Support vector machines (SVM)

Ruxandra Stoean

Additional bibliography

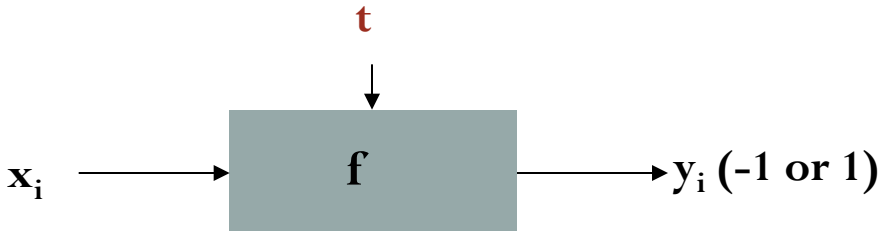
- Simon O. Haykin , Neural Networks and Learning Machines (3rd Edition), Prentice Hall, 2008
- Bernhard Scholkopf, Alexander J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, 2018
- Catalin Stoean, Ruxandra Stoean, Support Vector Machines and Evolutionary Algorithms for Classification: Single or Together?, Intelligent Systems Reference Library, Volume 69, Springer, 2014.

What is an SVM?

- Very powerful tools in:
  - Classification
    - Handwriting recognition
    - Object recognition
    - Speech recognition
    - Text categorization
    - Generically binary – methods to extend to more classes
  - Regression
- A type of supervised learning machine

- **Training phase: learning** *(example -> label)*
- **Test phase: prediction** *(new example -> ?)*

Binary classification



- A training data set

$$\{(x_i, y_i)\}_{i=1,2,\ldots,m} \qquad x_i \in R^n \qquad y_i \in \{-1,+1\}$$

- A family of functions

$$\{f_t \mid t \in T\} \qquad f_t : \Re^n \to \{-1,+1\}$$

- Find parameters t to learn the optimal correspondence between every x and y

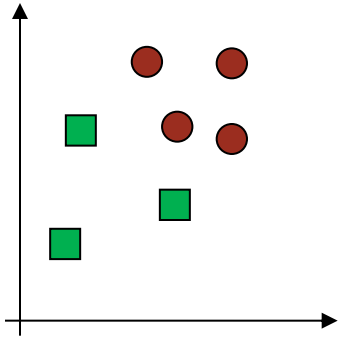
Learning within SVM

Problem: Will I get a good mark in this exam?

Yes

No

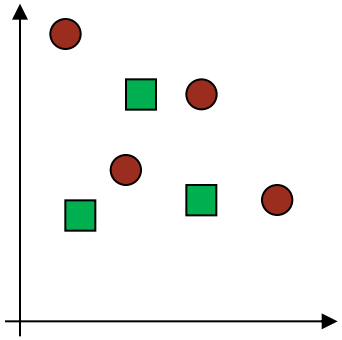
y = Hours spent on final project



x = Number of tasks solved for homework

Linear separable

Linear support vector machines



denotes +1

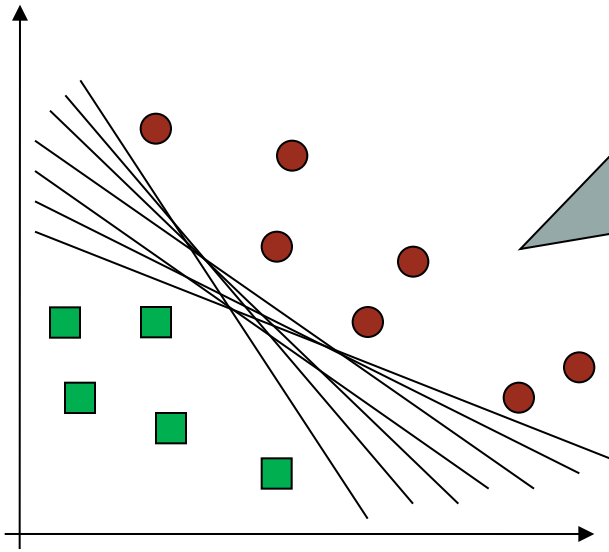
denotes -1

Linear non-separable

Linear support vector machines

Nonlinear support vector machines

Linear SVM for separable data



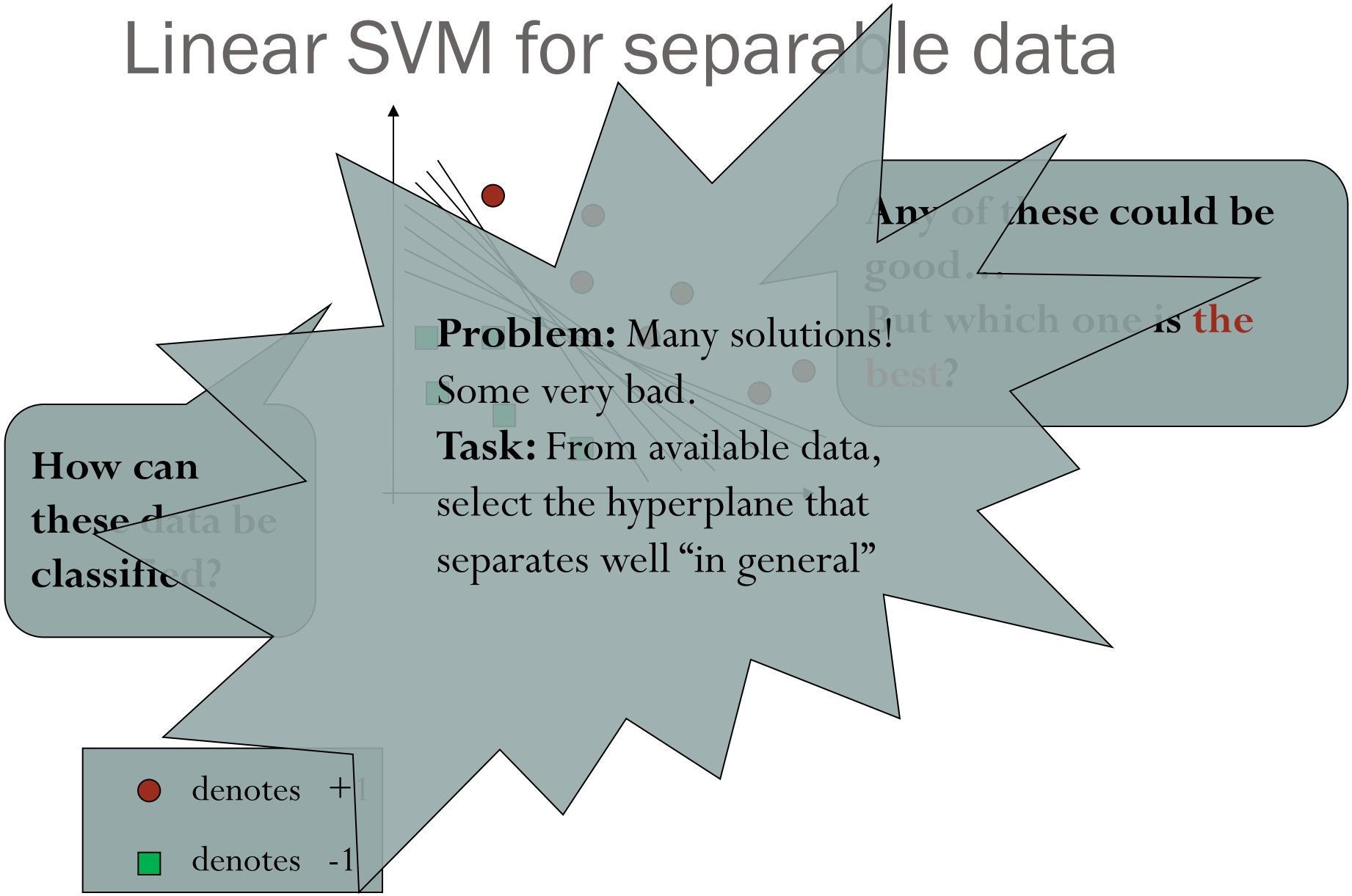
How can these data be classified?

Any of these could be good…
But which one is the best?

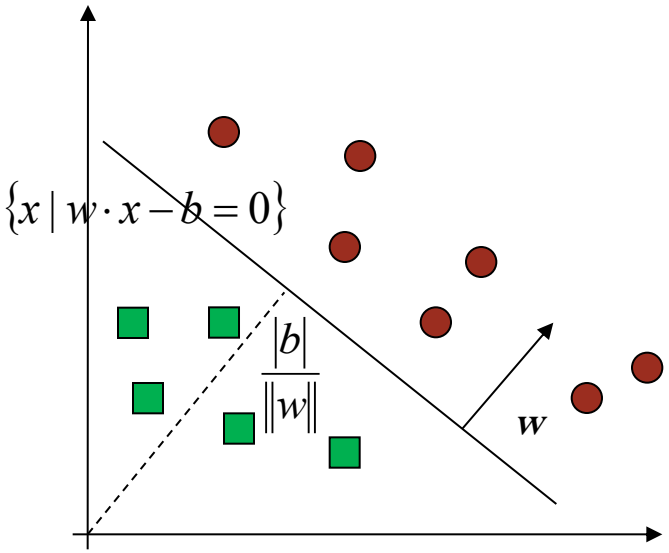
denotes +1

denotes -1

Linear SVM for separable data



Linear SVM for separable data



The existence of a hyperplane of separation with the equation:

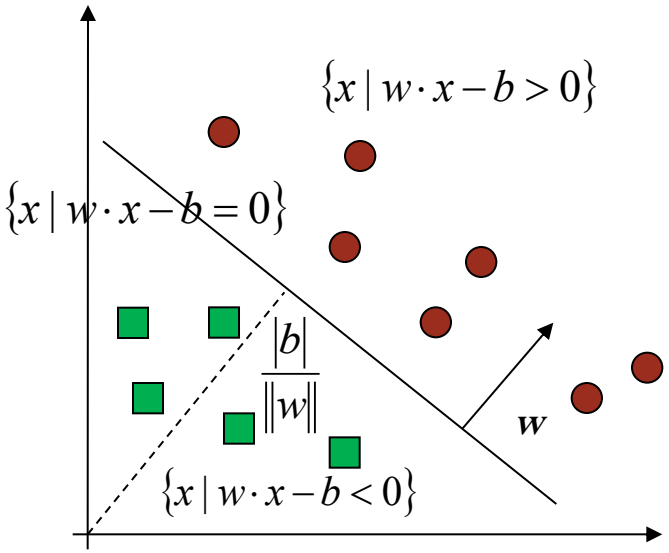
$$\langle w, x \rangle - b = 0$$

denotes +1

denotes -1

- w – vector of weights
- b – bias

Linear SVM for separable data



Two subsets are linearly separable iff there exists $w \in R^n$ and $b \in R$ such that:

$$\begin{cases} w \cdot x_i - b \geq 0, y_i = +1 \\ w \cdot x_i - b \leq 0, y_i = -1 \end{cases}$$

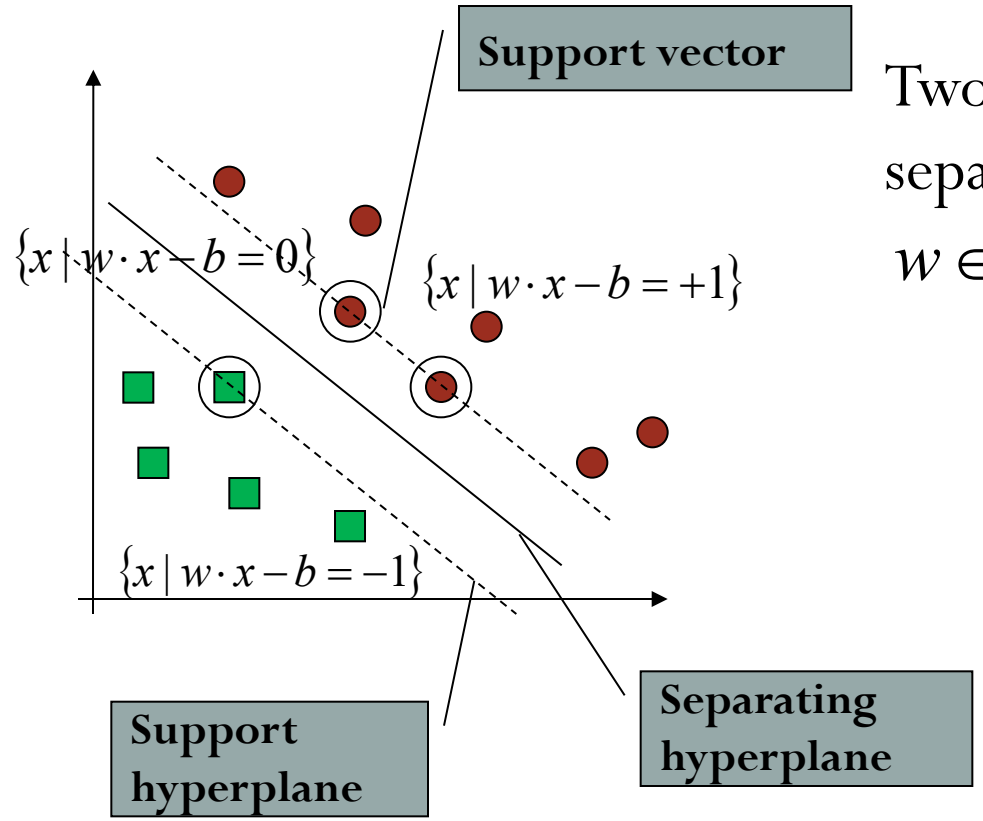
$$i = 1,2,\ldots,m$$

denotes +1

denotes -1

- w – vector of weights
- b – bias

Linear SVM for separable data



Two subsets are linearly separable iff there exists $w \in R^n$ and $b \in R$ such that:

$$\begin{cases} w \cdot x_i - b \geq +1, y_i = +1 \\ w \cdot x_i - b \leq -1, y_i = -1 \end{cases}$$

$$i = 1,2,\ldots,m$$

Linear SVM for separable data

- A family of functions:

$$\left\{f_{w,b} \mid w \in \Re^n, b \in \Re\right\} \qquad f_{w,b} : \Re^n \to \{-1, 1\} \qquad f_{w,b}(x) = \operatorname{sgn}\left(w \cdot x - b\right)$$

- A training data set:

$$\left\{(x_i, y_i)\right\}_{i=1,2,\ldots,m} \qquad x_i \in R^n \qquad y_i \in \{-1,+1\}$$

- Learns the optimal corespondence

Principle of risk minimization

- For a learning task, with a finite quantity of training data, a machine is optimal if:
  - It has a good accuracy on the given training data set
  - It has the capacity to learn other training sets without error (good generalization ability)

Optimal hyperplane in linear SVM for separable data

$w = ?$ and $b = ?$ *a.i.*

1. **Accuracy** $y_i(w \cdot x_i - b) - 1 \geq 0 \quad i = 1,2,\ldots,m$

2. **Maximal margin of separation**

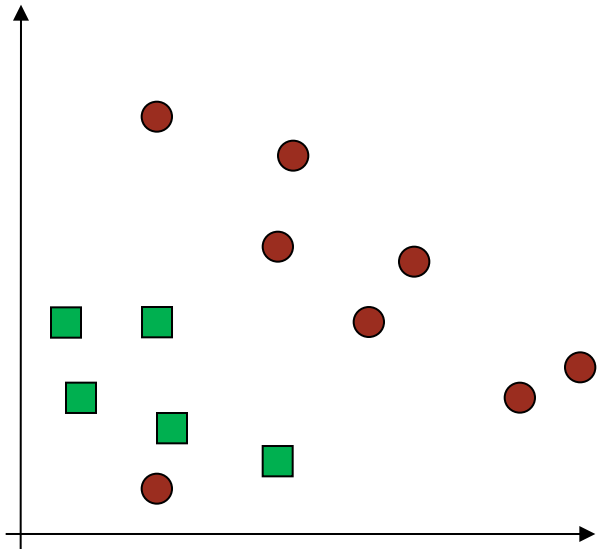
The width by which the border between classes can be enlarged until hitting an example

⇨ **minimize** $\frac{\|w\|^2}{2}$ **Why?**

The distance from each closest point to the separating hyperplane from both of its sides is:

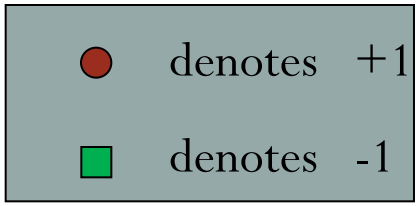
$$\frac{|w \cdot x_i - b|}{\|w\|} = \frac{1}{\|w\|} \quad \Rightarrow \frac{2}{\|w\|} \longrightarrow \text{maximum}$$

Linear SVM for non-separable data

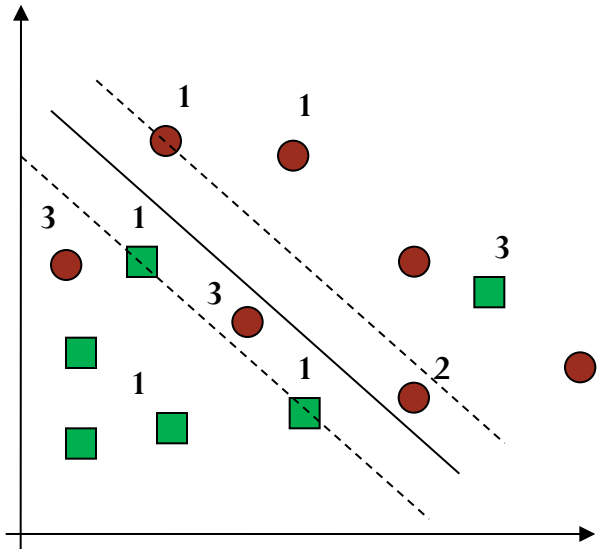


How does one separate these new data?

The separation condition can be relaxed.



Linear SVM for non-separable data



Each training vector has a deviation from its support hyperplane of

$$\pm \frac{\xi_i}{\|w\|} \qquad \xi_i \geq 0$$

$\xi_i = 0$	**1 Correct**
$\xi_i < 1$	**2 Correct (in margin)**
$\xi_i > 1$	**3 Error**

denotes +1

denotes -1

Optimal hyperplane in linear SVM for non-separable data

- The constraints are relaxed:

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i$$

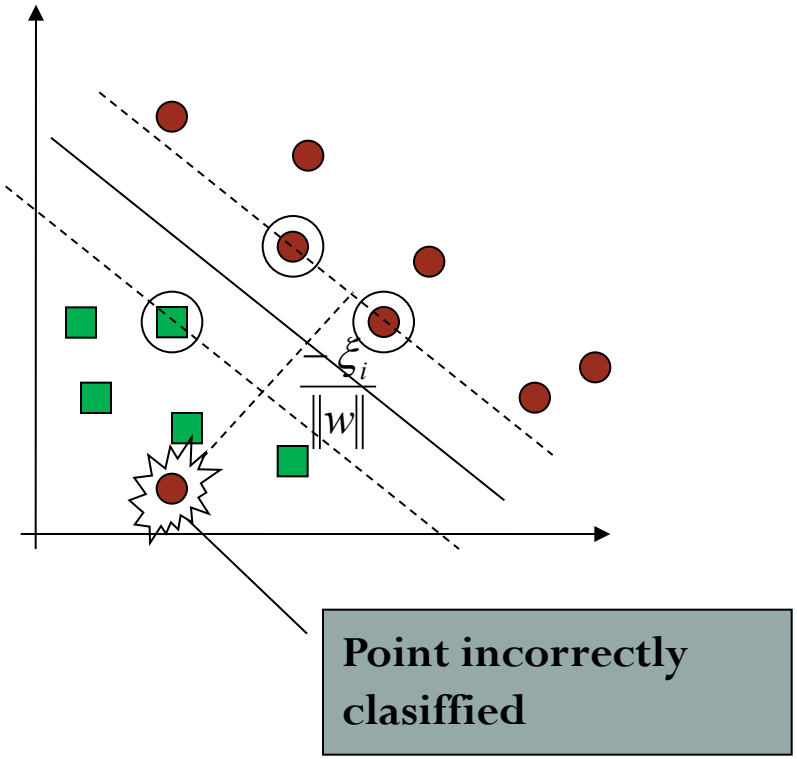
$$\xi_i \geq 0 \quad i = 1,2,\ldots,m$$

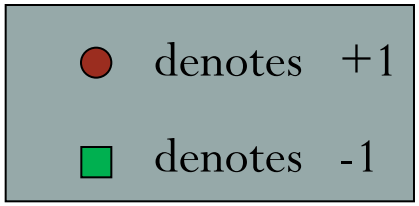
- The sum of misclassifications must be minimized together with maximizing the margin of separation:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$$

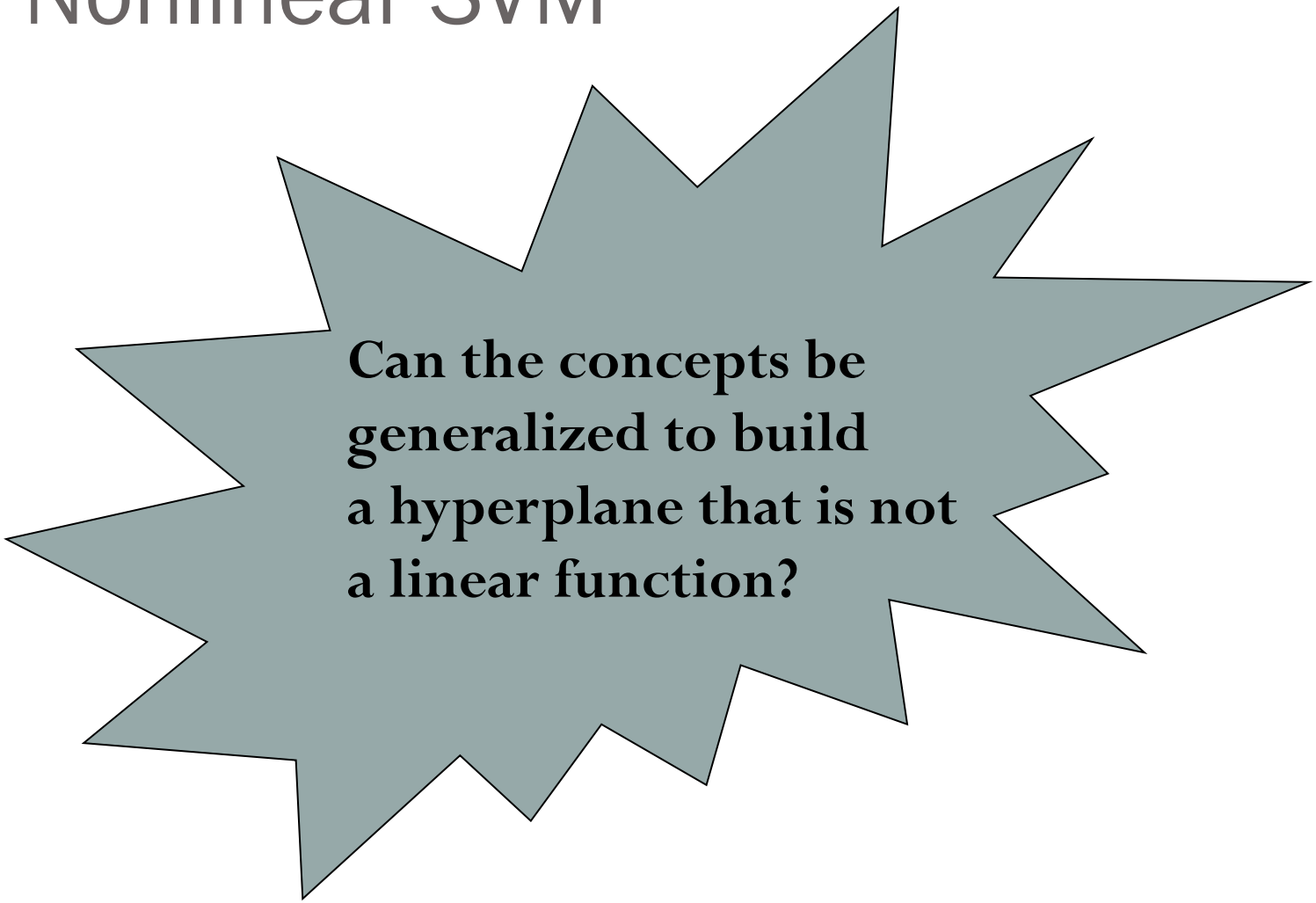
  - Regularization parameter $C > 0$
    - Trade-off between achieving a low training error and high generalizability
    - High value: smaller margin, higher training accuracy
    - Low value: larger margin, lower training accuracy

Linear SVM for non-separable data





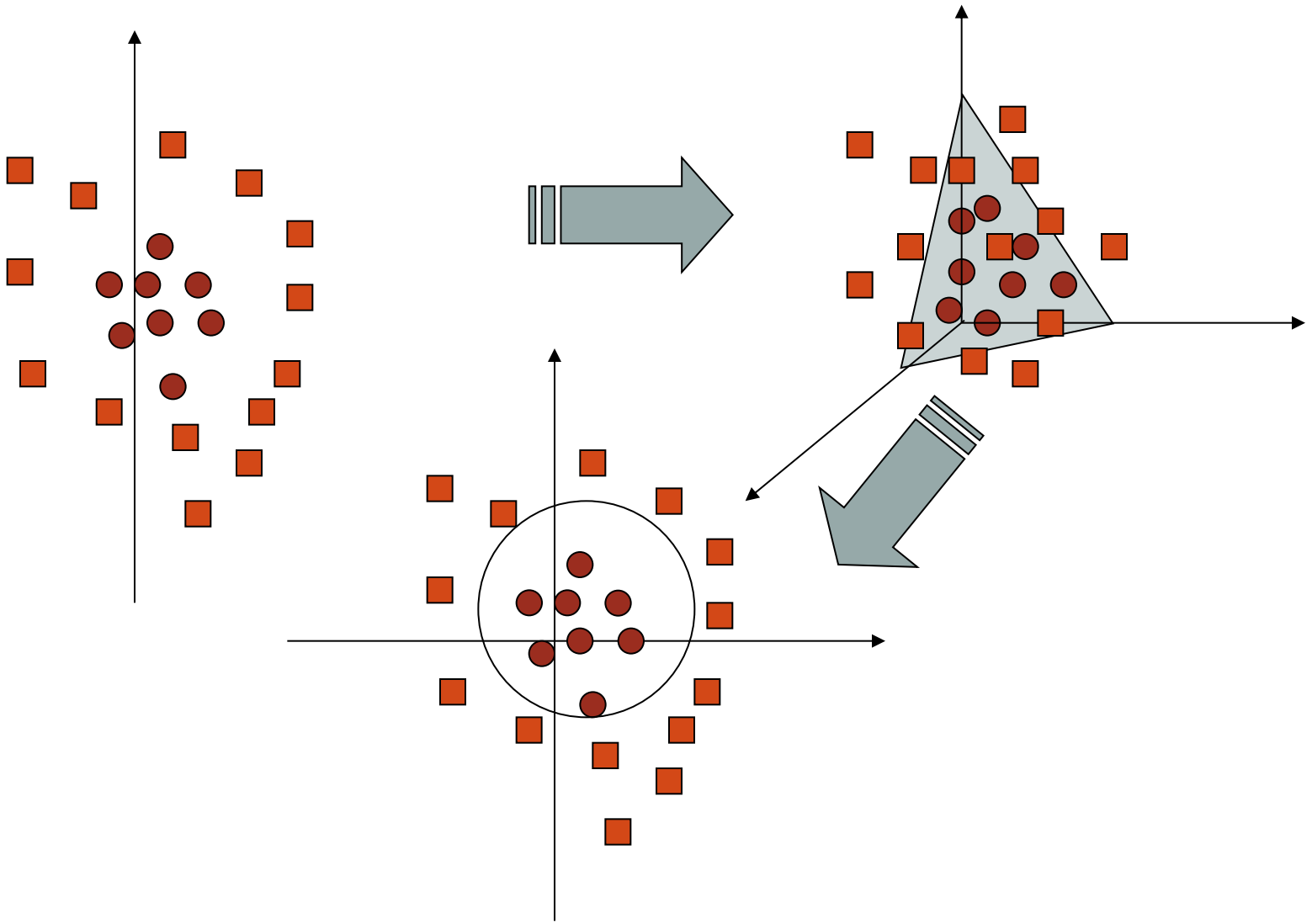
Nonlinear SVM



Nonlinear SVM

- An input space can be transformed in a new feature space, where data are linearly separable if:
  - the transformation is nonlinear
  - the dimensionality of the new space is sufficiently large

Nonlinear SVM



Nonlinear transformation

x – example from the input space

$\Phi : \Re^n \rightarrow H$ - non-linear transformation

- The optimization problem is transformed into a dual formulation.
- In the dual expression, data appear only as part of dot products.
- In the expression for the determination of the optimal hyperplane in H, data will appear, as well, as part of dot products.

Nonlinear transformation

The existence of a kernel function *K* would be possible…

$$K(x, y) = \Phi(x) \cdot \Phi(y)$$

How will the transformation be perceived at the level of the optimization problem?

Nonlinear transformation

- Kernel – a function that expresses a dot product in a feature space.

- The convergence of support vector machines towards a solution imposes that such a function be positively (semi-)defined.
- Such a kernel is one that satisfies Mercer's theorem.
- However, it makes the detection of efficient surfaces that do not obey the theorem impossible.

Kernel

- Kernel – function that expresses a dot product in a feature space.
- There are classical kernels that have been demonstrated to obey Mercer's condition:
  - The polynomial kernel: $K(x, y) = (x \cdot y + c)^p$

  The radial kernel: $K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$
    - Gamma $\gamma = \frac{1}{2\sigma^2}$
      - High value: decision boundary tight around points -> possible overfitting
      - Low value: the influence of each sample reaches far -> underfitting
      - Default: the inverse of the number of features in the dataset

Solving the optimization problem

- It is based on notions of convexity.
- The Lagrangian is built and the Karush-Kuhn-Tucker-Lagrange (KKTL) conditions are applied.
- The dual of the optimization problem is built.
- The gradient of the new objective function is equaled to 0 and the resulting system is solved.
- ***w*** and ***b*** result from the KKTL conditions, if they can be computed;
  - otherwise, the support vector machines determines the class directly for the test vectors.
- The sign of $f_{w,b}$ gives the final class – positive or negative.

SVM for multiple classes

- In their standard use, SVM is a supervised machine for binary problems (with two classes).
- In extending to several classes (*k*), there are two classical techniques:
  - One-against-all
  - One-against-one

One-against-all (1aa)

- k SVM classifiers are built.
  - For the i-th SVM, class i is positive and the others together represent the negative class.

- Every i-th SVM determines the optimal hyperplane of coefficients w^i si b^i.

- The label for a new example is given by the class with the maximum value for the learnt function f_{w^i,b^i}.

One-against-one (1a1)

- k(k-1)/2 SVM classifiers for every two classes in turn are built.
  - Class i if the positive one, j the negative one.

- The decision hyperplane between every two classes i si j with coefficients w^{ij} si b^{ij} is determined.

- The label for a new example is determined by voting.
  - For every SVM the class of the example is computed and receives a vote.
  - The class with most votes obtained wins.

SVM for regression

- In this case, data are constrained to lie on a hyperplane that
  - Permits a certain error ϵ from the real output
  - Has a high generalization ability

- For the linear case:

$$\begin{cases} y_i - w \cdot x_i + b \le \varepsilon \\ w \cdot x_i - b - y_i \le \varepsilon \end{cases}$$

$$i = 1,2,\ldots,m$$

  - and minimizes

$$\|w\|^2$$

The optimization problem - regression

- For the general nonlinear case:

$$\begin{cases} y_i - w \cdot x_i + b \le \varepsilon + \xi_i \\ w \cdot x_i - b - y_i \le \varepsilon + \xi_i^* \end{cases} \qquad \xi_i, \xi_i^* \ge 0$$

  - and minimizes $\|w\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \qquad C > 0$

- Kernel application is similar to that for classification.
- Resulting function $f_{w,b}$ gives the predicted output for a new example.

SVM in R. Classification

```r
library(e1071)
library(mlbench)
data(PimaIndiansDiabetes)
dat <- PimaIndiansDiabetes

repeats <- 30
classColumn <- 9
accuracies <- vector(mode="numeric", length = 30)
index <- 1:nrow(dat)

for (i in 1:repeats){
  testindex <- sample(index, trunc(length(index) / 4))
  testset <- dat[testindex, ]
  trainset <- dat[-testindex, ]

  svm_model <- svm(diabetes ~ ., data = trainset, kernel = "linear", cost = 1)
  svm_pred <- predict(svm_model, testset[, -classColumn])
  contab <- table(pred = svm_pred, true = testset[, classColumn])
  accuracies[i] <- classAgreement(contab)$diag
}

summary(svm_model)

print(accuracies)
print(mean(accuracies))
print(sqrt(var(accuracies)))
print(summary(accuracies))

print("Confusion matrix of last run")
print(contab)
```

Results

```
 [1] 0.7916667 0.7812500 0.7812500 0.8489583 0.7239583 0.8125000 0.7708333
 [8] 0.7604167 0.7552083 0.7604167 0.7500000 0.7395833 0.7708333 0.8072917
[15] 0.8281250 0.7916667 0.7864583 0.7604167 0.7656250 0.7760417 0.8177083
[22] 0.7760417 0.8333333 0.7500000 0.7916667 0.7812500 0.7291667 0.7812500
[29] 0.8020833 0.7604167
[1] 0.7795139
[1] 0.02982043
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.7240  0.7604  0.7786  0.7795  0.7917  0.8490
[1] "Confusion matrix of last run"
     true
pred  neg pos
  neg 108  33
  pos  13  38
```

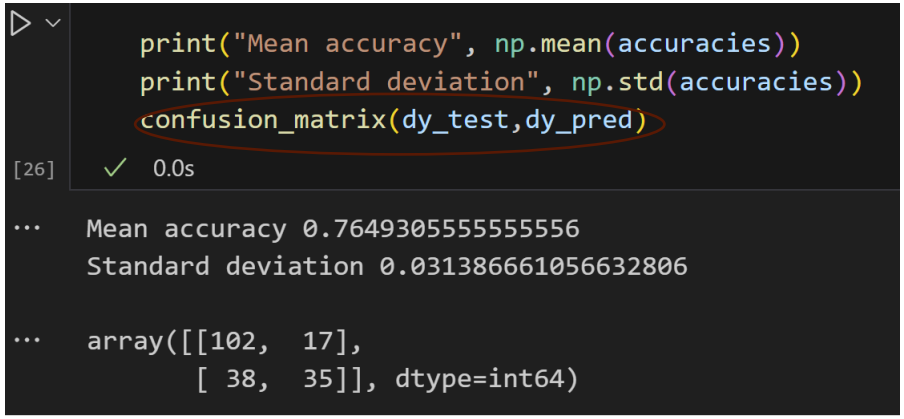
- Result of the program:
  - The accuracies in the 30 runs
  - Accuracy in mean
  - Standard deviation
  - Summary regarding accuracy in the 30 runs
  - The confusion matrix

SVM for classification in Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn import svm

repeats = 30
accuracies = []

data = pd.read_csv("diabetes.csv")

#take first n-1 columns for x and last column for y
dx = data.iloc[:, :-1]
dy = data.iloc[:, -1]

for i in range(0, repeats):
    #random generation of training and test, 75-25%
    dx_train, dx_test, dy_train, dy_test = train_test_split(dx, dy, test_size = 0.25)
    svmm = svm.SVC(kernel='linear')
    svmm.fit(dx_train,dy_train)
    dy_pred = svmm.predict(dx_test)
    accuracies.append(accuracy_score(dy_test, dy_pred))

print(accuracies)
```

Results

[0.8125, 0.7395833333333334, 0.7708333333333334, 0.78125, 0.7604166666666666, 0.71875, 0.78125, 0.7708333333333334, 0.7291666666666666, 0.8072916666666666, 0.7760416666666666, 0.7239583333333334, 0.796875, 0.75, 0.78125, 0.7395833333333334, 0.8072916666666666, 0.765625, 0.734375, 0.796875, 0.703125, 0.8072916666666666, 0.7291666666666666, 0.7552083333333334, 0.7708333333333334, 0.7916666666666666, 0.7395833333333334, 0.7916666666666666, 0.8020833333333334, 0.7135416666666666]



Variable importance

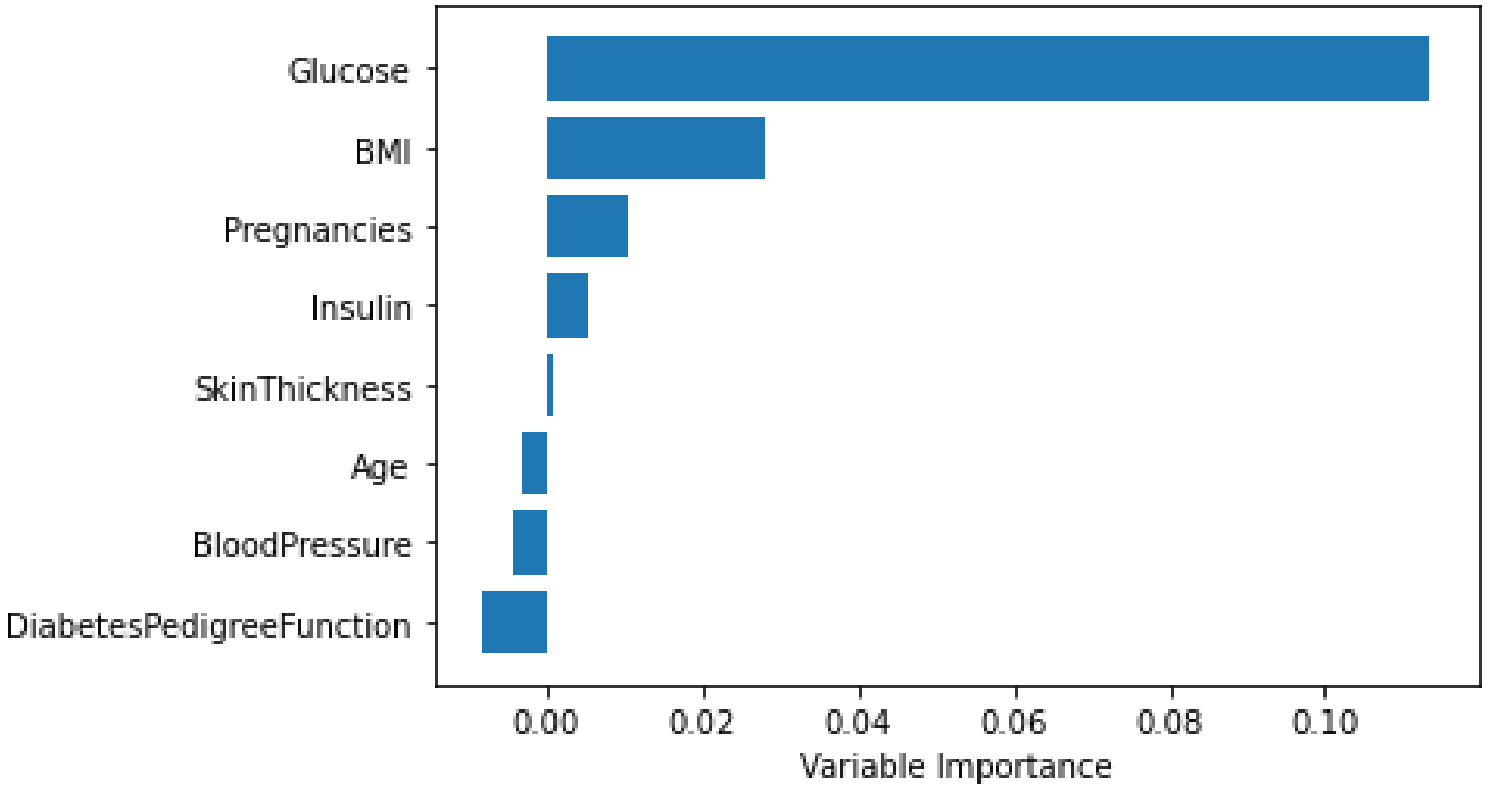
- According to studies (also the linear model in the previous lecture), glucose and BMI are the most important variables.
- Permutation importance measures the decrease in model test performance when the values (column) of each feature are randomly shuffled
  - The shuffling of the column values is repeated *k* times
  - The score of the model on the distorted data is computed
  - The importance of each variable (performance deterioration) is computed as the difference between the initial score and the mean over the scores of the corruption
  - The default scorer for the estimator is used
  - Multiple simultaneous scorers can be specified for parameter *scoring*

Variable importance

- Possible to be applied to a validation set unlike impurity-based importance of decision trees built on training and can give importance to variables not predictive on unseen data
- Also, it does not bias numerical features as the latter
- Thirdly, independent of the model
- Negative values further show that the feature is not contributing to the prediction and random chance within shuffling led to better performance.

```
from sklearn.inspection import permutation_importance
features = data.columns

perm_importance = permutation_importance(svmm, dx_test, dy_test)

sorted_idx = perm_importance.importances_mean.argsort()
plt.barh(features[sorted_idx], perm_importance.importances_mean[sorted_idx])
plt.xlabel("Variable Importance")
```

Variable importance visualized

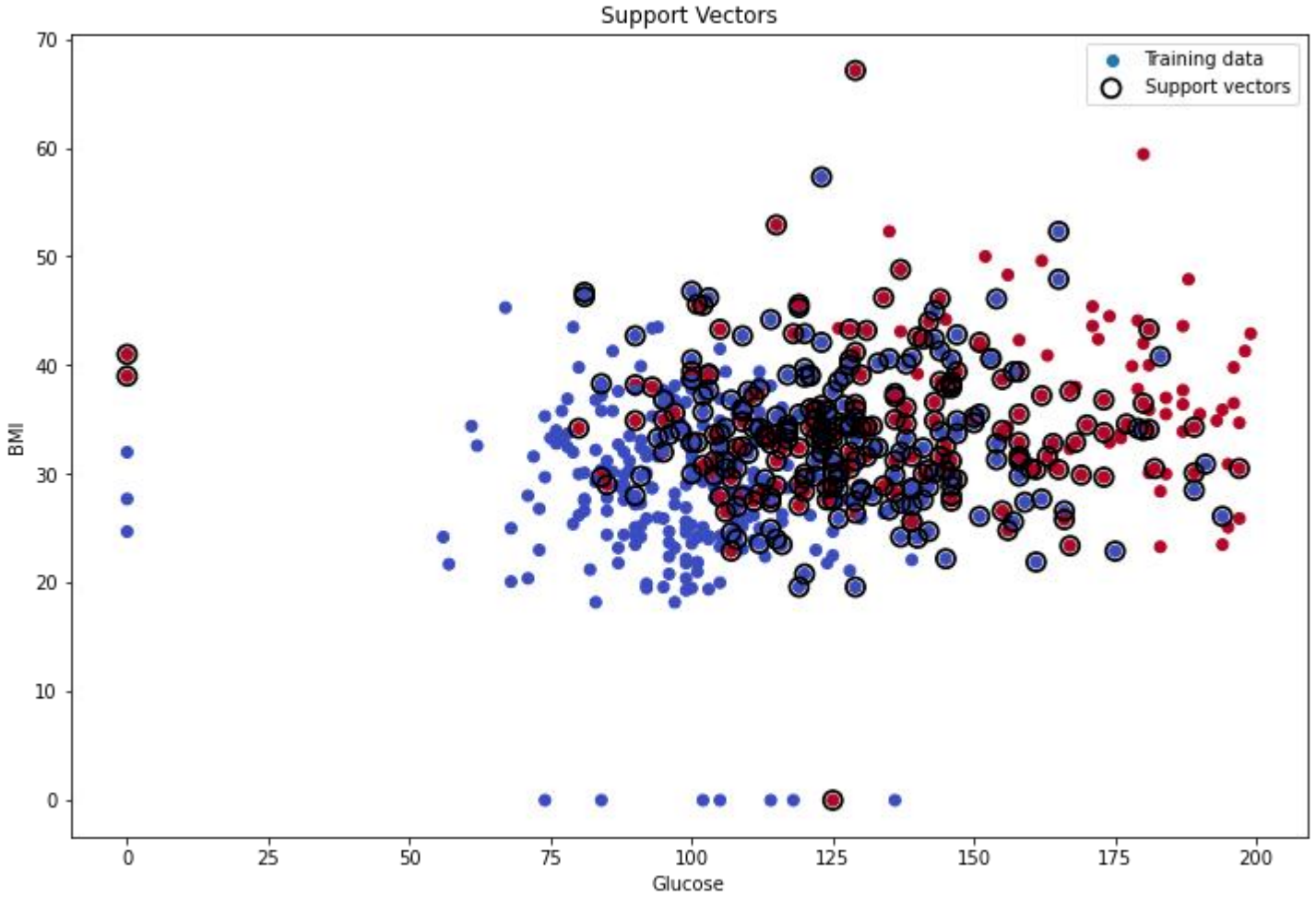


Support vectors

- The discovered support vectors can be plotted against the values for these two variables.

```
# support vectors against the training data

support_vectors = svmm.support_vectors_

plt.figure(figsize=(12, 8))
plt.scatter(dx_train.iloc[:,1], dx_train.iloc[:,5], c = dy_train, cmap='coolwarm', label='Training data')
plt.scatter(support_vectors[:,1], support_vectors[:,5], s = 100, linewidth=1.8, facecolors='none', edgecolors='k', label='Support vectors')
plt.title('Support Vectors')
plt.xlabel('Glucose')
plt.ylabel('BMI')
plt.legend()
plt.show()
```

Visualization training

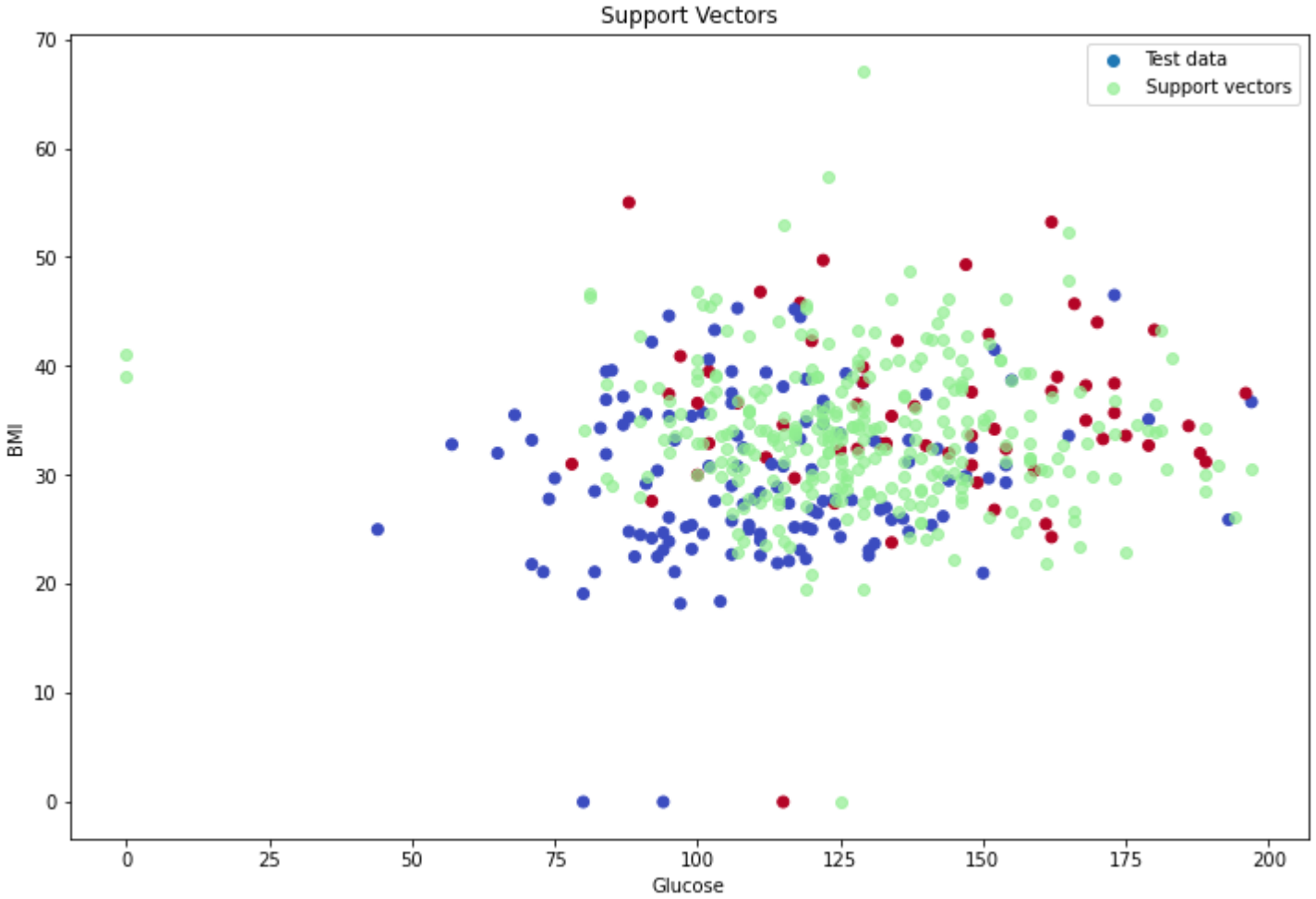


What samples can be removed from the data leaving the decision boundary unchanged?

SV against test data

```
# support vectors against the test data

support_vectors = svmm.support_vectors_

plt.figure(figsize=(12, 8))
plt.scatter(dx_test.iloc[:,1], dx_test.iloc[:,5], c = dy_test, cmap='coolwarm', label='Test data')
plt.scatter(support_vectors[:,1], support_vectors[:,5], color='lightgreen', alpha = 0.7, label='Support vectors')
plt.title('Support Vectors')
plt.xlabel('Glucose')
plt.ylabel('BMI')
plt.legend()
plt.show()
```

Visualization test



SVM for regression in R

```r
library(e1071)
library(mlbench)

data(BostonHousing)

repeats <- 30
classColumn <- 14
mses <- vector(mode="numeric", length = 30)
index <- 1:nrow(BostonHousing)


for (i in 1:repeats){
  testindex <- sample(index, trunc(length(index) / 4))
  testset <- BostonHousing[testindex, ]
  trainset <- BostonHousing[-testindex, ]
  svrm <- svm(medv ~ ., data = trainset, kernel = "radial", epsilon = 0)
  pred <- predict(svrm, testset[, -classColumn])
  mses[i] <- mean((pred - testset[, classColumn])^2)
}

print(mses)
print(mean(mses))
print(sqrt(var(mses)))
```

Results

- Rezultatul programului:
  - Mean squared error in cele 30 de rulari.
  - Mean of MSE.
  - Standard deviation.

```
 [1]  6.085508 10.032190  9.892823 10.413884 12.214825 15.579678 24.119295
 [8] 19.796761  9.477194  9.767336  7.902491  9.858571 15.305895  7.339980
[15]  6.026287 16.124503 25.321344 16.692788 21.799579 16.354292 10.649699
[22] 10.781551 17.095841 26.352994 12.998777 16.465062 14.085417 19.593361
[29] 21.342196  8.230289
[1] 14.25668
[1] 5.770945
```

The performance of an SVM depends upon:

Data scaling

- In SVM, scaling is not mandatory, but recommended.

- In R e1071 package, scaling is done by default.
- Scaling has to be done explicitly in sklearn.

- The scaler is fit on the training data, i.e. the information is taken from the available records only.
- Both the training and test data are transformed by the scaler.

Scaling

- MinMax scaler
  - Takes the data to the [0, 1] range
  - Normalization
  - Uniform distribution
- Standard Scaler
  - Takes the data to 0-mean and 1-variance
  - Standardization
  - Normal distribution

Normalization vs standardization

MinMax Scaler	Standard Scaler
Simplicity	Insensitivity to outliers
Relationship preservation	Lack of impact on data distribution (e.g., if there were extreme values present)
Interpretability	Independency of data range (e.g., if initially a narrow range)
Compatibility with distance-based models	No fixed range for analysis

SVR in Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import datasets
from sklearn.preprocessing import MinMaxScaler

repeats = 30
mses = []
r2s = []

bx, by = datasets.load_boston(return_X_y=True)

scaler = MinMaxScaler()

for i in range(0, repeats):
    #random generation of training and test, 75-25%
    bx_train, bx_test, by_train, by_test = train_test_split(bx, by, test_size = 0.25)

    #scale data for SVM
    fit_scalar = scaler.fit(bx_train)
    bx_train_scaled = fit_scalar.transform(bx_train)

    svmm = svm.SVR(kernel='rbf', gamma = 1, C = 10)
    svmm.fit(bx_train_scaled,by_train)

    bx_test_scaled = fit_scalar.transform(bx_test)
    by_pred = svmm.predict(bx_test_scaled)
    mses.append(mean_squared_error(by_test, by_pred))
    r2s.append(r2_score(by_test, by_pred))

print(mses)
print(r2s)
```

Results

```
    print("Mean MSE:", np.mean(mses))
    print("Standard deviation:", np.std(mses))
    print("R^2:", np.mean(r2s))
```

[51] ✓ 0.0s

```
Mean MSE: 15.313684936245918
Standard deviation: 5.612257978996744
R^2: 0.8171483423250877
```

Homework

- Implement in R or Python an SVM model
  - EITHER for Wisconsin breast cancer diagnosis
    - Drop first column as non-informative
    - Treat missing values (delete, change to 0, average etc.)
  - OR for the Tips data set from the previous lecture
- (Optional) Choose a data set of interest and apply an SVM.

Note: For all problems, try different scalers, variable importance and visualization of support vectors.


I SUPPORT
VECTOR MACHINES