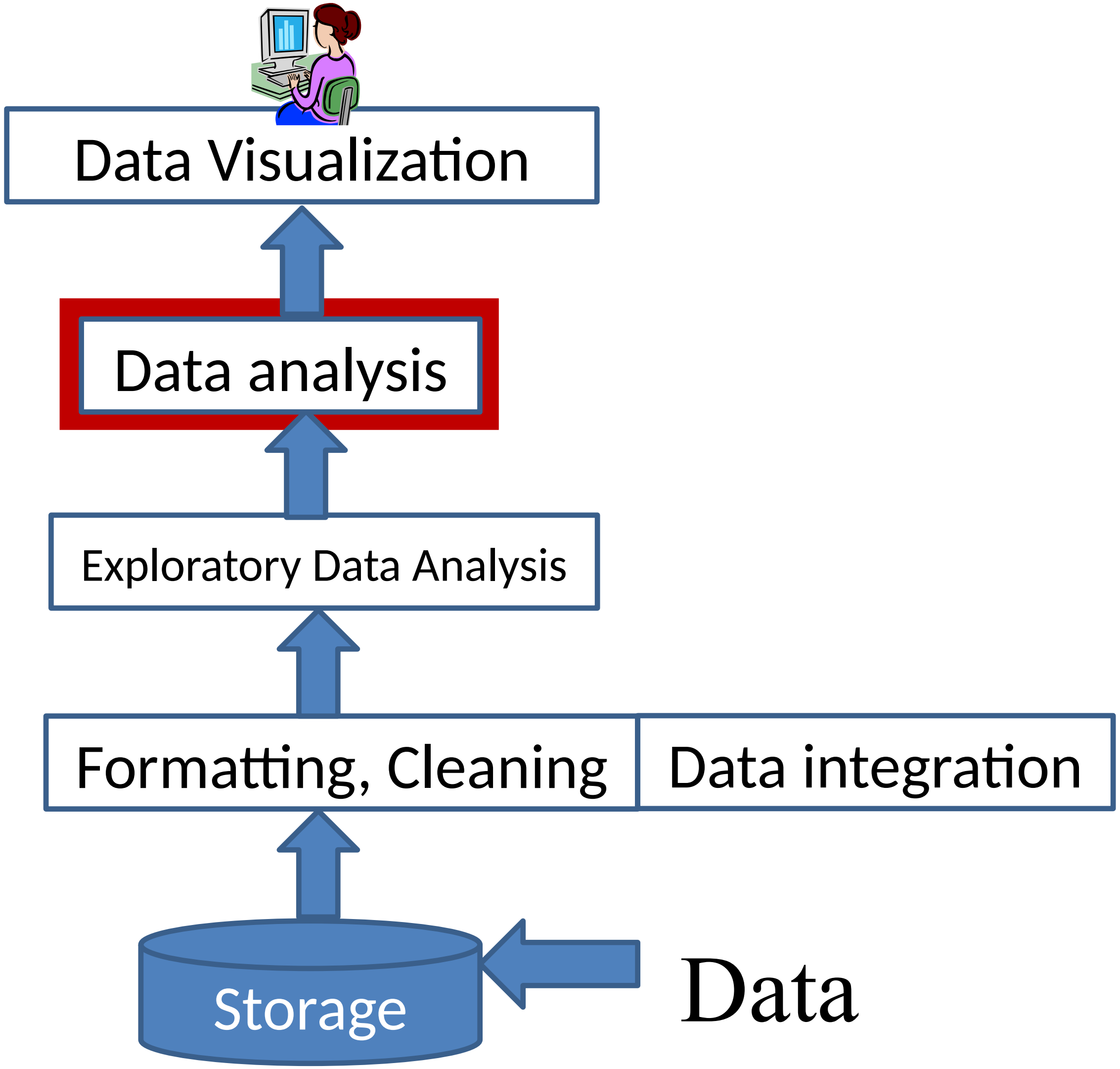
CPSC 483
Support Vector Machines

Anand Panangadan
apanangadan@fullerton.edu

What we will cover this week

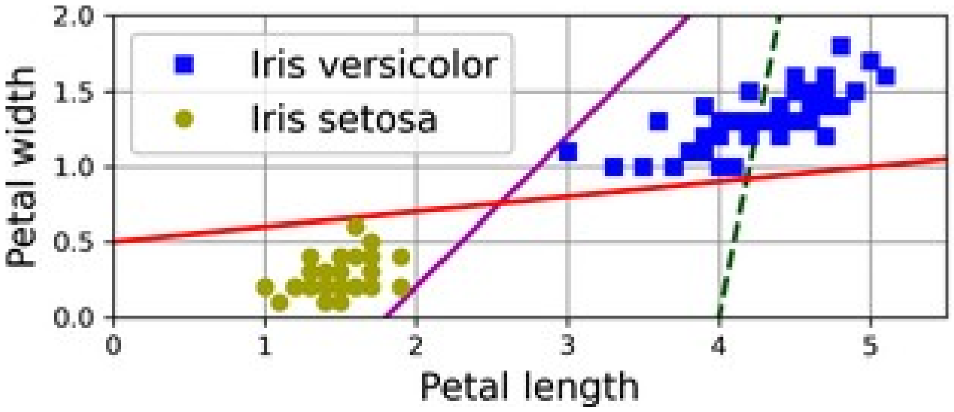
- Support Vector Machines
- The “kernel trick”
- Support Vector Regression

The Data Science Process



SUPPORT VECTOR MACHINES (SVM)

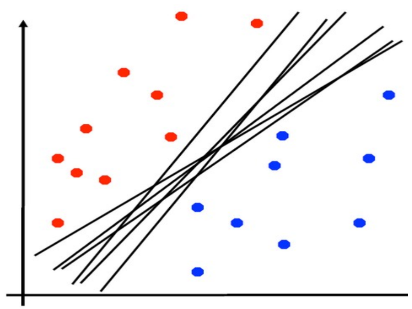
Hard margin classification

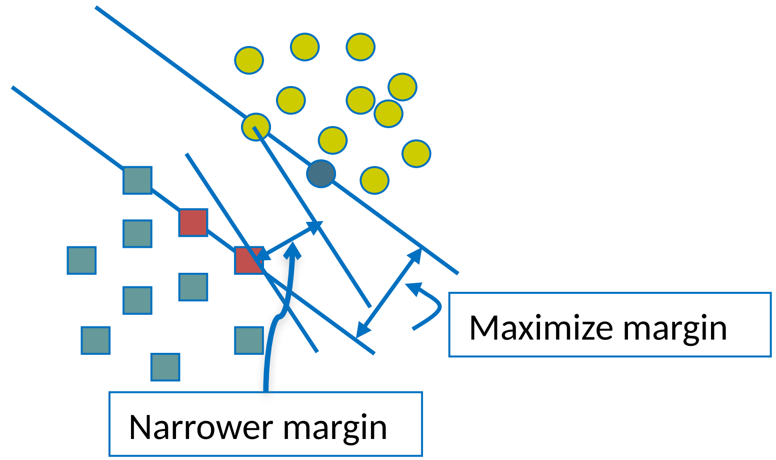


Which line “best” separates the points to two classes?

Support Vector Machine (SVM)

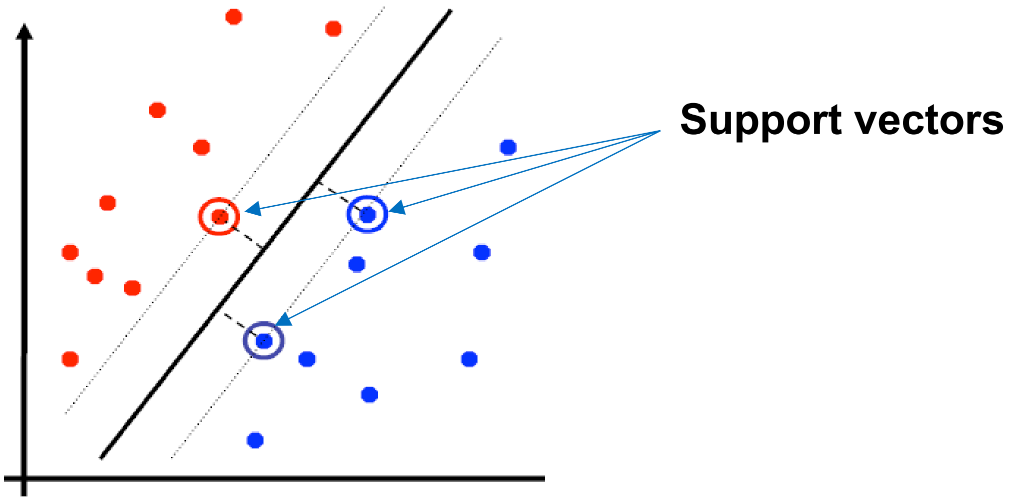
Linear separators





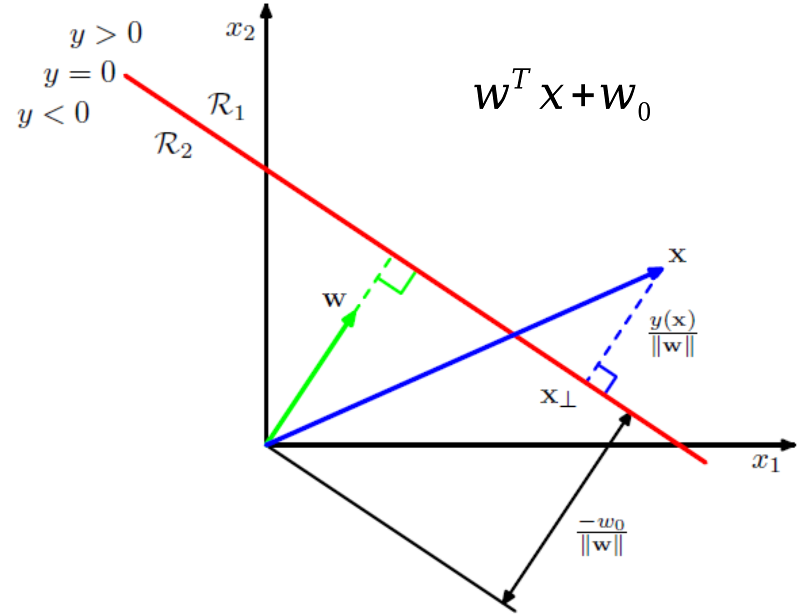
- **What are the optimal linear separators?**
  - The linear decision surface is expressed by .
  - Find **, ,** such that for red points and for blue points.
  - When there are many possible solutions for , , , the optimal separator is the line that separates the groups of data as clearly as possible.

Support Vector Machine (SVM) Vapnik, 1990



- A method invented by Vladimir Vapnik and colleagues at AT&T Bell Labs in the 1990s
- **Support vectors and margin**
  - Support vectors are the **data points** closest to the hyperplane. They have direct bearing on the optimal location of the decision surface and are the most difficulty to classify.
  - Margin is the width of separation between the support vectors.
- **SVM finds the hyperplanes that maximizes the margin.**
  - The decision function is fully specified by support vectors (a subset of training data).

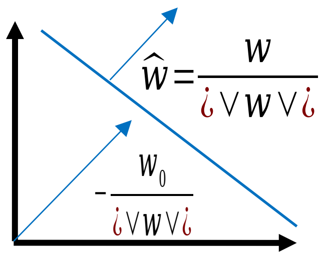
Distance to Hyperplane

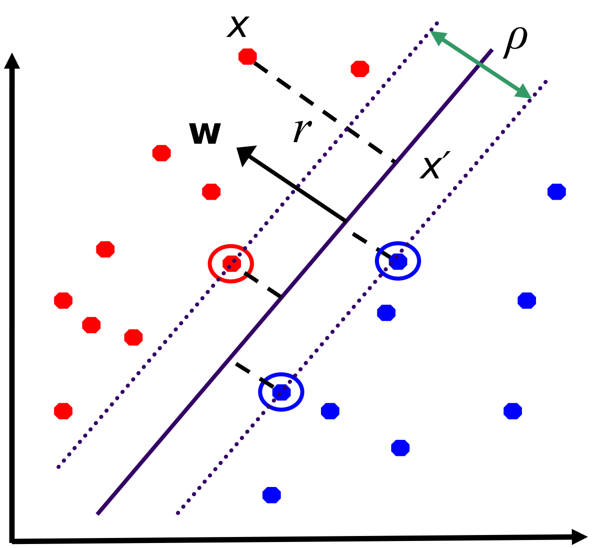


- **Data projection**
  - If the input data vector is projected onto the normal vector to the hyperplane, , what is the **distance** between the data point and hyperplane?
- **The distance to a hyperplane**
  - The **signed orthogonal distance** of a vector from the hyperplane:
    - If we use the hyperplane for **classification**, it is considered as decision surface.
  - The distance from origin to the decision surface is
    - The bias determines the location of the decision surface.

Distances and Hard Margin

$$w^T x + w_0 = 0$$





X: data points in training data set

X': Any points on the hyperplane

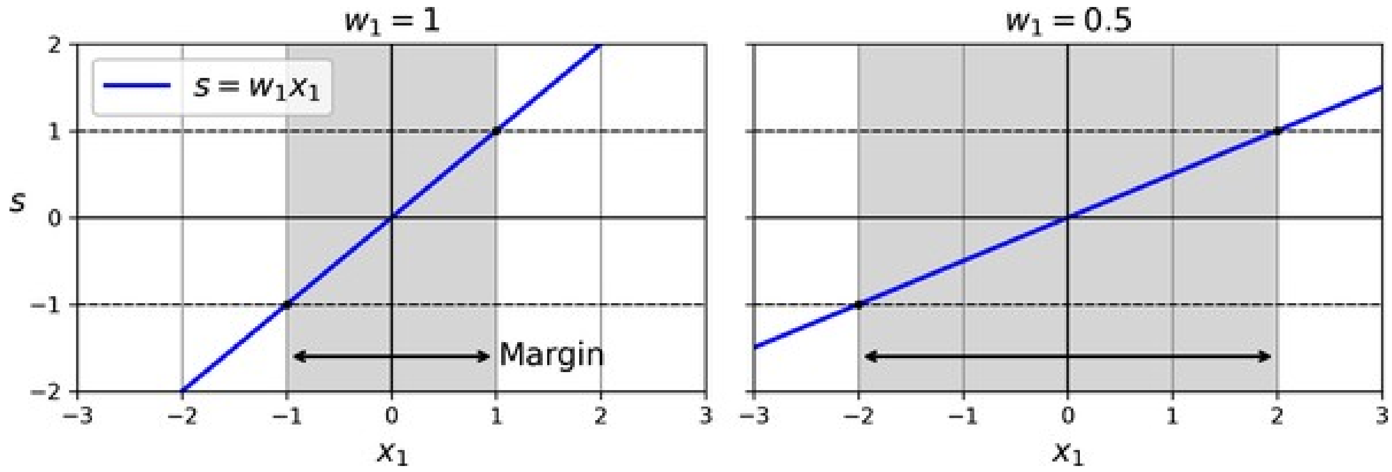
: *normal vector* to the decision hyperplane

: the location of the hyperplane

y: a class value +1 or -1 in training data

- If **y = 1**, and if **y =** , (for each side hyperplane)
  - **Support vectors** are **on** or
- **Combining both and – cases**, check
  - The distance between +/ point and decision hyperplane is or
  - The distance between origin and decision hyperplane is .
  - The distance between any data point and the decision hyperplane is
- The width of the margin area, called "**hard margin**" is

Width of margin



A smaller weight vector results in a larger margin

Primal Form of Linear SVM (Hard margin)

Find such that is **maximized** **subject** to (s.t)

for all .

Primal Form of Linear SVM (Hard margin)

- means (for mathematical convenience from)
  - can be considered as s L2 regularization term.

Find such that is **minimized** s.t

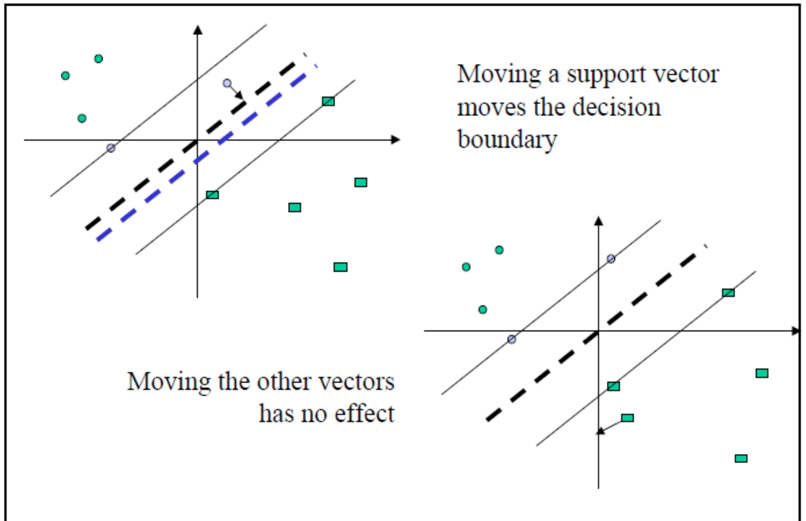
for all .

This is a constrained optimization problem.

- We cannot directly use gradient descent since this is a **constrained** optimization problem
- **Quadratic programming** can be used to solve the constrained optimization problems by converting into an equivalent unconstrained optimization problem.
  - Method of Lagrange multipliers

Functional Margin and Support Vectors

- **Implication of minimizing**
  - We want smaller **w**. To see the effects of w, draw the following parallel lines:



The algorithm to generate the weights proceeds in such a way that only the support vectors determine the weights and thus the boundary

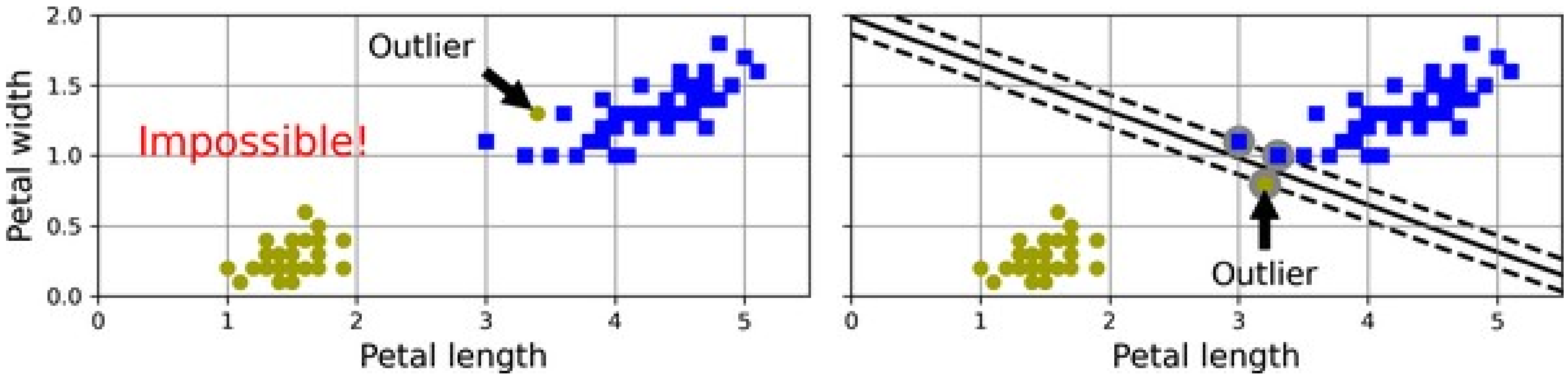
- The smaller w (coefficients), the larger margin; The larger w, the smaller margin.
  - From the functional margin of x, y(), we can increase this margin simply by scaling w and .

- What do we do with the support vectors?
  - For classification, we only need the support vectors.

Class work

- Write code to classify setosa vs virginica species in the Iris dataset using only Petal length and Petal width
  - Ignore the instances of versicolor
- What is the equation of the decision boundary?
- How many support vectors are needed to “support” this boundary?

Hard margin classification

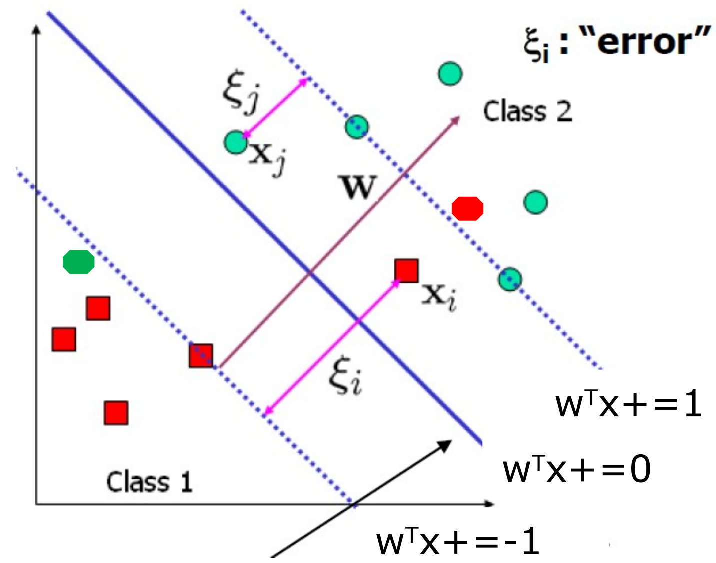


Problems with hard margin

- **Not realistic** to assume that the data are perfectly separable by the hard margin while **all the constraints** satisfied
- sensitivity to outliers

Soft Margin with Slack Variables

- Solution
  - Add *slack variables* (for each $\boldsymbol{x}$) to allow misclassification (violation of constraints) of difficult or noisy examples:
    - still on correct side of hyperplane but within the margin
    - wrong side of hyperplane
    - We still want to place a hyperplane with large margin to minimize the number of violations. How can we do that?



  - Idea:

For each data point, if margin 1, don't care but if <1, pay a linear penalty, ***C***.

Objective Function for Soft Margin

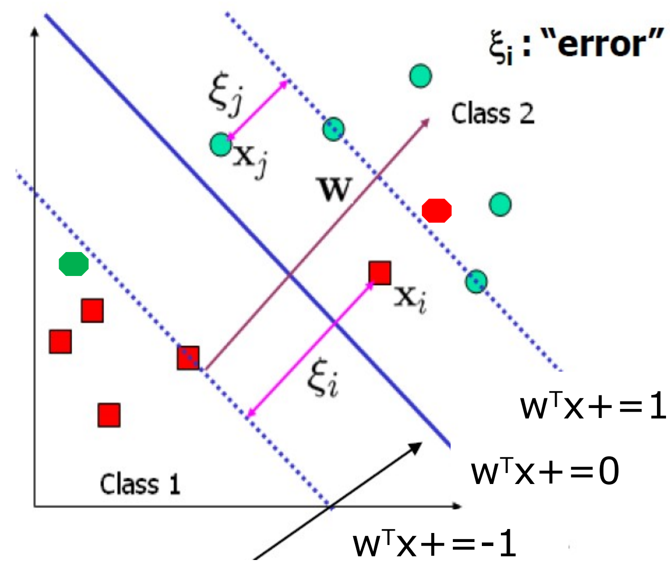
- **Soft margin objective function**:
  -

    s.t and

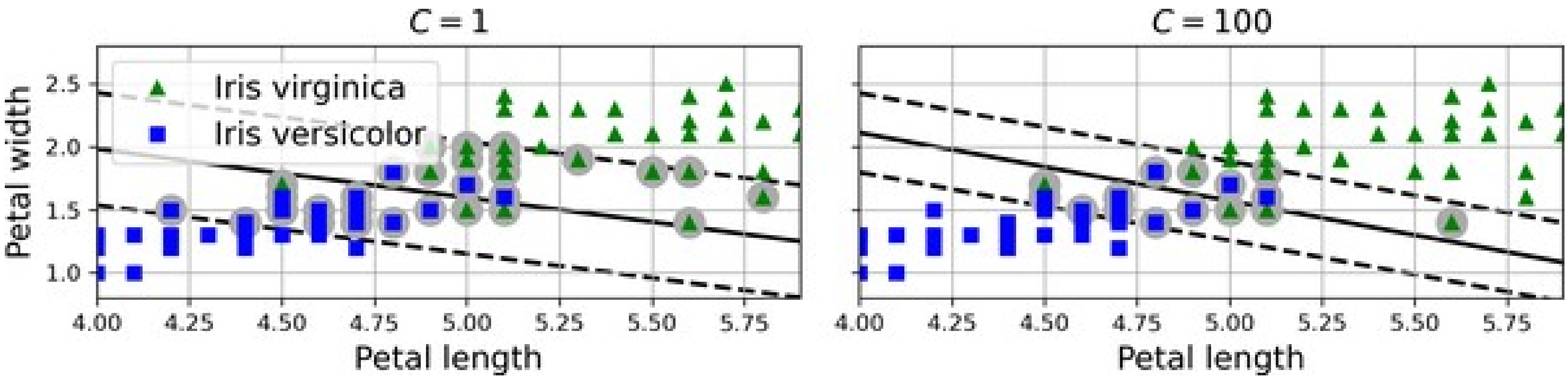
Maximize margin

Separating hyperplane

    - Smaller ||w||, wider margin for smoother decision boundary at the cost of misclassification errors (may underfit).
    - Larger ||w||, smaller margin for complex decision boundary as less tolerant to outliers (may overfit).
- **Hyper parameters *C* and**
  - is a regularization parameter to **control** the **penalty** for misclassification errors.
    - Increasing decreases the tolerance for breaking the margin.
    - have to separate the data (hard margin)
    - ignore the data entirely as value can be anything.
    - Choose the penalty using cross-validation.
  - is a slack variable that allows misclassification.
    - If margin $\geq$ 1, don't care, <1 pay linear penalty.
    - If = 0, the hard margin.

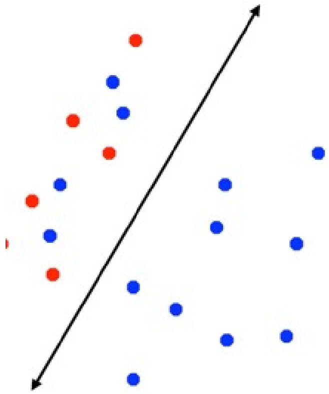


Soft margin classification



A hyperparameter to control how much margin violations are tolerated.
Larger C: more violations allowed, less likely to overfit

Dealing with Imbalanced Data

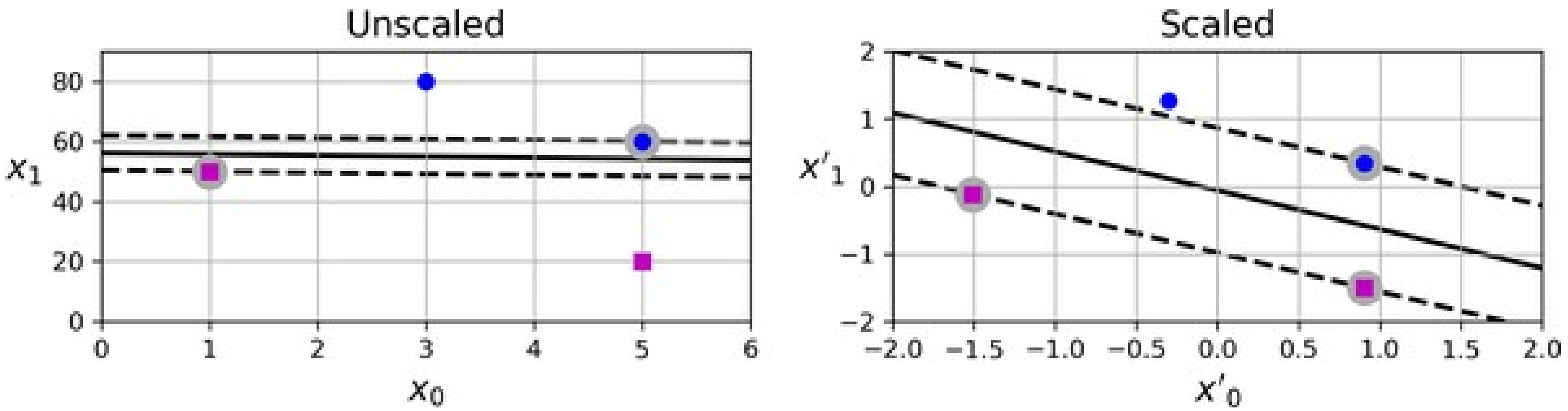


- **Classifying imbalanced data**
  - In many practical applications we may have imbalanced data. In this case we want errors to be equally distributed between the positive and negative classes
  - A slight modification to the SVM objective does the trick:

$$\min_{w,\, w_0} \; ||w||_2^2 + \frac{CN}{2N_+} \sum_{j:y_j=+1} \xi_j + \frac{CN}{2N_-} \sum_{j:y_j=-1} \xi_j$$

Class-specific weighting of the slack variables

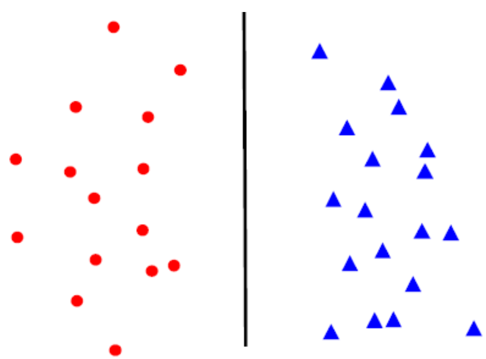
Scaling features

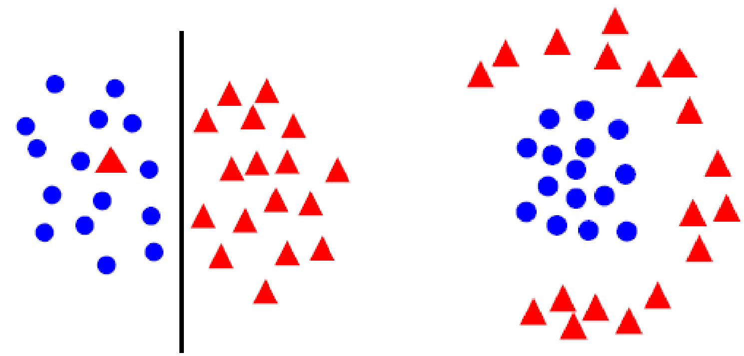


Note: changing the scale of the features (axes) will change the effect of the decision boundary.
Scale features before using margin-based classification

KERNEL METHODS

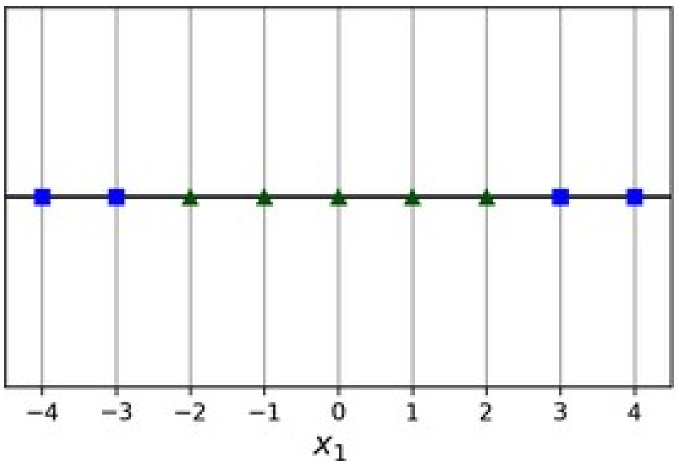
Linearly Separable and Not Separable





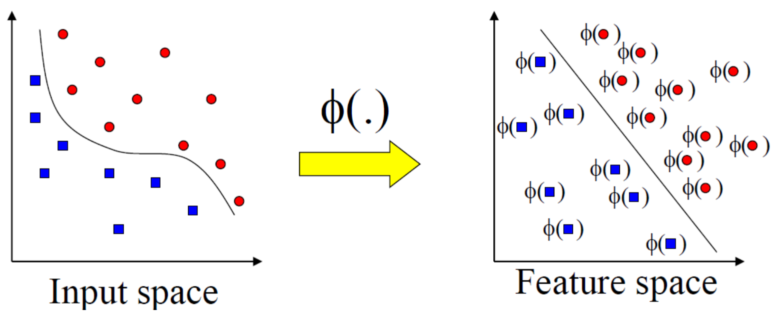
- Linearly separable and not separable
  - A linear decision boundary exists to separate the data.
  - No linear decision boundary exists to separate the data.
- **SVM** is a **<u>linear</u> classifier** using a hyperplane.
  - Hyperplane is a linear subspace of a vector space.
  - How can SVM classify the data that is not linearly separable?

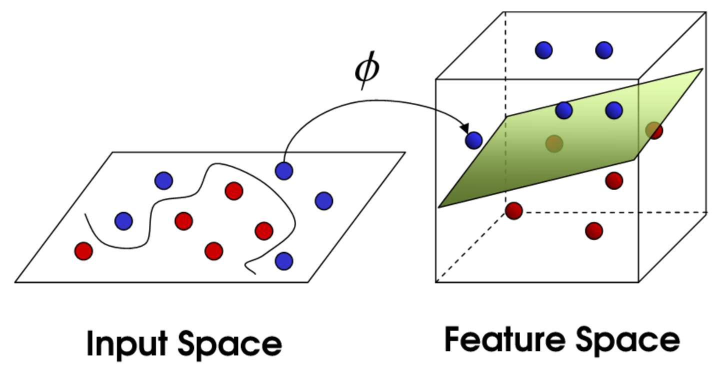
Mapping Data to High Dimension

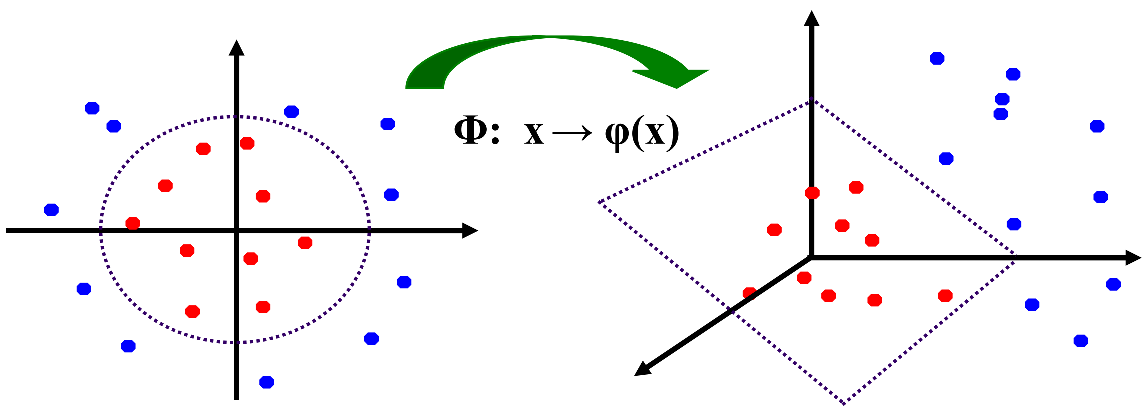


Points that were linearly non-separable on a line (1-D) **become** linearly separable in 2-D (provided an appropriate transformation was applied to the points:).

Mapping Data to High Dimension

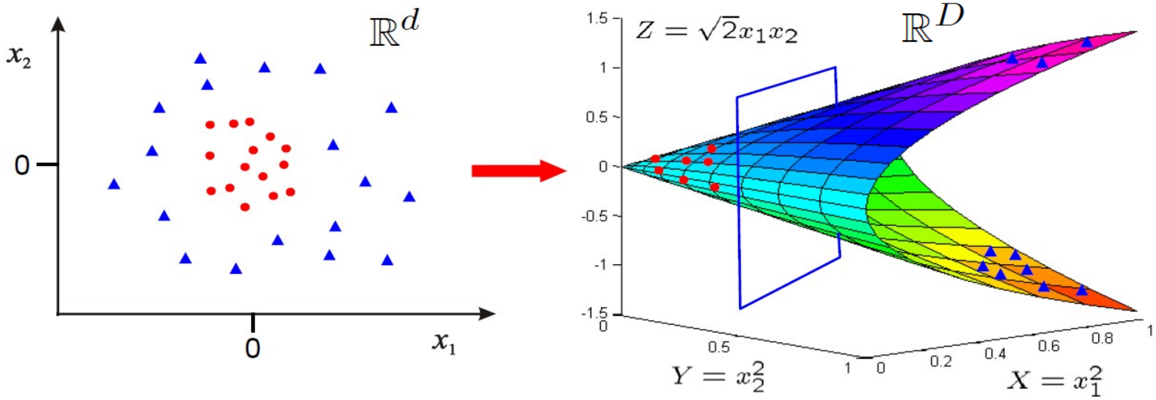






Rationale:
Data that is linearly non-separable in low dimensions **may become** linearly separable in high dimensions (provided sensible features are chosen).

Mapping Data to Higher Dimensions



Data is linearly separable in 3D.

The problem can still be solved by a linear classifier.

- (d < D)
  - **maps** to another dimensional space called “feature space”.
    - Input data are transformed into a higher-dimensional feature space that can be separable.
  - An example mapping function (**from 2D to 3D**)
  - General quadratic mapping
- **Learn a linear classifier** in higher dimension :

Kernel Function (Feature Map)

- A **kernel function** where and
  - The **dot product** of is called "kernel", "kernel function", or "feature map" in "reproducing kernel Hilbert space".
  - Example kernel: ,
  - **Kernel function** as a **dot product** (), is a similarity measure between **two objects** and (for the same object)
    - If two vectors are independent (dissimilar), zero; if parallel = 1.
  - This idea can be generally applied (not just to SVM), but one problem:
    - If D >> d, many more parameters to learn for w. How can this be avoided?

Very high number of dimension

- Let the original data be *m*-dimensional
- Transforming to a polynomial space of degree *d* will result in how many features?

The "Kernel Trick"

- In the previous kernel example:

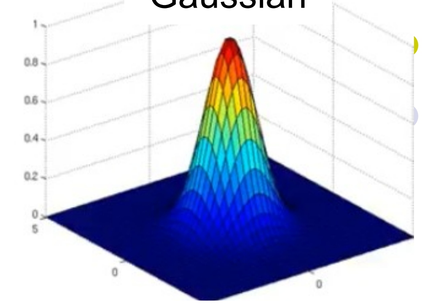
  - **Finding**:
    - Instead of that takes lots of computation in the feature space, just one dot product and square in input space without computing and .
    - Complexity of learning depends on # of examples N, typically for SVM not on the dimension D.

- **The kernel trick** claims :
  - Computing in the **original input space** **without** a feature transformation.
    - A **computational trick** in computing dot products in higher dimensional feature spaces

Polynomial Kernels

- Polynomials of degree **exactly** degree :
  - (linear kernel)
  - 2 (quadratic kernel)
  - For any , (proof is skipped)
- All polynomial terms **up to** degree :
  - ,

Common Kernels

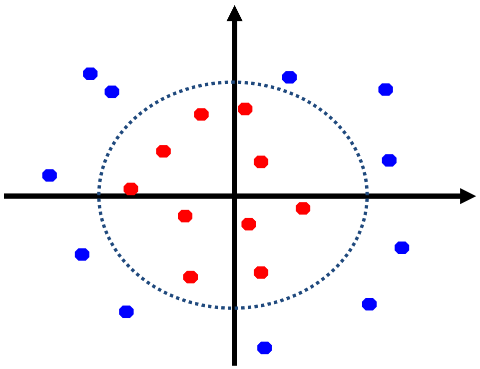


- **Polynomials of degree exactly d**
- **Polynomials of degree up to d**
- **Radial Basis Function (RBF) kernels**
  - Gaussian:
    - is the squared Euclidean distance; when close; when dissimilar
    - The small (is large) behaves like linear SVM, large may cause overfitting.
  - Exponential:
- **Sigmoid kernel**

Building New Kernels

- **Building new kernels using valid kernels**
  - If and are two valid kernels, the **following kernels are valid**:
    - Linear combination:
    - Exponential:
    - Product:
    - Polynomial transformation: where is polynomial with non-negative coefficients.
    - Function product: where is any function.

Class work



- For the above 2-D dataset, can you think of a transformation
  - to (another) 2-D space
  - to 1-D space
- Where the blue/red points become linearly separable?

Kernel “trick”

- Why a “trick”?
- The dual problem formulation of the SVM optimization problem can make use of the kernel

Primal and Dual Forms of SVM to Solve the Optimization Problem

- From the original SVM problem:
  - => s.t for all

- **The primal problem with Lagrange multipliers :**
  - ,

    ,
    - The primal form of the optimization problem that can be solved by solving for the **dual** of this original problem if we can transform the primal problem into a simpler form.

- **The dual problem to solve the primal problem:**
  -
  - s.t ,
    - **Why not just solve the original primal problem?** Because we can solve the problem by computing just the **dot products of pairs** of samples and .

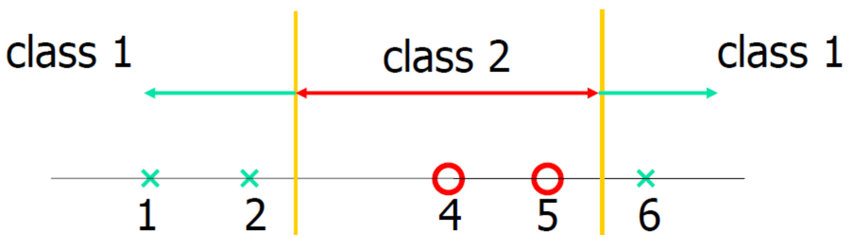
Solving the Dual Problem of SVM

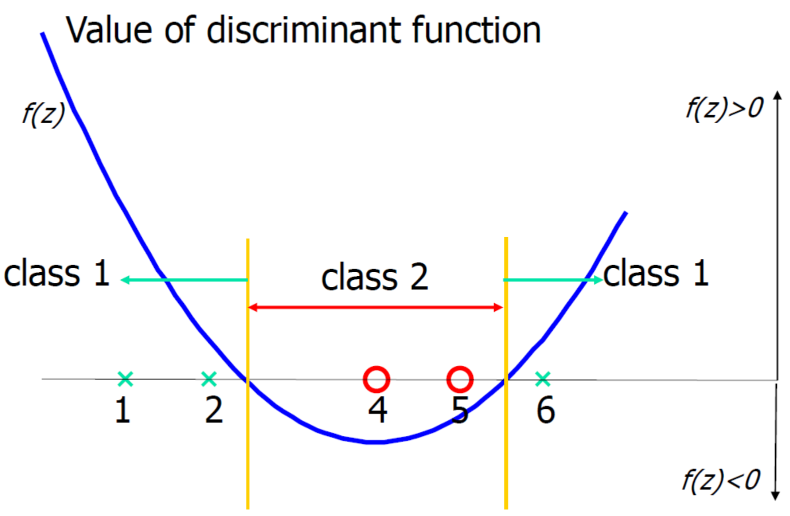
- **Solving**
- s.t ,
  - This is a quadratic programming (QP) problem. A global maximum of can be found.
  - Non-zero positive Lagrange coefficients corresponds to the support vectors.
    - **Most will be zero**. That means the other points play no role.
    - We only need to retain support vectors once model is trained.
- How do we know this solution is the optimal solution?
  - Karush-Kuhn-Tucker (KKT) conditions
    - , , , ,

SVM using Kernel Functions

- **Training to learn w** (by learning)
  -
    - s.t , for
  - **Classifier**
- **Predicting**
  - where and
    - Only need to compute dot products between training examples and the new example . This is true even if we map examples to a high dimensional space:
  - Prediction with high dimensional space using kernel trick

Interpreting the value of SVM discriminant function as probability





- Performing logistic regression on the SVM output of a set of data (validation set) that is not used for training

Steps for SVM Classification

- Prepare the pattern matrix.
- Select the kernel function.
- Select the parameter of the kernel function and the value of .
  - This is the trickiest part.
  - You can use the values suggested by the SVM software or set aside a validation set to determine the values of the parameter.
- Execute the training algorithm and obtain the .
- Prediction
  - Unseen data can be classified using the and the support vectors.

Overfitting in SVM

- SVM objective seeks a solution with large margin
  - Theory says that large margin leads to good generalization.
  - But everything overfits sometimes.
    - Large feature space with kernels (larger d in polynomial kernels)

- Model complexity can be controlled by:
  - Setting the parameter C
  - A better kernel
  - Varying parameters of the kernel (e.g., Gaussian)

Strength and Weakness of SVM

- **Strength of SVM**
  - Training is relatively easy
    - No local optimal unlike in neural networks
  - Scale well to high dimensional data
    - Not suffer from the curse of dimensionality.
    - The model complexity is independent of the dimensionality with support vectors.
    - Tradeoff between classifier complexity and error can be controlled explicitly.
  - Non-traditional data like strings and trees can also be used as input data.

- **Weakness of SVM**
  - Need to choose a “good” kernel function
  - How to determine hyperparameters (C, ε)
    - Hyperparameter optimization by empirical studies (Xu et al. 2009), genetic algorithms, simulated annealing (Pai and Hong 2005), k-fold cross-validation.
  - How to incorporate uncertainty (e.g., relevance vector machine)

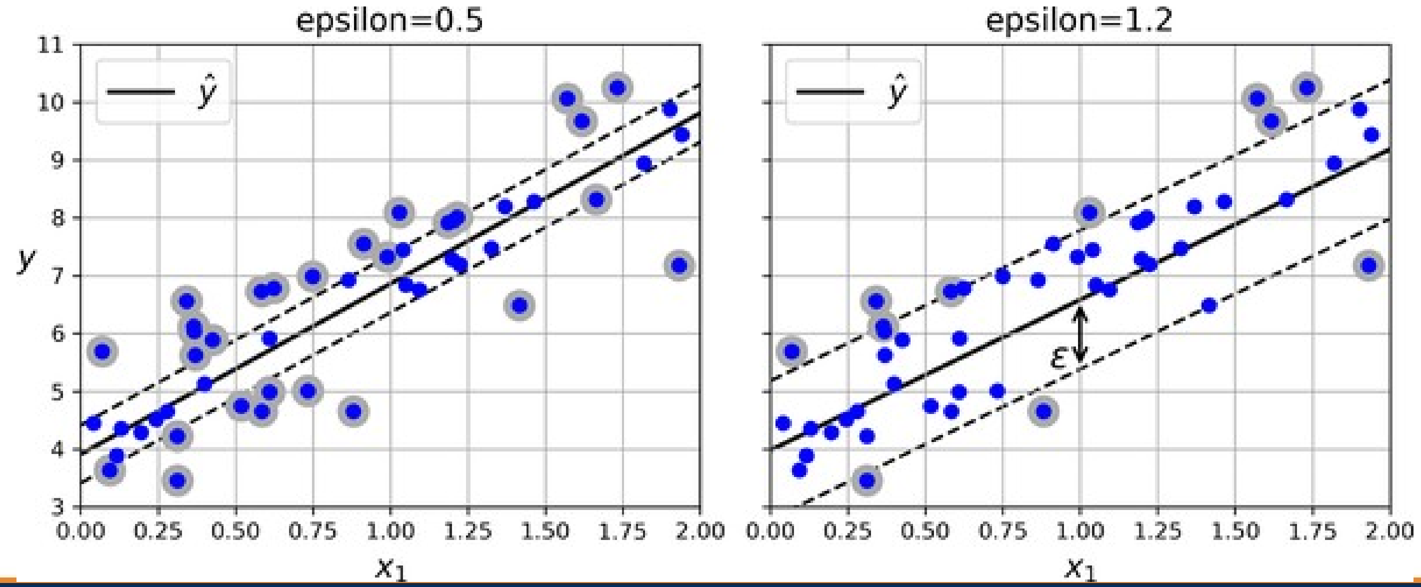
Class work

- Write code to classify setosa vs virginica species in the Iris dataset using only Petal length and Petal width
  - Ignore the instances of versicolor
- Try different kernels (each with its own hyperparameters)
- Which kernel needed the least support vectors?

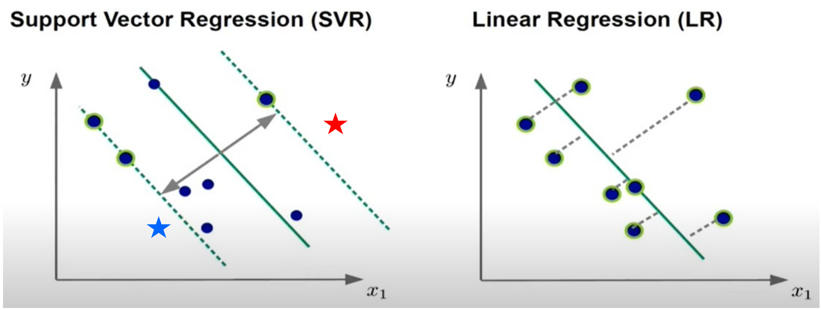
SUPPORT VECTOR REGRESSION (SVR)

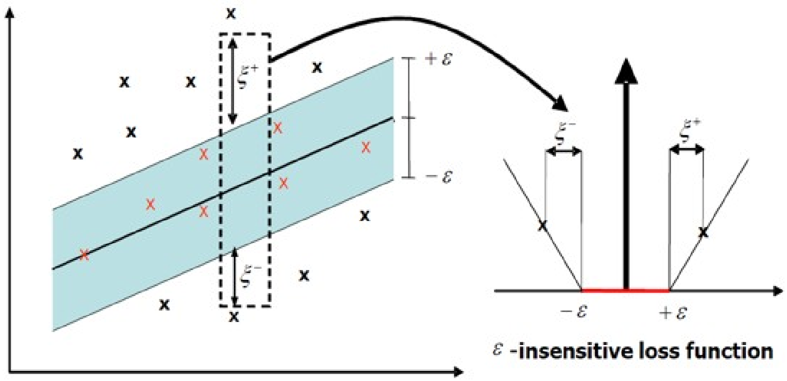
Support Vector Regression (SVR)

- To use SVMs for regression instead of classification, tweak the objective:
- instead of maximizing the margin ***between two classes***,
- SVM regression tries to fit as many instances as possible ***inside*** the margin.



Support Vector Regression (SVR)





- **Basic idea:**
  - Margin lines are chosen to cover all the data (hard margin) or allow for some violation (soft margin).
  - Use slack variables for the data points outside of margin (red and blue **stars**)
  - The rest is analogous to the classification.
- **SVR vs Linear Regression (LR)**
  - Unlike LR, SVR considers the support vectors (the outermost points).
    - SVR is less sensitive than LR for those data not on the regression line.
  - Unlike LR, the error function is ε-insensitive loss function.
    - Any deviations < ε is ignored (**ε-SVR**). This leads to a sparse solution like SVM.
  - Non-linear regression in SVR is more efficient using Kernel trick.

Epsilon-Support Vector Regression

- **The Primal Problem for ε-SVR:**
  -
  - Subject to
  - Like SVM, we can solve it using Quadratic Programming methods

Class work

- Load the iris dataset
- The goal is to classify instances as Species="virginica" or not given Petal.Length, Sepal.Length, Sepal.Length, and Sepal.Width
- Use logistic regression and experiment with different combinations of the features
- Use 3-fold cross-validation to evaluate models
  - Which combination of features gives the highest precision/recall?

```
from sklearn.linear_model import LogisticRegression
X = iris.data[["petal width (cm), ???"]].values
y = iris.target_names[iris.target] == 'virginica'

log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
log_reg.predict(..., ...)
```

Textbook code

- [Textbook code]
- [Textbook code on Google Colab]
- Open `05_support_vector_machines.ipynb`

Acknowledgement

- Many slides from Dr. Christopher Ryu
- Content based on “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow,” Aurélien Géron, 3rd Edition (October 2022), O'Reilly Media, Inc.