Supervised Learning: Classification with Support Vector Machine (SVM)

Dr. Yuzana Win (Nagasaki University, Japan)
Lecturer
Department of Computer Engineering and Information Technology

Think smart, be different.

Lecture Objectives

- To introduce
  - What is Support Vector Machine (SVM)?
  - How does the SVM works?
  - Advantages and Disadvantages

Classification Techniques or Methods

- Naïve Bayes Classifier *
- Support Vector Machines *
- Neural Networks (Deep Learning)
- Decision Tree based Methods
- Rule-based Methods
- K-Nearest Neighbors *

What is Support Vector Machine (SVM)?

- SVM is a set of supervised learning methods used for classification, regression and outliers detection
- It is known for its kernel trick to handle nonlinear input spaces
- It is used in a variety of applications
  - Face detection
  - Intrusion detection
  - Classification of emails
  - News articles and web pages
  - Classification of genes
  - Handwriting recognition

CEIT
Think smart, be different.

Linear Model

- Linear function

$$f_{\mathbf{w}}(\boldsymbol{x}) = b + w_1 x_1 + \cdots + w_d x_d$$

f(*x*) is can be written as $w^T x$

$w = b, w_1, \ldots, w_d$

$x = 1, x_1, \ldots, x_d$

- Linear model is a model that using a linear function
- It is a simple model but it works pretty as well
- Search for a good model = find the $(b, w_1, \ldots, w_d)$ that matches the problem you want to solve
  - $(b, w_1, \ldots, w_d)$ are called model parameters

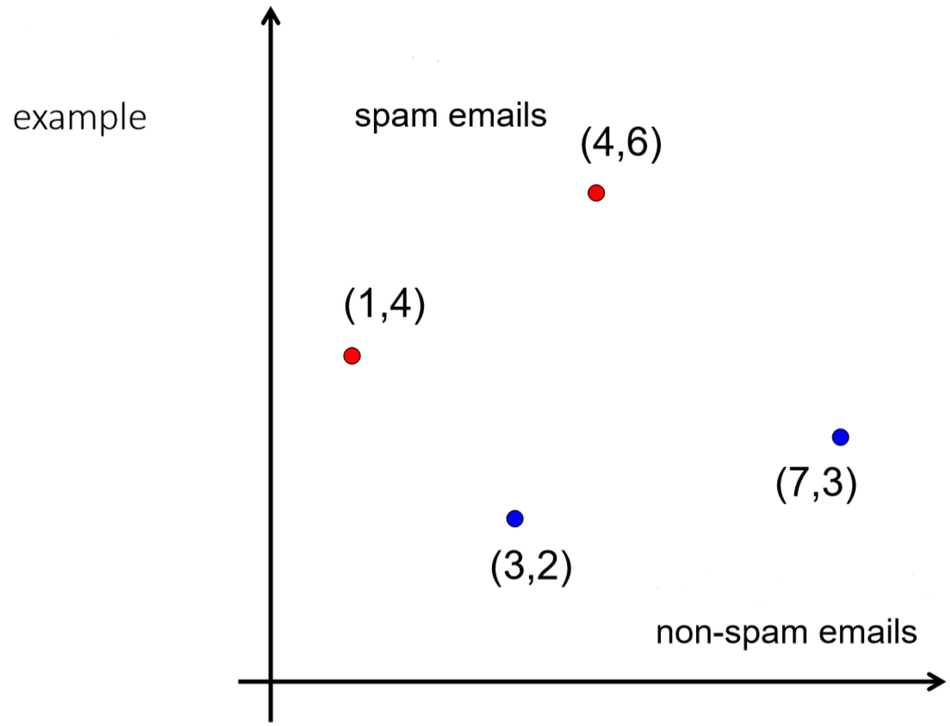
Linear Model (uses linear function as model)

- Classification
  - The analysis target is divided into two groups
- Choose a linear function so that the two groups can be distinguished depending on whether f(x) is positive or not

$$f(\boldsymbol{x}) \geq 0 \text{ or } f(\boldsymbol{x}) \leq 0$$

Linear Model (uses linear function as model)

- Classification
  - Spam mail/regular mail (x is a vectorization of information obtained from mail)

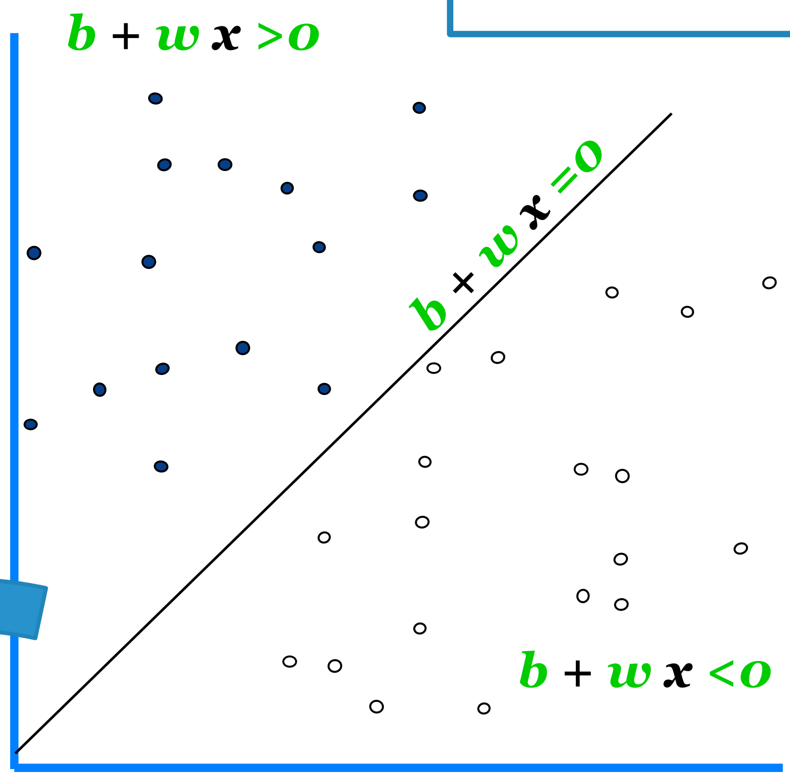


How would you classify this data?

SVM Linear Model (uses linear function as model)

$$f_{\mathbf{w}}(\boldsymbol{x}) = b + w_1 x_1 + \cdots + w_d x_d$$

- • denotes $f(\boldsymbol{x}) \geq 0$
- ◦ denotes $f(\boldsymbol{x}) \leq 0$



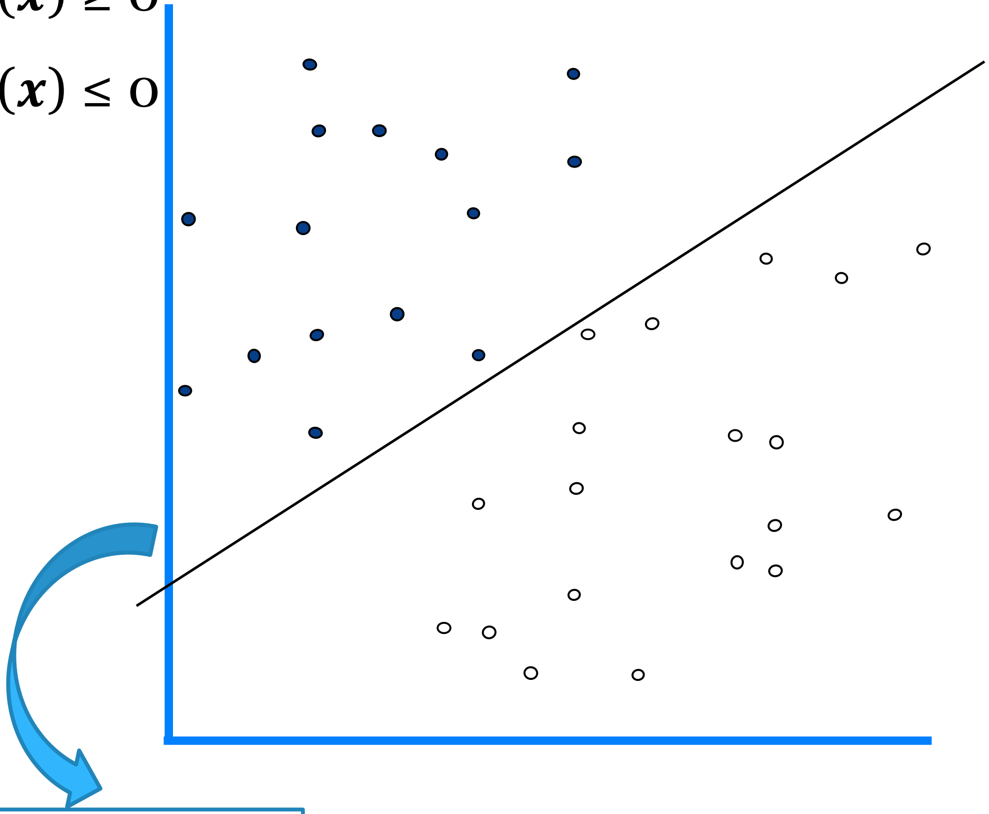
How would you classify this data?

Find a linear hyperplane (decision boundary) that will separate the data

SVM Linear Model (uses linear function as model)

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = b + w_1 x_1 + \cdots + w_d x_d$$

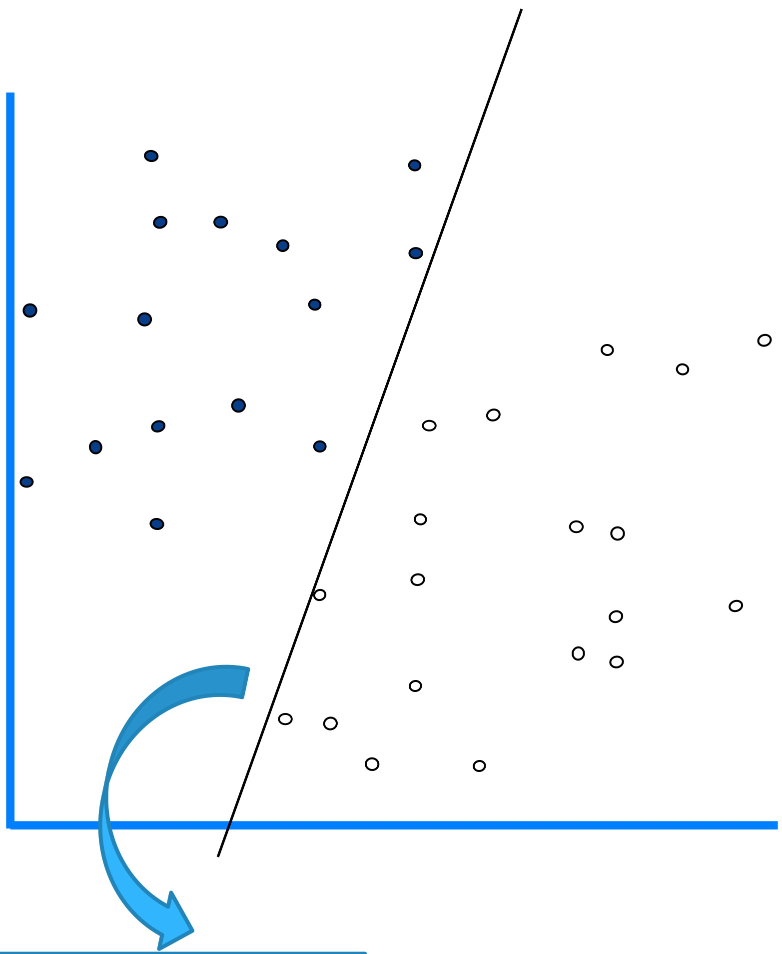
- • denotes $f(\boldsymbol{x}) \geq 0$
- ∘ denotes $f(\boldsymbol{x}) \leq 0$



One Possible Solution

SVM Linear Model (uses linear function as model)

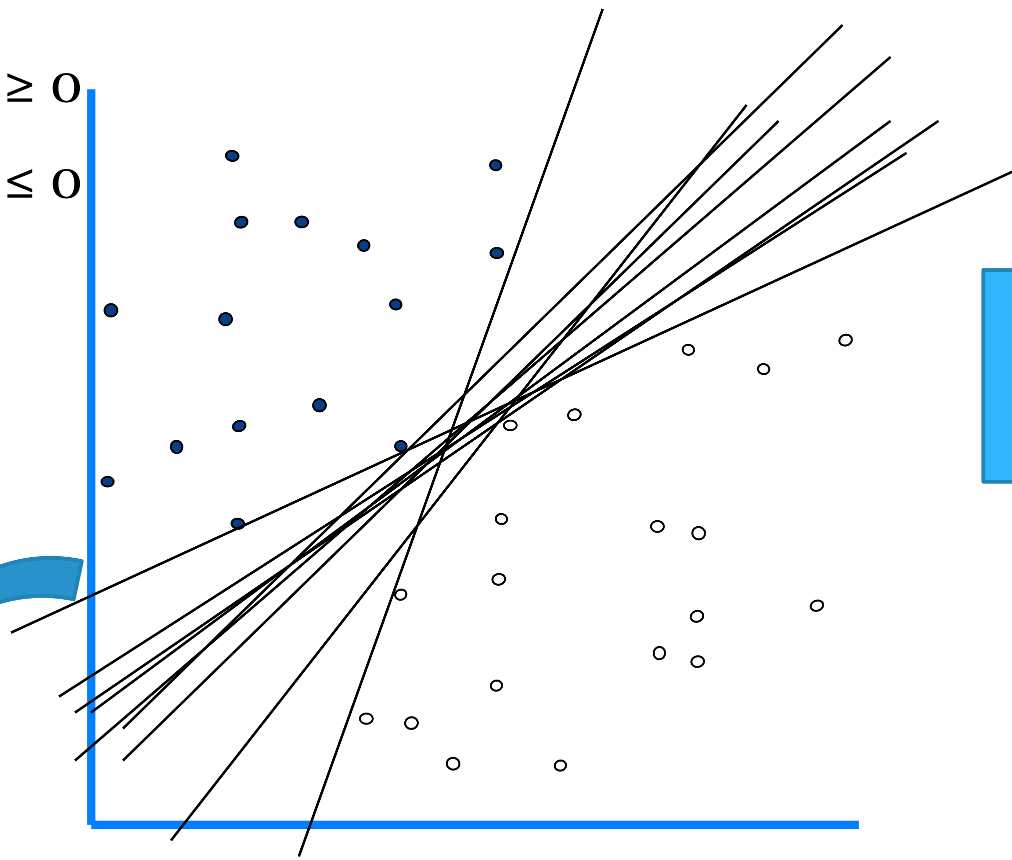
- • denotes $f(x) \geq 0$
- ∘ denotes $f(x) \leq 0$



Another possible solution

SVM Linear Model (uses linear function as model)

- • denotes $f(\boldsymbol{x}) \geq 0$
- ∘ denotes $f(\boldsymbol{x}) \leq 0$

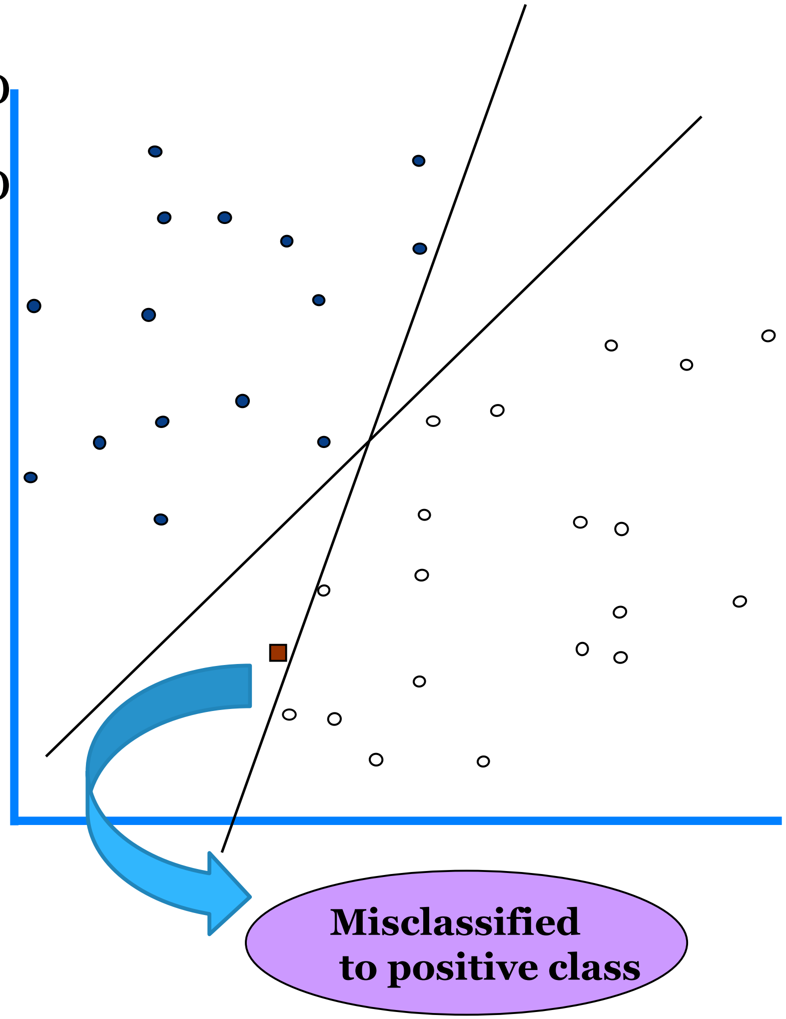


Which is the best?????

Other possible solutions

SVM Linear Model (uses linear function as model)

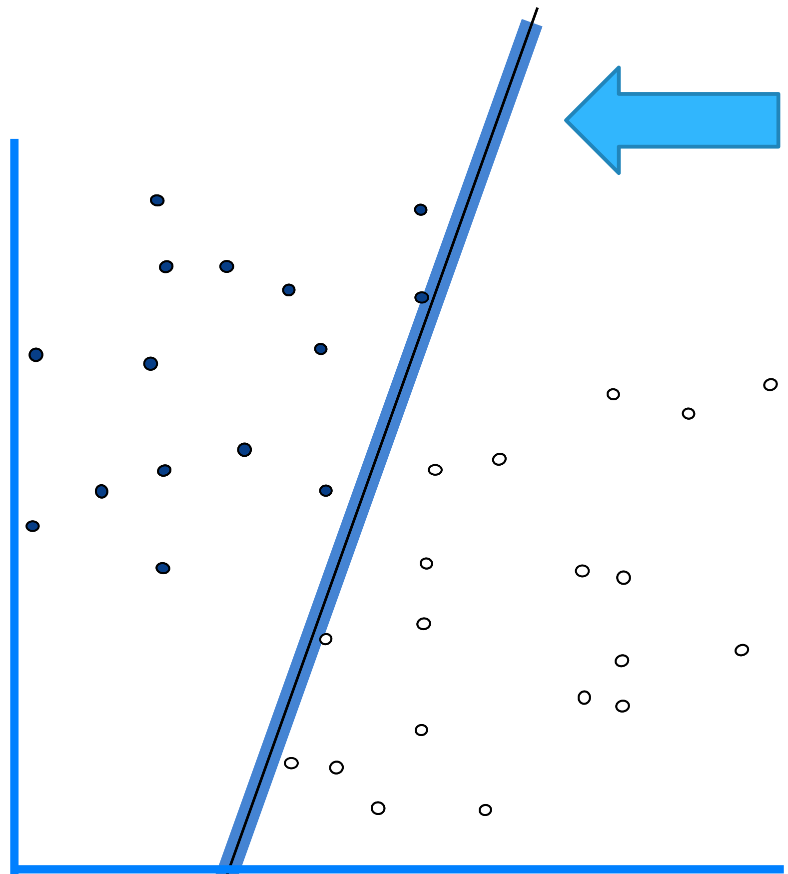
- • denotes $f(\boldsymbol{x}) \geq 0$
- ◦ denotes $f(\boldsymbol{x}) \leq 0$



How would you define better?

Solution: Define the Margin

- • denotes $f(x) \geq 0$
- ∘ denotes $f(x) \leq 0$

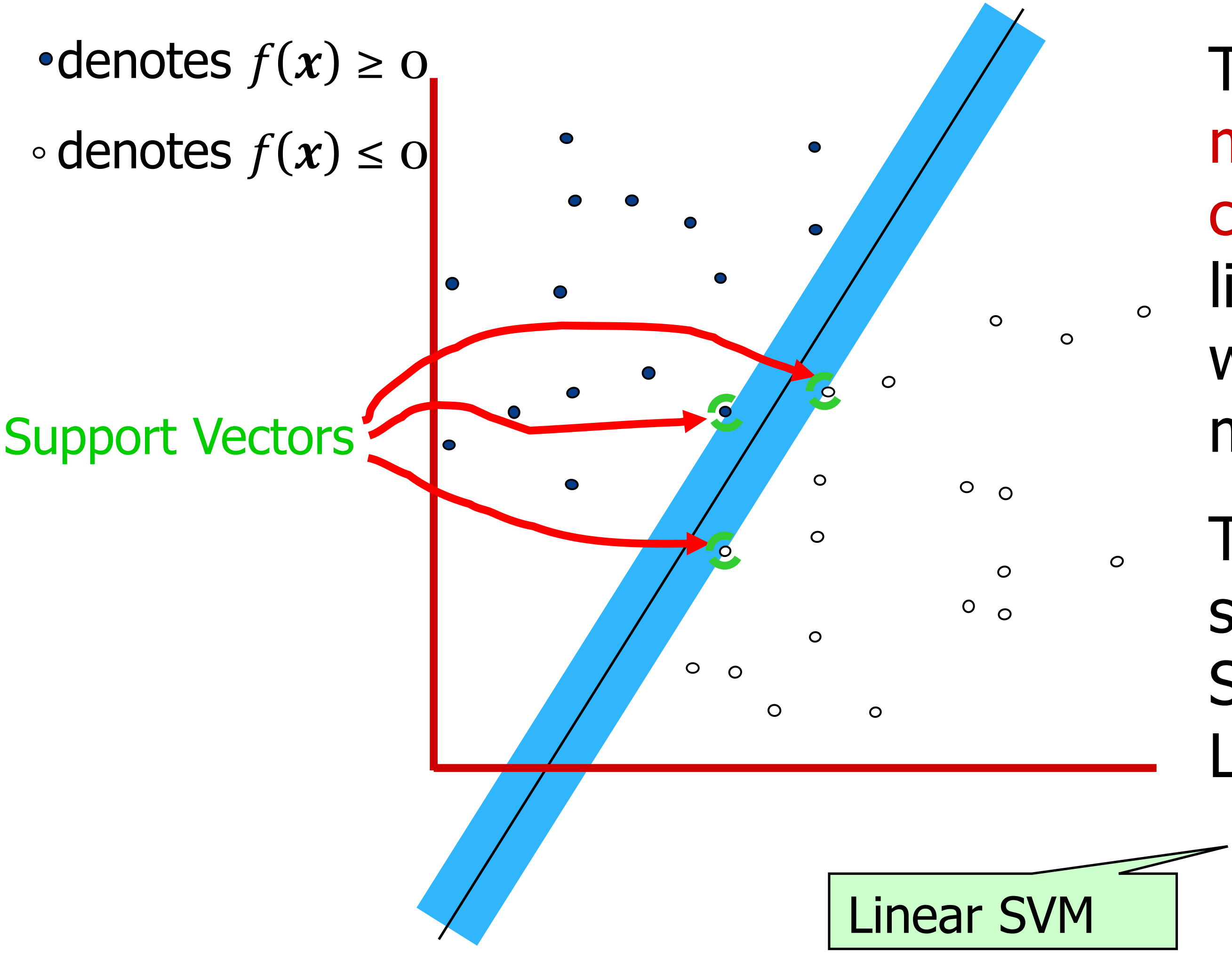


Define the **margin** of a linear classifier

Find hyperplane **maximizes** the margin

Margin => width of the boundary that could be increased before hitting a data point

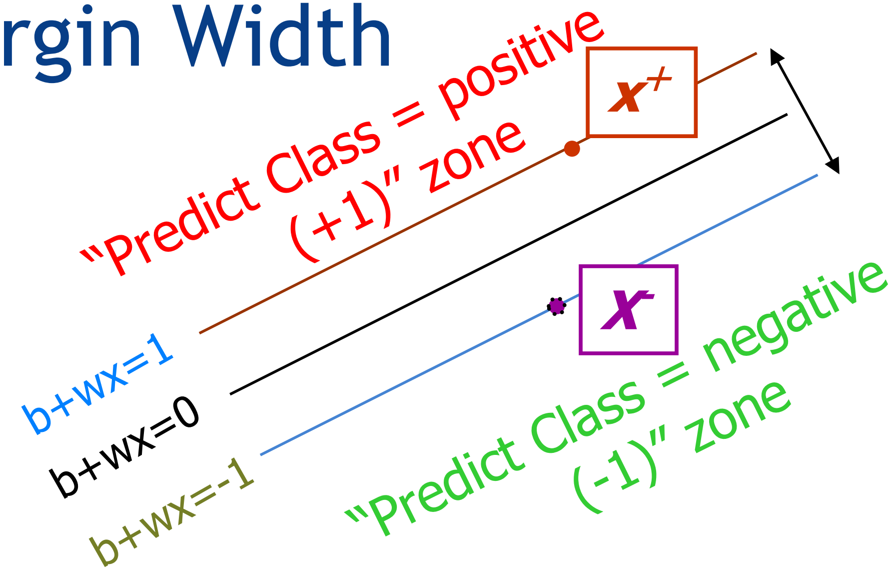
Solution: Maximum Margin



The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Find Margin Width



M=Margin Width

By calculating,

$b + \boldsymbol{w} \cdot \boldsymbol{x}^{+} = +1$

$b + \boldsymbol{w} \cdot \boldsymbol{x}^{-} = -1$

$\boldsymbol{w} \cdot (\boldsymbol{x}^{+} - \boldsymbol{x}^{-}) = 2$

$$M = \frac{(x^{+} - x^{-}) \cdot w}{|w|} = \frac{2}{|w|}$$

Linear SVM Mathematically

If all training data are correctly classify,

$$wx_i + b \geq 1 \quad \text{if } y_i = +1$$

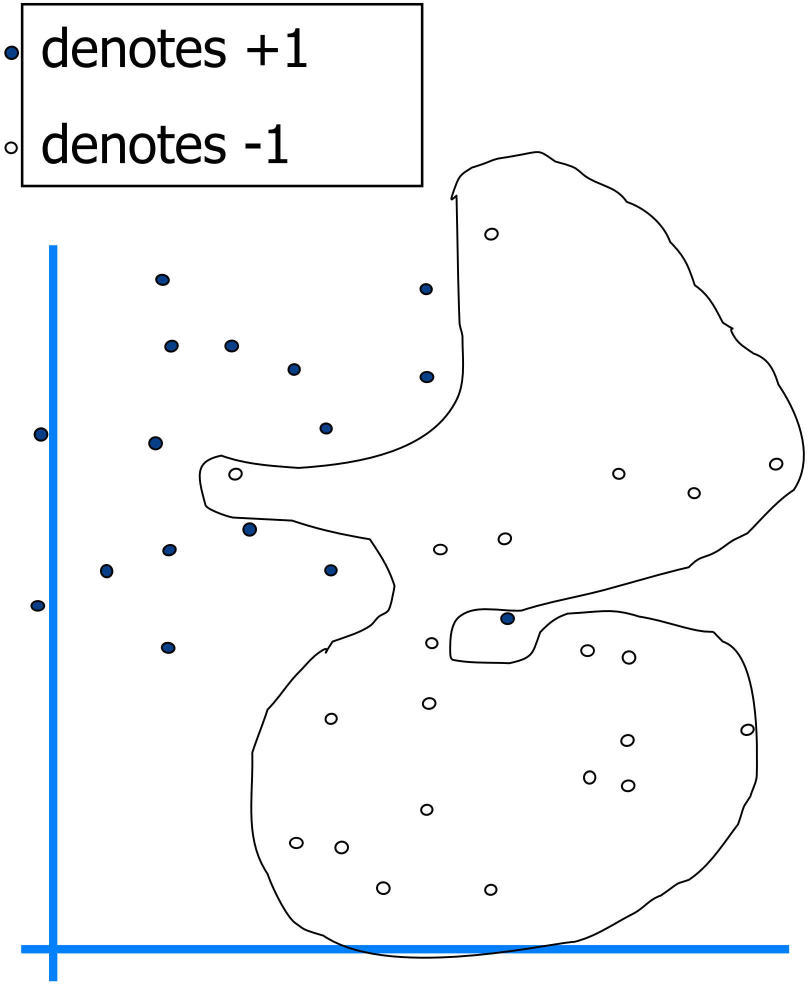
$$wx_i + b \leq 1 \quad \text{if } y_i = -1$$

$$y_i(wx_i + b) \geq 1 \quad \text{for all i}$$

Maximize the Margin $$M = \frac{2}{|w|}$$

same as minimize $$\frac{1}{2} w^t w$$

If the problem is not linearly separable?



- **If the training set is noisy or not linearly separable, the solutions are,**

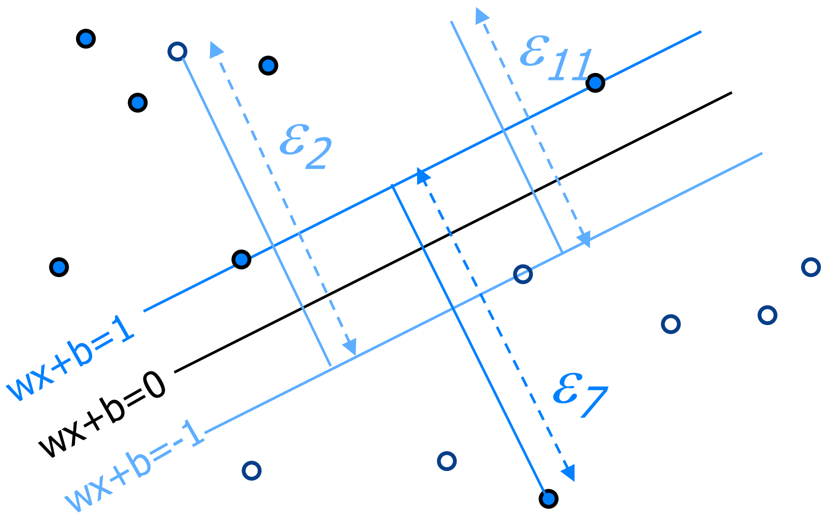
Use slack variable ξ_i

Use very powerful kernels

to avoid

OVERFITTING!

If the problem is not linearly separable?



Need to Minimize:

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

Control overfitting

Subject to:

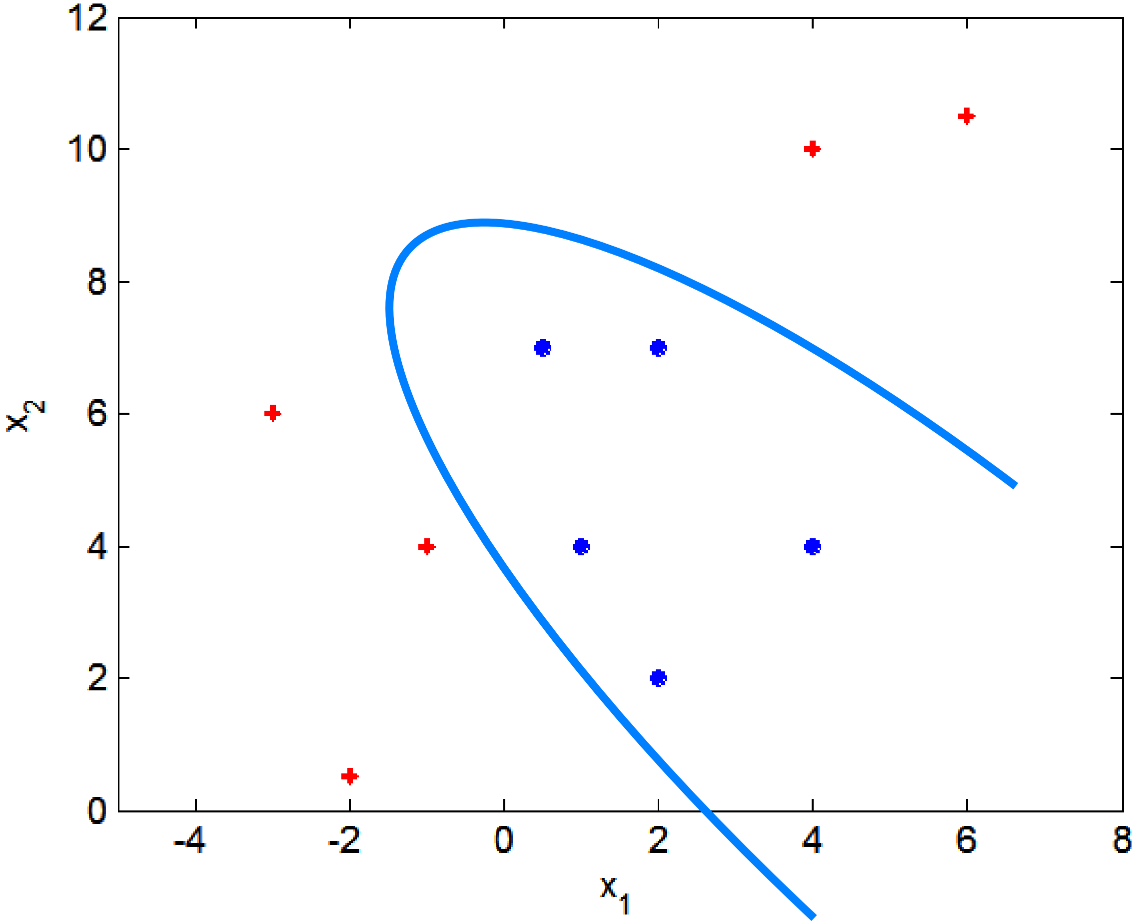
$$y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + \mathrm{b}) \geq 1$$

$$y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geq 1 - \xi_i$$

Slack variable

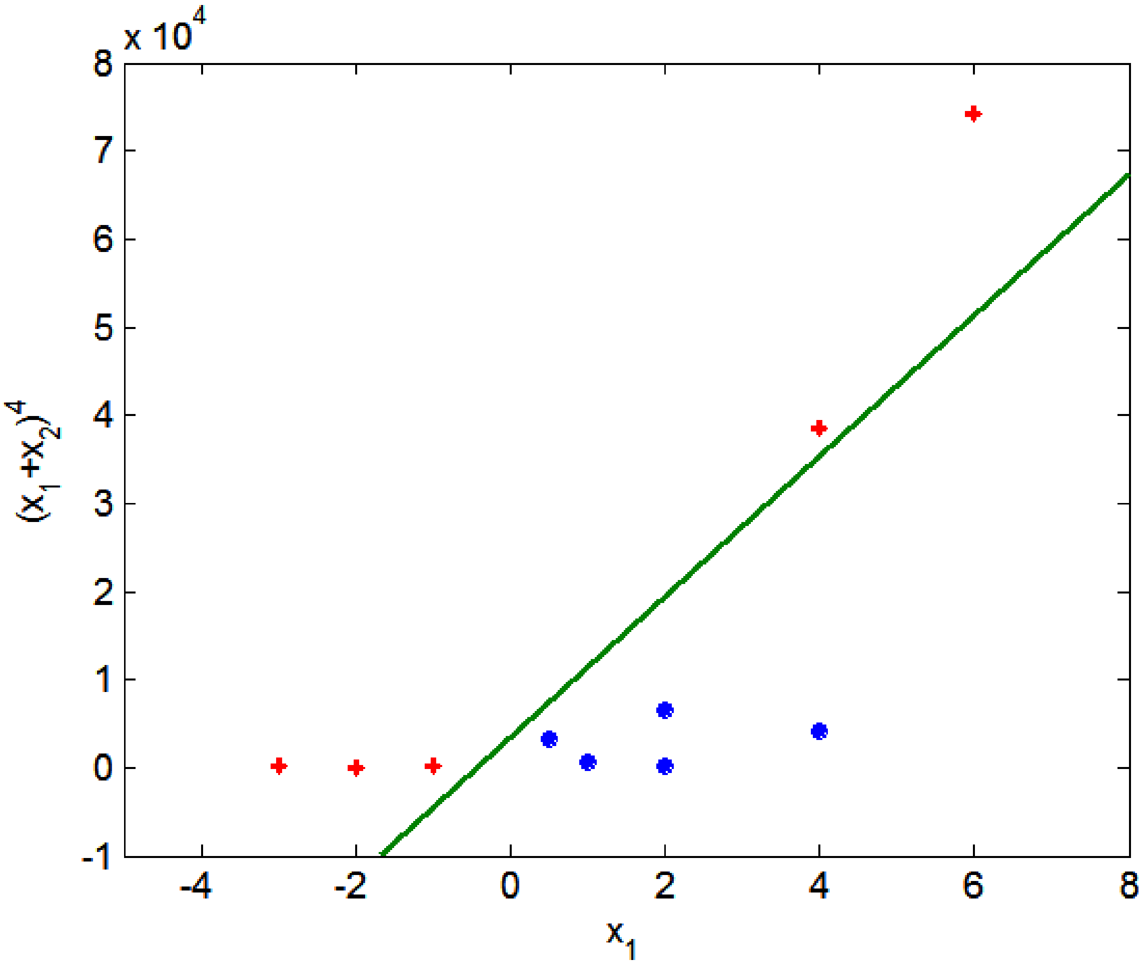
Nonlinear SVM

- If decision boundary is not linear?



Nonlinear SVM

- Solution: Transform data into higher dimensional space



Think smart, be different.

The "Kernel Trick"

- The linear classifier relies on dot product between vectors

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

- If every data point is mapped into high-dimensional space via some transformation

$$\Phi: \; x \rightarrow \varphi(x),$$

the dot product becomes:

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

- A kernel function is some function that corresponds to an inner product in some expanded feature space.

Examples of Kernel Functions

- Linear: $K(\mathbf{x_i},\mathbf{x_j}) = \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j}$

- Polynomial of power *p*: $K(\mathbf{x_i},\mathbf{x_j}) = (1 + \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j})^p$

- Gaussian (radial-basis function network):

$$K(\mathbf{x_i},\mathbf{x_j}) = \exp\left(-\frac{\|\mathbf{x_i}-\mathbf{x_j}\|^2}{2\sigma^2}\right)$$

- Sigmoid: $K(\mathbf{x_i},\mathbf{x_j}) = \tanh(\beta_0 \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j} + \beta_1)$

CEIT
Think smart, be different.

Advantages of Support Vector Machine

- SVM classifiers offer good accuracy and perform faster prediction compared to other algorithms
- They also use less memory
- SVM works well with a clear margin of separation and with high dimensional space

Disadvantages of Support Vector Machine

- SVM is not suitable for large datasets because of its high training time
  - A relatively small number of mislabeled examples can dramatically decrease the performance

- It works poorly with overlapping classes
- It is sensitive to the type of kernel used

Practical: Python Coding Environment

- Anaconda

  https://www.anaconda.com/distribution/#linux

- Google Colaboratory

  https://colab.research.google.com

  - Requrie G-mail Account

Cancer Classification using SVM

- Required dataset

UCI Machine Learning Library or scikit-learn library

Cancer Classification using SVM

- Required dataset

UCI Machine Learning Library or scikit-learn library

- Loading Data

```
#import scikit-learn dataset library
from sklearn import datasets

#load dataset
cancer = datasets.load_breast_cancer()
```

Cancer Classification using SVM

- Exploring Data

```
# print the names of the 13 features
print("Features: ", cancer.feature_names)

# print the label type of cancer('malignant' 'benign')
print("Labels: ", cancer.target_names)
```

```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels:  ['malignant' 'benign']
```

- Check the data size

```
# print data(feature)shape
cancer.data.shape
```

```
(569, 30)
```

Cancer Classification using SVM

- Check top 5 records of the feature set

```
# print the cancer data features (top 5 records)
print(cancer.data[0:5])
```

```
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
  3.613e-01 8.758e-02]
 [1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
  1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
  9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
  2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
  6.638e-01 1.730e-01]
 [2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
  1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
  1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
  1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
  2.364e-01 7.678e-02]]
```

CEIT
Think smart, be different.

Cancer Classification using SVM

- Check the records of the target set (0: malignant, 1:benign)

```python
# print the cancer labels (0:malignant, 1:benign)
print(cancer.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

Cancer Classification using SVM

- Splitting Data

```
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3,random_state=109)
# 70% training and 30% test
```

- Generating Model

```
#Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Cancer Classification using SVM

- Evaluating the Model

```python
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9649122807017544
```

The classification rate is 96.49%, it is considered as very good accuracy.

Therefore, SVM classifier model can predict the breast cancer of patients according to the evaluation result.

References

- Pang-Ning Tan, Michael Steinbach, Vipin Kumar:*Introduction to Data Mining*, Addison-Wesley
- https://towardsdatascience.com/breast-cancer-classification-using-support-vector-machine-svm-a510907d4878

Think smart, be different.

Next Week Lecture

- Supervised Learning: Simple Regression

Thank You