Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Support Vector Machine (SVM)

RADI608: Data Mining and Machine Learning
RADI602: Data Mining and Knowledge Discovery

Lect. Anuchate Pattanateepapon, D.Eng.

Section of Data Science for Healthcare

Department of Clinical Epidemiology and Biostatistics

Faculty of Medicine Ramathibodi Hospital, Mahidol University

Wisdom of the Land

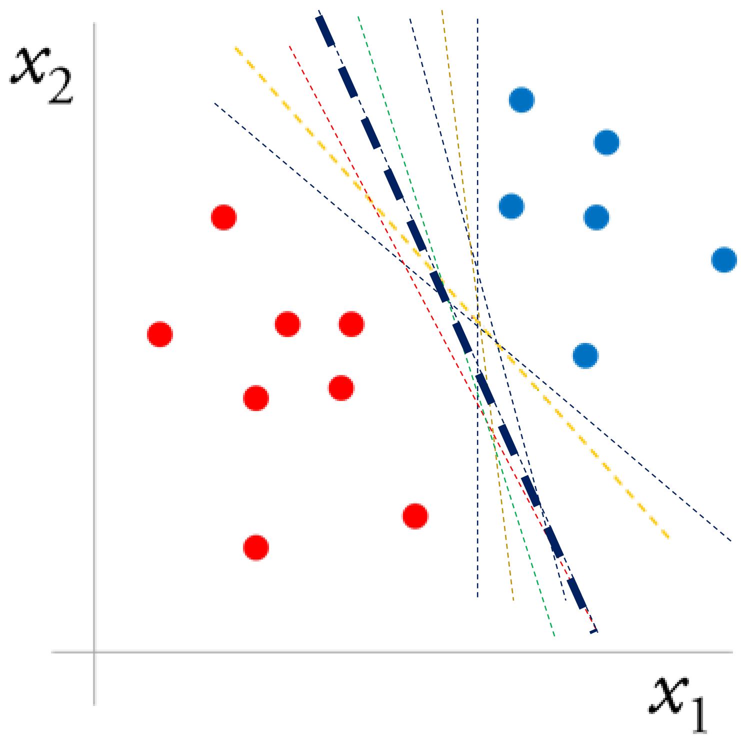
Introduction of Support Vector Machine

Support Vector Machine (SVM) was invited by Boser, Guyon, and Vapnik in 1992. SVM uses a machine learning methodology to enlarge the accuracy of classification and regression predictors with avoiding a model overfitting.

The SVM trained the input data based on optimization theory which constructed from statistical learning concepts and created a linear function in a high dimensional feature space.

Moreover, the objective of SVM is to search the best separating line or hyperplane that leaves the maximum margin from both classes (binary classification).

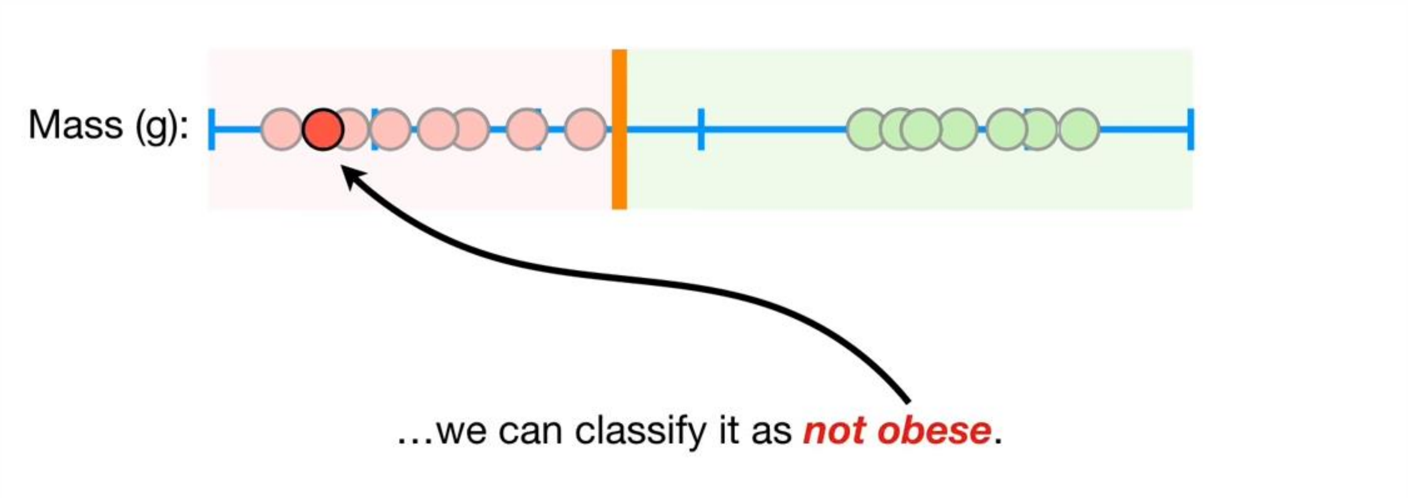
How does a support vector machine work?



find the best separating line that leaves the maximum margin from both classes or keeps ● and ● as far away from each other as possible

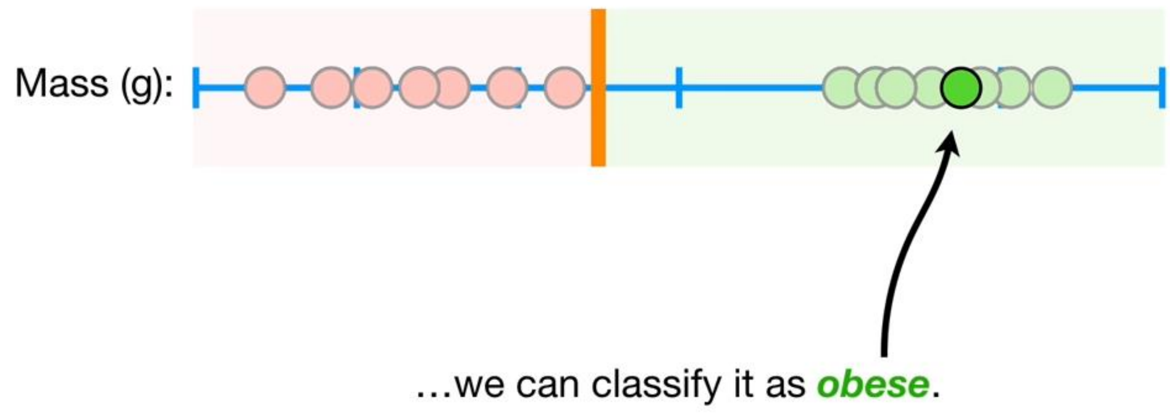
http://efavdb.com/wp-content/uploads/2015/05/binaryclass_2d.png

How does a support vector machine work?



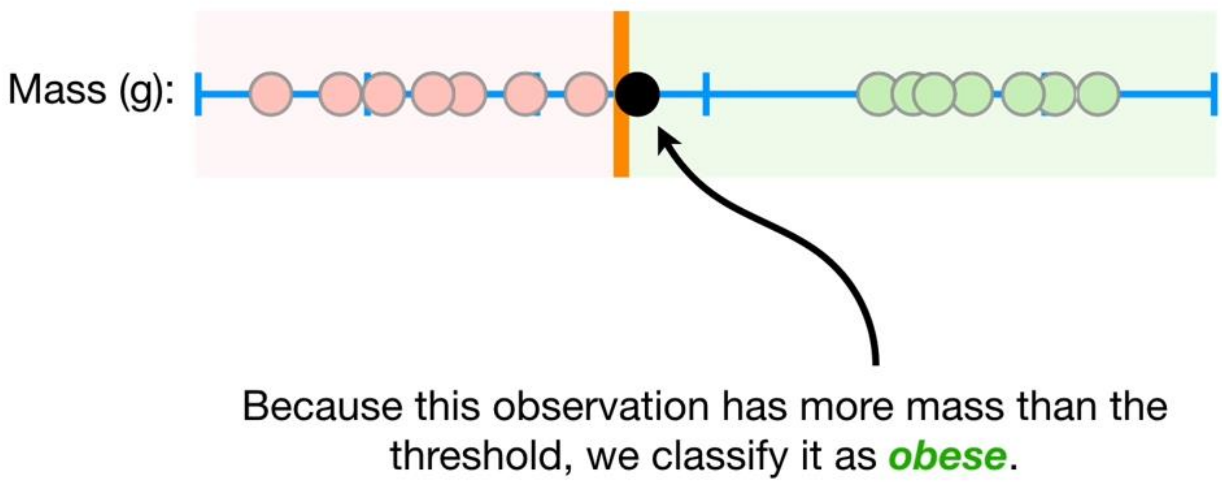
X = mass
Y = Obese, Not Obese

How does a support vector machine work?

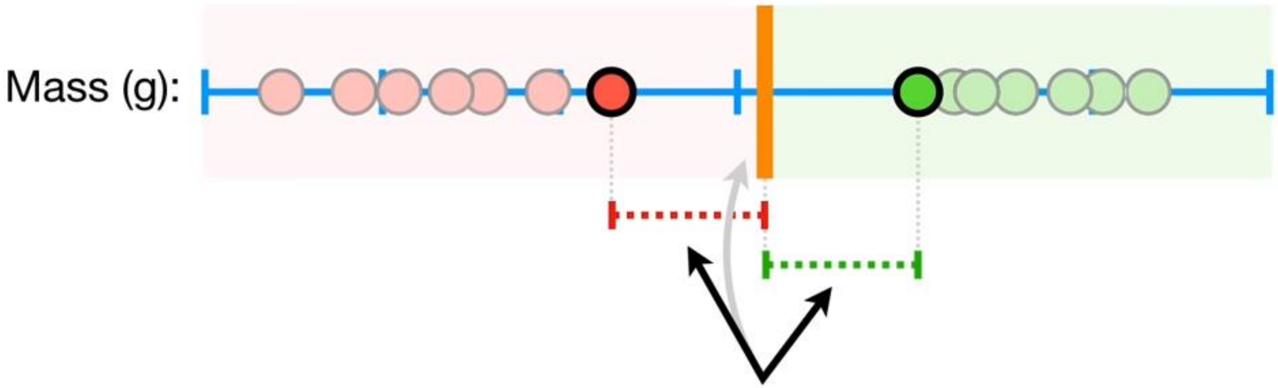


X = mass
Y = Obese, Not Obese

How does a support vector machine work?

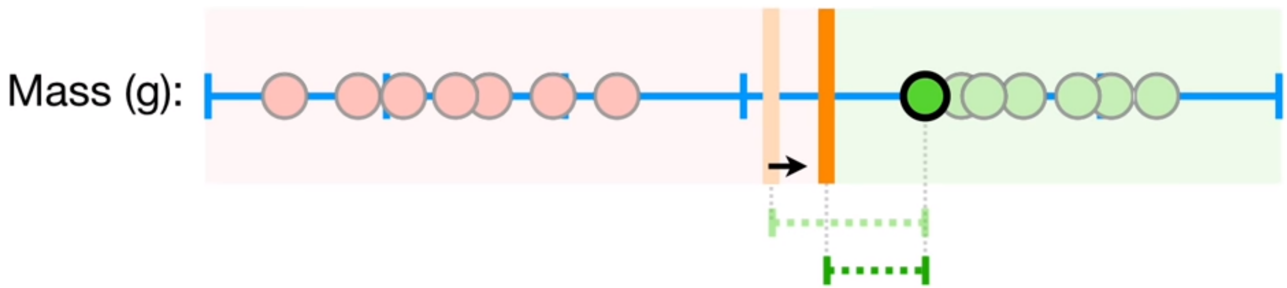


How does a support vector machine work?



When the threshold is halfway between the two observations, the **margin** is as large as it can be.

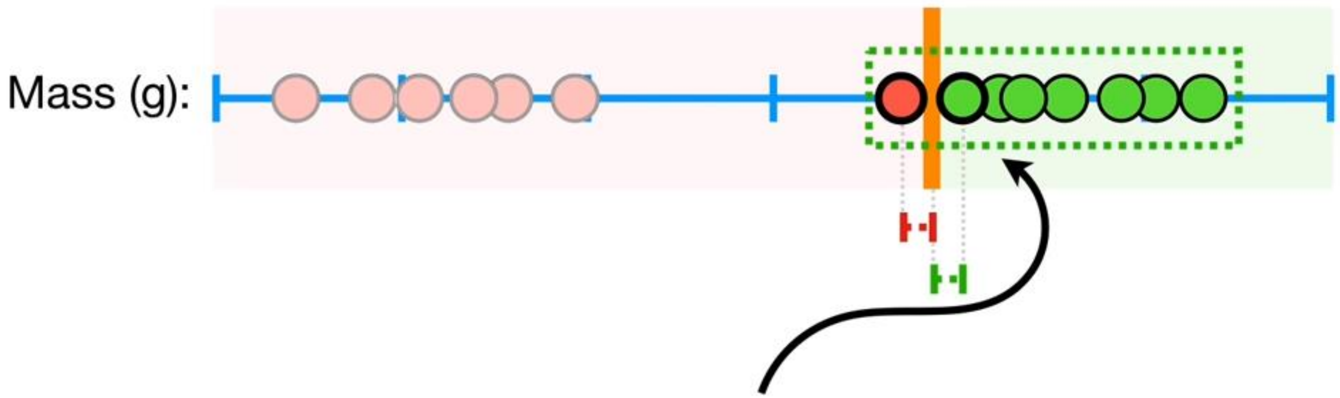
How does a support vector machine work?



If we shift the threshold to the right

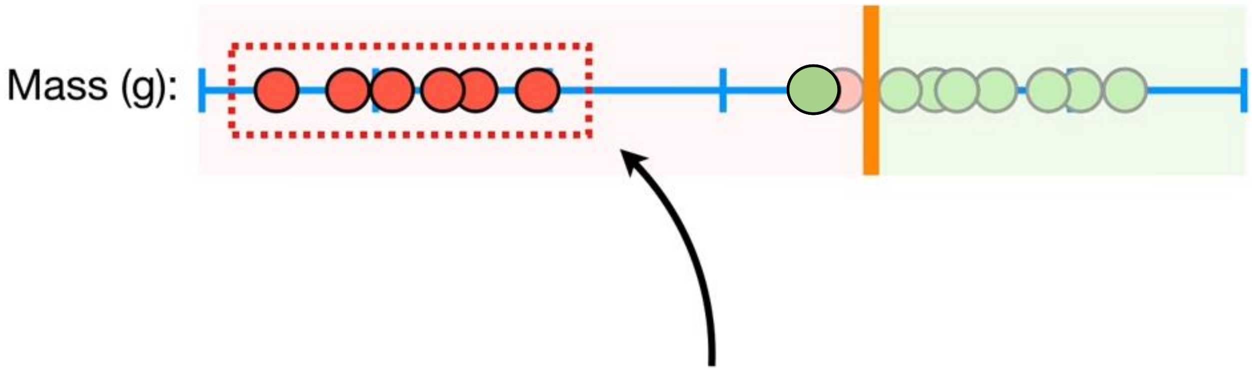
…then the distance between the ***obese*** observation and the threshold would get smaller…

How does a support vector machine work?



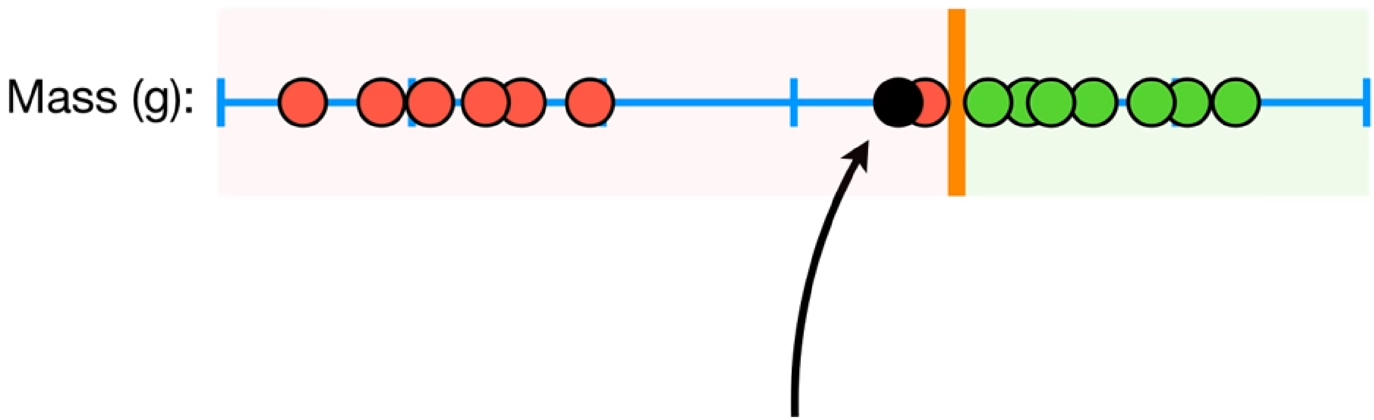
In this case, the **Maximum Margin Classifier** would be super close to the ***obese*** observations…

How does a support vector machine work?



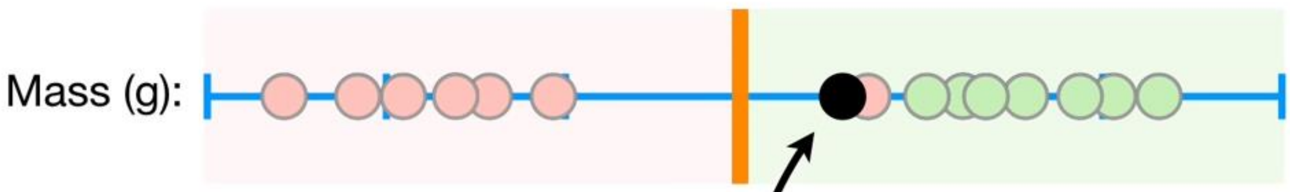
…we would classify it as ***not obese***, even though most of the ***not obese*** observations are much further away than the ***obese*** observations.

How does a support vector machine work?



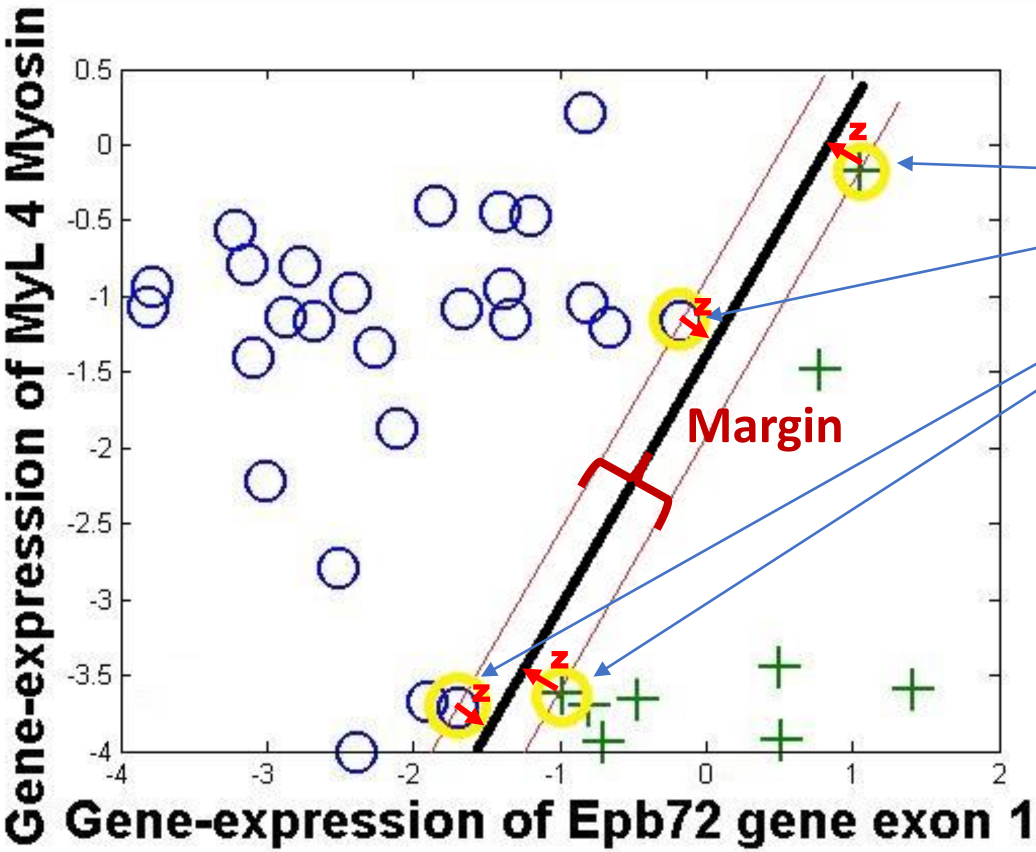
…and it performed poorly when we got new data (high variance).

How does a support vector machine work?



…it performed better when we got new data (low variance).

How to create a discriminant line



searches for the closest point

(support vectors)

find the maximum margin

Maximum margin: Formalization

Classifiers: $f(x_i) = sign(w^\top x_i + b)$

Functional margin of x_i : $y_i(w^\top x_i + b)$

where w is a decision hyperplane normal vector, x_i is a data point $\boldsymbol{i}$
and y_i is a class of data point $\boldsymbol{i}$ (+1 or -1)

How to find the width of margin (1)

Since $w^\top x + b = 0$ and $c(w^\top x + b) = 0$ define the same plane, we could choose the normalization of w

Choose normalization such that $w^\top x + b = +1$ and $w^\top x + b = -1$ for the positive and negative support vectors respectively

How to find the width of margin (2)

Margin = Unit Vector . Difference Vector

$$= \frac{\mathbf{w}}{||\mathbf{w}||} \cdot (\mathbf{x}_{+} - \mathbf{x}_{-})$$

$$= \frac{\mathbf{w}^{\top}\mathbf{x}_{+} - \mathbf{w}^{\top}\mathbf{x}_{-}}{||\mathbf{w}||}$$

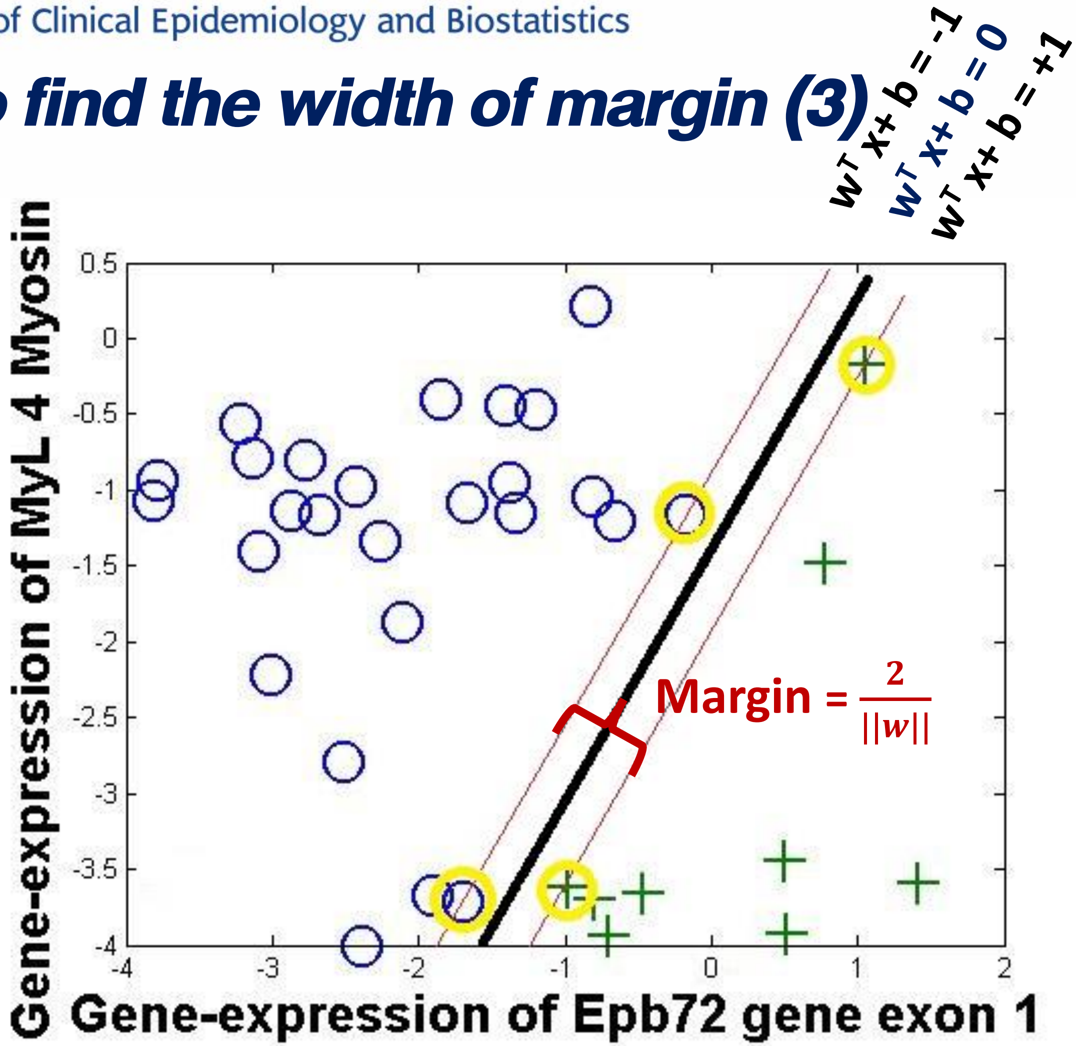
$$\mathbf{w}^{\top}\mathbf{x}_{+} + b = +1 \qquad \mathbf{w}^{\top}\mathbf{x}_{-} + b = -1$$

$$\Downarrow \qquad \Downarrow$$

$$\mathbf{w}^{\top}\mathbf{x}_{+} = 1 - b \qquad \mathbf{w}^{\top}\mathbf{x}_{-} = -1 - b$$

$$= \frac{1 - b + 1 + b}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

How to find the width of margin (3)



Find the maximum margin ***by minimizing w***

Learning the SVM can be formulated as an optimization:

$\max_{w} \frac{2}{||\boldsymbol{w}||}$ subject to $(w^\top x_i + b) \geq 1$ if $y_i = +1$ and $(w^\top x_i + b) \leq 1$ if $y_i = -1$ for $i = 1, \dots, N$

Or equivalently

$\min_{w} ||\mathrm{w}||^2$ subject to $y_i(w^\top x_i + b) \geq 1$ for $i = 1, \dots, N$

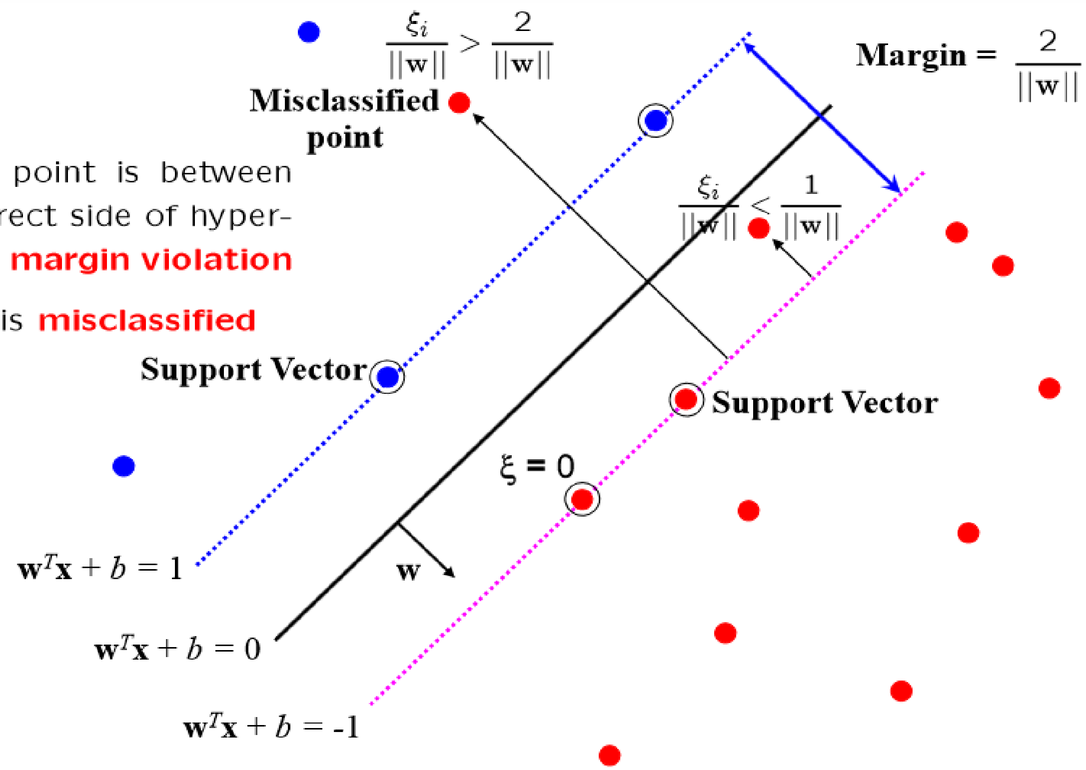
This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

The number of mistakes: slack variables

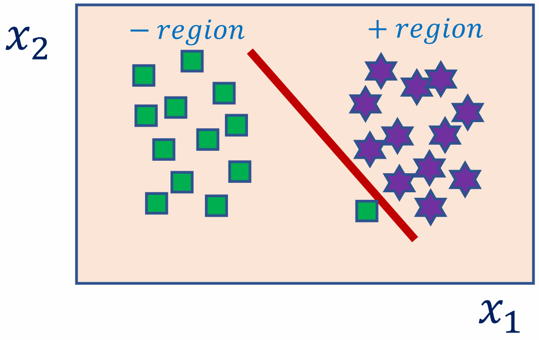
$$\xi_i \geq 0$$

for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a **margin violation**

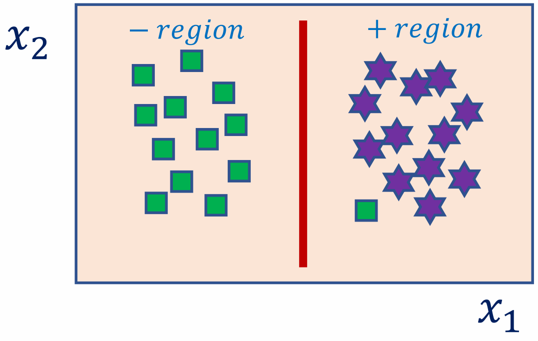
for $\xi > 1$ point is **misclassified**



The trade off between the maximum margin and the number of mistakes

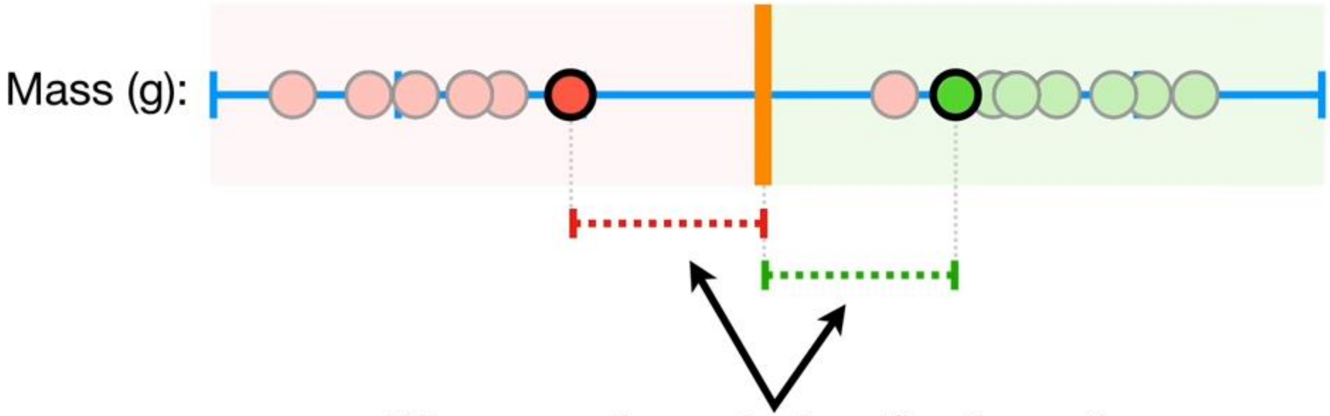


all points can be separated by a discriminant line with narrow margin or hard margin



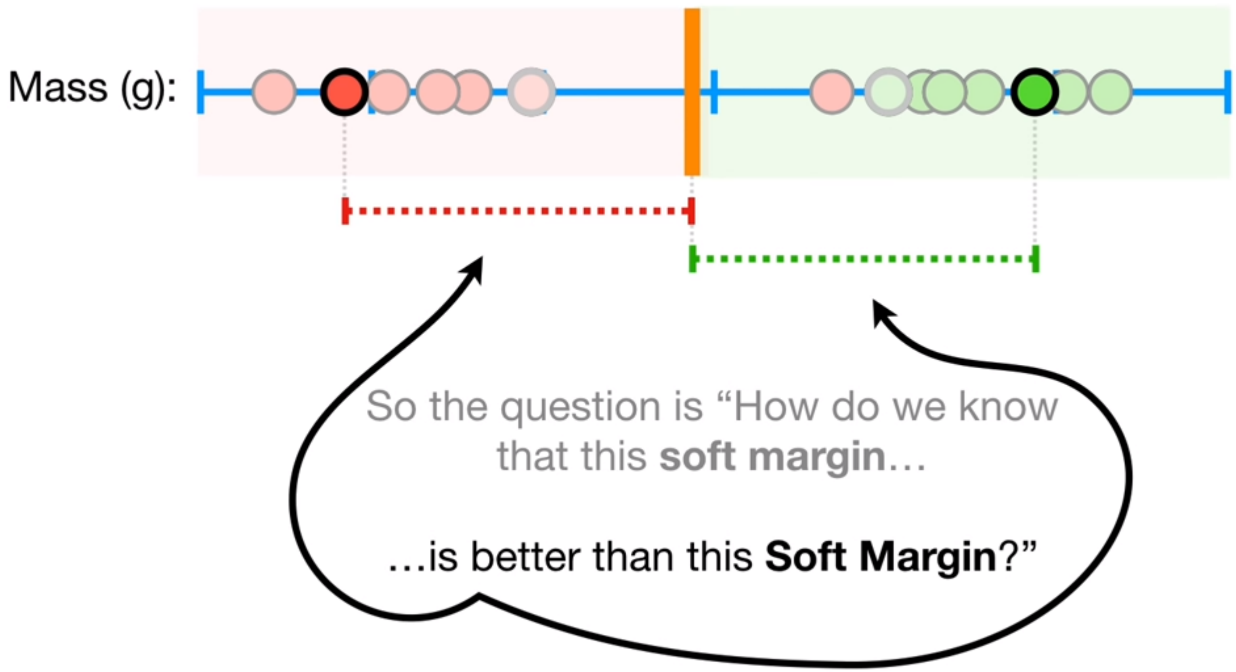
one points can't be separated by a discriminant line but possibly the large margin or soft margin

The trade off between the maximum margin and the number of mistakes

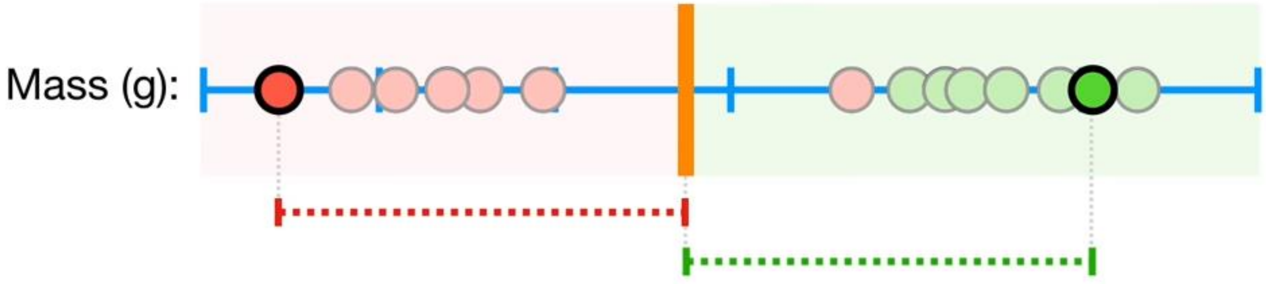


When we allow misclassifications, the distance between the observations and the threshold is called a **Soft Margin**.

The trade off between the maximum margin and the number of mistakes



The trade off between the maximum margin and the number of mistakes



The answer is simple: We use **Cross Validation** to determine how many misclassifications and observations to allow inside of the **Soft Margin** to get the best classification.

What is a large and a narrow margin (1)

Every constraint can be satisfied if is sufficiently large

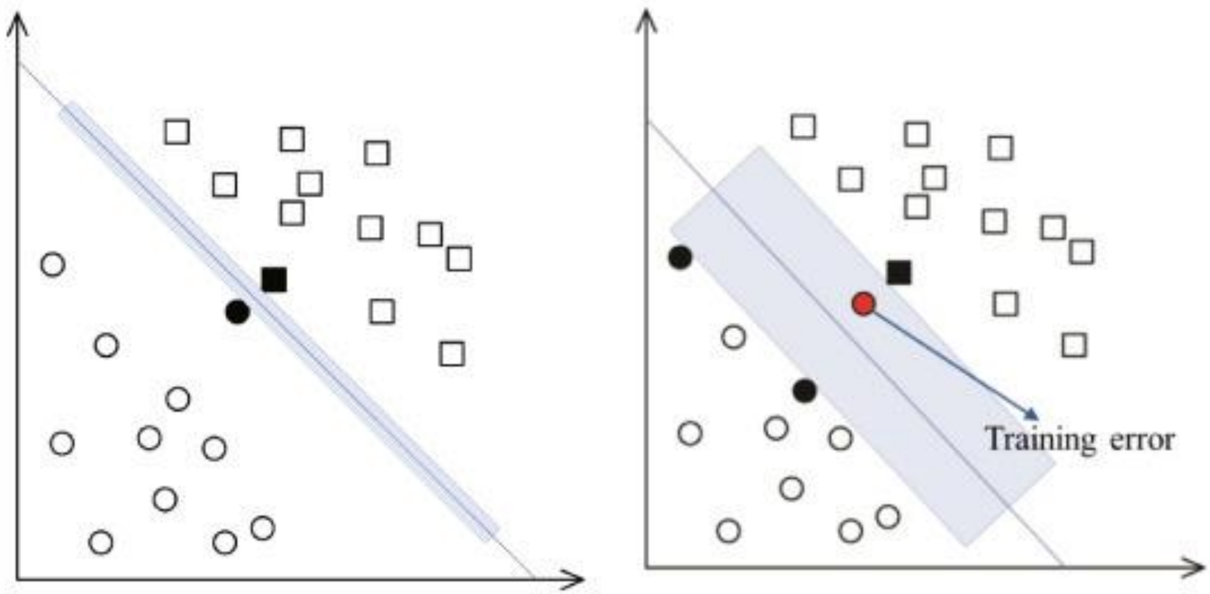
C is a regularization parameter:

- Small C allows constraints to be easily ignored a large margin
- Large C makes constraints hard to ignore a narrow margin
- $\mathrm{C} = \infty$ enforces all constraints to a hard margin

What is a large and a narrow margin (2)

C = infinity, Hard Solution

C = 10, Soft Solution



(a) LSVM with hard margin

(b) LSVM with soft margin

https://ars.els-cdn.com/content/image/1-s2.0-S0926580516301297-gr4.jpg

The SVM Optimization problem

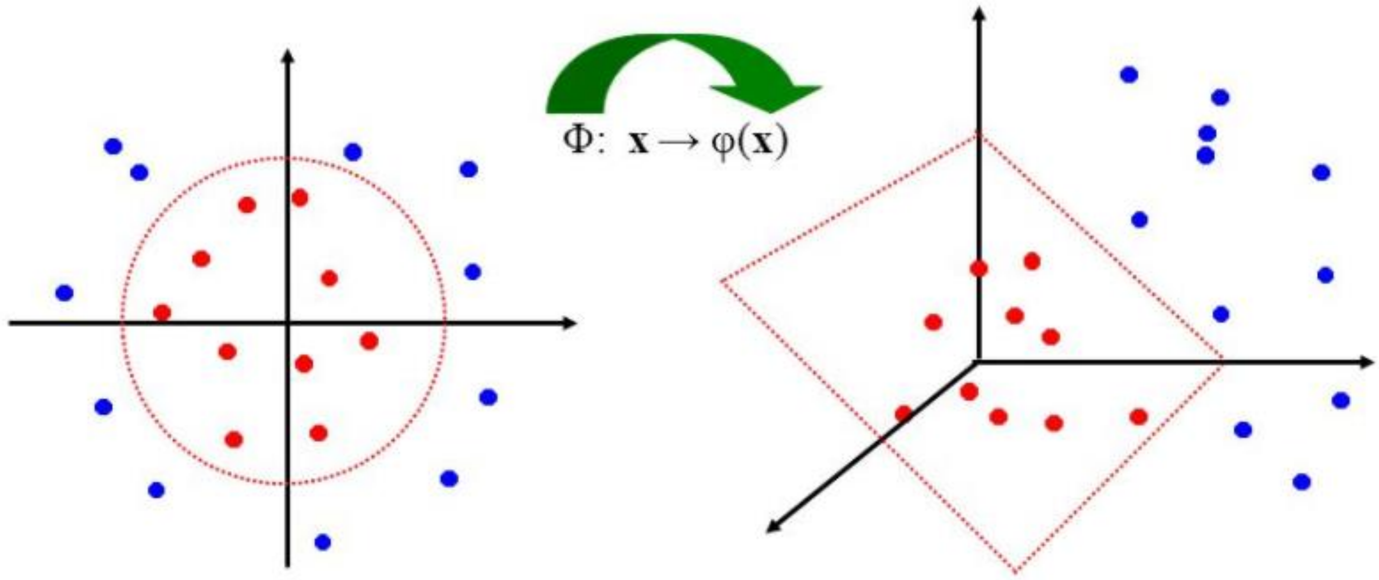
$$\min_{w,b} \frac{\|w\|}{2} + C \sum_{i=1}^{N} \xi_i$$

Subject to: $y_i(w^\top x_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$, $i = 1, 2, \ldots, N$

Could add even more flexibility by introducing a function ϕ that maps the original feature space to a higher dimensional feature space

Subject to: $y_i(w^\top \phi(x_i) + b) \geq 1 - \xi_i$, $\xi_i \geq 0$, $i = 1, 2, \ldots, N$

Maps the original feature space to a higher dimensional feature space

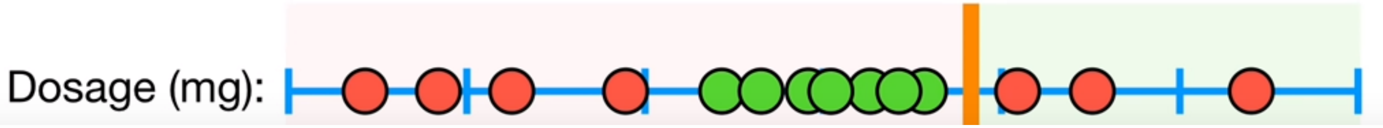


The original space becomes a linear problem in high-dimensional space

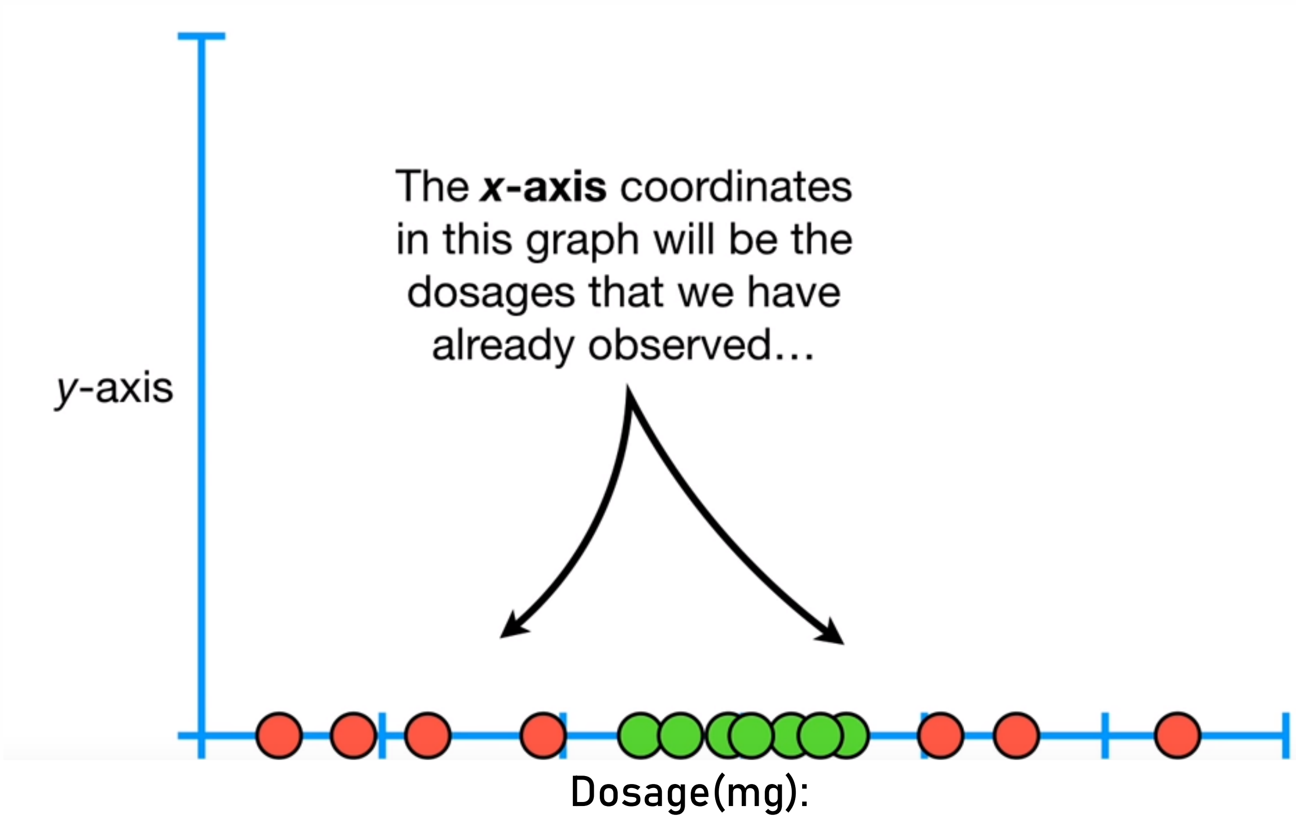
Picture credit: Andrew W. Moore, http://www.cs.cmu.edu/~aarti/Class/10701_Spring14/slides/kernel_methods.pdf

Maps the original feature space to a higher dimensional feature space

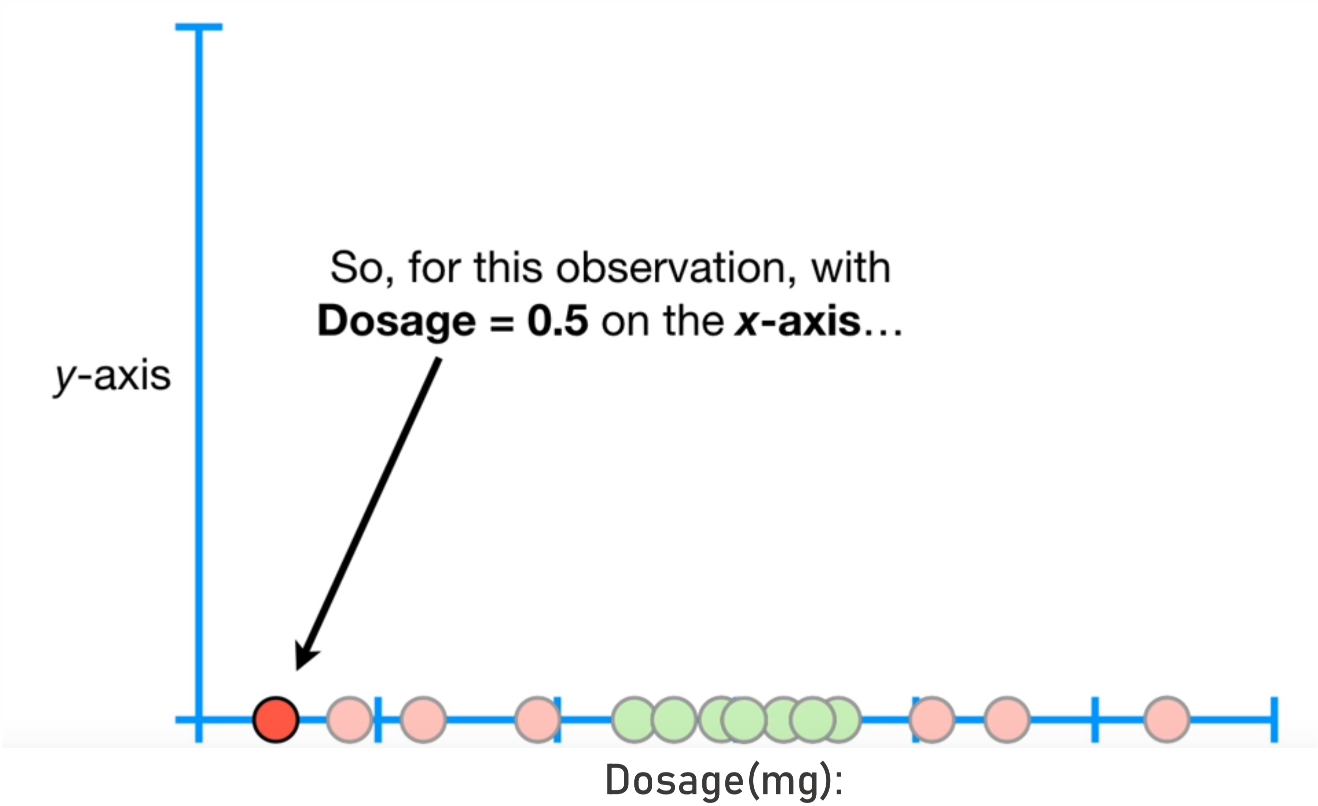
Since **Maximal Margin Classifiers** and **Support Vector Classifiers** can't handle this data, it's high time we talked about…



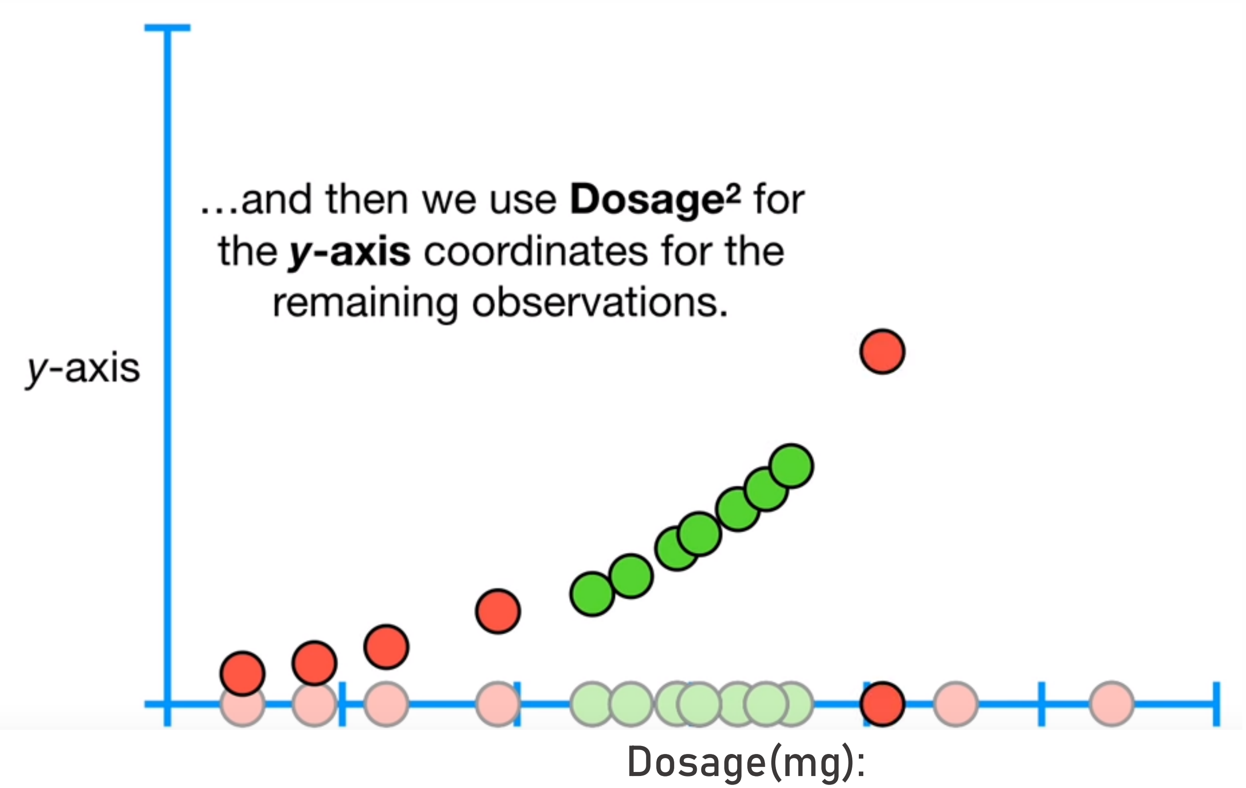
Maps the original feature space to a higher dimensional feature space



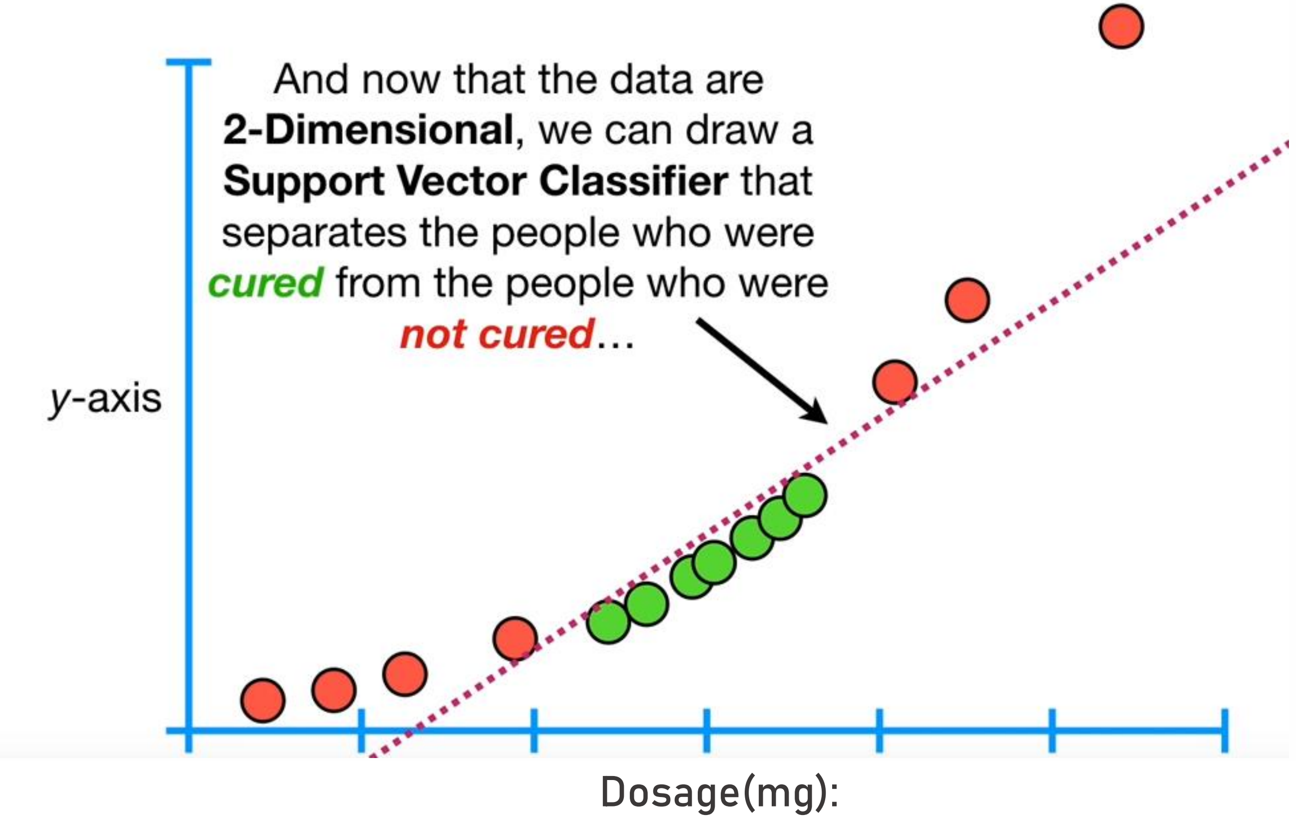
Maps the original feature space to a higher dimensional feature space



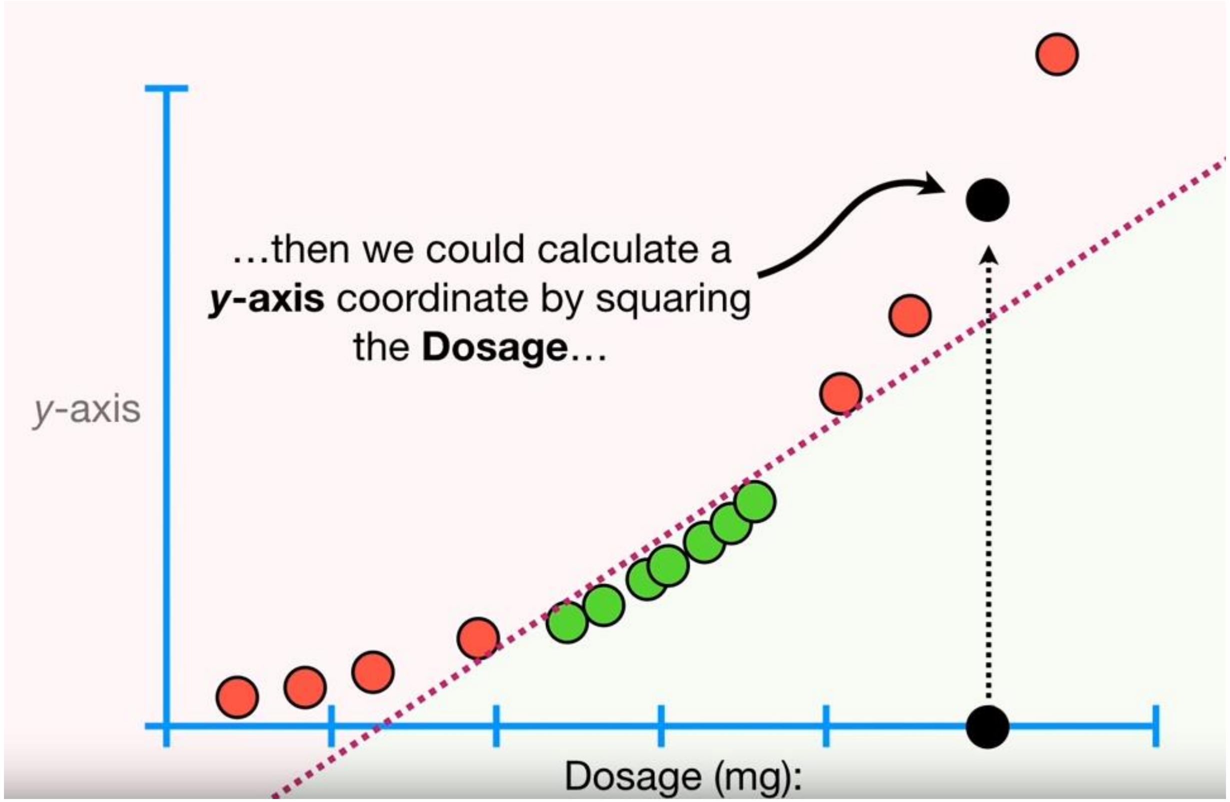
Maps the original feature space to a higher dimensional feature space



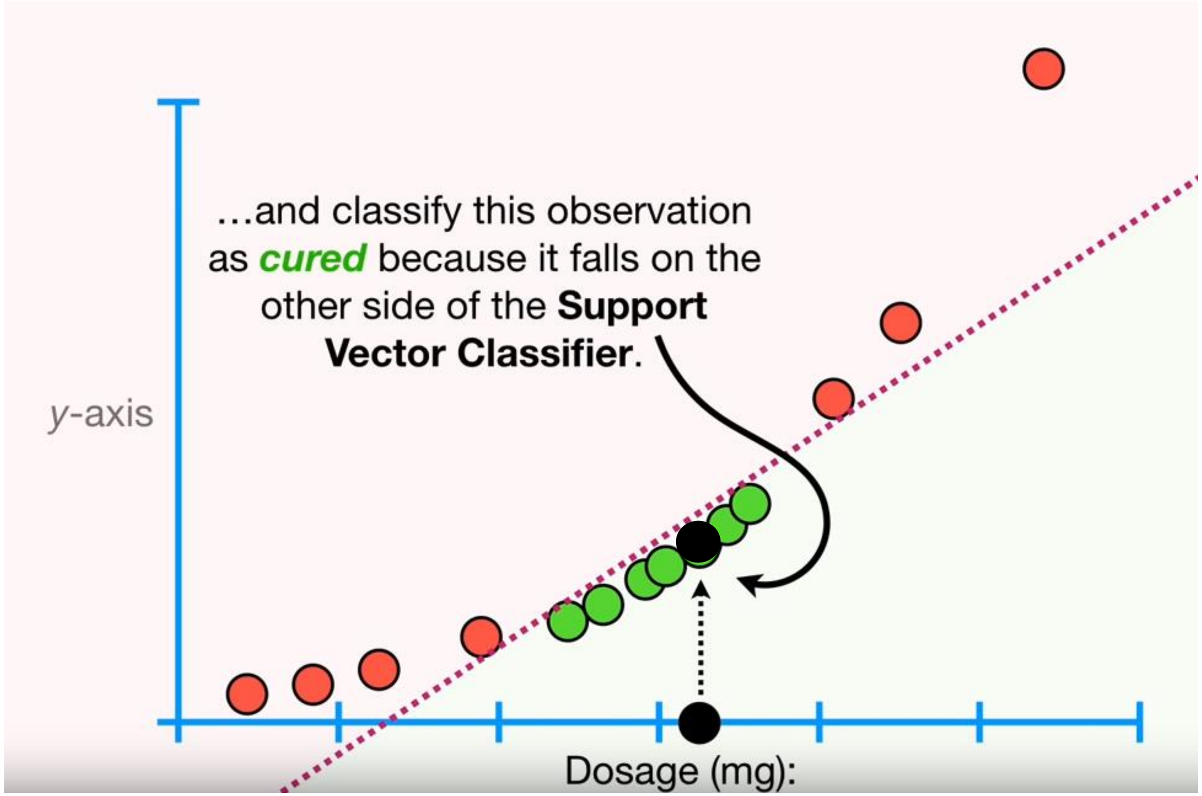
Maps the original feature space to a higher dimensional feature space



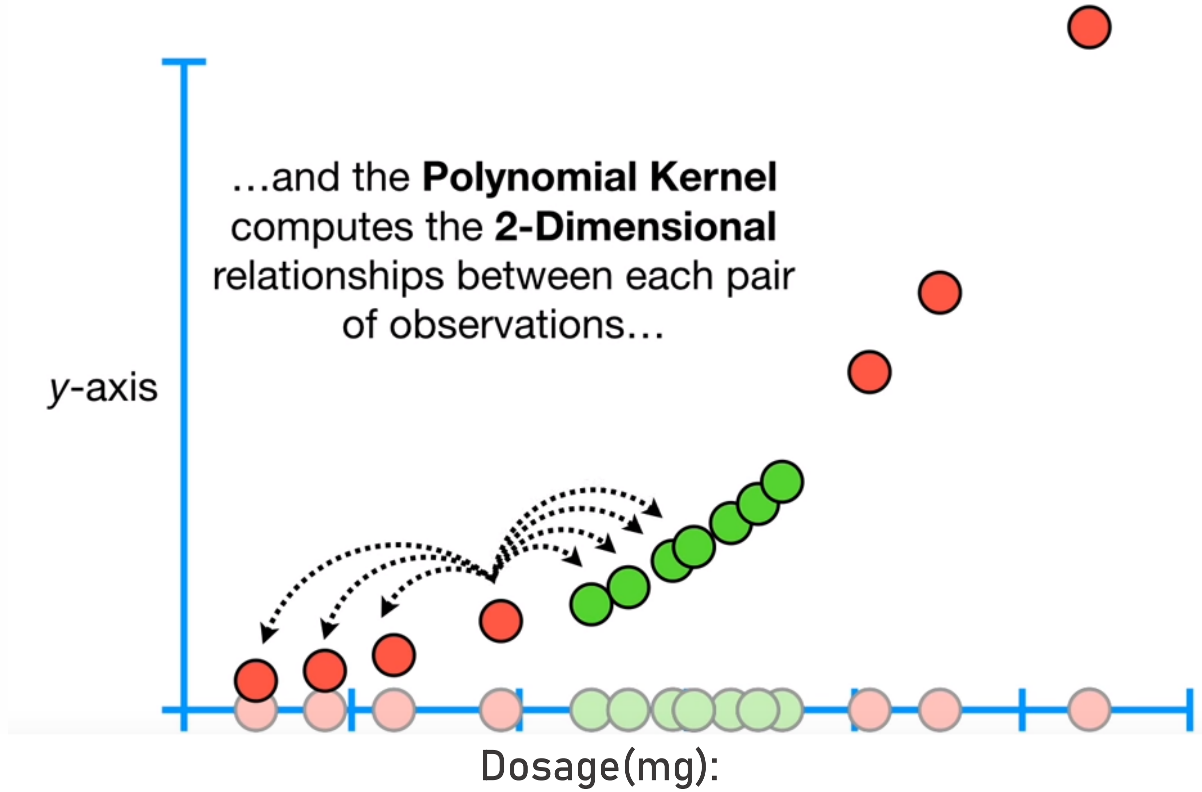
Maps the original feature space to a higher dimensional feature space



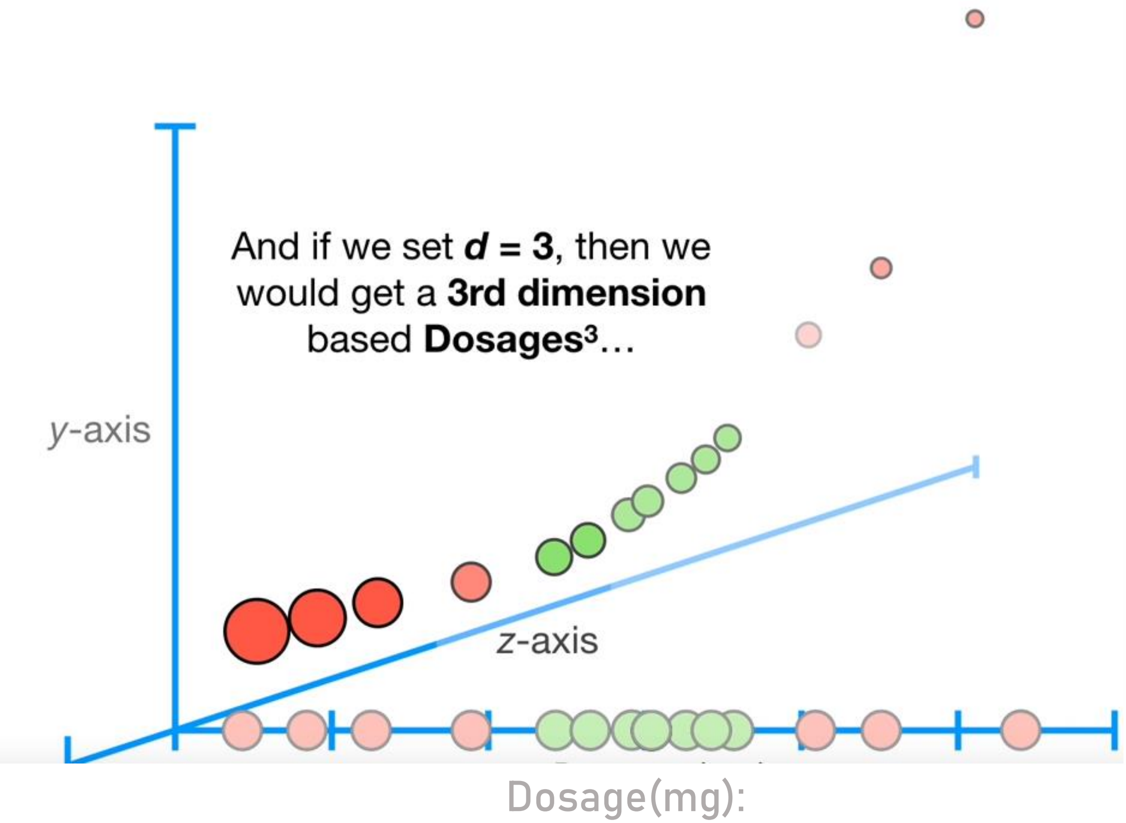
Maps the original feature space to a higher dimensional feature space



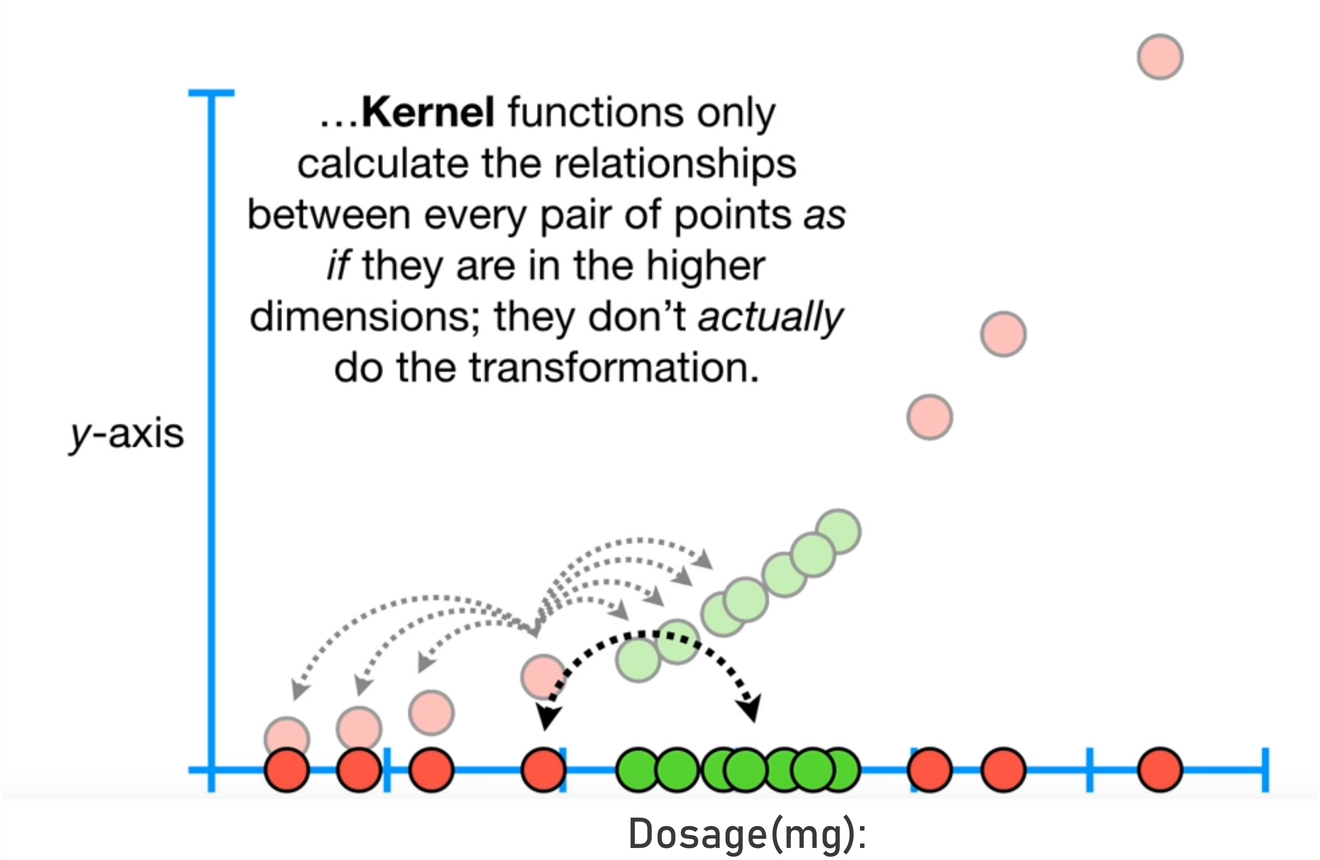
Maps the original feature space to a higher dimensional feature space



Maps the original feature space to a higher dimensional feature space



Maps the original feature space to a higher dimensional feature space



Transforms the quadratic optimization problem

Can be transformed into another optimization problem called "the Lagrangian dual problem"

$$\max_{\alpha} \min_{w,b} \frac{\|w\|}{2} + C \sum_{i=1}^{N} \alpha_i (1 - w^{\top} \phi(x_i) + b)$$

Or

Subject to: $w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)$, $0 \leq \alpha_i \leq C, i = 1, \dots, N$

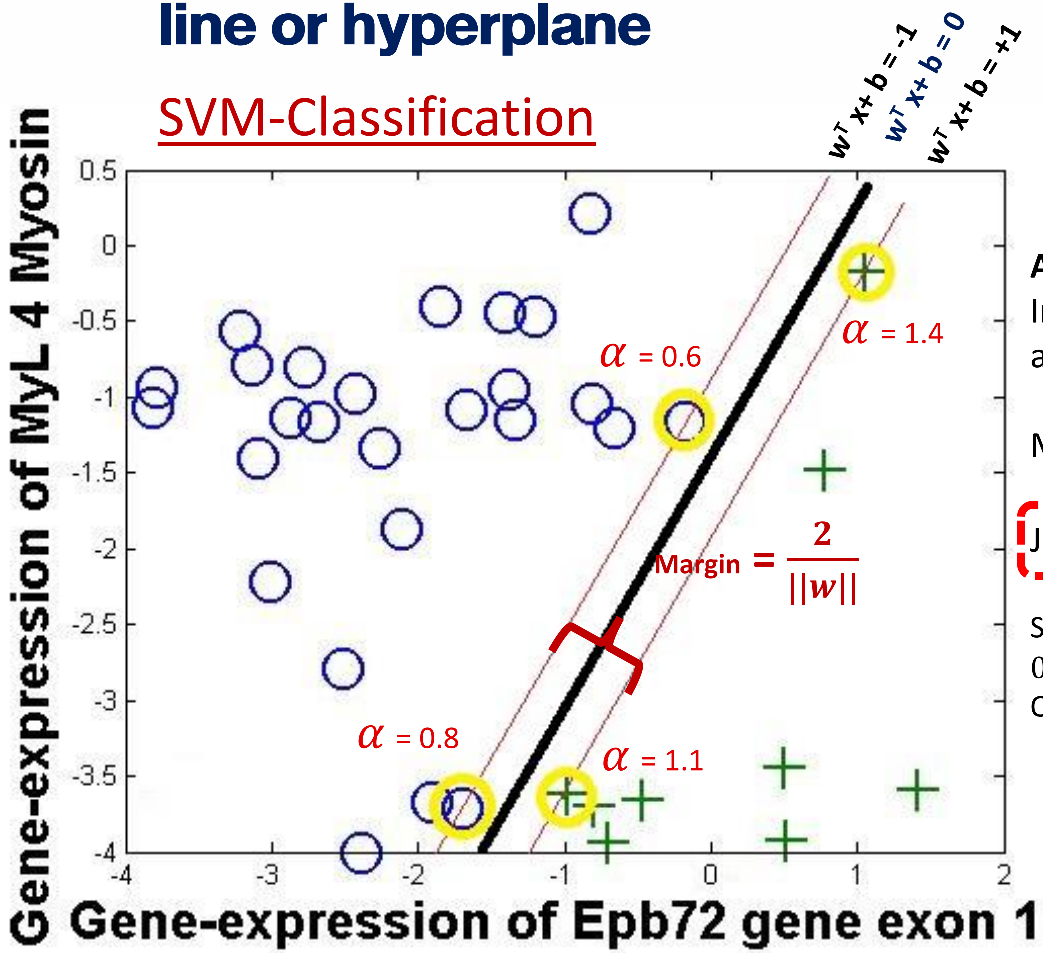
$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N} \sum_{j=1}^{N} (y_i \alpha_i \phi(x_i)^{\top} \phi(x_j) y_j \alpha_j)$$

$$\min_{\alpha} (\frac{1}{2}) \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left(\phi(x_i).\phi(x_j) + \lambda \delta_{ij}\right) - \sum_{i=1}^{N} \alpha_i$$

Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

SVM classify the two labeled-data based on the discriminant line or hyperplane

SVM-Classification



The primal form of the optimization problem, α and w are related as the dual problem

Algorithm SVM-train:
Inputs: Training examples $\{ x_1, x_2, \dots, x_i, \dots\dots x_l \}$
and class labels$\{ y_1, y_2, \dots, y_i, \dots\dots y_l \}$

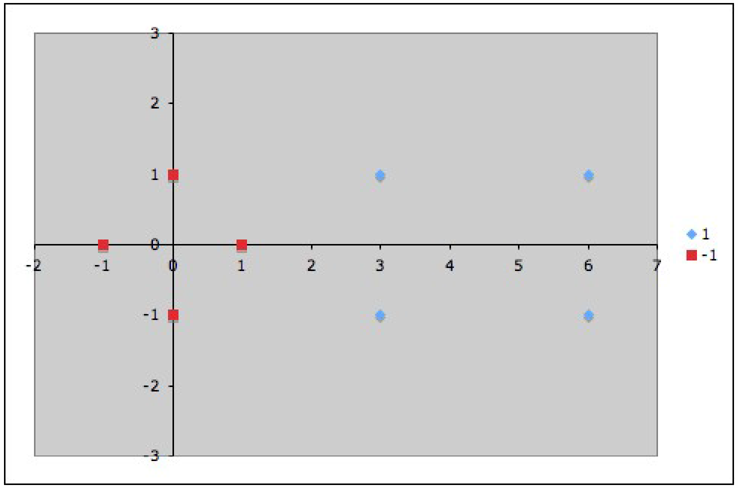
Minimize over α_i:

$$J = \left(\frac{1}{2}\right) \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left(\phi(x_i) . \phi(x_j) + \lambda \delta_{ij} \right) \quad - \sum_{i=1}^{N} \alpha_i$$

Subject to:
$0 \le \alpha_i \le C$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$
Outputs: Parameter α_i

$$w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)$$

An example 1, of linear classifiable dataset



A sample data point in $\Re^2$ *(Dan Ventura, 2009)*

Training Set : $\left\{\binom{3}{1}, \binom{3}{-1}, \binom{6}{1}, \binom{6}{-1}, \binom{1}{0}, \binom{0}{1}, \binom{0}{-1}, \binom{-1}{0}\right\}$

Training Set Labeled : { 1, 1, 1, 1, -1, -1, -1,-1 }

Testing Set: $\left\{\binom{4}{1}, \binom{1.9}{-0.8}, \binom{2.3}{1}\right\}$

Testing Set Labeled : { 1,-1, 1}

An example of linear classifiable dataset

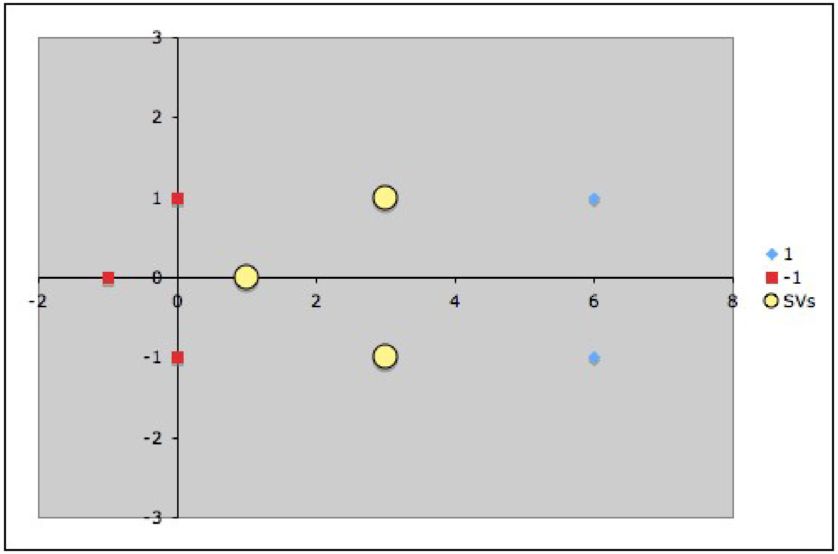
1. *Find the support vectors by random-creating lines or hyperplane based on the decision function $w \cdot x + b = 0$, and the support decision function $w \cdot x + b = 1$ for class {+1}, and the support decision function $w \cdot x + b = -1$ for class{ -1}.*

An example of linear classifiable dataset

2. By the results from step 1, the 3 support vectors are

S1 = (1,0), S2 = (3,1) and S3 = (3,-1) . Next, compute the alphas by

using the equations in step 3



(Dan Ventura, 2009)

An example of linear classifiable dataset

3. Use the alphas from step 2 to compute the w and create a hyperplane equation.

S1 = (1,0) class -1, S2 = (3,1) class +1 and S3 = (3,-1) class +1

$$\alpha_1\phi(s_1).\phi(s_1) + \alpha_2\phi(s_2).\phi(s_1) + \alpha_3\phi(s_3).\phi(s_1) = -1$$
$$\alpha_1\phi(s_1).\phi(s_2) + \alpha_2\phi(s_2).\phi(s_2) + \alpha_3\phi(s_3).\phi(s_2) = +1$$
$$\alpha_1\phi(s_1).\phi(s_3) + \alpha_2\phi(s_2).\phi(s_3) + \alpha_3\phi(s_3).\phi(s_3) = +1$$

let $\phi(\,) = I$, *and reduce to*

$$\alpha_1\tilde{s_1}.\tilde{s_1} + \alpha_2\tilde{s_2}.\tilde{s_1} + \alpha_3\tilde{s_3}.\tilde{s_1} = -1$$
$$\alpha_1\tilde{s_1}.\tilde{s_2} + \alpha_2\tilde{s_2}.\tilde{s_2} + \alpha_3\tilde{s_3}.\tilde{s_2} = +1$$
$$\alpha_1\tilde{s_1}.\tilde{s_3} + \alpha_2\tilde{s_2}.\tilde{s_3} + \alpha_3\tilde{s_3}.\tilde{s_3} = +1$$

add the bias inputs = 1 to $\tilde{s_1}, \tilde{s_2}, \tilde{s_3}$

$$\tilde{s_1} = \{1, 0, 1\}, \tilde{s_2} = \{3, 1, 1\} \; and \; \tilde{s_3} = \{3, -1, 1\}$$

An example of linear classifiable dataset

$\alpha_1\tilde{s_1}.\tilde{s_1} + \alpha_2\tilde{s_2}.\tilde{s_1} + \alpha_3\tilde{s_3}.\tilde{s_1} = -1$
$\alpha_1\tilde{s_1}.\tilde{s_2} + \alpha_2\tilde{s_2}.\tilde{s_2} + \alpha_3\tilde{s_3}.\tilde{s_2} = +1$
$\alpha_1\tilde{s_1}.\tilde{s_3} + \alpha_2\tilde{s_2}.\tilde{s_3} + \alpha_3\tilde{s_3}.\tilde{s_3} = +1$

$\tilde{s_1} = \{1, 0, 1\}, \tilde{s_2} = \{3, 1, 1\}$ *and* $\tilde{s_3} = \{3, -1, 1\}$

compute the dot products results

$\alpha_1((1 \times 1) + (0 \times 0) + (1 \times 1)) + \alpha_2((3 \times 1) + (1 \times 0) + (1 \times 1)) + \alpha_3((3 \times 1) + (-1 \times 0) + (1 \times 1)) = -1$
$\alpha_1((1 \times 3) + (0 \times 1) + (1 \times 1)) + \alpha_2((3 \times 3) + (1 \times 1) + (1 \times 1)) + \alpha_3((3 \times 3) + (-1 \times 1) + (1 \times 1)) = +1$
$\alpha_1((1 \times 3) + (0 \times -1) + (1 \times 1)) + \alpha_2((3 \times 3) + (1 \times -1) + (1 \times 1)) + \alpha_3((3 \times 3) + (-1 \times -1) + (1 \times 1)) = +1$

$2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1$
$4\alpha_1 + 11\alpha_2 + 9\alpha_3 = +1$
$4\alpha_1 + 9\alpha_2 + 11\alpha_3 = +1$

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad \left|\begin{array}{ccc|c} 2 & 4 & 4 & -1 \\ 4 & 11 & 9 & 1 \\ 4 & 9 & 11 & 1 \end{array}\right|$$

Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

An example of linear classifiable dataset

$$\left|\begin{array}{ccc|c|} 1 & 2 & 2 & -1/2 \\ 4 & 11 & 9 & 1 \\ 4 & 9 & 11 & 1 \end{array}\right. r_1/2 \quad \Leftarrow \quad \left|\begin{array}{ccc|c} \frac{2}{2}=1 & \frac{4}{2}=2 & \frac{4}{2}=2 & -\frac{1}{2} \\ 4 & 11 & 9 & 1 \\ 4 & 9 & 11 & 1 \end{array}\right.$$

$$\left|\begin{array}{ccc|c|} 1 & 2 & 2 & -1/2 \\ 0 & 3 & 1 & 3 \\ 0 & 1 & 3 & 3 \end{array}\right. \begin{array}{l} \\ r_2 - 4r_1 \\ r_3 - 4r_1 \end{array} \quad \Leftarrow \quad \left|\begin{array}{ccc|c} 1 & 2 & 2 & -1/2 \\ 4-4(1)=0 & 11-4(2)=3 & 9-4(2)=1 & 1-4(-1/2)=3 \\ 4-4(1)=0 & 9-4(2)=1 & 11-4(2)=3 & 1-4(-1/2)=3 \end{array}\right.$$

$$\left|\begin{array}{ccc|c|} 1 & 2 & 2 & -1/2 \\ 0 & 1 & 1/3 & 1 \\ 0 & 1 & 3 & 3 \end{array}\right. r_2/3$$

$$\left|\begin{array}{ccc|c|} 1 & 2 & 2 & -1/2 \\ 0 & 1 & 1/3 & 1 \\ 0 & 0 & 8/3 & 2 \end{array}\right. r_3 - r_2$$

$$\left|\begin{array}{ccc|c|} 1 & 2 & 2 & -1/2 \\ 0 & 1 & 1/3 & 1 \\ 0 & 0 & 1 & 3/4 \end{array}\right. r_3 \times 3/8$$

An example of linear classifiable dataset

$$\left|\begin{array}{ccc|c} 1 & 2 & 0 & -2 \\ 0 & 1 & 0 & 3/4 \\ 0 & 0 & 1 & 3/4 \end{array}\right| \begin{array}{c} r_1 - 2r_3 \\ r_2 - (1/3)r_3 \\ \\ \end{array}$$

$$\downarrow$$

$$\left|\begin{array}{ccc|c} 1 & 0 & 0 & -7/2 \\ 0 & 1 & 0 & 3/4 \\ 0 & 0 & 1 & 3/4 \end{array}\right| \begin{array}{c} r_1 - 2r_2 \\ \\ \\ \end{array}$$

$$\downarrow$$

$$\alpha_1 = -3.5,\ \alpha_2 = 0.75,\ \alpha_3 = 0.75$$

$$\downarrow$$

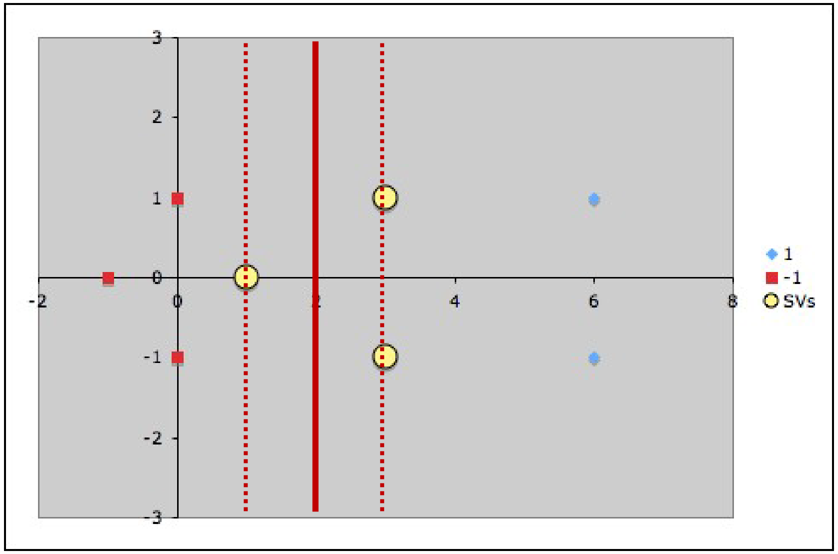
$$w = \sum_{i=1}^{N} \alpha_i \tilde{s}_i = -3.5\begin{pmatrix}1\\0\\1\end{pmatrix} + 0.75\begin{pmatrix}3\\1\\1\end{pmatrix} + 0.75\begin{pmatrix}3\\-1\\1\end{pmatrix} = \begin{pmatrix}1\\0\\-2\end{pmatrix}$$

$$\downarrow$$

From $y = w^\top x_i + b$ then $w = \binom{1}{0}$ and b = -2

An example of linear classifiable dataset

$w = \binom{1}{0}$ and b = -2



How to apply SVM to categorical data

2 sub-types of categorical features: Ordinal and nominal

Ordinal features example:

- a patient satisfaction metric {'satisfied', 'neutral', 'dissatisfied'}

is a ordinal variable since we can order it: 'satisfied' > 'neutral' > 'dissatisfied'

we can simply map the 'string' notation into an integer notation, for example

'satisfied'=1, 'neutral' =0, and 'dissatisfied'= -1.

How to apply SVM to categorical data

2 sub-types of categorical features: Ordinal and nominal

Nominal features example:

- *think of 'color'; there are some cases in image processing where ordering color values makes sense, but for simplicity, we can't say 'red > blue > yellow'*

- *To deal with such variables in SVM classification, we typically do a "one-hot" encoding*

A "one-hot" encoding for SVM

Nominal features: 'red > blue > yellow'

- *Create one dummy variable for each possible value of that nominal feature variable*
- *Our color variable can have one of the three values: 'red,' 'blue,' 'yellow.'*

	blue	red	yellow
sample 1	1	0	0
sample 2	0	0	1
sample 3	0	1	0
sample 4	0	0	1

Note:

- For numerical data

  no consider about create more dimension

- For nominal category data – sex = [male, female]

  need to create more 2-dimensions male {1,0} female {0,1}

- For ordinal category data salary = [low, medium, high]

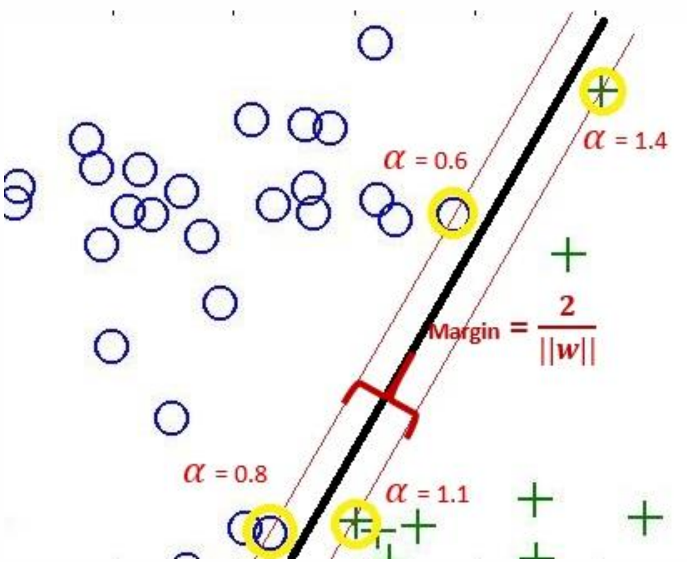
  no need to create another 3-dimensions

  can apply dummy data = 0, 1, 2

Pros and Cons of SVM

Pros:

1. SVM accurate in high dimensional spaces.
2. SVM uses a subset of training points in the decision function (called support vectors), so it's also memory efficient.



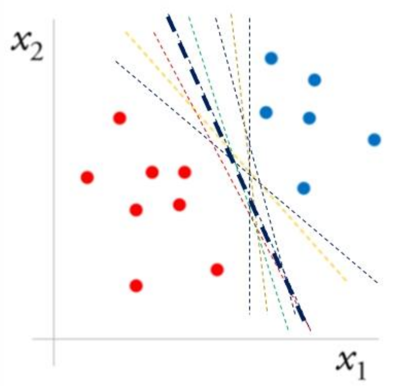
Minimize α_i to find w

$$\boxed{w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)}$$

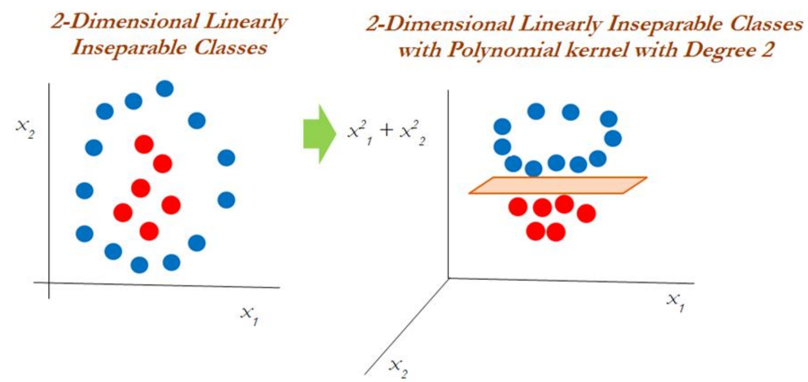
Pros and Cons of SVM

Pros:

3. SVM can guarantee optimality.



4. SVM is useful for both Linearly Seperable(hard margin) and Non-linearly Separable(soft margin) data



https://hub.packtpub.com/what-is-a-support-vector-machine/

Pros and Cons of SVM

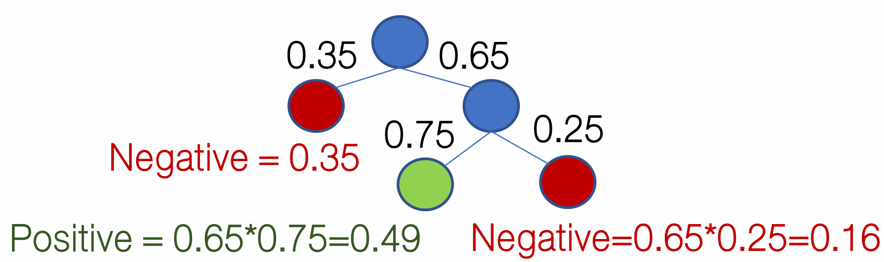
Cons:

1. SVM is prone for over-fitting, if the number of features is much greater than the number of samples.
2. SVM do not directly provide probability estimates, which are desirable in most classification problems.

SVM

$$\mathbf{w}^{T}\,\mathbf{x} + \mathbf{b} = \begin{cases} -1 \rightarrow class - 1 \\ 0 \rightarrow class + 1 \\ 1 \rightarrow class + 1 \end{cases}$$

Decision Tree



3. SVM is not very efficient computationally, if your dataset is very big, such as when you have more than one thousand rows.

Implementing SVM with Python

https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/

Implementing SVM with Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df_colon = pd.read_csv(r'H:\Coding_python\colon.csv')
df_colon.shape
df_colon.head()
```

(62, 2001)

	H55933	R39465	R39465.1	R85482	...	H40891	R77780	T49647	Class
0	3.62	3.31	2.986154	2.71	...	-0.315	-1.764190	-2.75	1
1	3.47	3.68	3.425553	3.05	...	-1.210	-1.062064	-2.13	1
2	3.02	2.78	2.569772	3.21	...	-1.010	-2.260031	-1.50	1
3	3.10	2.86	2.772942	3.19	...	-1.610	-1.223450	-1.07	1
4	3.01	2.91	2.560548	3.25	...	-1.210	-1.232686	-1.62	1

Data Preprocessing

Data preprocessing involves

(1) Dividing the data into attributes and labels

```
X = df_colon.drop('Class', axis=1)
y = df_colon['Class']
```

```
X.shape
```

```
(62, 2000)
```

```
X.head()
```

	H55933	R39465	R39465.1	R85482	...	H40891	R77780	T49647
0	3.62	3.31	2.986154	2.71	...	-0.315	-1.764190	-2.75
1	3.47	3.68	3.425553	3.05	...	-1.210	-1.062064	-2.13
2	3.02	2.78	2.569772	3.21	...	-1.010	-2.260031	-1.50
3	3.10	2.86	2.772942	3.19	...	-1.610	-1.223450	-1.07
4	3.01	2.91	2.560548	3.25	...	-1.210	-1.232686	-1.62

Data Preprocessing

(2) dividing the data into training and testing sets.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

X_train.shape	(49, 2000)
X_test.shape	(13, 2000)
y_train.shape	(49,)
y_test.shape	(13,)

Training the Algorithm

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
```

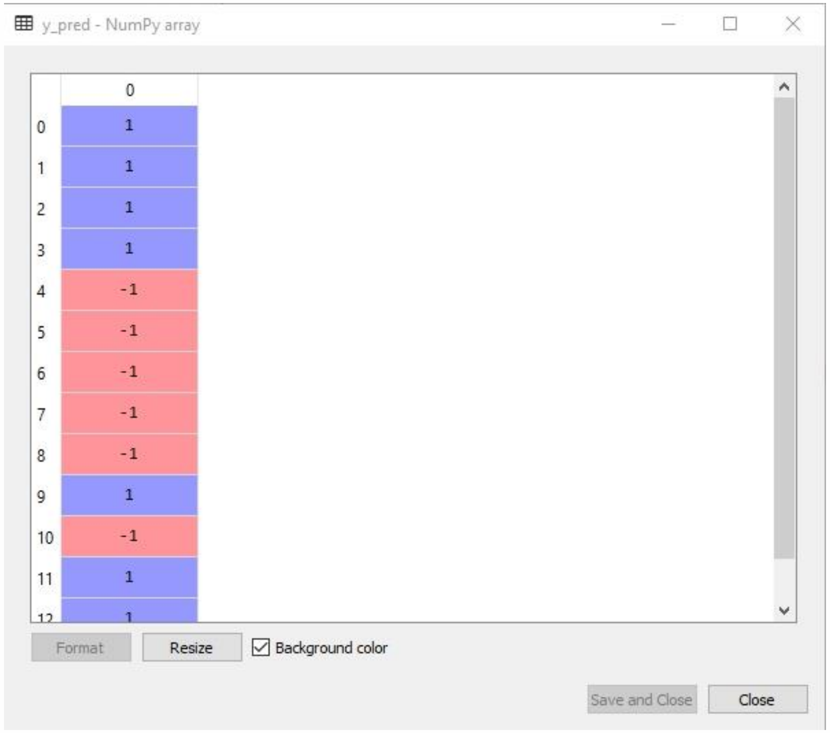
support vector classifier class

Linear classifier

Making Predictions

y_pred = svclassifier.predict(X_test)

Predict y by using X_test



Evaluating the Algorithm

```
from sklearn.metrics import classification_report, confusion_matrix
confusion = confusion_matrix(y_test,y_pred)
print(confusion)
print(classification_report(y_test,y_pred))
```

```
[6 3]
[0 4]
```

	precision	recall	f1-score	support
-1	1.00	0.67	0.80	9
1	0.57	1.00	0.73	4
micro avg	0.77	0.77	0.77	13
macro avg	0.79	0.83	0.76	13
weighted avg	0.87	0.77	0.78	13

Evaluating the Algorithm

```
#edit target name
target_names = ['Cancer', 'Healthy']
print(classification_report(y_test, y_pred, target_names=target_names))
```

	precision	recall	f1-score	support
Cancer	1.00	0.67	0.80	9
Healthy	0.57	1.00	0.73	4
micro avg	0.77	0.77	0.77	13
macro avg	0.79	0.83	0.76	13
weighted avg	0.87	0.77	0.78	13

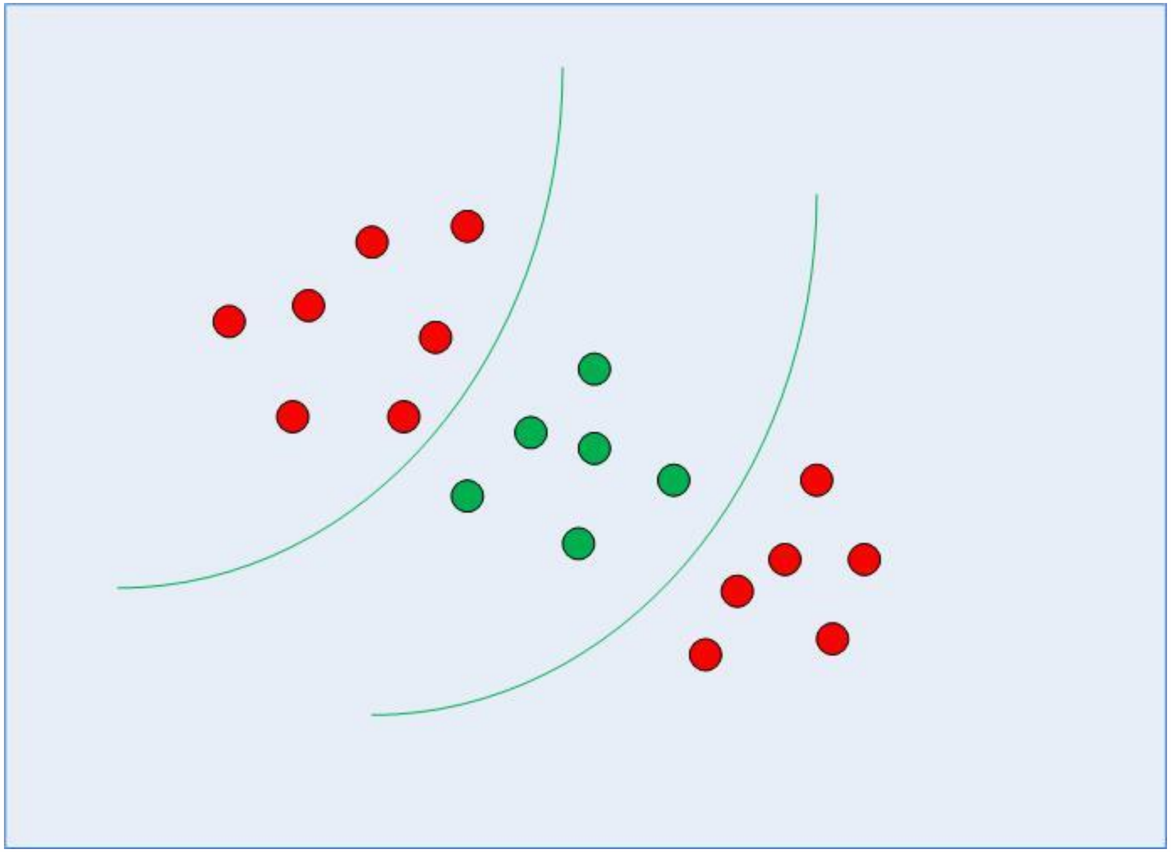
Evaluating the Algorithm

```
# True Positives
TP = confusion[1, 1]
# True Negatives
TN = confusion[0, 0]
# False Positives
FP = confusion[0, 1]
# False Negatives
FN = confusion[1, 0]
print('accuracy: ', (TP + TN) / float(TP + TN + FP + FN))
print('sensitivity: ',TP / float(TP + FN))
print('specificity: ',TN / float(TN + FP))
```

```
accuracy:  0.7692307692307693
sensitivity:  1.0
specificity:  0.6666666666666666
```

Implementing Kernel SVM with Scikit-Learn

Non-linearly Separable Data



https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/

Kernel SVM with Scikit-Learn

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Assign colum names to the dataset
colnames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']

# Read dataset to pandas dataframe
irisdata = pd.read_csv(url, names=colnames)
```

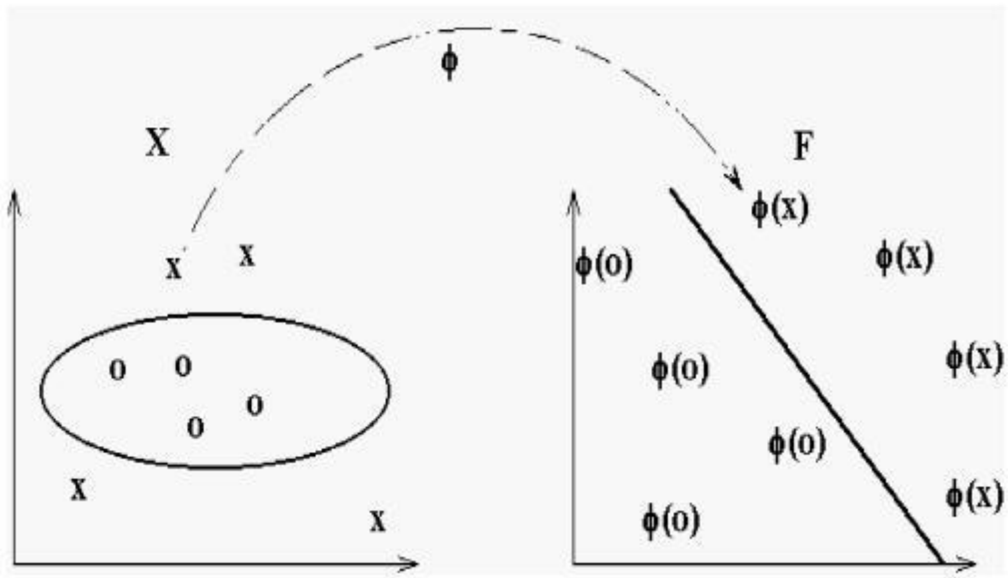
URL for downloading iris.data

Kernel SVM with Scikit-Learn

```
#Preprocessing
X = irisdata.drop('Class', axis=1)
y = irisdata['Class']

#Train Test Split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

Polynomial Kernel SVM with Scikit-Learn



https://en.wikipedia.org/wiki/Polynomial_kernel#/media/File:Svm_8_polinomial.JPG

Polynomial Kernel SVM with Scikit-Learn

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='poly', degree=8)
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Kernel = Polynomial

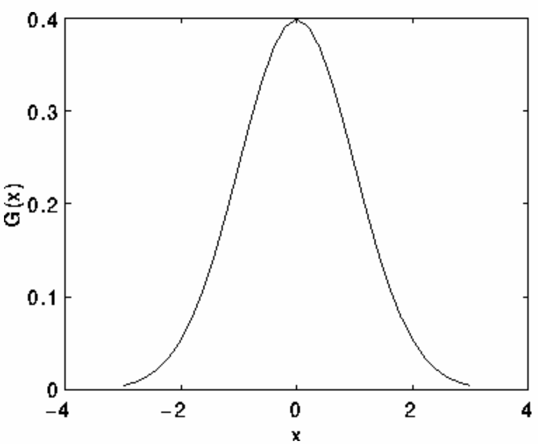
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

```
[[ 7  0  0]
 [ 0 11  0]
 [ 0  3  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.79	1.00	0.88	11
Iris-virginica	1.00	0.75	0.86	12
micro avg	0.90	0.90	0.90	30
macro avg	0.93	0.92	0.91	30
weighted avg	0.92	0.90	0.90	30

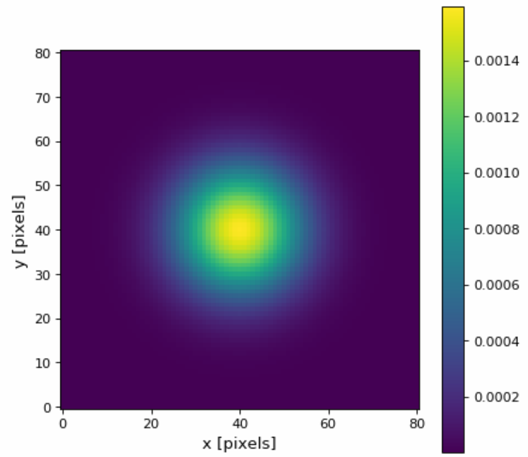
Gaussian Kernel SVM with Scikit-Learn

1 Dimension



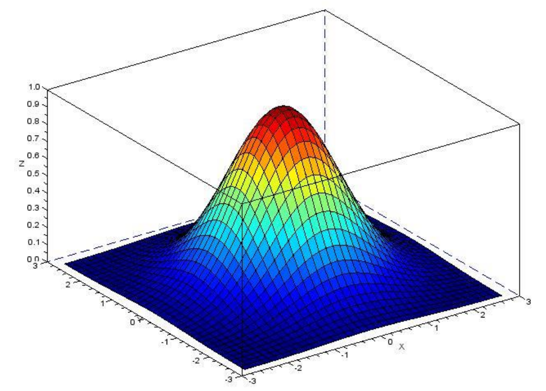
https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

2 Dimensions



https://docs.astropy.org/en/stable/api/astropy.convolution.Gaussian2DKernel.html

3 Dimensions



https://jamesmccaffrey.files.wordpress.com/2014/01/gaussiankernel.jpg

Gaussian Kernel SVM with Scikit-Learn

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Kernel = the radial basis function

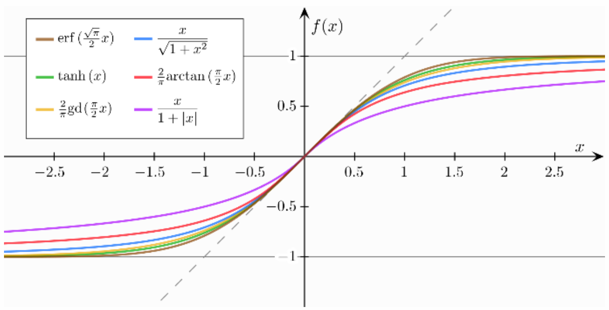
```
[[ 7  0  0]
 [ 0 10  1]
 [ 0  0 12]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	1.00	0.91	0.95	11
Iris-virginica	0.92	1.00	0.96	12
micro avg	0.97	0.97	0.97	30
macro avg	0.97	0.97	0.97	30
weighted avg	0.97	0.97	0.97	30

Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

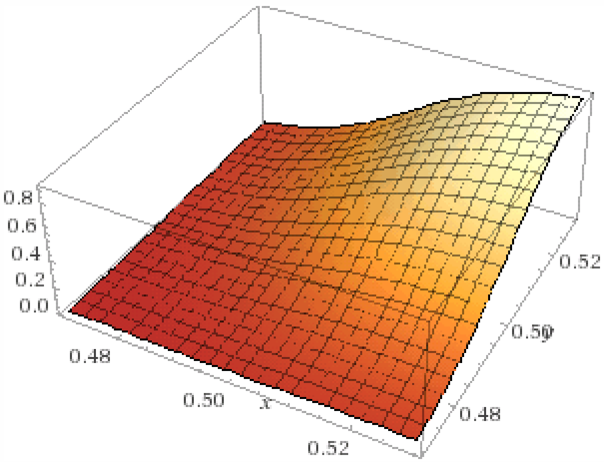
Sigmoid Kernel SVM with Scikit-Learn

1 Dimension



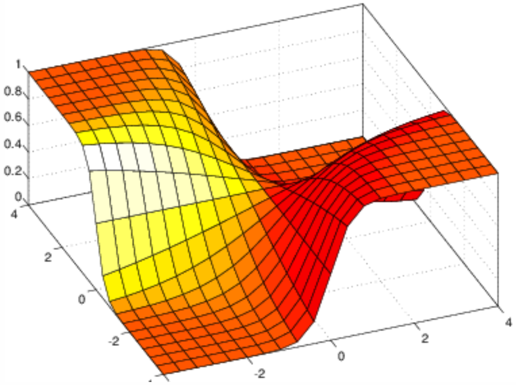
https://en.wikipedia.org/wiki/Sigmoid_function#/media/File:Gjl-t(x).svg

2 Dimensions



https://math.stackexchange.com/questions/863662/need-function-for-2d-sigmoid-shaped-monotonic-surface

3 Dimensions



https://www.researchgate.net/figure/3D-classical-sigmoid-function-f-x-T_fig1_221165140

Wisdom of the Land

Sigmoid Kernel SVM with Scikit-Learn

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='sigmoid')
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Kernel = sigmoid

```
[[ 7  0  0]
 [11  0  0]
 [12  0  0]]
```

	precision	recall	f1-score	support
Iris-setosa	0.23	1.00	0.38	7
Iris-versicolor	0.00	0.00	0.00	11
Iris-virginica	0.00	0.00	0.00	12
micro avg	0.23	0.23	0.23	30
macro avg	0.08	0.33	0.13	30
weighted avg	0.05	0.23	0.09	30

Comparison of Kernel Performance

Kernel = Polynomial

[[7 0 0]
[0 11 0]
[0 3 9]]

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.79	1.00	0.88	11
Iris-virginica	1.00	0.75	0.86	12
micro avg	0.90	0.90	0.90	30
macro avg	0.93	0.92	0.91	30
weighted avg	0.92	0.90	0.90	30

Kernel = Gaussian

[[7 0 0]
[0 10 1]
[0 0 12]]

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	1.00	0.91	0.95	11
Iris-virginica	0.92	1.00	0.96	12
micro avg	0.97	0.97	0.97	30
macro avg	0.97	0.97	0.97	30
weighted avg	0.97	0.97	0.97	30

Kernel = sigmoid

[[7 0 0]
[11 0 0]
[12 0 0]]

	precision	recall	f1-score	support
Iris-setosa	0.23	1.00	0.38	7
Iris-versicolor	0.00	0.00	0.00	11
Iris-virginica	0.00	0.00	0.00	12
micro avg	0.23	0.23	0.23	30
macro avg	0.08	0.33	0.13	30
weighted avg	0.05	0.23	0.09	30

Tuning Hyper Parameters

Tuning Hyper Parameters

```
from sklearn.model_selection import GridSearchCV
# Set the parameters by cross-validation
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-2, 1e-3, 1e-4, 1e-5],
                     'C': [0.001, 0.10, 0.1, 10, 25, 50, 100, 1000]},
                    {'kernel': ['sigmoid'], 'gamma': [1e-2, 1e-3, 1e-4, 1e-5],
                     'C': [0.001, 0.10, 0.1, 10, 25, 50, 100, 1000]},
                    {'kernel': ['linear'], 'C': [0.001, 0.10, 0.1, 10, 25, 50, 100, 1000]}
                   ]

scores = ['precision', 'recall']
```

Tuning Hyper Parameters

```
for score in scores:
    print("# Tuning hyper-parameters for %s" % score)
    print()

    clf = GridSearchCV(SVC(C=1), tuned_parameters, cv=5,
                       scoring='%s_macro' % score)
    clf.fit(X_train, y_train)
```

Tuning Hyper Parameters

```
print("Best parameters set found on development set:")
print()
print(clf.best_params_)
print()
print("Grid scores on development set:")
print()
means = clf.cv_results_['mean_test_score']
stds = clf.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, clf.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))
print()
```

Tuning Hyper Parameters

```
Best parameters set found on development set:

{'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}

Grid scores on development set:

0.306 (+/-0.026) for {'C': 0.001, 'gamma': 0.01, 'kernel': 'rbf'}
0.306 (+/-0.026) for {'C': 0.001, 'gamma': 0.001, 'kernel': 'rbf'}
0.306 (+/-0.026) for {'C': 0.001, 'gamma': 0.0001, 'kernel': 'rbf'}
…..
```

Tuning Hyper Parameters

```
# Tuning hyper-parameters for recall


Best parameters set found on development set:
{'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
Grid scores on development set:


0.500 (+/-0.000) for {'C': 0.001, 'gamma': 0.01, 'kernel': 'rbf'}
0.500 (+/-0.000) for {'C': 0.001, 'gamma': 0.001, 'kernel': 'rbf'}
0.500 (+/-0.000) for {'C': 0.001, 'gamma': 0.0001, 'kernel': 'rbf'}
…
```

Assignment:

SVM - due on 11 November, 2022 (10 points)

1. (3 points)

From data: (-2,1) class 1, (-2,-1) class -1, (-1,-1.5) class -1,
(1,1) class 1, (1.5, -0.5) class 1, (2,-2) class -1

Find a vector *w* and bias *b, please show the calculation step by step as same as example 1*

If the support vectors are (1.5, -0.5) and (2,-2)

2. (3 points)

Create a SVM-model and plot a 2D-SVM classification by using Python and colon data set (use only two genes, T62947 and H64807), and find your best hyper-parameters for precision, recall, and accuracy. (Training:Testing = 80:20)

3. (4 points)

Train a SVM-model by using colon-data set and tuning the hyper-parameters, and select the best model. (Training:Testing = 80:20) and give your comments.