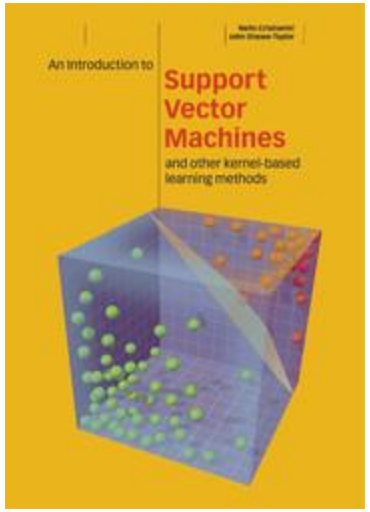
# Support Vector Machines and Other Kernel Methods

**Assoc. Prof. Karl Ezra Pilario, Ph.D.**
Process Systems Engineering Laboratory
Department of Chemical Engineering
University of the Philippines Diliman

© 2023 Karl Ezra S. Pilario, PhD.

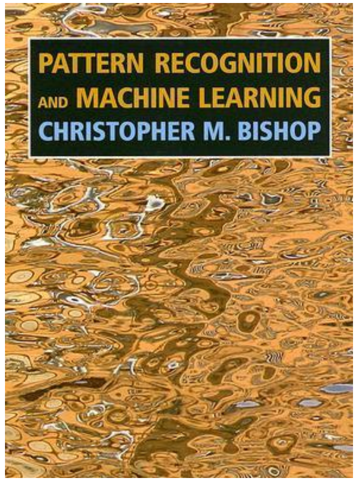
# Outline

- Support Vector Machines
  - Large-Margin Classifiers
  - Quadratic Optimization Problem
  - Nonlinear SVM using Kernels
  - Mercer’s Theorem
  - Multiple Kernel Learning
- Multi-class Classification
- Kernel Methods for Regression
  - Support Vector Regression
  - Kernel Ridge Regression




Cristianini & Shawe-Taylor (2000)
*An Introduction to Support Vector Machines and other kernel-based learning methods.*
Cambridge University Press.

Bishop (2006)
*Pattern Recognition and Machine Learning*. Springer.




© 2023 Karl Ezra S. Pilario, PhD.

# Review

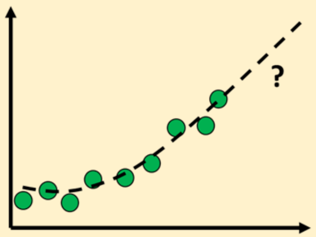
**Supervised Learning**

Learn a mapping or a function:

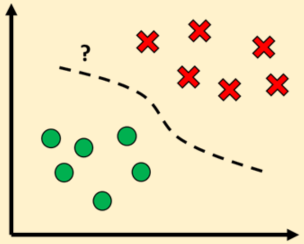
$$y = f(x)$$

from inputs ($x$) to outputs ($y$) , *given a labelled set of input-output examples* ( or ).

**Regression**




**Classification**

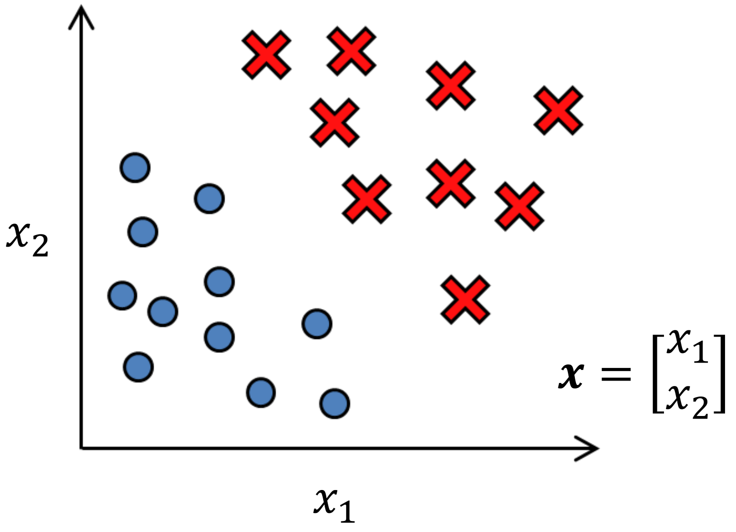



| | Prediction Model | Cost Function |
|---|---|---|
| **Linear Regression** | $y = w_0 + w_1 x$ <br> $y = \boldsymbol{w}^T \boldsymbol{\phi}(x)$ | $\min_{\boldsymbol{w}} \; f(\boldsymbol{w}) = \sum_{i=1}^{N} \left(y_i - \boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)^2$ <br> Sum-of-squares loss |
| **Logistic Regression** | $y = g(w_0 + w_1 x)$ <br> $y = g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x)\right)$ <br> $g(z) = \frac{1}{1 + e^{-z}}$ | $\min_{\boldsymbol{w}} \; f(\boldsymbol{w}) = \sum_{i=1}^{N} -y_i \log\left(g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)\right) - (1 - y_i) \log\left(1 - g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)\right)$ <br> Cross-entropy loss or log-loss |
| **Support Vector Machine** | ??? | ??? |

© 2023 Karl Ezra S. Pilario, PhD.

# Large-margin Classifiers

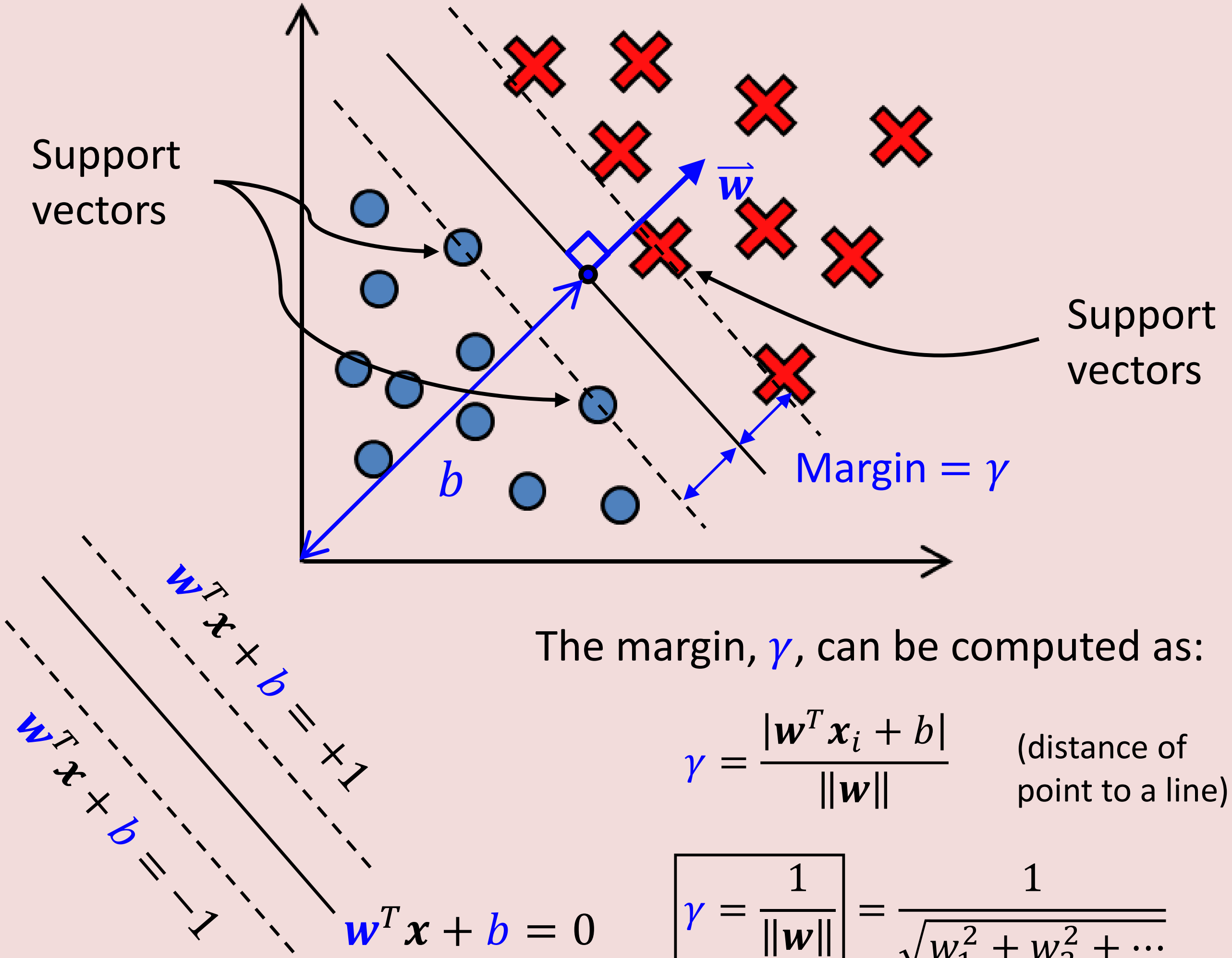
Consider a classification data set whose samples from different classes *can be separated by a line* (linearly separable):




There exists many straight lines that can separate the positive from the negative samples.

What is the equation of the *best-separating line*?

**One possible answer:** The one with the *largest margin for error*.




The margin, $\gamma$, can be computed as:

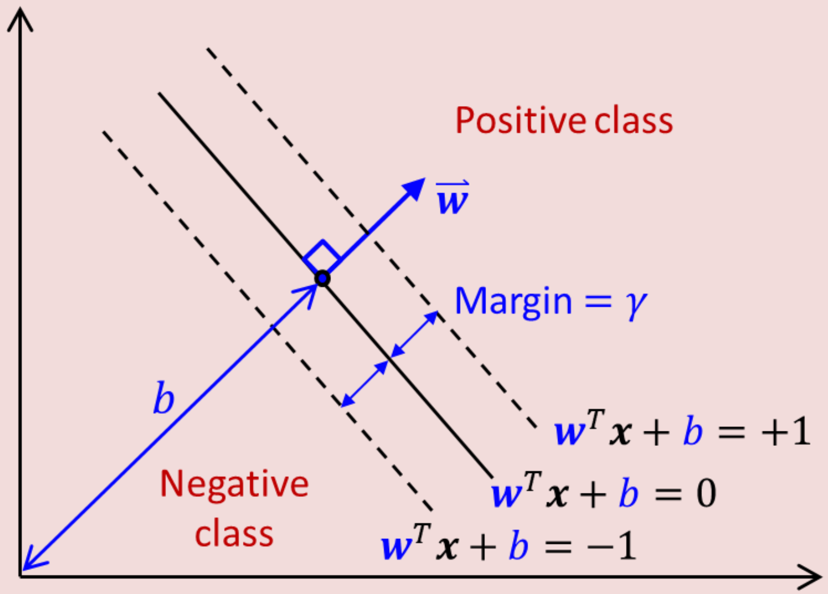
$$\gamma = \frac{|\boldsymbol{w}^T\boldsymbol{x}_i + b|}{\|\boldsymbol{w}\|}$$ (distance of point to a line)

$$\boxed{\gamma = \frac{1}{\|\boldsymbol{w}\|}} = \frac{1}{\sqrt{w_1^2 + w_2^2 + \cdots}}$$

© 2023 Karl Ezra S. Pilario, PhD.

# Large-margin Classifiers

**Large-margin Classifiers:**




Margin: $\boxed{\gamma = \dfrac{1}{\|\boldsymbol{w}\|}} = \dfrac{1}{\sqrt{w_1^2 + w_2^2 + \cdots}}$

If we want to maximize the margin, we can form the following quadratic minimization problem:

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

Subject to the following constraints, for all $i = 1, \dots, N$:

If a sample is from the *positive* class ($y_i = 1$):

$$\boldsymbol{w}^T\boldsymbol{x}_i + b \geq +1$$

If a sample is from the *negative* class ($y_i = -1$):

$$\boldsymbol{w}^T\boldsymbol{x}_i + b \leq -1$$

This solves the maximal margin classifier for the linearly separable case:

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

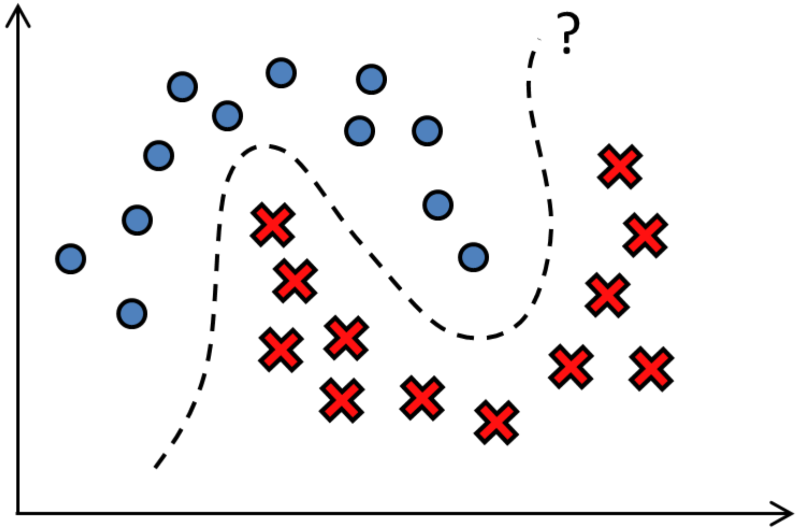
Subject to: $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$

for $i = 1, 2, \dots, N$

© 2023 Karl Ezra S. Pilario, PhD.

# Large-margin Classifiers

How about samples that cannot be separated by a line?




**Kernel functions (or kernels):**

- A *measure of similarity* between a sample $\boldsymbol{x}$ and any point in its surrounding space, $\boldsymbol{x}'$.
- The **linear** kernel is given by:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$$

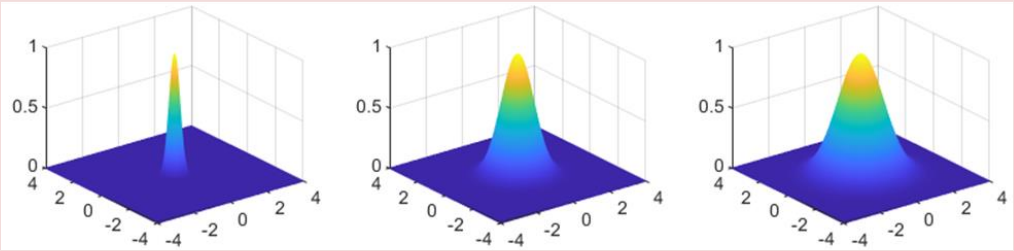
- A commonly used *nonlinear* kernel is the Gaussian radial basis function (**RBF**) kernel:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{\sigma}\right)$$

where $\sigma$ = kernel width

If, say, $\boldsymbol{x} = [0, 0]$:

$\sigma = 0.1$ $\sigma = 1$ $\sigma = 2$




© 2023 Karl Ezra S. Pilario, PhD.

# Large-margin Classifiers

Primal form:

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

Subject to: $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$

for $i = 1, 2, \dots, N$

This optimization problem has a dual form that can be derived using Lagrange multipliers:

Let $\alpha$ be the Lagrange multipliers (or dual variables):

(Eq. 1) $$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \alpha_i[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1]$$

Differentiating with respect to $\boldsymbol{w}$ and $b$:

$$\frac{\partial\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial\boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{N} y_i\alpha_i\boldsymbol{x}_i = 0$$

$$\frac{\partial\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{N} y_i\alpha_i = 0$$

$$\boldsymbol{w} = \sum_{i=1}^{N} y_i\alpha_i\boldsymbol{x}_i \quad \text{(Eq. 2)}$$

$$\sum_{i=1}^{N} y_i\alpha_i = 0 \quad \text{(Eq. 3)}$$

Substituting Eq. 2 and 3 into Eq. 1:

(Eq. 4) $$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j\, \boldsymbol{x}_i^T\boldsymbol{x}_j$$

We can now write the dual form:

Dual form:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j\, \boldsymbol{x}_i^T\boldsymbol{x}_j$$

Subject to: $\sum_{i=1}^{N} y_i\alpha_i = 0$

$\alpha_i \geq 0$, for $i = 1, 2, \dots, N$

The data set $\boldsymbol{x}$ now only appears as a dot product, $\boldsymbol{x}_i^T\boldsymbol{x}_j$. This is the **linear kernel**!

© 2023 Karl Ezra S. Pilario, PhD.

# Support Vector Machines

## Support Vector Machines

**Linear SVM algorithm:**

This is the linear kernel!

Dual form:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

Subject to: $\sum_{i=1}^{N} y_i \alpha_i = 0$

$\alpha_i \geq 0, \quad$ for $i = 1, 2, \ldots, N$

**Kernel SVM algorithm:**

Can be any kernel, e.g. RBF

Find $\boldsymbol{\alpha_i}$ that solves the optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Subject to: $\sum_{i=1}^{N} y_i \alpha_i = 0$

$\alpha_i \geq 0, \quad$ for $i = 1, 2, \ldots, N$

## Mercer's Theorem

A necessary and sufficient condition for a symmetric function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to be a kernel is that it should be *positive definite*.

i.e. For any set of samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l$ and any set of real numbers $\lambda_1, \ldots, \lambda_l$, a positive definite function must satisfy:

$$\sum_{i=1}^{l}\sum_{j=1}^{l} \lambda_i \lambda_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

In that case, any Mercer kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can act as a dot product in nonlinear space.

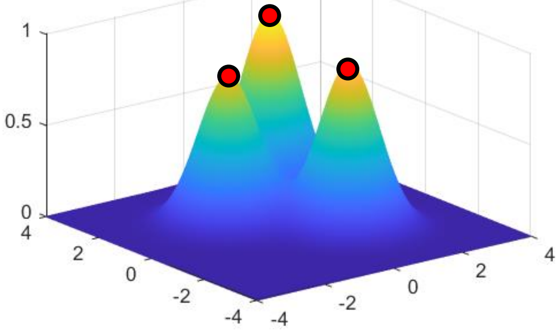
***"kernel trick"***

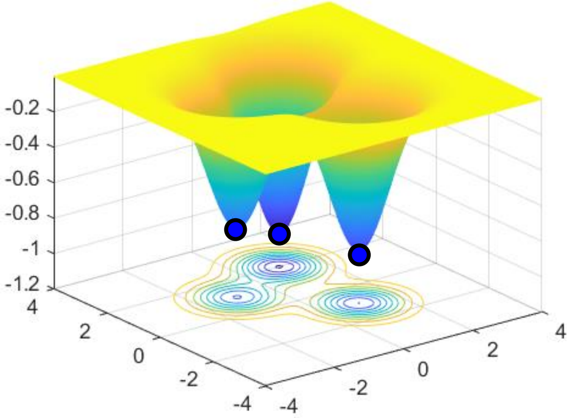
© 2023 Karl Ezra S. Pilario, PhD.

# Support Vector Machines

Why does the **RBF kernel function** lead to nonlinear decision boundaries?

Each point in the positive class is fitted to a “mountain” RBF.

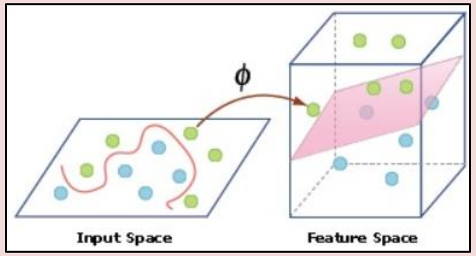




Each point in the negative class is fitted to a “valley” RBF.

**Main idea:**

Kernel functions project the original data onto a *higher-dimensional feature space* where a *linear hyperplane* can more likely separate them (Cover, 1965).

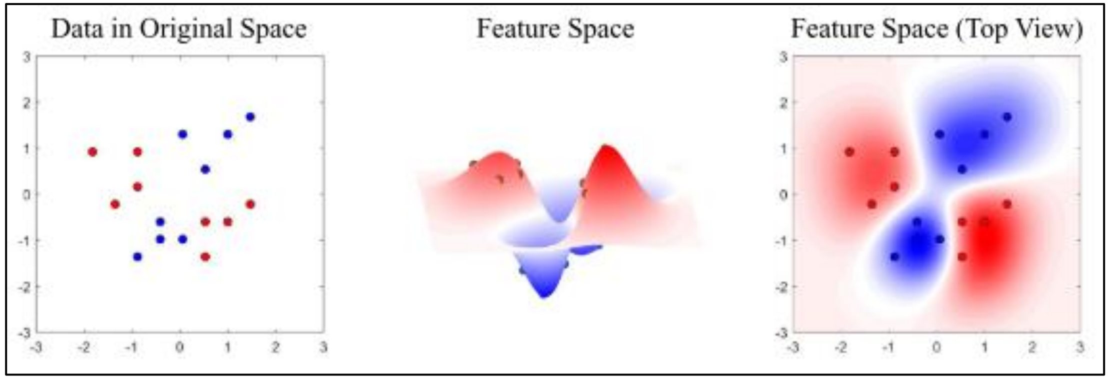


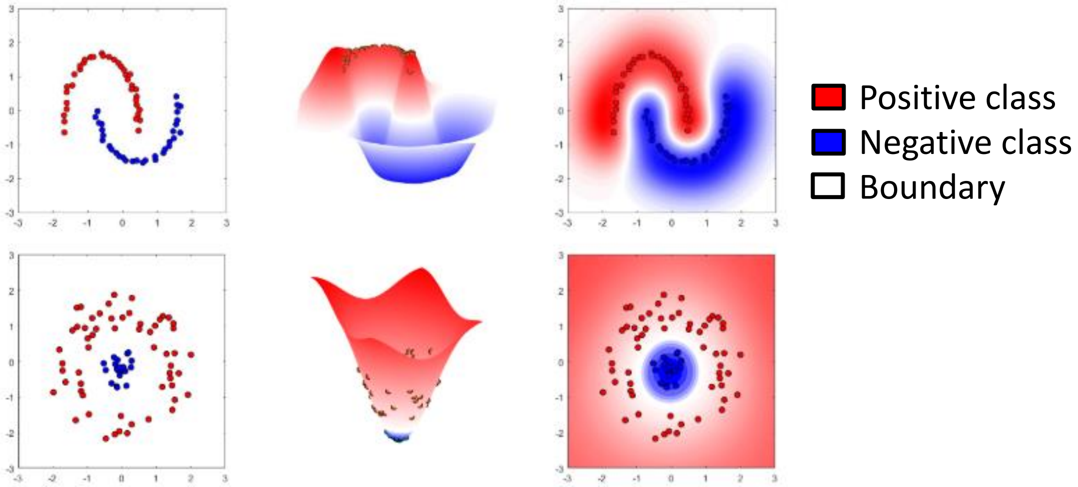

The *prediction model* in SVM is the **SUM** of all the mountains $+\alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}')$ and valleys $-\alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}')$ :

$$f(\boldsymbol{x}) = \sum_{i \in SV} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}') + b^*$$

(Representer Theorem)

*where:*
$\boldsymbol{x}_i$ are the input data
$\boldsymbol{y}_i \in \{-1, +1\}$ are their labels
$\alpha_i^*, b^*$ are SVM parameters
$K(\boldsymbol{x}, \boldsymbol{x}')$ is the kernel
$SV$ = set of support vectors




© 2023 Karl Ezra S. Pilario, PhD.

# Support Vector Machines

| | Prediction Model | Cost Function |
|---|---|---|
| **Linear regression** (**Regression**) | $y = \boldsymbol{w}^T \boldsymbol{\phi}(x_i)$ | $\min_{\boldsymbol{w}} \sum_{i=1}^{N} \left(y_i - \boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)^2$ (sum-of-squares loss) |
| **Logistic regression** (**Classification**) | $y = \dfrac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{\phi}(x_i)}}$ | $\min_{\boldsymbol{w}} \sum_{i=1}^{N} -y_i \log\left(\dfrac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{\phi}(x_i)}}\right) - (1 - y_i)\log\left(1 - \dfrac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{\phi}(x_i)}}\right)$ (cross-entropy loss or log-loss) |
| **Support Vector Machines** (**Classification**) | $y = \sum_{i \in SV} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}') + b^*$ | $\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ (maximum margin or hinge loss) Subject to: $\sum_{i=1}^{N} y_i \alpha_i = 0,\ \alpha_i \geq 0,$ for $i = 1, 2, \dots, N$ |
| **Artificial Neural Networks** | ???? | ???? |
| **Etc…** | | |

© 2023 Karl Ezra S. Pilario, PhD.

# Parametric vs. Non-parametric Models

| | | Parametric Model | Non-Parametric Model |
|---|---|---|---|
| | | • Model parameters ($\boldsymbol{w}$) are visible in the model, and they are explicitly fitted to data.<br>• Number of parameters are fixed with respect to the size of the training set. | • Only dual variables $\alpha_i$ are optimized, but the exact $\boldsymbol{w}$ are unknown.<br>• Number of parameters $\alpha_i$ can grow with the size of the training data set. |
| **Linear regression** (**Regression**) | $y = \boldsymbol{w}^T \boldsymbol{\phi}(x_i)$ | ✓ | |
| **Logistic regression** (**Classification**) | $y = \dfrac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{\phi}(x_i)}}$ | ✓ | |
| **Support Vector Machines** (**Classification**) | $y = \sum_{i \in SV} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}') + b^*$ | | ✓ |

Note: Locally Weighted Linear Regression is also a non-parametric model.

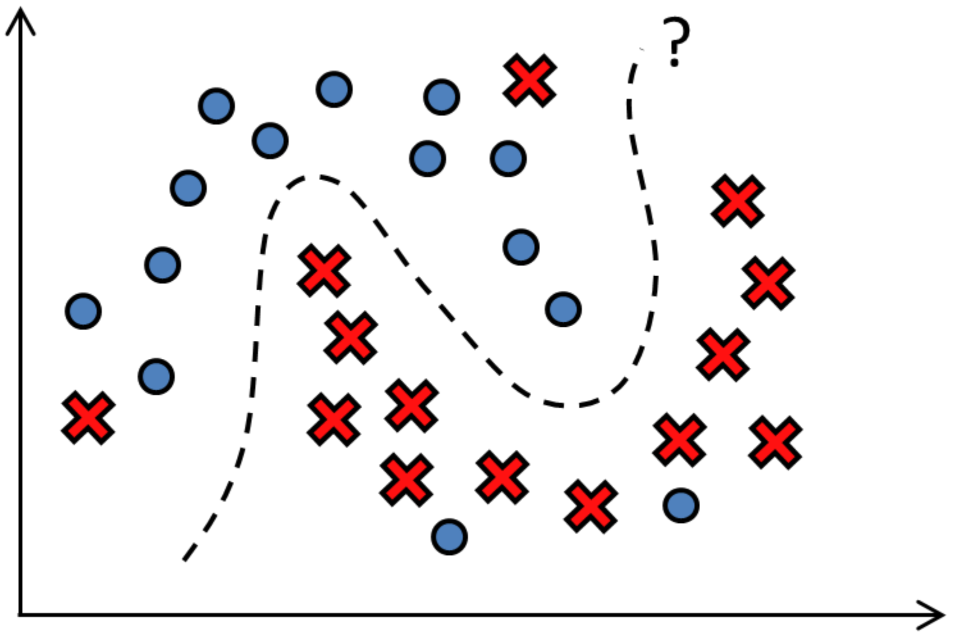
At test time, we need all the training data $\boldsymbol{x}'$ to measure their similarity with test data, $\boldsymbol{x}_i$

© 2023 Karl Ezra S. Pilario, PhD.

# Support Vector Machines

## What about outliers / noise in the data?

Real-world data are **noisy**. If we acknowledge that fact, we must tell the SVM that *a few misclassifications is okay*.




## **Solution:** *Soft-margin SVM*

The 1-norm C-SVM employs a **box constraint,** $C$, on the values of $\alpha_i$ in the optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Subject to: $\sum_{i=1}^{N} y_i \alpha_i = 0$

$$0 \leq \alpha_i \leq C, \quad \text{for } i = 1, 2, \dots, N$$

The $C$ is a parameter that can be adjusted:

- If $C = \infty$, then hard-margin SVM is employed:
  no misclassifications are tolerated.
- If $C < \infty$, then soft-margin SVM is employed:
  the smaller the $C$, the more misclassifications are *tolerated*.

© 2023 Karl Ezra S. Pilario, PhD.

# Support Vector Machines

## **Support Vector Machine:** Summary

**Given:** Training Data, $\{\boldsymbol{x}_i, y_i\}$, $y_i \in \{-1,1\}$, $i = 1, \dots, N$

**Initialization:** Set the value of box constraint, $C$
Set a kernel function (e.g. RBF)
Set the kernel parameter value, $\sigma$

**1:** Solve the maximization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Subject to: $\sum_{i=1}^{N} y_i \alpha_i = 0$

$$0 \leq \alpha_i \leq C, \quad \text{for } i = 1, 2, \dots, N$$

**2:** Compute the bias, $b^*$, such that on average, $y_i[\alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}') + b^*] = 1$.

**3:** Return the classifier: $y_{\text{pred}} = \sum_{i \in SV} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}') + b^*$
Decision rule: If $y_{pred} > 0$, predict positive class.
If $y_{\text{pred}} < 0$, predict negative class.

In Python Scikit-learn, the **C-SVM** is implemented as sklearn.svm.**SVC**, with the following *defaults*:

- Box constraint, $C = 1$
- Kernel = RBF (Radial Basis Function)

  List of possible kernels:

  - linear: $\langle x, x' \rangle$.
  - polynomial: $(\gamma\langle x, x'\rangle + r)^d$, where $d$ is specified by parameter `degree`, $r$ by `coef0`.
  - rbf: $\exp(-\gamma\|x - x'\|^2)$, where $\gamma$ is specified by parameter `gamma`, must be greater than 0.
  - sigmoid $\tanh(\gamma\langle x, x'\rangle + r)$, where $r$ is specified by `coef0`.

- Gamma, $\gamma = \dfrac{1}{\text{no. of features} \times var(\boldsymbol{x})}$

Other implementations of SVM in Scikit-learn are sklearn.svm.**NuSVC** and sklean.svm.**LinearSVC**.

The maximization problem in SVMs is mainly solved using **SQP** (sequential quadratic programming) as implemented in `libsvm`.

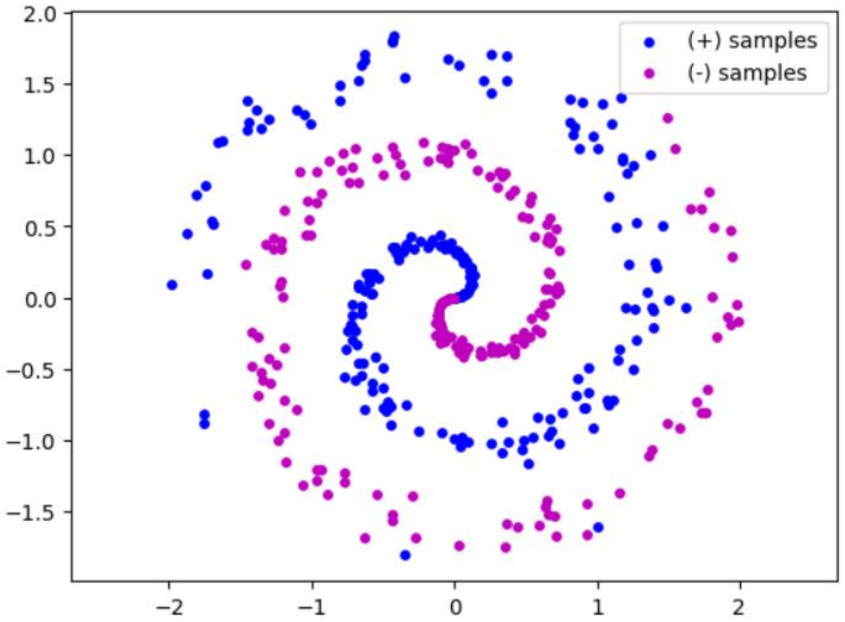
© 2023 Karl Ezra S. Pilario, PhD.

# Support Vector Machines

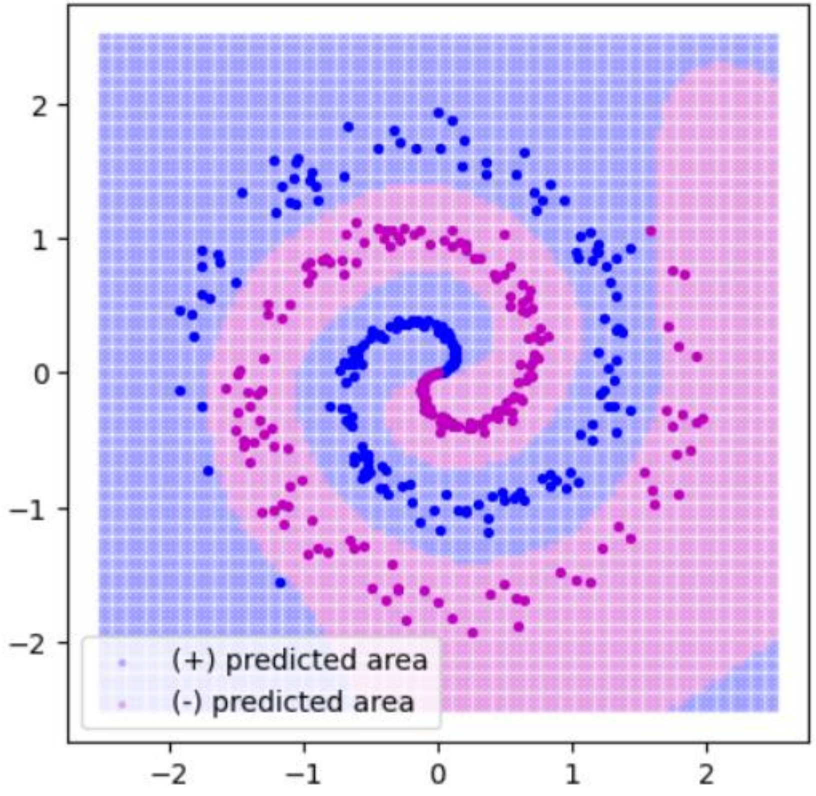
**Example 1:** Spiral Data Set

Find a maximal margin boundary between the blue and red points in the spiral data set below. Use an RBF kernel with a width of 1 and C = 5.

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{1.0}\right)$$




**Answer:**




$$f(\boldsymbol{x}) = \sum_{i \in SV} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}') + b^*$$

*where:*
$\boldsymbol{x}_i$ are the input data
$\boldsymbol{y}_i \in \{-1, +1\}$ are their labels
$\alpha_i^*, b^*$ are SVM parameters
$K(\boldsymbol{x}, \boldsymbol{x}')$ is the kernel
*SV* = set of support vectors

© 2023 Karl Ezra S. Pilario, PhD.

# Underfitting vs. Overfitting

## Underfitting

- Performance is too low on the training data set.
- Patterns were not learned at all.
- The model is too simple (high bias, low variance).
- The model cannot generalize to unseen test data.

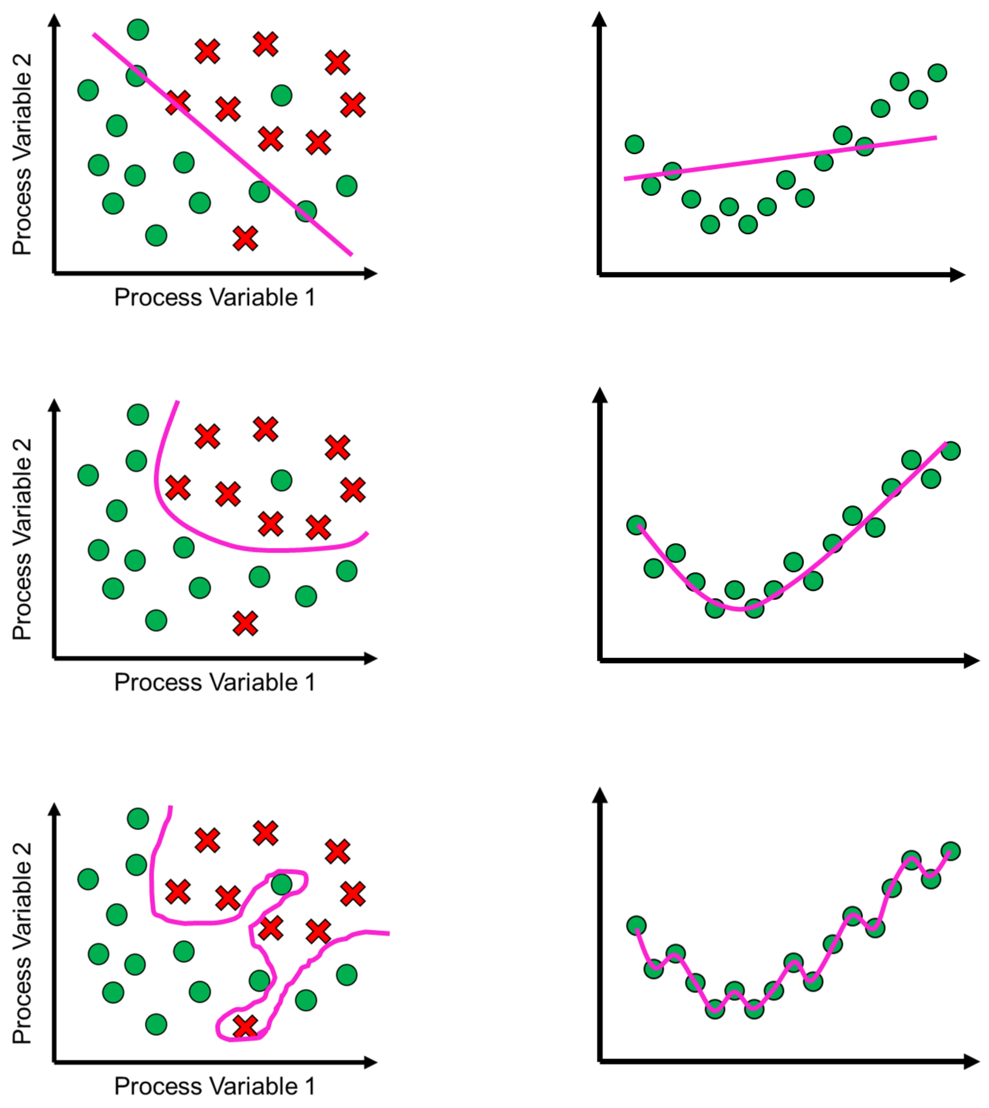
## Good fit

- Performance on training data is decent.
- The pattern was learned.
- Bias and variance are balanced.
- Model is not too complex nor simple.
- The model *may* generalize well to unseen test data.

## Overfitting

- Performance is too good to be true on the training data set.
- Patterns "learned" from noise.
- The model is too complex (low bias, high variance).
- The model cannot generalize to unseen test data.

**Classification** | **Regression**




© 2023 Karl Ezra S. Pilario, PhD.

# Multiple Kernel Learning

- The name "kernel" is derived from *integral operator theory* in math.
- There are many kernel functions, other than the Gaussian RBF.
- One can create new Mercer kernels by combining base Mercer kernels.

**Common kernels:**

Gaussian RBF

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{\sigma}\right)$$

Sigmoid kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tanh(a\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + b)$$

Polynomial kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1)^d$$

Linear kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$$

**Creating kernels:** If $K_1(\boldsymbol{x}, \boldsymbol{x}')$ and $K_2(\boldsymbol{x}, \boldsymbol{x}')$ are valid Mercer kernels, then the following functions are also valid Mercer kernels.

- $K(\boldsymbol{x}, \boldsymbol{x}') = K_1(\boldsymbol{x}, \boldsymbol{x}') + K_2(\boldsymbol{x}, \boldsymbol{x}')$
- $K(\boldsymbol{x}, \boldsymbol{x}') = K_1(\boldsymbol{x}, \boldsymbol{x}')K_2(\boldsymbol{x}, \boldsymbol{x}')$
- $K(\boldsymbol{x}, \boldsymbol{x}') = aK_1(\boldsymbol{x}, \boldsymbol{x}')$
- $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\big(K(\boldsymbol{x}, \boldsymbol{x}')\big)$
- $K(\boldsymbol{x}, \boldsymbol{x}') = p\big(K(\boldsymbol{x}, \boldsymbol{x}')\big)$, $p$ is a polynomial with non-negative coefficients.

**Related readings:**
- Multiple kernel learning
- Compositional kernel
- Mixed kernels
- Hierarchical kernels
- Neural kernel blocks

**Other kernels in other fields:**

- String kernels are used in text mining and bioinformatics (gene analysis).
- Matern kernels are used for Gaussian process regression / kriging in geostatistics.
- Clinical kernels are used in analyzing clinical data / survival analysis in medicine.

© 2023 Karl Ezra S. Pilario, PhD.

# Outline

- Support Vector Machines
  - Large-Margin Classifiers
  - Quadratic Optimization Problem
  - Nonlinear SVM using Kernels
  - Mercer’s Theorem
  - Multiple Kernel Learning

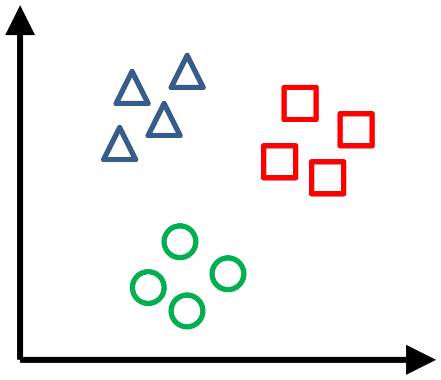
- **Multi-class Classification**

- Kernel Methods for Regression
  - Support Vector Regression
  - Kernel Ridge Regression

© 2023 Karl Ezra S. Pilario, PhD.

# Multi-class Classification

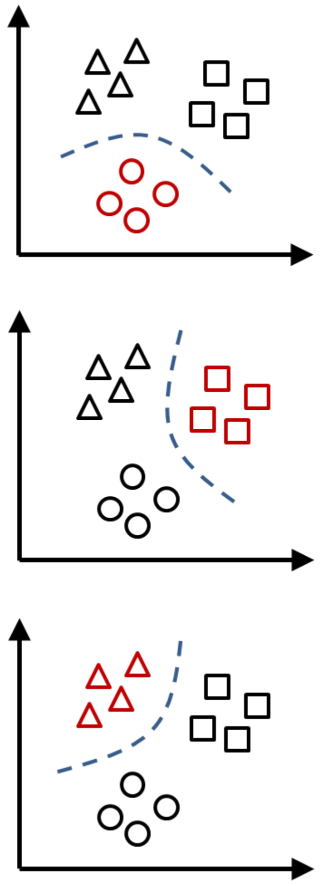
Sample data set:



## One-vs.-Rest

(OvR, One-vs.-All, 1-v-R)

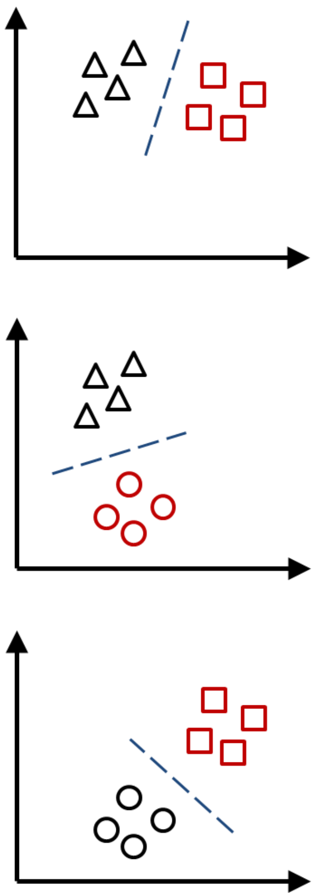
Given $K$ classes, train $K$ binary classifiers, each one treating one class as positive and the rest as negative.



## One-vs.-One

(OvO, 1-v-1)

Given $K$ classes, train $K(K-1)/2$ binary classifiers, one for each pair of classes. Predict by majority voting.

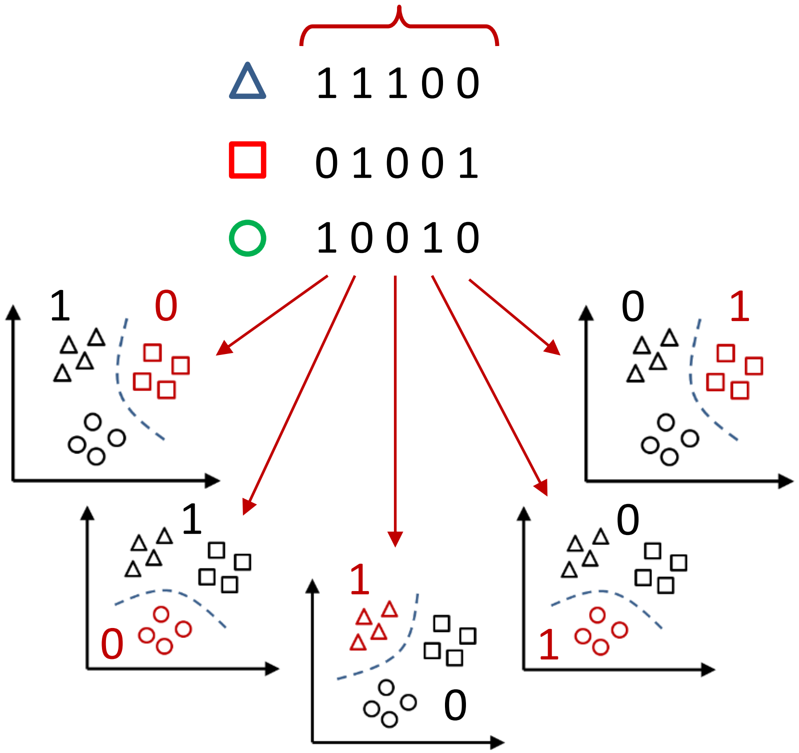


## Output Codes

(Error-correcting output codes, ECOC)

Given $K$ classes, assign a unique binary codeword for each class, then train a binary classifier for each bit position in the codeword. Predict by generating the output codeword of a new sample, then report the class with the closest codeword to it.

e.g. 5 classifiers




**Note:** The default multi-class strategy of sklearn.svm.SVC is OvO.

© 2023 Karl Ezra S. Pilario, PhD.

# Multi-class Classification

## **Example 2:** Iris Flowers Data Set

- 150 flowers from 3 different Iris species (50 each)
- Measurements of sepal and petal lengths and widths.

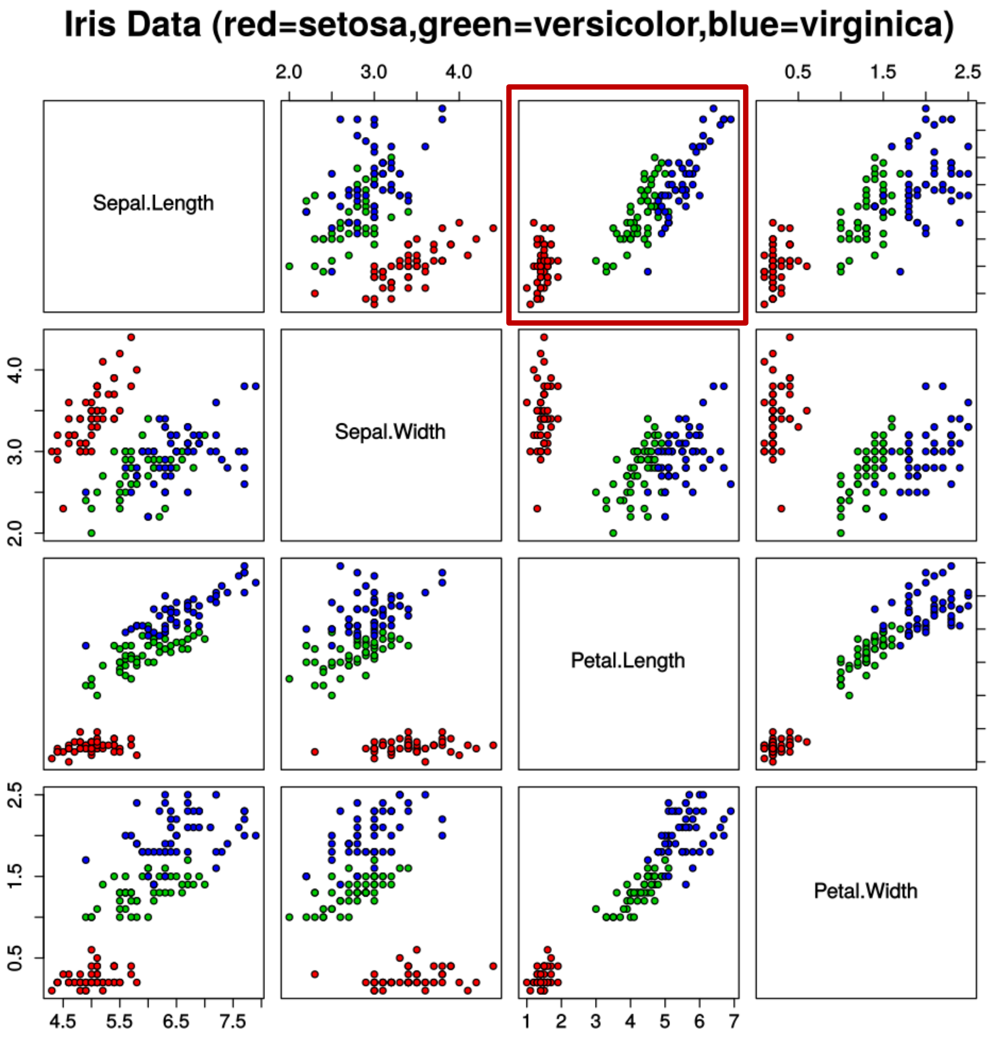
Using only the sepal length and petal length features, train an SVM to classify all 150 flowers into their correct species.

Compare the results between the OvO, OvR, and ECOC strategies.

Use default SVM settings. For ECOC, use codewords of length 9.

| Sepal Length | Sepal Width | Petal Length | Petal Width | Classification |
|---|---|---|---|---|
| cm | cm | cm | cm | |
| 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 4.9 | 3 | 1.4 | 0.2 | Iris-setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| … | … | … | … | … |
| | | | | |
| 7 | 3.2 | 4.7 | 1.4 | Iris-versicolor |
| 6.4 | 3.2 | 4.5 | 1.5 | Iris-versicolor |
| 6.9 | 3.1 | 4.9 | 1.5 | Iris-versicolor |
| … | … | … | … | … |
| | | | | |
| 6.3 | 3.3 | 6 | 2.5 | Iris-virginica |
| 5.8 | 2.7 | 5.1 | 1.9 | Iris-virginica |
| 7.1 | 3 | 5.9 | 2.1 | Iris-virginica |
| … | … | … | … | … |
| | | | | |


Versicolor Setosa Virginica




© 2023 Karl Ezra S. Pilario, PhD.

# Multi-class Classification

## **Example 2:** Iris Flowers Data Set

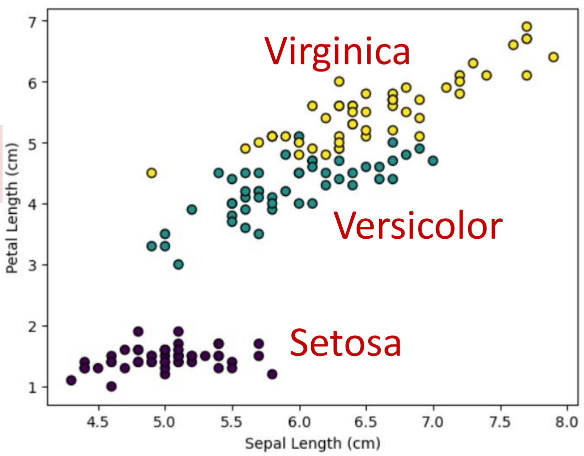
- 150 flowers from 3 different Iris species (50 each)
- Measurements of sepal and petal lengths and widths.

Using only the sepal length and petal length features, train an SVM to classify all 150 flowers into their correct species.

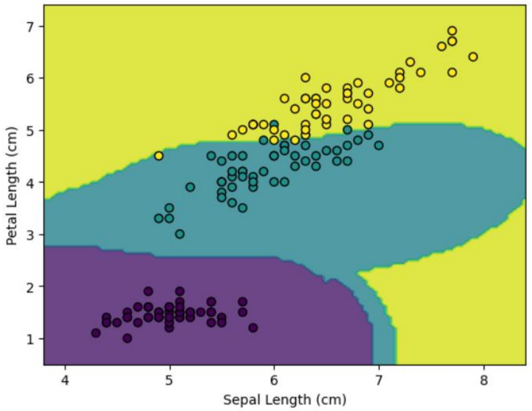
Compare the results between the OvO, OvR, and ECOC strategies.

Use default SVM settings. For ECOC, use codewords of length 9.

**Data Set:**




### OvO Strategy

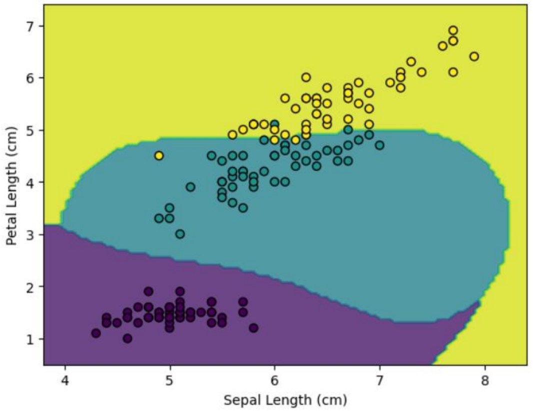



Actual

| | | |
|---|---|---|
| 50 | 0 | 0 |
| 0 | 47 | 3 |
| 0 | 3 | 47 |

Predicted

### OvR Strategy

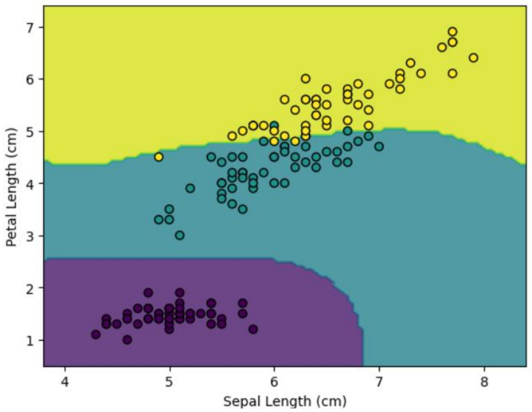



Actual

| | | |
|---|---|---|
| 50 | 0 | 0 |
| 0 | 48 | 2 |
| 0 | 4 | 46 |

Predicted

### ECOC Strategy




Actual

| | | |
|---|---|---|
| 50 | 0 | 0 |
| 0 | 48 | 2 |
| 0 | 4 | 46 |

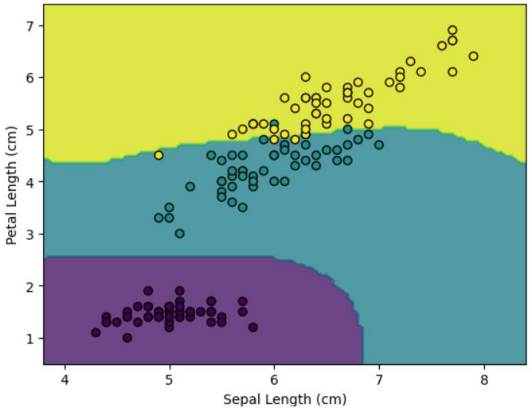
Predicted

© 2023 Karl Ezra S. Pilario, PhD.

# Multi-class Classification

## **Example 2:** Iris Flowers Data Set

### ECOC Strategy




Actual

| | setosa | versicolor | virginica |
|---|---|---|---|
| setosa | 50 | 0 | 0 |
| versicolor | 0 | 48 | 2 |
| virginica | 0 | 4 | 46 |

Predicted

### Performance Metrics for Multi-class Classification

**Classification Report:**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| setosa | 1.00 | 1.00 | 1.00 | 50 |
| versicolor | 0.92 | 0.96 | 0.94 | 50 |
| virginica | 0.96 | 0.92 | 0.94 | 50 |
| accuracy | | | 0.96 | 150 |
| macro avg | 0.96 | 0.96 | 0.96 | 150 |
| weighted avg | 0.96 | 0.96 | 0.96 | 150 |

**Sample calculations:**

Precision (Versicolor) $= \frac{48}{48+4} = 0.92$

| | | |
|---|---|---|
| 50 | 0 | 0 |
| 0 | 48 | 2 |
| 0 | 4 | 46 |

Recall (Versicolor) $= \frac{48}{48+2} = 0.96$

| | | |
|---|---|---|
| 50 | 0 | 0 |
| 0 | 48 | 2 |
| 0 | 4 | 46 |

**Overall Metrics**

- Macro Average (mean of metrics)
- Weighted Average (mean weighted by number of samples)

© 2023 Karl Ezra S. Pilario, PhD.

# Multi-class, multi-label, multi-output

In Python Scikit-learn, there are other kinds of classification learning tasks available:

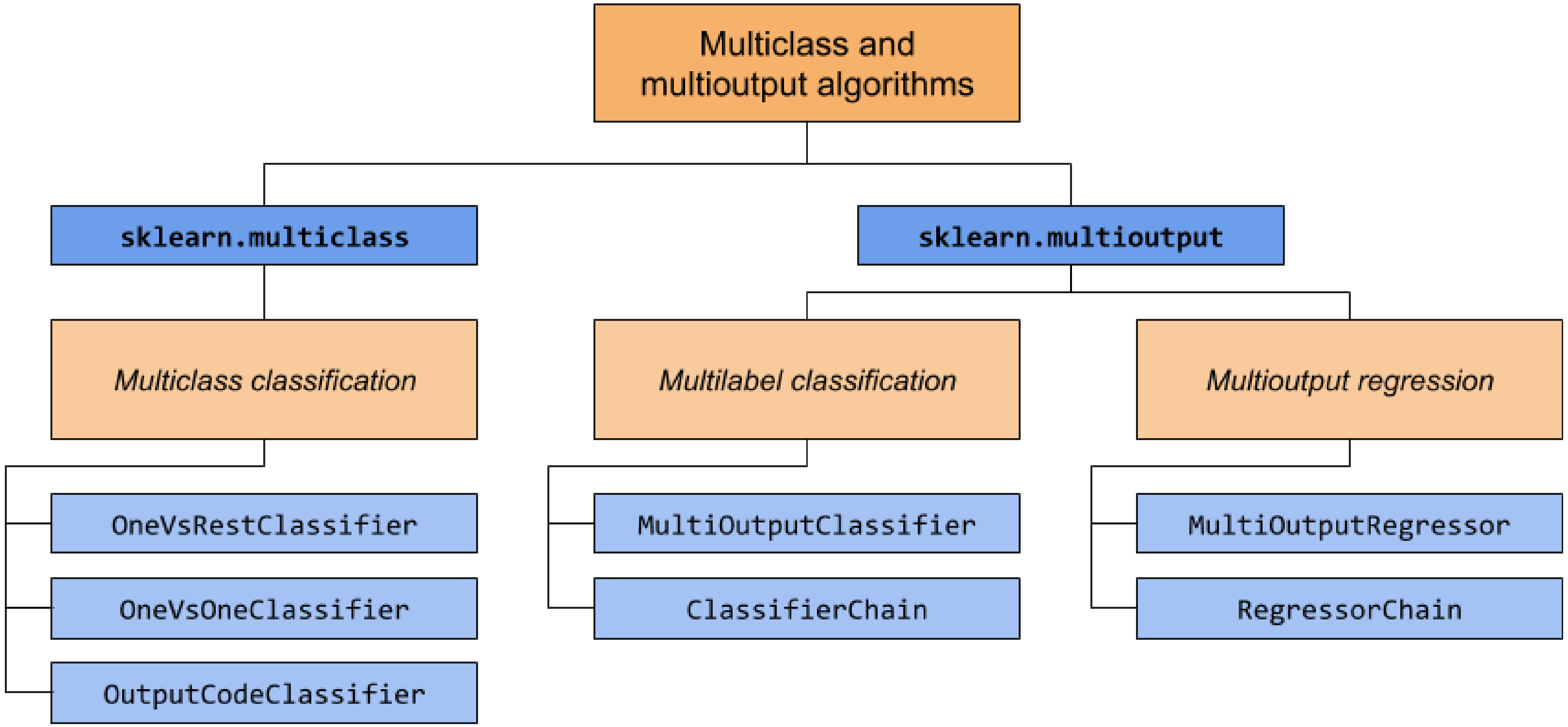
- **Multi-class:** Two or more classes are available; each sample is labeled to one and only one class.
- **Multi-label:** Two or more classes are available; a sample can be labelled to one or more classes at once.
- **Multi-class, Multi-output:** Two or more classes are available;
  A model can predict multiple outputs for one sample simultaneously.

Multi-class, multi-output classification is also known as **multi-task classification**.

One-vs.-Rest

One-vs.-One

Output Codes

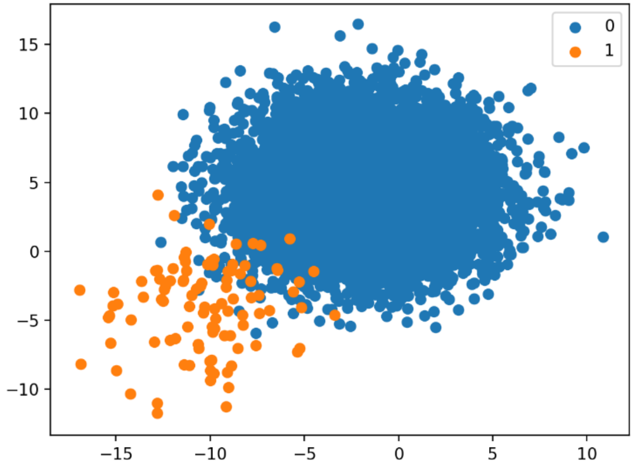



© 2023 Karl Ezra S. Pilario, PhD.

# What about class imbalance?

Class imbalance occurs when classes are not approximately equally represented.

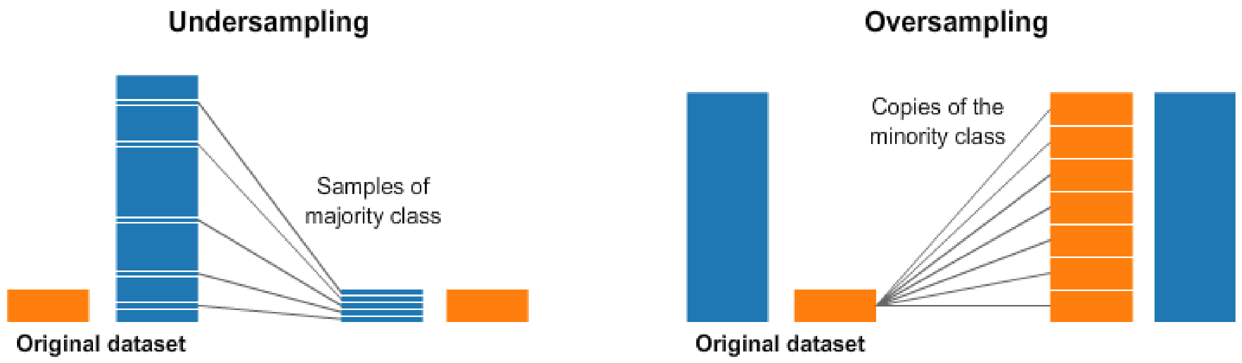
The majority class becomes easy to predict, while the minority class becomes hard.



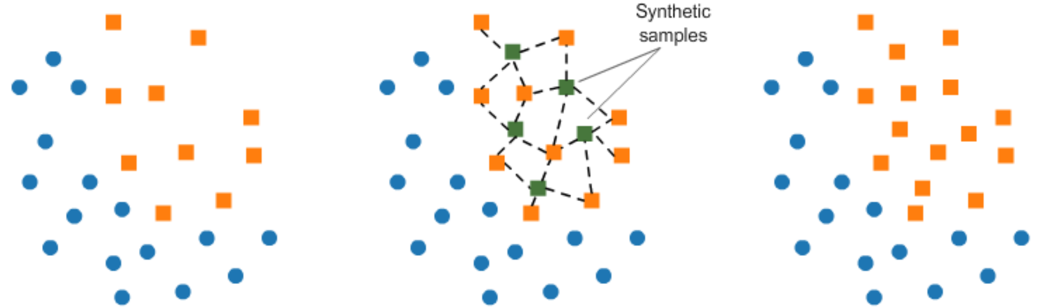

**Examples:**

- Some images are rare.
- There are only a few fraudulent checks.
- There are only a few faulty equipment data.
- There are more SFW content than NSFW content.
- Some words appear more often than others.

**Solution 1:** Random undersampling and oversampling




**Solution 2:** SMOTE (Synthetic Minority Oversampling Technique)




Other solutions:

- **Do nothing.**
- Collect more data.
- Treat the problem as anomaly detection rather than classification.
- Include *class weights* in the classifier's cost function.
- Consider other evaluation metrics, like F1-score.

© 2023 Karl Ezra S. Pilario, PhD.

# Outline

- Support Vector Machines
  - Large-Margin Classifiers
  - Quadratic Optimization Problem
  - Nonlinear SVM using Kernels
  - Mercer’s Theorem
  - Multiple Kernel Learning

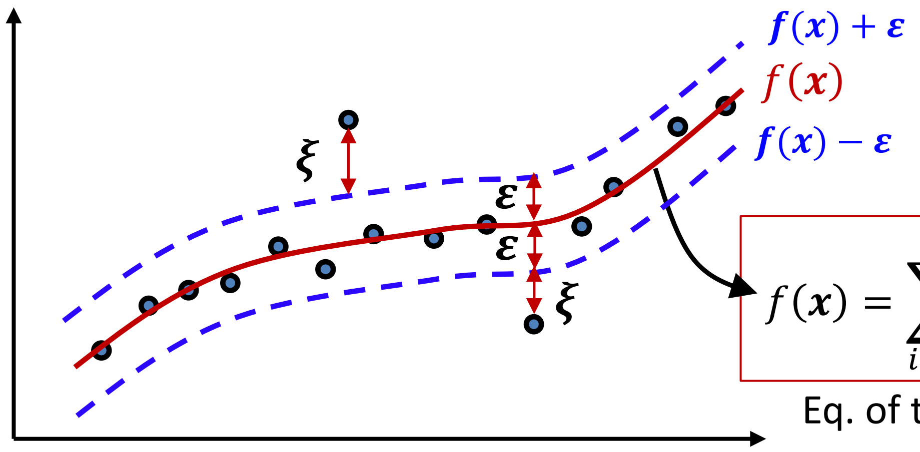
- Multi-class Classification

- **Kernel Methods for Regression**
  - Support Vector Regression
  - Kernel Ridge Regression

© 2023 Karl Ezra S. Pilario, PhD.

# Support Vector Regression

- Support Vector Machines can also be formulated to solve regression instead of classification.
- Large-margin intuition →$\varepsilon$-insensitive cost function ($\varepsilon$ is 'epsilon')




$$\boxed{f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}') + b^*}$$

Eq. of the best-fit curve

In Support Vector Regression, a certain subset of points are also support vectors.

## **SVR:** Dual Form of the Optimization Problem

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} y_i \alpha_i - \varepsilon \sum_{i=1}^{N} |\alpha_i| - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Subject to: $\sum_{i=1}^{N} \alpha_i = 0$

$$-C \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, N$$

This optimization problem was derived from minimizing the *linear* $\boldsymbol{\varepsilon}$*-insensitive loss / cost function*:

$$L_\varepsilon = \begin{cases} 0 & \text{if } |y - f(x)| \leq \varepsilon \\ |y - f(x)| - \varepsilon & \text{otherwise} \end{cases}$$

A more compact way to write it:

$$L_\varepsilon = \max(0, |y - f(x)| - \varepsilon)$$

The *linear* $\boldsymbol{\varepsilon}$*-insensitive cost function* <u>ignores errors that are within $\boldsymbol{\varepsilon}$ distance of the observed value</u> by treating them as equal to zero.

© 2023 Karl Ezra S. Pilario, PhD.

# Kernel Ridge Regression

## From Linear Ridge Regression to Kernel Ridge Regression:

**Linear Basis Function Model:**

$$y = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \cdots + w_m\phi_m(x)$$

$$y = \boldsymbol{w}^T\boldsymbol{\phi}(x) = \boldsymbol{\phi}(x)^T\boldsymbol{w}$$

By minimizing the cost function with penalty:

$$\min_{\boldsymbol{w}} \ (\boldsymbol{y} - \boldsymbol{\Phi w})^T(\boldsymbol{y} - \boldsymbol{\Phi w}) + \lambda(\boldsymbol{w}^T\boldsymbol{w})$$

The solution is given by:

$$\boxed{\widehat{\boldsymbol{w}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}}$$

Substituting the $\widehat{\boldsymbol{w}}$ into the model:

$$y_{\text{pred}} = \boldsymbol{\phi}(x)^T\left((\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}\right)$$

Using the Woodbury matrix identity:

$$y_{\text{pred}} = \boldsymbol{\phi}(x)^T\boldsymbol{\Phi}(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}$$

Applying the kernel trick:

$$K(\boldsymbol{x}', \boldsymbol{x}') = \boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$$

Kernel matrix or Gram matrix

$$k(x, \boldsymbol{x}') = \boldsymbol{k}(x) = \boldsymbol{\phi}(x)^T\boldsymbol{\Phi}$$

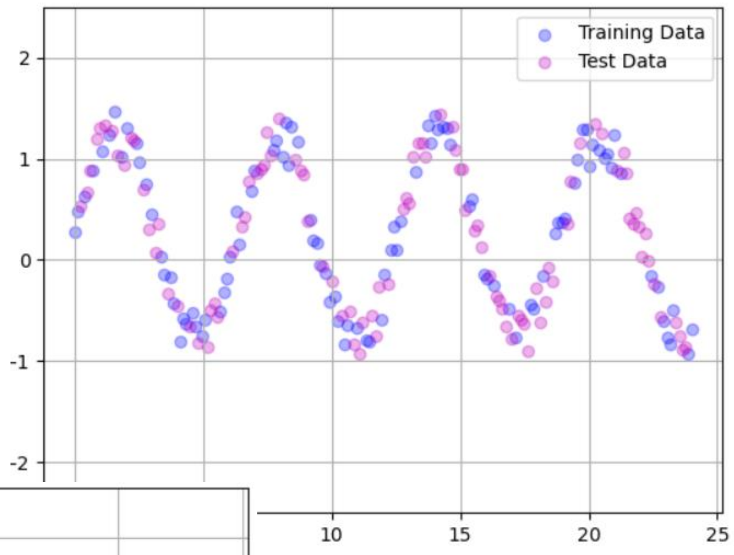
We have now derived Kernel Ridge Regression:

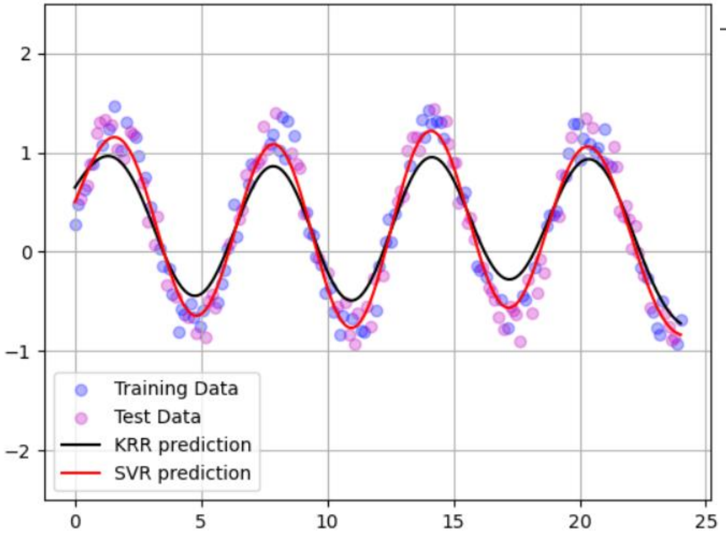
$$\boxed{y_{\text{pred}} = \boldsymbol{k}(x)^T(\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}}$$

## **Example 3:** Sine Data Set (again)

Train an SVR and a KRR model on the blue training data, then predict the targets on the pink test data. Use RBF kernel on both with gamma=5 and all other defaults retained.

Report the training and testing MSE for both.




KRR, $R^2$ (Training): 0.86725
KRR, $R^2$ (Test): 0.85191
SVR, $R^2$ (Training): 0.95738
SVR, $R^2$ (Test): 0.94552

© 2023 Karl Ezra S. Pilario, PhD.

# A closer look at the RBF kernel…

We learned that the reason why the **kernel trick** works is that the kernel function acts as a dot product of some feature map $\boldsymbol{\phi}(\boldsymbol{x})$ of the input $\boldsymbol{x}$:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}) \rangle$$

So for the RBF kernel, what does the $\boldsymbol{\phi}(\boldsymbol{x})$ correspond to?

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{\sigma}\right) = \langle\, ?\,, ?\, \rangle$$

**Derivation:**

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2}\right) = \exp\left(-\frac{1}{2}(\|\boldsymbol{x}\|^2 + \|\boldsymbol{x}'\|^2 - 2\langle \boldsymbol{x}, \boldsymbol{x}' \rangle)\right)$$

$$= \exp\left(-\frac{1}{2}(\|\boldsymbol{x}\|^2 + \|\boldsymbol{x}'\|^2 - 2\langle \boldsymbol{x}, \boldsymbol{x}' \rangle)\right)$$

$$= \exp\left(-\frac{1}{2}(\|\boldsymbol{x}\|^2 + \|\boldsymbol{x}'\|^2)\right) \exp\left(-\frac{1}{2}(-2\langle \boldsymbol{x}, \boldsymbol{x}' \rangle)\right)$$

$$= c \exp\left(-\frac{1}{2}(-2\langle \boldsymbol{x}, \boldsymbol{x}' \rangle)\right) = c \exp(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle)$$

$$= c \sum_{n=0}^{\infty} \frac{\langle \boldsymbol{x}, \boldsymbol{x}' \rangle^n}{n!}$$

Since the Gaussian RBF kernel can be expressed as a power series, it corresponds to an infinite-dimensional feature map, $\boldsymbol{\phi}(\boldsymbol{x}) \in \Re^{\infty}$. More formally, the underlying feature map is a **Reproducing Kernel Hilbert Space (RKHS)**.

**Smaller example:** Given $\boldsymbol{x} = [x_1, x_2]$, the quadratic kernel $(\boldsymbol{x} \cdot \boldsymbol{x}')^2$ can be derived from the feature map:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} x_1^2, & x_2^2, & \sqrt{2}\, x_1 x_2 \end{bmatrix}.$$

**Proof:** $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}) \boldsymbol{\phi}(\boldsymbol{x}')^T$

$$= \begin{bmatrix} x_1^2, & x_2^2, & \sqrt{2}\, x_1 x_2 \end{bmatrix} \begin{bmatrix} {x_1'}^2 \\ {x_2'}^2 \\ \sqrt{2} x_1' x_2' \end{bmatrix}$$

$$= x_1^2 {x_1'}^2 + x_2^2 {x_2'}^2 + 2 x_1 x_2 x_1' x_2'$$

$$= (x_1 x_1' + x_2 x_2')^2$$

$$= \left( \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \right)^2 = \boxed{(\boldsymbol{x} \cdot \boldsymbol{x}')^2}$$

Quadratic kernel

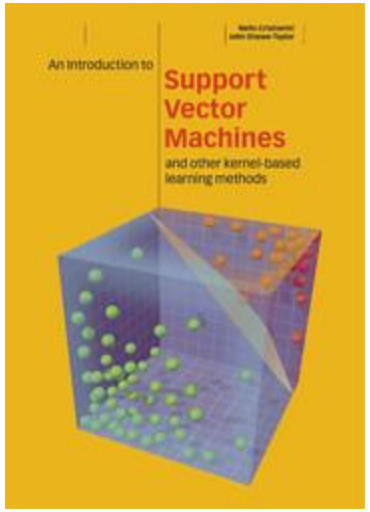
© 2023 Karl Ezra S. Pilario, PhD.

# Outline

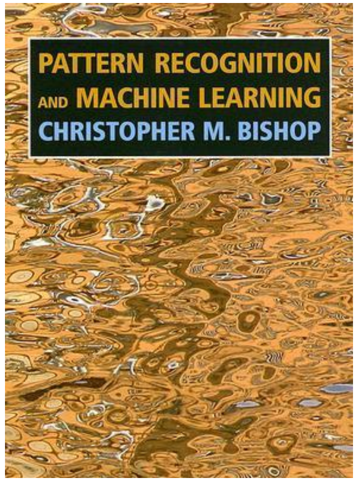
- Support Vector Machines
  - Large-Margin Classifiers
  - Quadratic Optimization Problem
  - Nonlinear SVM using Kernels
  - Mercer's Theorem
  - Multiple Kernel Learning

- Multi-class Classification

- Kernel Methods for Regression
  - Support Vector Regression
  - Kernel Ridge Regression




Cristianini & Shawe-Taylor (2000)
*An Introduction to Support Vector Machines and other kernel-based learning methods.*
Cambridge University Press.




Bishop (2006)
*Pattern Recognition and Machine Learning.* Springer.

Other Kernel Machines you may be interested in:

- Least-squares SVM (LSSVM)
- Relevance Vector Machine (RVM)
- Core Vector Machines (CVM)
- Twin Support Vector Machines
- One-class SVM

© 2023 Karl Ezra S. Pilario, PhD.

# Further Reading


- Cristianini & Shawe-Taylor (2000). *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press.
- Bishop (2006). *Pattern Recognition and Machine Learning*. Springer.
- Cortes and Vapnik (1995). Support-Vector Networks. *Machine Learning*, vol. 20, pp. 273-297.
- Andrew Ng CS 229 ML: https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU
- Multi-class, Multi-label, Multi-output: https://scikit-learn.org/stable/modules/multiclass.html
- ECOC paper: https://arxiv.org/pdf/cs/9501101.pdf
- SMOTE paper: https://arxiv.org/pdf/1106.1813.pdf
- https://www.kdnuggets.com/2020/01/5-most-useful-techniques-handle-imbalanced-datasets.html
- KRR vs. SVR: https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_kernel_ridge_regression.html
- K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf (2001). “An introduction to kernel-based learning algorithms”, IEEE Transactions on Neural Networks, 12(2), 181-201. https://ieeexplore.ieee.org/document/914517
- Kernel Cookbook: https://www.cs.toronto.edu/~duvenaud/cookbook/
- Gonen and Alpaydin (2011). Multiple Kernel Learning Algorithms. Journal of Machine Learning Research, 12. https://jmlr.org/papers/volume12/gonen11a/gonen11a.pdf
- RKHS: https://ngilshie.github.io/jekyll/update/2018/02/01/RKHS.html
- Hofmann (2008). Kernel Methods in Machine Learning: https://arxiv.org/pdf/math/0701907.pdf ← More theory on Kernel Machines
- Tipping (1999). Relevance Vector Machines. https://proceedings.neurips.cc/paper/1999/file/f3144cefe89a60d6a1afaf7859c5076b-Paper.pdf


© 2023 Karl Ezra S. Pilario, PhD.