Machine Learning (ML)

Chapter 8:

Classification

Support Vector Machines (SVM)

Saeed Saeedvand, Ph.D.

Outline

In this Chapter:

- ✓ Support Vector Machines (SVM)
- ✓ Support Vector Classification (SVC)
  - Linear SVC
  - Nonlinear SVC
    - Polynomial Kernel SVC
    - RBF Kernel SVC
    - Sigmoid Kernel SVC
- ✓ Hinge loss
- ✓ Kernel function
- ✓ Kernel Trick

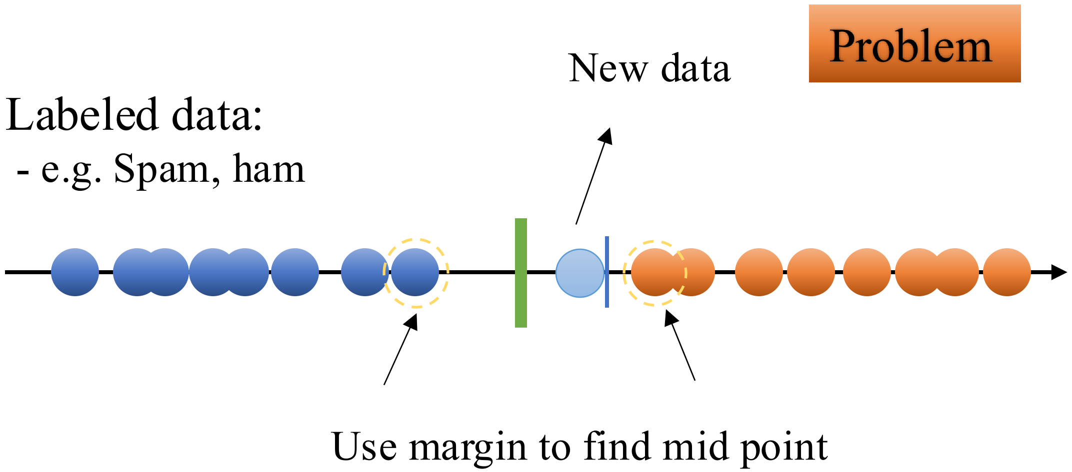
Aim of this chapter:

- ✓ Understanding two important concepts of SVM in classification applications, as well as the concept of the kernel trick and using different kernel functions.

Classification

Support Vector Machines (SVM)

- ✓ Widely used supervised machine learning algorithm for classification and regression tasks.
- ✓ It is a supervised machine learning model.
- ✓ Generally SVM is to find the optimal hyperplane (decision boundary).
- ✓ SVM finds a decision boundary that maximizes the margin between different classes in the feature space.



Support Vector Machines (SVM)

What is the Support Vector Classification (SVC)

✓ SVC is a specific instance of the SVM algorithm for classification tasks.

SVM algorithms

Support Vector Machine (SVM)

- Support Vector Classification (SVC)
  - Linear SVC
  - Nonlinear SVC
    - Polynomial Kernel SVC
    - RBF Kernel SVC
    - Sigmoid Kernel SVC

In this chapter.

- Support Vector Regression (SVR)
  - Linear SVR
  - Nonlinear SVR
    - RBF Kernel SVR
    - Polynomial Kernel SVR
    - Sigmoid Kernel SVR
- Least Squares SVM (LS-SVM)
  - LS-SVC (Classification)
  - LS-SVR (Regression)
- Twin Support Vector Machine (TWSVM)
  - TWSVM for Classification
  - TWSVM for Regression

Support Vector Machines (SVM)

The key concepts in SVM

- Hyperplane
- Margin
- Support Vectors
- Regularization
- Kernel trick

Support Vector Machines (SVM)

Hyperplane:

✓ Feature space that has one dimension less than the number of the feature, also called (decision boundary):

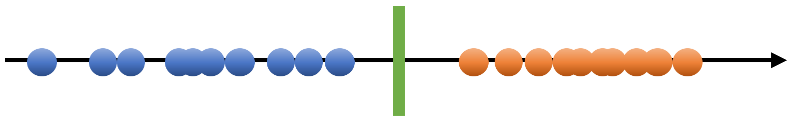
- In one-dimensional space, we have a point.
- In two-dimensional space, we have a line.
- In three-dimensional space, we have a 2D surface.
- In more than four-dimensional space, it is a hyperplane.

Note: we call all of them hyperplane, especially when we can’t see it visually.

Support Vector Machines (SVM)

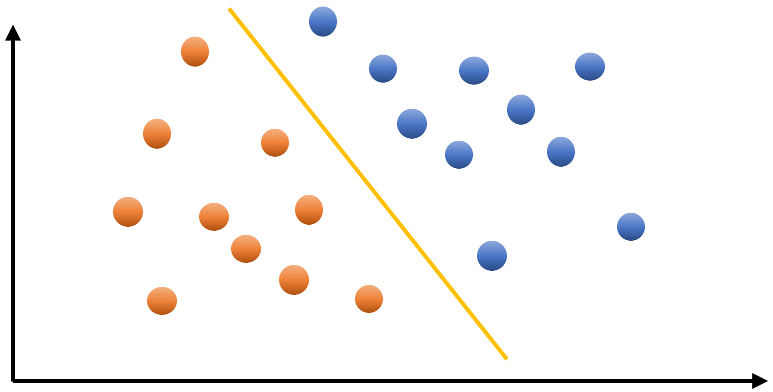
Hyperplane:

✓ Feature spaces:

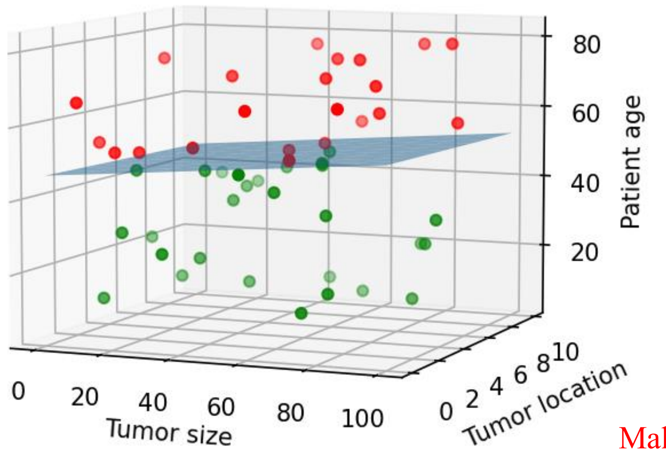


Point

line



Surface or hyperplane

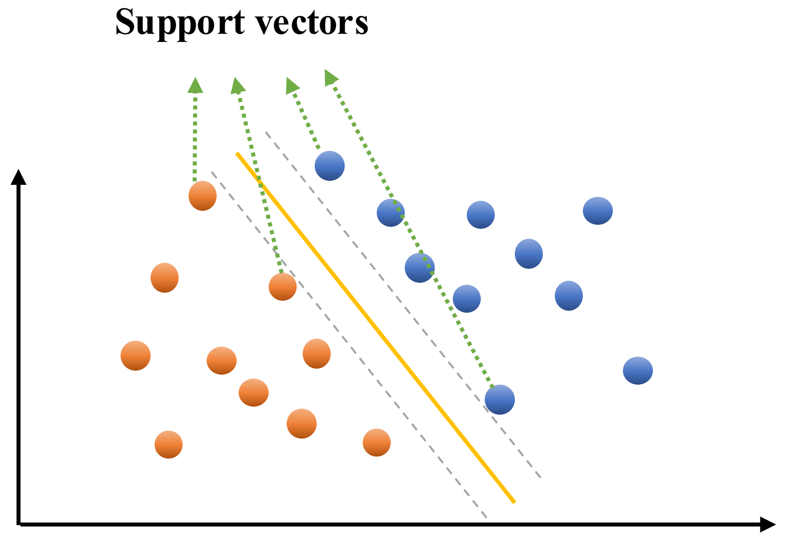


Malignant
Benign (safe)

Support Vector Machines (SVM)

Support Vectors:

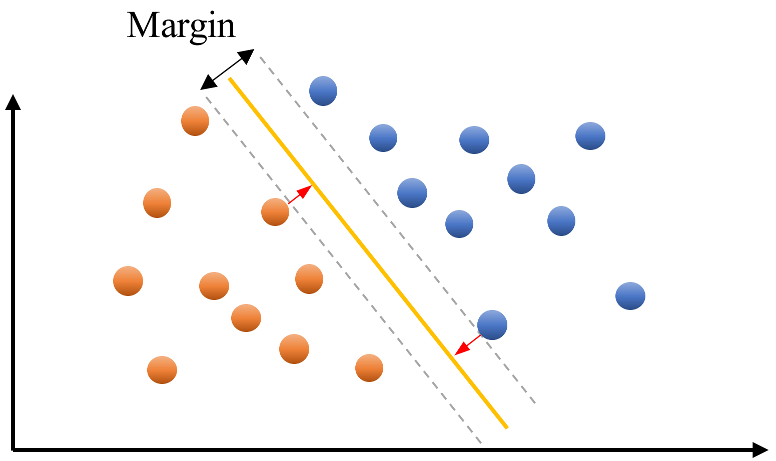
- ✓ The **data points** that are closest to the hyperplane or decision boundary.
- ✓ The points that are used to define the margin.
- ✓ Critical in determining the **optimal hyperplane**.



Support Vector Machines (SVM)

Margin:

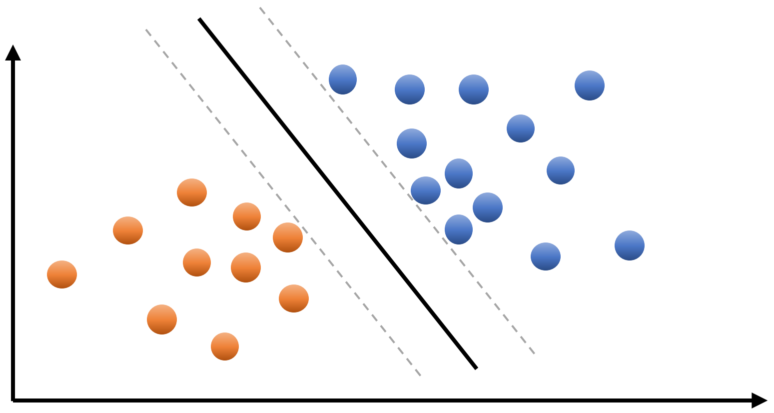
- ✓ Distance between the closest points of different classes **to the hyperplane.**
- ✓ In SVM we aim to maximize the margin from each class:
  - **Called** Maximum Margin Classifier.
- ✓ **Maximum Margin Classifier is to improve:**
  1. The classification performance.
  2. Generalization of the model.



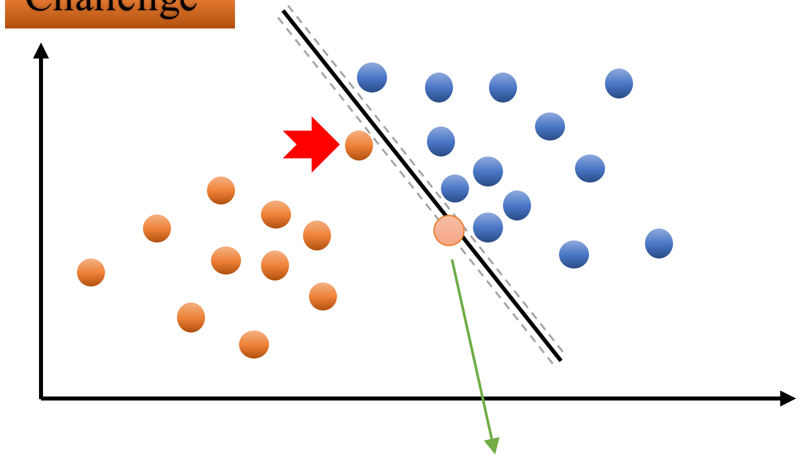
Support Vector Machines (SVM)

Margin:

- ✓ Now with maximum margin classifier the hyperplane may get very close to one class!
- ✓ So maximum margin classifier is super sensitive to outlier data.
- ✓ This can be considered also as overfit!



Challenge



New data!
Wrong class

Support Vector Machines (SVM)

Regularization:

- ✓ The solution is to allow misclassifications.
- ✓ We **need to have a** bias-variance tradeoff in SVC similar to other ML algorithms.
- ✓ In **SVC,** we can **allow the regularization** using a parameter called "C".
- ✓ With regularization, we can balance the trade-off between overfitting and underfitting.

With a small 'C':

- Higher misclassifications accepted so that we can have a wider margin for training data.

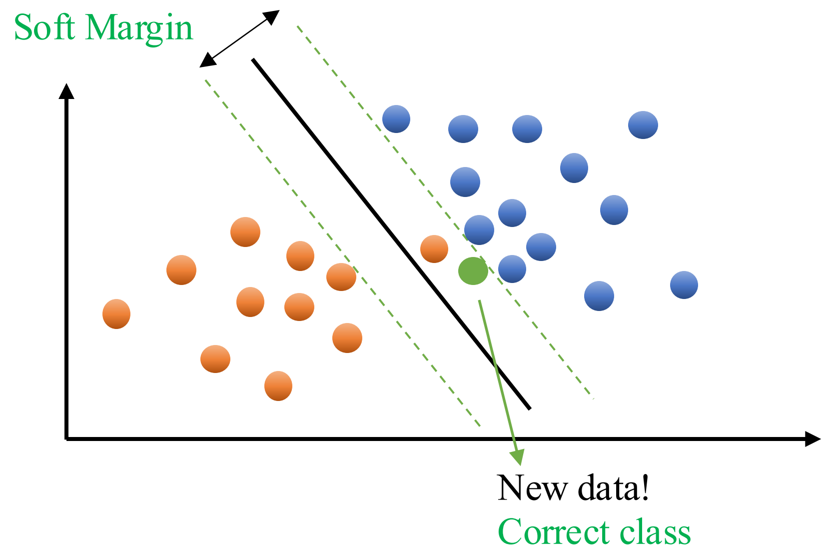
With big 'C':

- Lower misclassification accepted, and therefore, we have a narrow margin for training data.

Support Vector Machines (SVM)

Regularization:

✓ Allow misclassifications by ‘C’:



✓ **If we allow misclassification** we call it soft margin and the approach is Soft Margin Classifier in Support Vector Classifier (SVC).

Note: The value of 'C' is not directly related to the distance or the number of misclassifications.

Support Vector Machines (SVM)

Regularization:

✓'C' determines the balance between the margin and the classification error.

✓Classification error in the objective function such as:

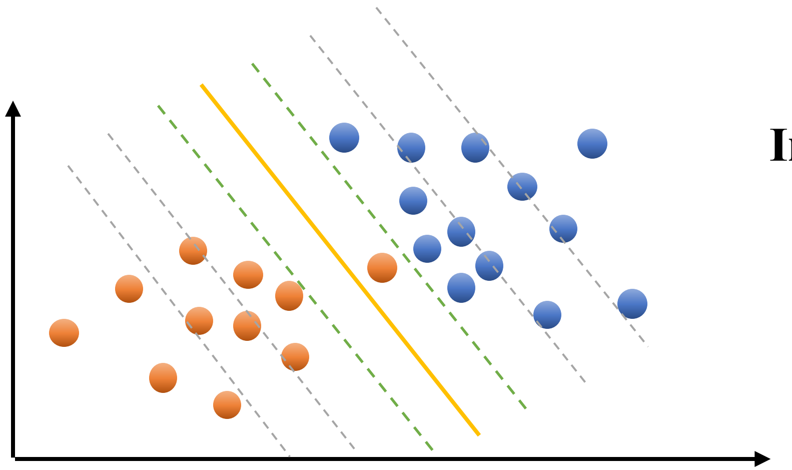
- Accuracy, precision, recall, or F1 score (to optimize during training).

Support Vector Machines (SVM)

Chose best soft margin:

✓ How to know how to chose best soft margin by ‘C’?

- We can use cross-validation based on metric that we have (accuracy, precision, recall, or F1 score, …)
- Then change ‘C’ and evaluate by allowing misclassification inside the soft margin to find best classification.



In case we use library it is very simple:

```
# 5 folds Cross-validation
from sklearn.model_selection import cross_val_score
scores = cross_val_score(classifier, X, Y, cv=5, scoring='accuracy')
# scoring= accuracy precision  recall f1_macro f1 roc_auc
average_precision
```

Support Vector Machines (SVM)

Basic steps of the SVM algorithm

✓ Data Preparation:

- **Collecting and organizing** the data and splitting it into **training** and **testing** sets.

✓ Feature Selection :

- We **select the features** that are most important for our classification task.

✓ Model Training:

- We train the SVM model on our training dataset (**to find the hyperplane that separates the data** into different classes in best way**).**

Support Vector Machines (SVM)

Basic steps of the SVM algorithm

✓ Hyperparameter Tuning:

- We need to **tune these hyperparameters** to achieve the best performance, which includes regularization parameter and kernel parameters (we will discuss it).

✓ Model Evaluation:

- We **evaluate the performance** of the SVM model based on metrics on our test data (accuracy, precision, recall, F1-score, and …).

✓ Applying the Model on the Task:

- Use the model to **make predictions on new unseen data**.

Support Vector Machines (SVM)

Training the SVC:

✓ Similar to linear regression but objective function is different, as in following steps.

Step 1: In Linear SVC, the **goal is to find** the best hyperplane, but the objective function is different, as in that it separates two classes by maximizing the margin between them.

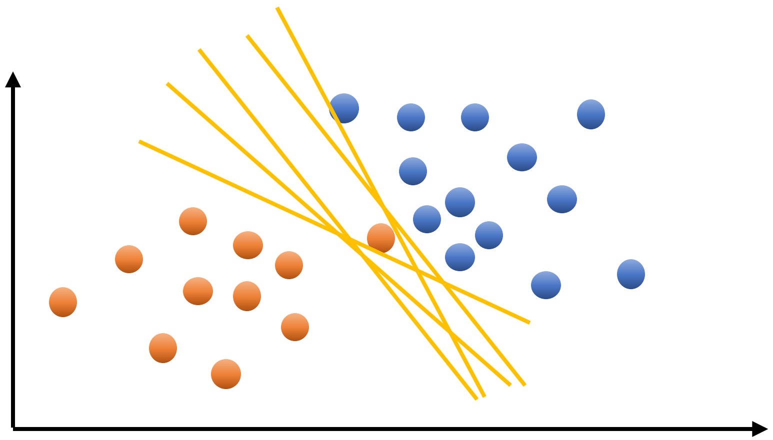
Bias term

$$f(x) = w.x + b$$

Weight vector

dot product

Input feature vector



Support Vector Machines (SVM)

Training the SVC:

Step 2: Use one optimization algorithm to minimize the loss function (GD, SGD, …)

Objective Function

Runs over all the training examples

$$L(w, b) = \frac{1}{2} * ||w||^2 + C * \sum (\max(0, 1 - y_i(w.x_i + b)))$$

Euclidean norm of the weight vector or sum of the squares of the elements

Regularization parameter (hyperparameter)

True class label (-1 or 1)

Feature vector of the i^{th} training example

$$||w||^2 = w_1^2 + w_2^2 + \cdots + w_n^2$$

Measure the magnitude of the weight vector (minimizing $||w||$ maximizes the margin).

Support Vector Machines (SVM)

Training the SVC:

Step 2:

To ensure that the loss is non-negative

The distance from the i^{th} training example to the decision boundary

$$L(w, b) = \frac{1}{2} * ||w||^2 + C * \sum (\max(0, 1 - y_i(w \,.\, x_i + b)))$$

Hinge Loss Function:

- ✓ It is the most commonly used function.
- ✓ In ML, **hinge loss boosts** the model to find the optimal separating hyperplane between classes **while** penalizing misclassified points.
- ✓ There are other version of hinge loss too that you can study (Squared hinge loss, Huberized, Logistic loss, Exponential loss, …).
- ✓ In most cases, the hinge loss works well for SVM

Support Vector Machines (SVM)

Training the SVC:

Step 2:

$$L(w, b) = \frac{1}{2} * ||w||^2 + C * \sum (\max(0, 1 - y_i(w \,.\, x_i + b)))$$

Hinge loss with linear Loss kernel:

$$\text{hing\_loss}_i = \max(0, 1 - y_i(w \,.\, x_i + b))$$

- If a point is correctly classified and is outside of the margin then:

$$\text{hing\_loss}_i = 0$$

- If a point is correctly classified but is inside of the margin then:

$$0 < \text{hing\_loss}_i < 1$$

- If a point is misclassified then:

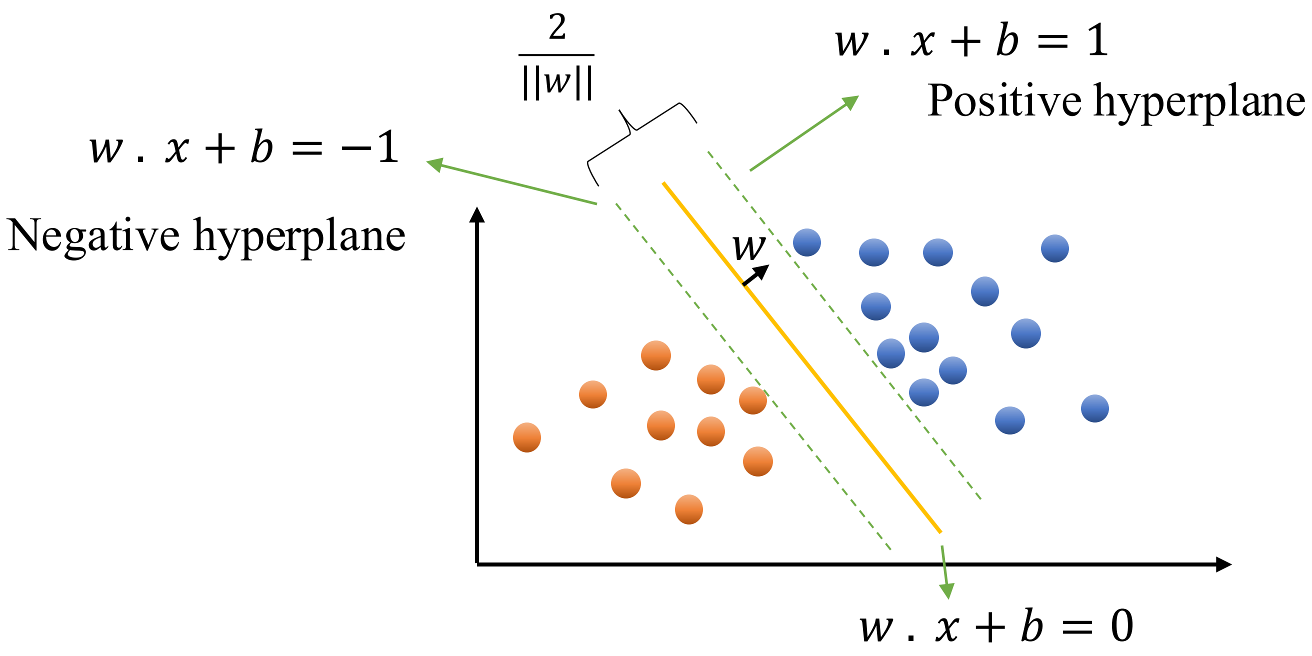
$$\text{hing\_loss}_i \geq 1$$

Support Vector Machines (SVM)

Training the SVC:

Step 2: We need to find w, and b for minimizing loss (visual view of objective function)

$$L(w, b) = \frac{1}{2} * ||w||^2 + C * \sum (\max(0, 1 - y_i(w.x_i + b)))$$



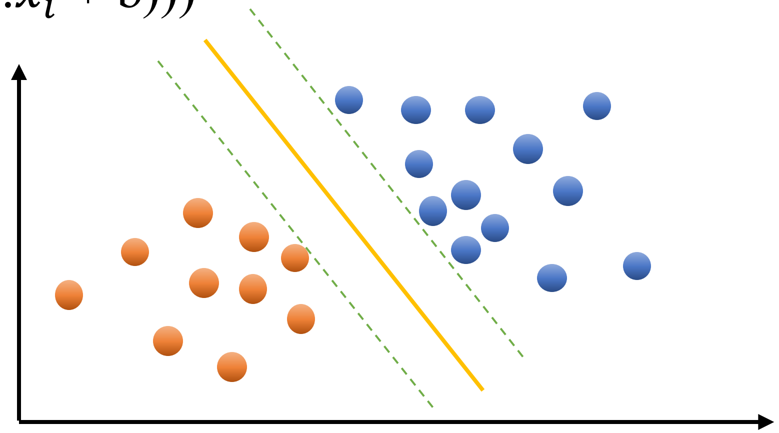
Support Vector Machines (SVM)

Training the SVC:

Step 3: Converge with the optimization algorithm and **define the decision boundary** (hyperplane) **by**:

- ✓ The slope of the decision boundary, which is the normalized weight vector w.
- ✓ The intercept, which is the bias term b.

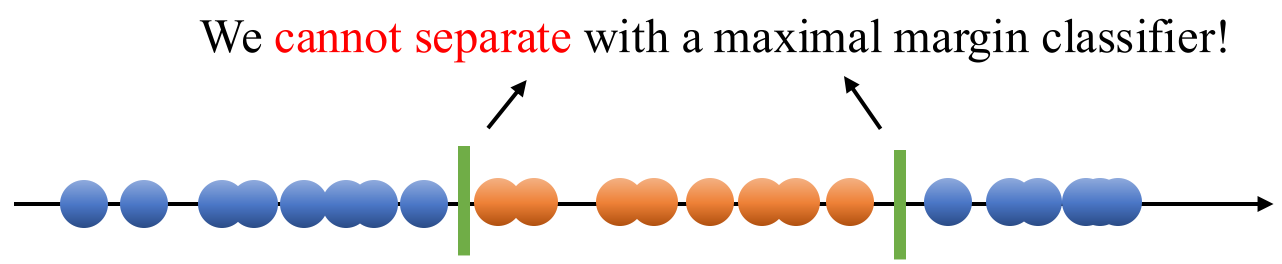
$$L(w, b) = \frac{1}{2} * ||w||^2 + C * \sum (\max(0, 1 - y_i(w.x_i + b)))$$



Support Vector Machines (SVM)

Challenge:

✓ For the cases that data points are not separatable linearly in the feature space.



✓ In such cases, in SVM we can use a kernel function to map the data into a higher-dimensional space to make it linearly separable.

✓ **Common kernel functions include**:

- Linear
- Polynomial
- Radial Basis Function (RBF)
- Sigmoid.

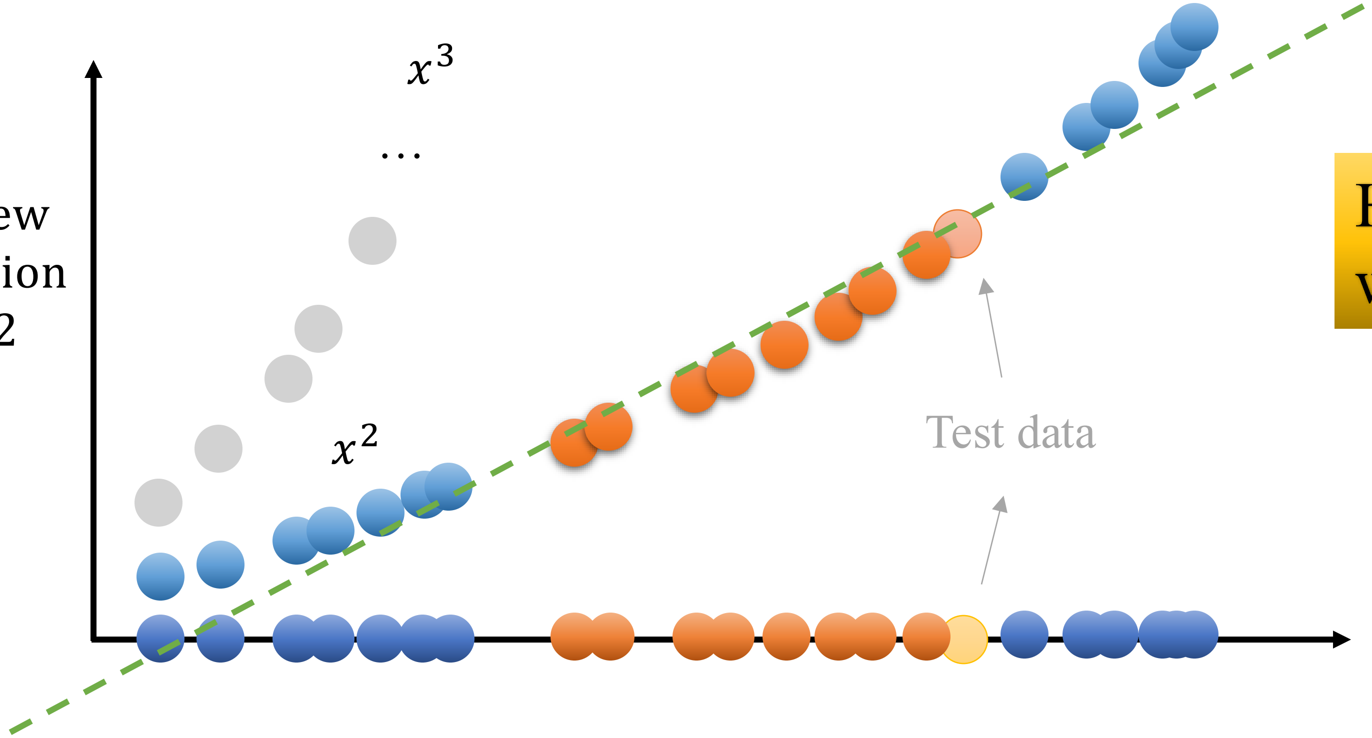
Support Vector Machines (SVM)

Solution:

✓ We can add one more dimension d and calculate:

- For example $x^2, x^3, \ldots$ (**note**: after defining kernel trick we don't need to define it)

Add new dimension $d = 2$



How many dimensions we should add?

✓ In Polynomial kernel we increase the dimentions d and select the best by cross-validation!

Support Vector Machines (SVM)

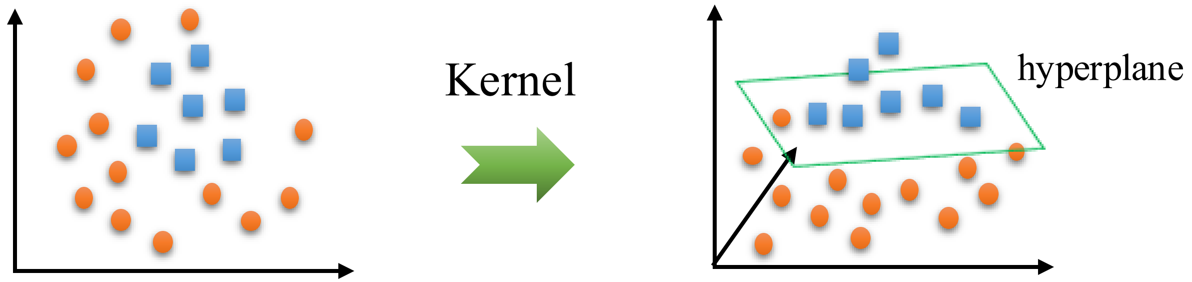
Kernel function:

✓ A kernel function is to map the data into a higher-dimensional space to make it linearly separable.

Kernel trick:

✓ If we do this process without explicitly transforming, we call it kernel trick (a function does that).

✓ Kernel trick allows the model to learn non-linear decision boundaries **efficiently** (**lower complexity**).



Support Vector Machines (SVM)

Popular kernel functions used in ML

i^{th} and j^{th} training examples
which is j^{th} is +1 here

Linear kernel:

$$K(x_i, x_j) = x_i . x_j$$

✓ With the linear kernel we compute the inner product between two data points x_i, x_j.

$$x_i = [1, 2, 3], x_j = [4, 5, 6], x_i . x_j = (1 * 4) + (2 * 5) + (3 * 6) = 32$$

Polynomial kernel:

$$K(x_i, x_j) = (\gamma * (x_i . x_j) + r)^d$$

✓ With the polynomial kernel we compute the similarity between two data points x_i, x_j that we raised to a specific polynomial degree.

hyperparameters of the kernel

γ (gamma): a constant that scales the dot product.
r: also a constant term added to the scaled dot product.
d: a constant that specifies the degree of the kernel function to the inputs.

Support Vector Machines (SVM)

Popular kernel functions used in ML

To control the spread of the Gaussian function

Radial Basis Function (RBF) kernel:

$$K(x_i, x_j) = \exp(-\gamma * ||x_i - x_j||^2)$$

✓ The **RBF kernel** computes the similarity between two data points x_i, x_j based on their Euclidean distance.

Scales the dot product

Scaled dot product

Sigmoid kernel:

$$K(x_i, x_j) = \tanh(\gamma * (x_i \, . \, x_j) + r)$$

✓ The **sigmoid kernel** computes the similarity between two data points x_i, x_j using the hyperbolic tangent function.

Support Vector Machines (SVM)

Adopt Kernel trick with loss function:

- ✓ The kernel function is not a loss function.
- ✓ As we saw kernel function is method for transforming input data points into a higher-dimensional space.
- ✓ So, we need to rewrite the chosen loss function based on the chosen kernel.
- ✓ We use it in **conjunction of kernel** with for example hinge loss function.

Support Vector Machines (SVM)

Adopt Kernel trick with loss function:

✓ Hinge loss function for optimization problem:

- A non-linear soft-margin SVM with a desired kernel function.

$$L(\alpha) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j * \boxed{K(x_i, x_j)} - \sum_{i=1}^{n}\alpha_i \quad \text{Where } 0 \leq \alpha_i \leq C$$

Lagrange Multipliers α_i:

- In SVM importance or weight of:
  - ➢ Each training example — only non-zero for support vectors.

✓ Now we can replace desired kernel $K(x_i, x_j)$

Note: Lagrange Multipliers is a strategy to find the local maxima or minima of a given function in mathematical optimization. It is subject to equality constraints.

Support Vector Machines (SVM)

Adopt Kernel trick with loss function:

$$\mathrm{L}(\alpha) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j * K(x_i, x_j) - \sum_{i=1}^{n}\alpha_i$$

How to implement?

```
from sklearn import svm

# We can Fit SVM model
classifier = svm.SVC(kernel='linear' , C=1.0) # poly  rbf   sigmoid  precomputed
classifier.fit(X, Y)

# Get weights and bias of hyperplane if it is linear
w = classifier.coef_[0]
b = classifier.intercept_[0]
```

Support Vector Machines (SVM)

Idea of the Support Vector Regression (SVR)

- ✓ Similar to the classification problem.
- ✓ In regression instead of the distance between the hyperplane and the training examples in classification we:
  - Use the margin to **measure** the deviation between **predicted** and **actual values**.
- ✓ So, in **SVM regression (SVR)**, we modify the objective function to minimize the distance between the **predicted** and **actual values (**subject some constraints).

Support Vector Machines (SVM)

SVM Advantages

- ✓ SVM is a powerful and flexible ML algorithm that can be effective in a **wide range of applications**.
- ✓ SVM is known for its effectiveness in high-dimensional spaces.
- ✓ SVM is robust against overfitting using regularization parameter C.
- ✓ SVM using kernel functions usually works well with non-linearly separable data (adding one more dimension).
- ✓ SVM still woks well when the number of features is greater than the number of samples.
- ✓ Less hyperparameters to tune compared to other ML methods.

Support Vector Machines (SVM)

SVM Advantages vs DL models

SVM advantage over DL:

- ✓ Data efficiency
- ✓ Faster training (spatially in medium-sized datasets)
- ✓ More robustness to noisy data
- ✓ Transparency and interpretability
- ✓ Scaled up to handle large datasets

DL advantage over SVM:

- ✓ Ability to learn complex representations.
- ✓ Handle unstructured data much better.

The choice between SVM and DL models mostly depends on the available data and the problem.

Support Vector Machines (SVM)

SVM Applications in image processing

- ✓ Object detection
- ✓ Image segmentation
- ✓ Face recognition
- ✓ Image classification
- ✓ Medical image analysis

SVM generally is useful in situations where the data is high-dimensional or the number of training samples is small.

Support Vector Machines (SVM)

SVM Challenges

- ✓ SVM needs for careful selection of kernel function.
- ✓ SVM needs the hyperparameters tuning ('C' and the kernel-specific parameters).
- ✓ SVM may face difficulty with very large datasets.
- ✓ The parameters can significantly affect its performance.

Practice

- Implement K-fold cross validation to determine the C with given example.
-Use SVM to solve previous chapter's SPAM detection example first

Assignment

then use SVM to solve one image processing task with small dataset from Kaggle.
Upload your code and dataset, (in case it is large provide used link).

Summery

- ✓ We understood **SVM's concepts** and understood what we need to optimize with approaches like SGD.
- ✓ We learnt **Support Vector Classification (SVC)**.
- ✓ We defined **Hinge loss and its use.**
- ✓ We **defined Kernel function** to use in SVM.
- ✓ We **understood Kernel Trick** in which we do mapping to higher dimension but without explicitly transforming.