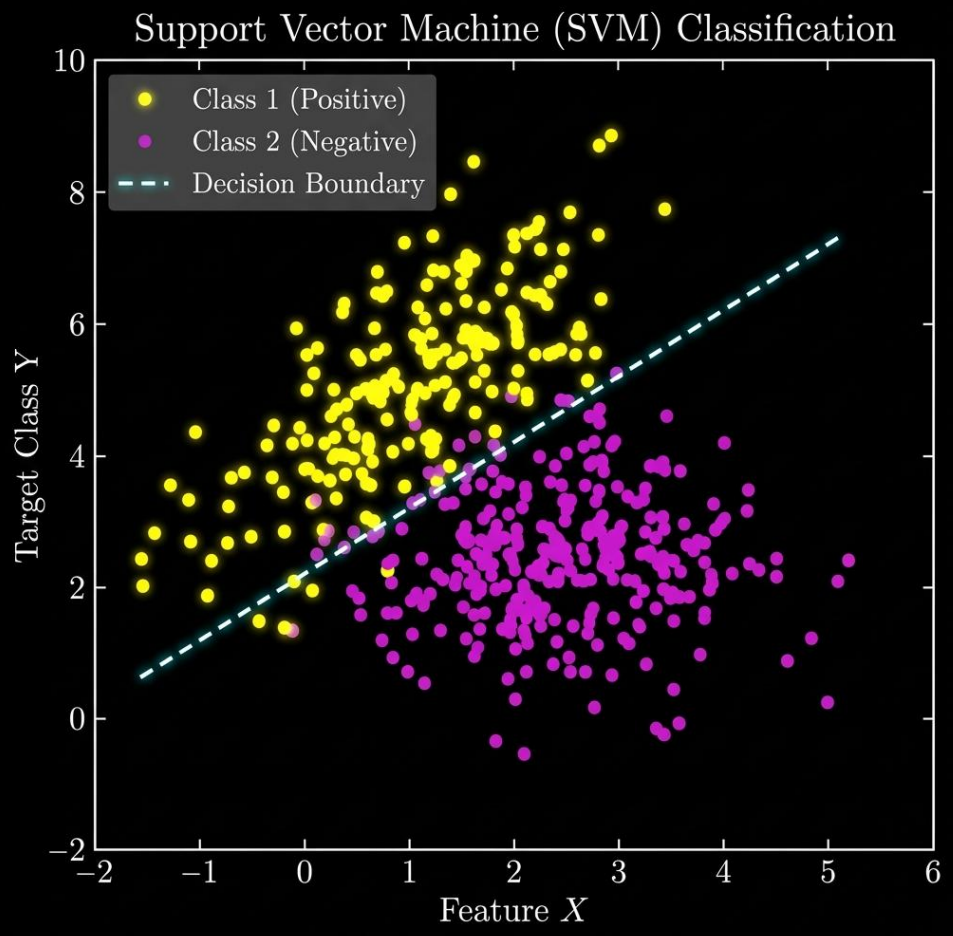
Prof. Giseop Noh (kafa46@hongik.ac.kr)

Contents


Recap
Linear Regression
Geometry in ML
Logistic Regression
Linear SVM
Margin
Slack Variables
Duality
Kernel Method
Exercise with Codes
We will cover.

Slack Vriables

Problematic Linear Separation

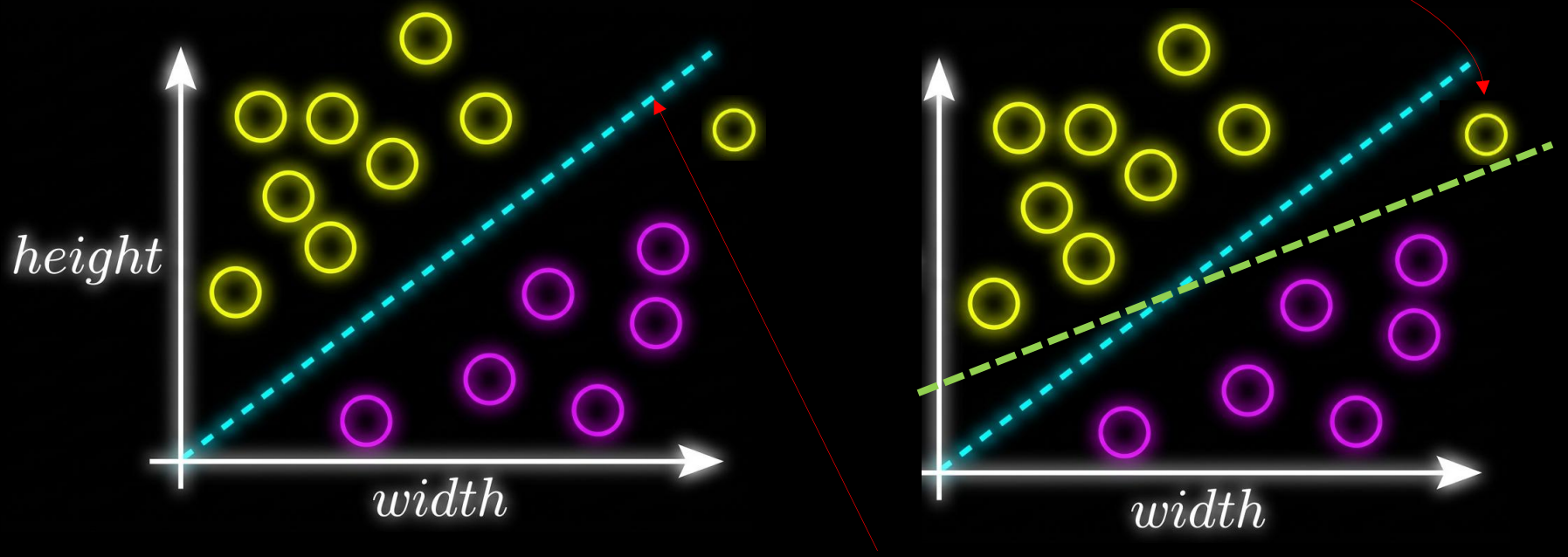


- ✓ Real-world data is often **not perfectly** linearly separable.
- ✓ Some points may be **outliers or overlapping** between classes.
- ✓ Enforcing 100% classification accuracy can lead to **overfitting**.
- ✓ If the constraints cannot be satisfied, **hard-margin SVM fails**.

Perfect Prediction vs. Overfitting

SVM can perfectly predict (separate) all points.

SVM is adjusted to outliers.



Even SVM misses some outliers,
this line is better!

How can we handle this?

Introduce "Slack Variable"

Recap: Adding Regularization

We introduce slack variable, "Soft-Margin SVM"

Make SVM allow for constraints to misclassify a few points

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i$$

Misclassification Penalty

subject to

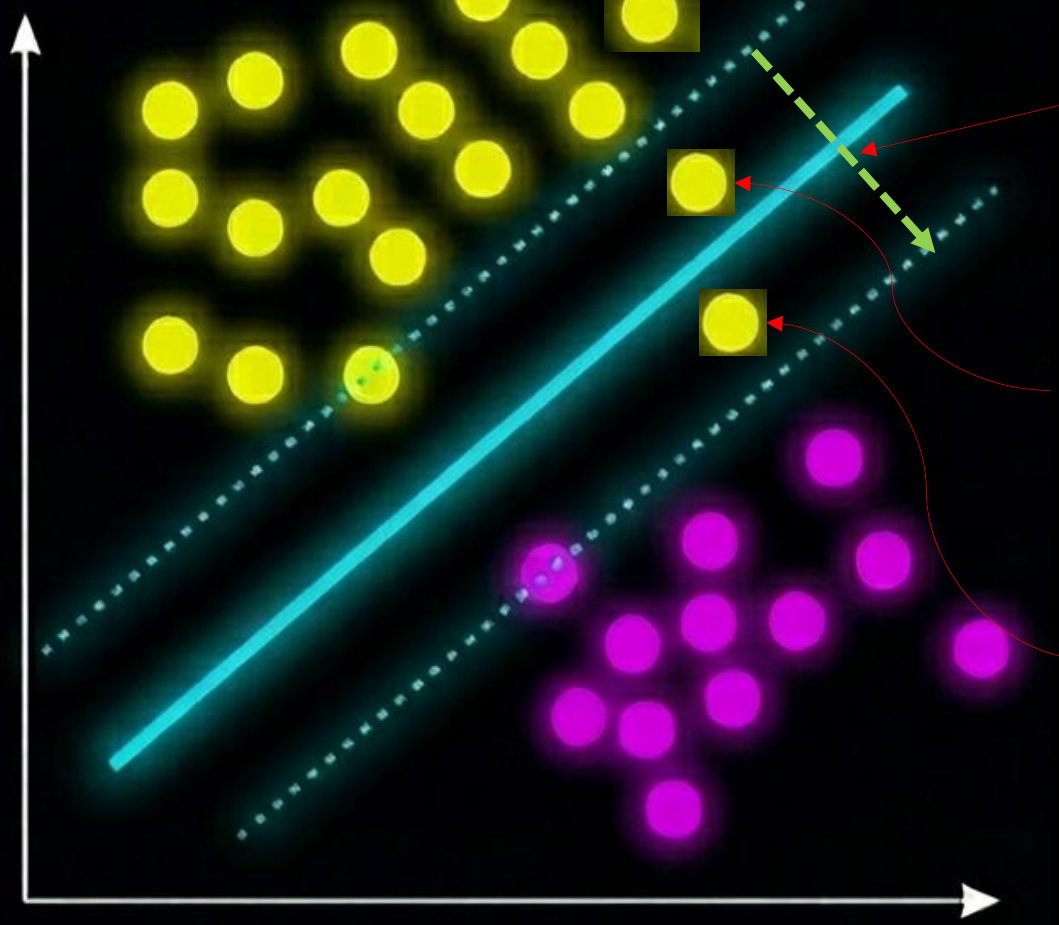
$$y_i(w^T x_i + b) > \hat{y}_i - \xi_i$$

$$\xi_i \geq 0\,, \forall i = 1, 2, \cdots, N$$

The more wrong classification, the bigger ξ_i

C controls relative strength of misclassification penalty

Understanding Slack Variable



$\xi_i = 0$: correctly classify

$\xi_i > 0$: violate margin
(not all misclassified)

$\xi_i = 0.5$: correctly classify
but x_i is inside of margin

$\xi_i > 1$: x_i wrong side of
decision boundary

Gradient Descent in Linear SVM

Hinge Loss

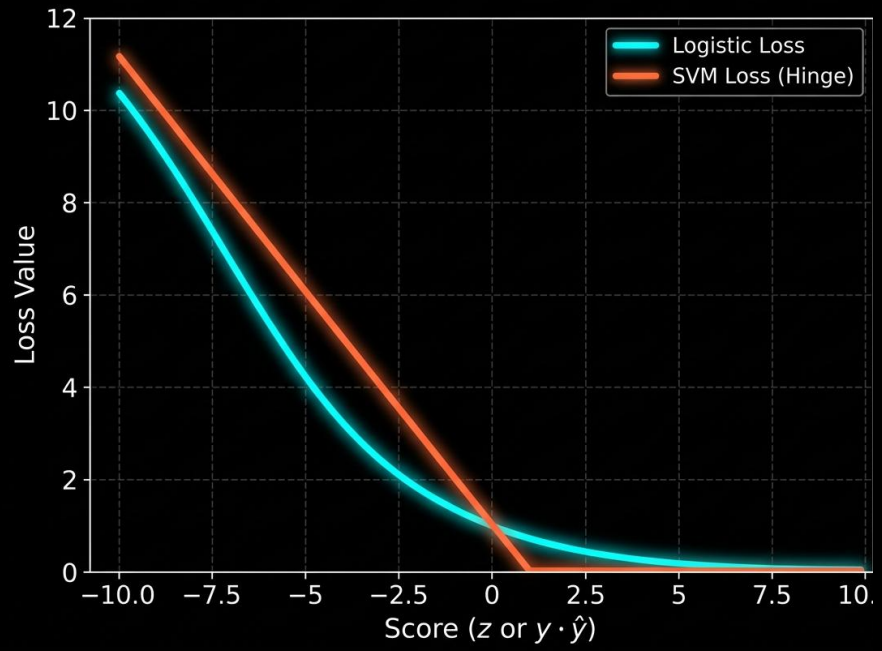
$$\xi_i \geq 0$$

$$y_i\left(w_i^T x_i\right) \geq 1 - \xi_i$$

Let's consider equal condition instead of inequality.

$$y_i\left(w_i^T x_i\right) = 1 - \xi_i$$

$$\xi_i = 1 - y_i\left(w_i^T x_i\right)$$



ξ is non-negative,

$$\xi_i = \max\left(0, 1 - y_i\left(w_i^T x_i\right)\right)$$

Implementation Approach

$$L = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \max\left(0, 1 - y_i\left(w_i^T + b\right)\right)$$

```
class SVM:

    def fit(X, Y):

        // do something

    def predict(X):

        return sign(X.dot(w) + b)
```

$$\nabla_w L = w - C \sum_{i:\xi_i > 0} y_i x_i$$

$$\frac{\partial L}{\partial b} = -C \sum_{i:\xi_i > 0} y_i$$

$$w \leftarrow w - \alpha \nabla_w L$$

$$b \leftarrow b - \beta \frac{\partial L}{\partial b}$$

Duality
(Linear → Non-linear SVM)

Lagrange Multipliers

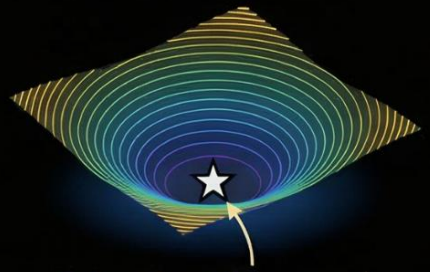
PRIMAL PROBLEM

$$\min_{x \in \mathbb{R}^n} f(x)$$

Subject to:

$$g_i(x) \leq 0, \; i = 1, \ldots, m$$

"Inequality constraints"



Primal optimum x^* (when it exists)

LAGRANGIAN

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

- Define **Lagrange multipliers**:
  $$\lambda \in \mathbb{R}^m, \; \lambda \geq 0$$
- **Key mechanism**: A large positive penalty is applied to L when any constraint is violated (e.g., $g_i(x) > 0$).
- **Complementary Slackness**: Feasibility gap is zero for optimal primal x^* and dual λ^*:
  $$\lambda_i^* g_i(x^*) = 0, \; \forall i$$

DUAL PROBLEM

Dual function:

$$g(\lambda) = \inf_{x \in \mathbb{R}^n} L(x, \lambda)$$

- $g(\lambda)$ is **Concave**: even if f and g_i are not convex!
- Pointwise infimum of affine functions of λ.

Dual optimization problem:

$$\max_{\lambda \in \mathbb{R}^m, \, \lambda \geq 0} g(\lambda)$$

Primal Problem: $f(x)$ → Construct Lagrangian: $L(x, \lambda)$ → Form Dual: $g(\lambda)$ → Maximize Dual to get dual-optimal λ^*

Key Idea: 1. Transform a **CONSTRAINED** minimization problem into an **UNCONSTRAINED** (though sometimes implicit) maximization problem using the dual function.
2. Primal feasibility is implicitly enforced by the non-negativity of multipliers λ.

Introduction to Duality

If we have two types algorithms such that:

- Algorithm P^*: takes 10 Hours to solve
- Algorithm Q^*: takes 10 Minuntes to solve

Absolutely, we prefer the algorithm Q^*

This paradigm also exists in QP as well as LP,

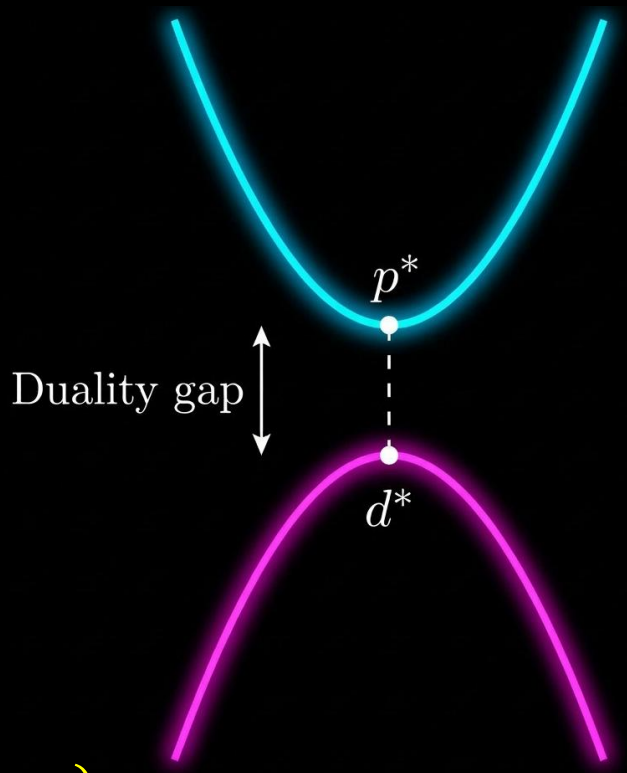
which means the paradigm can be applied to SVM

General Objective

p^*: minimizes primal objective

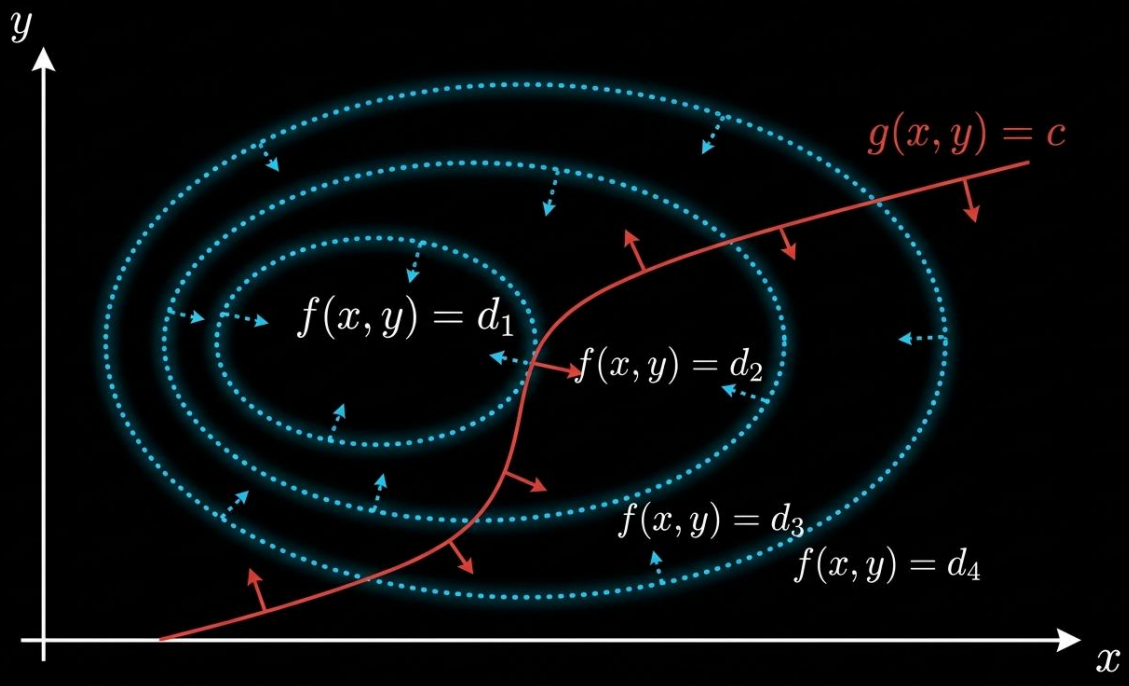
d^*: maximizes dual objective

$p^* - d^*$: duality gap



Our goal: $p^* = d^*$ (minimize duality gap)

Lagrange Mulitpliers (1/2)



$$\max_{x,y} f(x,y)$$

subject to $g(x,y) = 0$

The solution seems to be somewhere

f and g are parallel!

Lagrange Mulitpliers (2/2)

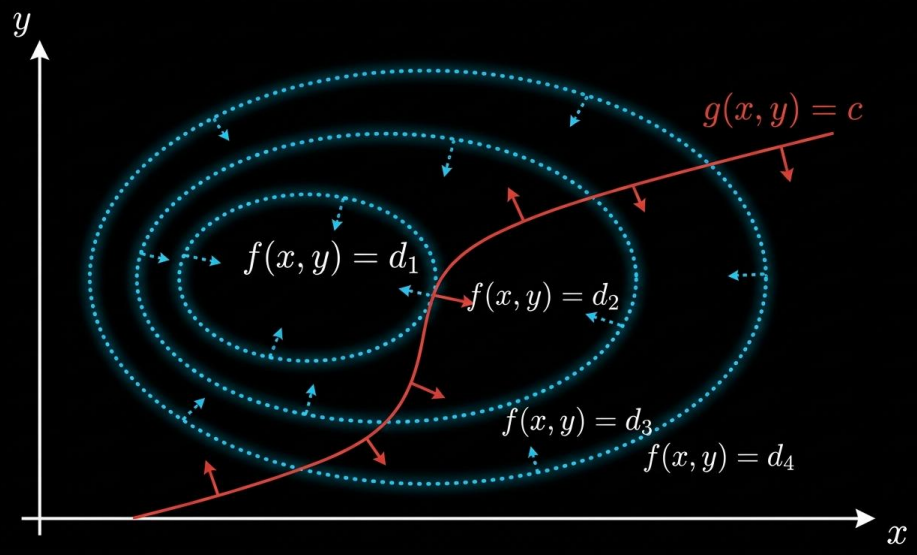
Introduce new variable

λ: Lagrange multiplier

Setup new function:

$$L(x, y, \lambda) = f(x, y) - \lambda g(x, y)$$

We call $L(x, y, \lambda)$ as "Lagrangian"



The solution to find $L(x, y, \lambda)$ such that $\nabla L = 0$

$$\nabla_{x,y,\lambda} L(x, y, \lambda) = 0$$

Derivation of Required Equations

$$L(x, y, \lambda) = f(x, y) - \lambda g(x, y)$$

$$\nabla_{x,y,\lambda} L(x, y, \lambda) = 0$$

$$\nabla_{x,y} L(x, y, \lambda) = 0$$

$$\nabla_{x,y} f(x, y) - \lambda \nabla_{x,y} g(x, y) = 0$$

$$\nabla_{x,y} f(x, y) = \lambda \nabla_{x,y} g(x, y)$$

$$\frac{\partial L}{\partial \lambda} = g(x, y) = 0$$

$$\nabla_{x,y,\lambda} L(x, y, \lambda) = 0$$

This generates
3 unknows and 3 equation.

We can easily find solution!

General Form of Constrained Optimization

$$\min_x f(x)$$

subject to: $g_i(x) \leq 0, \forall i = 1, \cdots, N$ Inequality constraint

$h_j(x) = 0, \forall i = 1, \cdots, M$ Equality constraint

$$L(x, \alpha, \lambda) = f(x) + \sum_{i=1}^{N} \alpha_i g_i(x) + \sum_{j=1}^{M} \lambda_j h_j(x), \alpha_i \geq 0, \lambda_j \in \mathbb{R}$$

$$\min_x f(x) \rightarrow \min_x L(x, \alpha, \lambda) \rightarrow \nabla_x L(x, \alpha, \lambda) = 0$$

Dual Problem:

$g(\alpha, \lambda) = \inf_x L(x, \alpha, \lambda)$ Fix α and λ, then find minimize $L(x, \alpha, \lambda)$ w.r.t x (Lower bound of x)

$\max_{\alpha \geq 0, \lambda \in \mathbb{R}} g(\alpha, \lambda)$ Function without x, dual variables: α, λ (We want maximum lower bound)

$$\nabla_x L(x, \alpha, \lambda) = \nabla_x f(x) + \sum_{i=1}^{N} \alpha_i \nabla_x g_i(x) + \sum_{j=1}^{M} \lambda_j \nabla_x h_j(x) = 0$$

Optimality Condition of the Lagrangian (Stationarity Condition)

$$\nabla_x L(x, \alpha, \lambda) = \nabla_x f(x) + \sum_{i=1}^{N} \alpha_i \nabla_x g_i(x) + \sum_{j=1}^{M} \lambda_j \nabla_x h_j(x) = 0$$

"Stationary": 더 이상 변하지 않는 상태
기울기(gradient)가 0 이라서 함수 값이 증가하거나 감소하지 않는 점

$$\nabla_x f(x) = -\sum_{i=1}^{N} \alpha_i \nabla_x g_i(x) - \sum_{j=1}^{M} \lambda_j \nabla_x h_j(x)$$

Primal problem's gradient = constraint gradient combination

At the optimum,

the gradient of the objective is balanced

by the gradients of the constraints.

KKT Conditions (Optimality Conditions)

KKT (Karush–Kuhn–Tucker) conditions

KKT conditions characterize the optimal solution of constrained optimization problems.

제약 조건이 있는 최적화 문제의 최적해가 만족해야 하는 필요 조건
(constraint optimization에서의 optimality 조건)

1. Stationary condition: $\nabla_x L(x, \alpha, \lambda) = 0$

2. Primal feasibility: $g_i(x) \leq 0, h_j(x) = 0$

3. Dual feasibility: $\alpha_i \geq 0$

4. Complementary slackness: $\alpha_i g_i(x) = 0$

When strong duality holds,
the optimal solutions of primal and dual are equal.

Applying Lagrangian Duality to SVM

$$L(w, b, \alpha) = f(w, b) + \sum_{i=1}^{N} \alpha_i g_i(w, b)$$

$$\text{, where } \quad f(w, b) = \frac{1}{2}\|w\|^2, \quad g_i(w, b) = 1 - y_i(w^T x_i + b) \leq 0$$

Put all together:

SVM objective + Duality + Lagrange + KKT + ⋯

We will omit all heavy mathematical works

$$L(w, b, \alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\Rightarrow \quad \max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

From Primal SVM to Dual SVM

Huge & heavy math are omitted

Only inner products between training samples appear

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

subject to:

$$y_i(w^T x_i + b) \geq 1, \quad \forall i = 1 \dots N$$

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to:

$$\alpha_i \geq 0 \ (i = 1 \dots N), \sum_{i=1}^{N} \alpha_i y_i = 0$$

α_i: Lagrange multiplier
(importance of i-th constraint)

Decision boundary:
Linear combination of support vectors!

Almost data points: $\alpha_i = 0$

Data points near decision boundary: $\alpha_i > 0$

Support vector

Train & Predict

Train

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to:

$$\alpha_i \geq 0, \forall i = 1 \ldots N, \qquad \sum_{i=1}^{N} \alpha_i y_i = 0$$

Predict

$$f(x) = w^T x + b = \sum_{i \in SV} \alpha_i y_i x_i^T x + b$$

x: input data

x_i: training data

SV: support vectors

where $w = \sum_{i \in SV} \alpha_i y_i x_i^T$ is linear combination of SV

Kernel Methods

Why Do We Need the Kernel Trick?

Problem: Linear Models Have Limited Expressiveness

Linear classifiers use a hyperplane:

$$f(x) = w^T x + b$$

- Decision boundary is linear
- Many real datasets are not linearly separable
- Examples: circular patterns, XOR structure

Idea: Map Data to a Higher-Dimensional Space

Transform the input features

$$x \rightarrow \phi(x)$$

Data becomes **linearly separable** in the new space.

Example:

$$\phi(x) = (x_1^2, x_2^2, x_1, x_2)$$

Introduction to Kernel Trick

Problem: Feature Explosion

Explicit feature expansion causes:

- Combinatorial growth of features
- High computational cost
- Memory inefficiency
- Increased overfitting risk

Solution: Kernel Trick

Instead of computing the mapping explicitly,

compute the inner product in feature space:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

How to Kernel Apply

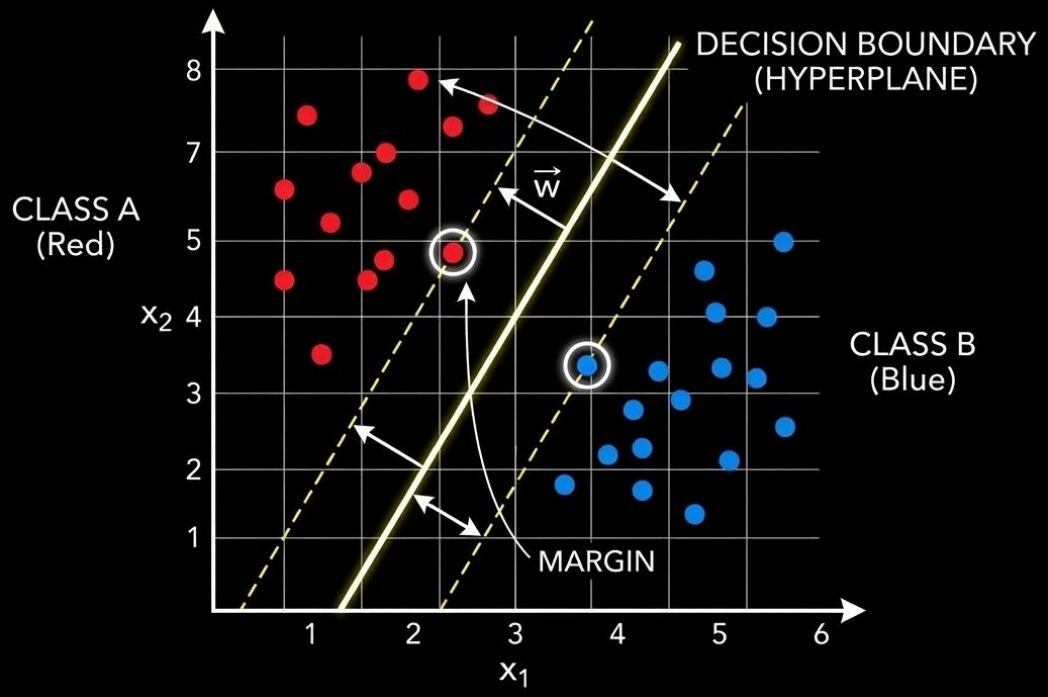
Training	Prediction
$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$	$w^T x + b = \sum_{i \in SV} \alpha_i y_i x_i^T x + b$
⇩	⇩
$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{K(x_i, x_j)}$	$w^T x + b = \sum_{i \in SV} \alpha_i y_i \boldsymbol{K(x_i, x)} + b$

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

Linear Kernel

$$K(x, z) = x^T z$$

- ✓ 데이터를 다른 차원으로 매핑하지 않는다.
- ✓ Kernel trick이 실제로 필요 없는 경우
- ✓ 일반적인 선형 SVM과 동일



장점

- ✓ 계산이 매우 빠름
- ✓ 메모리 사용 적음
- ✓ 대규모 데이터에 적합

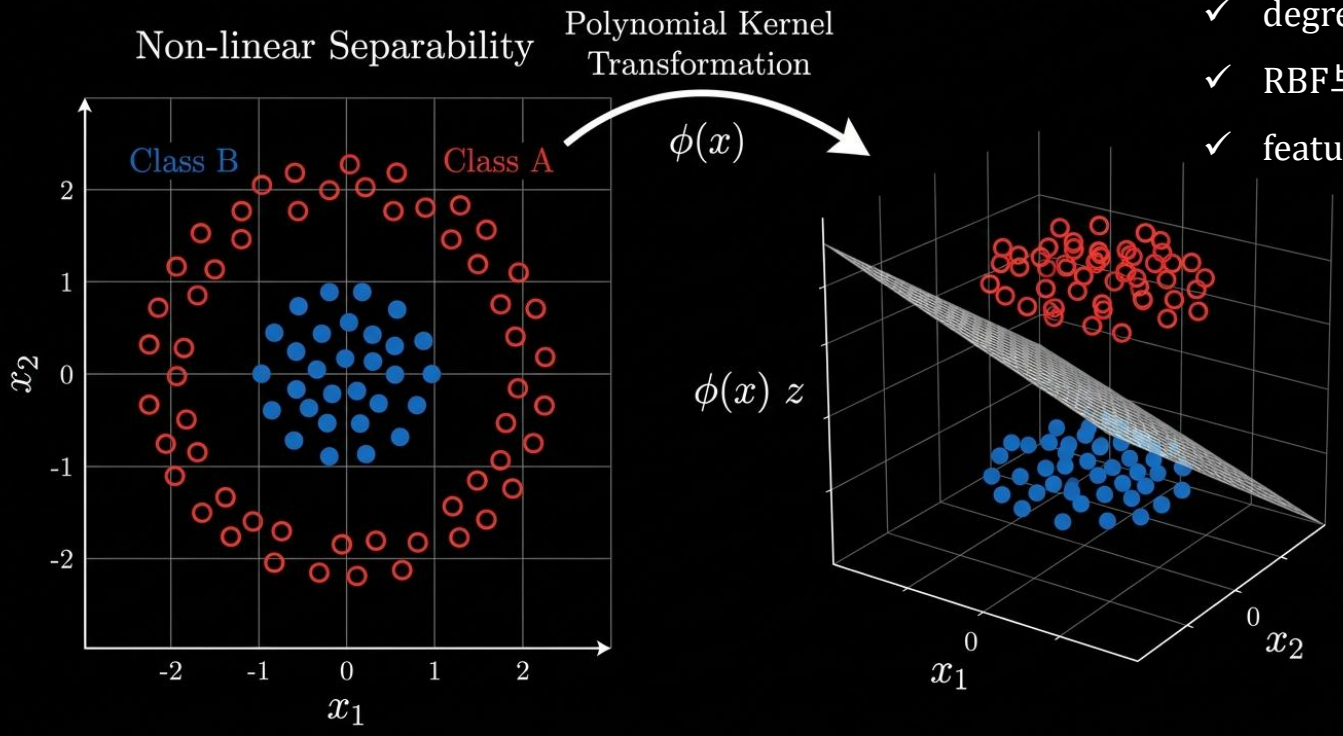
단점

- ✓ 비선형 패턴을 모델링하기 어려움
- ✓ 데이터가 선형적으로 분리되지 않으면 성능이 떨어질 수 있음

Polynomial Kernel

$$K(x,z) = (\gamma x^T z + c)^d$$

- ✓ 데이터의 다항식 조합을 자동으로 생성하는 효과
- ✓ 고차 특징 공간으로 확장하는 것과 같음



장점

- ✓ feature interaction 모델링 가능
- ✓ 비교적 직관적인 커널
- ✓ 특정 문제에서 매우 효과적

단점

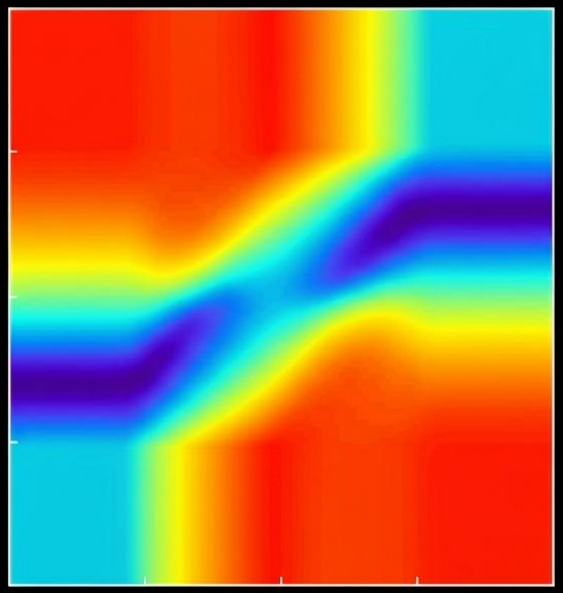
- ✓ degree가 커지면 overfitting 위험
- ✓ RBF보다 일반적으로 성능이 떨어짐
- ✓ feature scaling에 민감

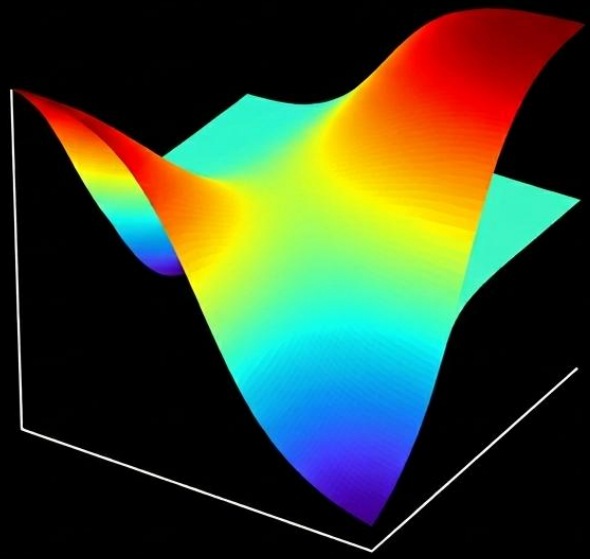
Sigmoid Kernel

$$K(x, z) = \tanh(\gamma x^T z + c)$$

- ✓ 신경망의 activation function과 동일
- ✓ Surface가 saddle 형태
- ✓ Neural Network와 SVM을 연결하는 커널로 제안

결정 경계가 S-shape 형태
입력 공간에서 곡선 형태의 boundary
두 데이터가 유사하면 → 1에 가까움
반대 방향이면 → -1에 가까움





학습이 불안정하고
SVM 최적화 convex가 깨질 수 있음
SVM에서는 거의 사용하지 않음

Gaussian Kernel (Radial Basis Function)

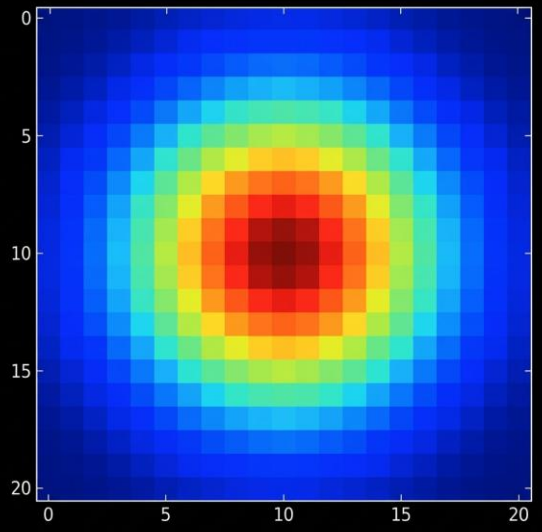
$$K(x, z) = \exp(-\gamma \|x - z\|^2)$$

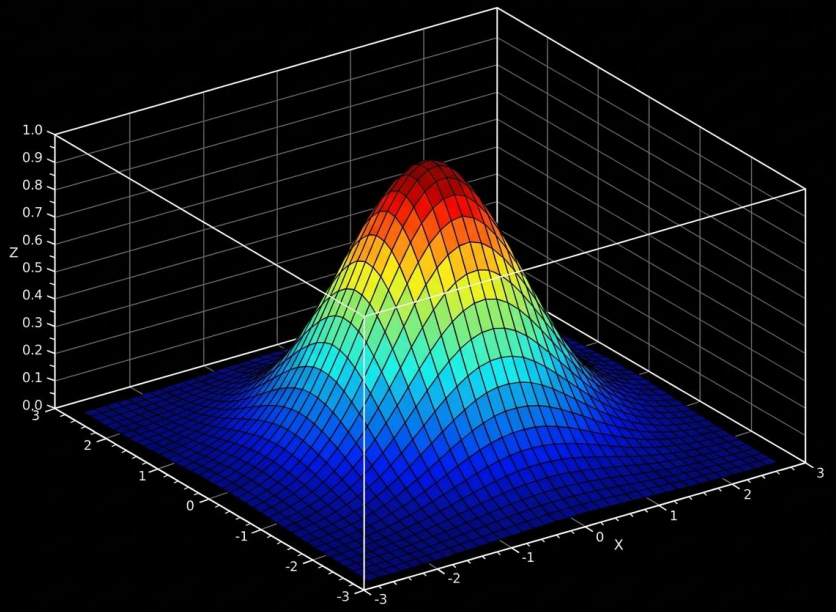
- ✓ 여기에 수식을 입력하십시오.두 점이 가까우면 커널 값이 1에 가깝다
- ✓ 두 점이 멀어지면 커널 값이 0에 가깝다.
- ✓ Local similarity를 아주 잘 표현한다
- ✓ 항상 convex 유지 → 최적해 보장

Also called
"Radial Basis Function (RBF) Kernel"

봉우리 중심에서 값이 가장 크고, 중심에서 멀어질수록 지수적으로 감소하는 형태

참고: $N(x|\mu, \sigma^2) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$





Excercise with Codes

Exercise SVM in Code Level

Ex #1. 2D Classification (Social Network Advertising)

Ex #2. Non-linear classification (Donut Problem)

Ex #3. Medical symptom prediction (Breast Cancer)

Ex #4. Handwriting image classification (MNIST)

Web link: https://www.deepshark.org/courses/machine_learning_2/

Thank you!