Support Vector Machines: An Introduction

By: Randy Jin and Jiawei Liu

Support Vector Machines (SVM)

SVM is binary classification algorithms.

To find a hyperplane separates classes
by the greatest distance (e.g. H_3 in Fig. 1)

Which is between two canonical hyperplanes
(dashed line in Fig. 2)

$$\mathbf{w} \cdot \mathbf{x} + b = 0,$$

So the distance between hyperplanes (*margin*) is maximized.

Points along the margin boundaries are ***support vectors***

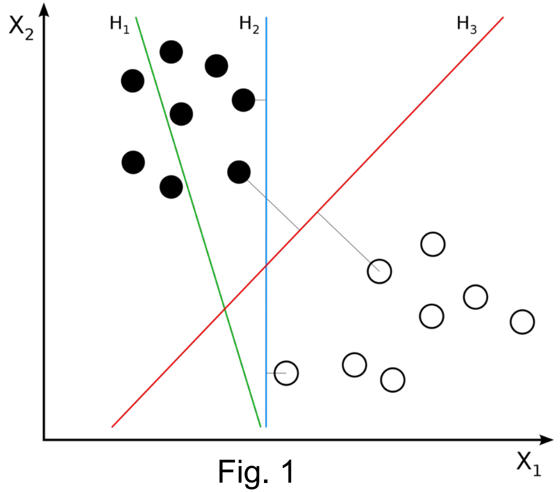


Fig. 1

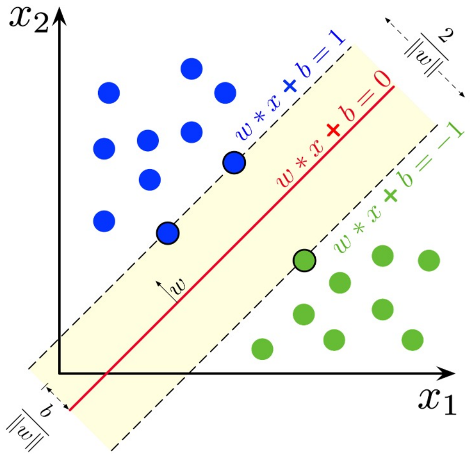


Fig. 2

Optimization

Correct classification: $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) > 0\ \forall i$

Let $\mathbf{x}_1$, $\mathbf{x}_2$ be two points inside the hyperplanes on the two classes, then:

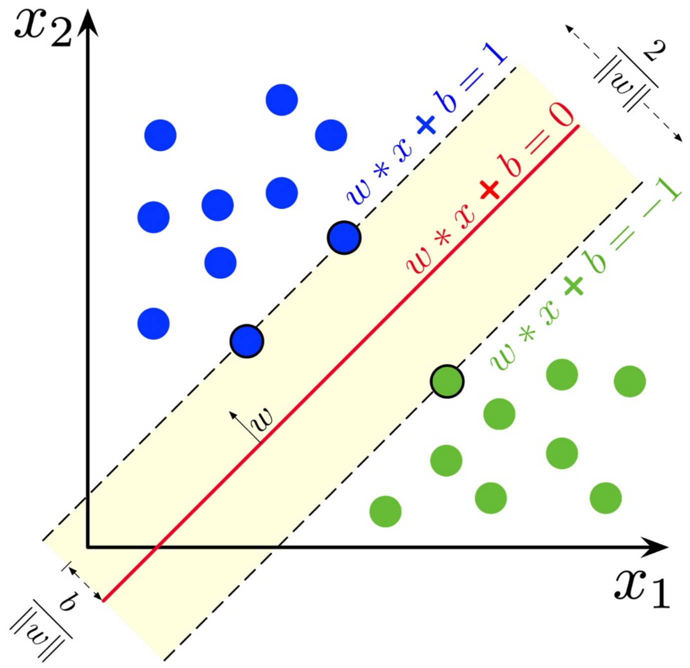
$$\mathbf{w}\cdot\mathbf{x}_1 + b = 1 \quad \mathbf{w}\cdot\mathbf{x}_2 + b = -1$$

The normal vector to the hyperplane:

$$\mathbf{w}/\,||\mathbf{w}||_2$$

The margin between two canonical hyperplanes is thus the projection of $\mathbf{x}_1 - \mathbf{x}_2$ onto $\mathbf{w}/\,||\mathbf{w}||_2$:

$$(\mathbf{x}_1 - \mathbf{x}_2)\cdot\mathbf{w}/\,||\mathbf{w}||_2 = 2/\,||\mathbf{w}||_2$$



Maximizing Margin Distance by Minimizing ||**w**||

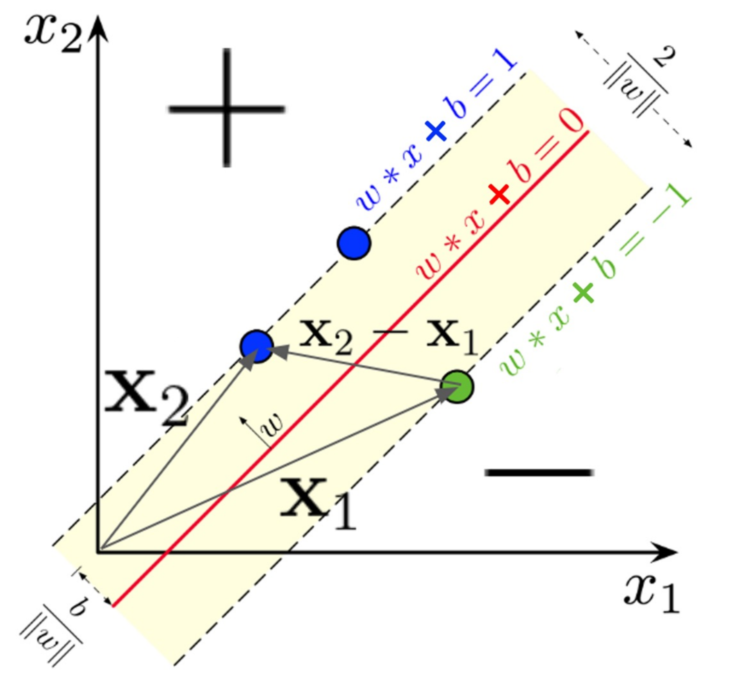
We have the function:

$$\max \frac{2}{\|\mathbf{w}\|} \rightarrow \min \frac{1}{2} \|\mathbf{w}\|^2$$

and the constraint for correctly classified subjects:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0$$

To solve a constrained optimization problem, we adopt the Lagrange multipliers



Lagrange multiplier

Maximizing the margin is therefore equivalent to:

- Minimizing objective function: $\frac{1}{2}||\mathbf{w}||_2^2$
- Subject to constraints: $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \qquad \forall i$

Equivalent to minimizing Lagrange function:

$$L(\mathbf{w}, b) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) - \sum_{i=1}^{m} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right)$$

where $\alpha_i \geq 0$ are the Lagrange multipliers, and *m* is the number of constraints.

Minimization of: $$L(\mathbf{w}, b) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) - \sum_{i=1}^{m} \alpha_i \left(y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right)$$

Solution: $$\frac{\partial L}{\partial b} = -\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = 0$$

$$\Longrightarrow \quad \mathbf{w} = \sum_{i} \alpha_i y_i \mathbf{x}_i$$

Plug into $L(\mathbf{w}, b)$, we have dual function with constraint:

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j \right) \qquad \alpha_i \geq 0 \qquad \sum_{i=1}^{m} \alpha_i y_i = 0$$

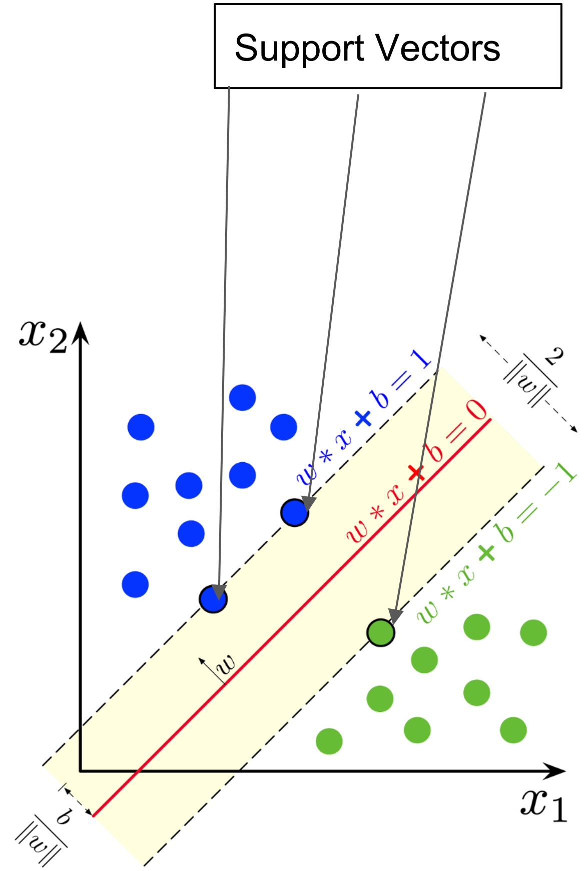
The dual function yields equivalent solutions to the Lagrange function!

Revisit Support Vectors

- Solving the Lagrange function yields:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

- When the maximal margin hyperplane is found, only those which lie closest to the hyperplane have $\alpha_i^* > .0$
- These points are the ***support vectors***.
- Removing non-support vectors does not affect the current separating hyperplane.
- Data points with large values of α_i^* have a large influence on the orientation of the hyperplane.



Soft Margin

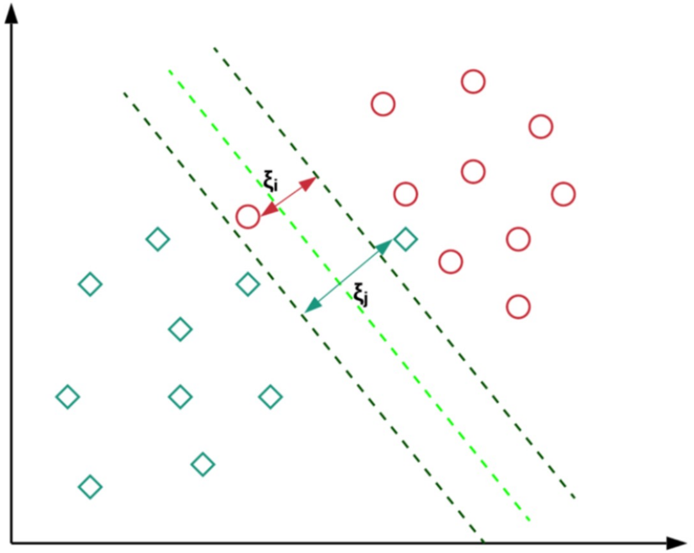
- What if data has noise or outliers?
  - Introduce soft margin to reduce effects of noise.
  - Non-negative slack variables ξ_i to relax the constraints and allow for classification errors:

$$y_i\,(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i$$

- The minimization problem is then the sum of errors $\sum_{i=1}^{m} \xi_i$ and $||\mathbf{w}||^2$:

$$\min\left[\frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i}\xi_i\right]$$

- C is a tuning parameter, and if $\xi_i > 0$ we allow for misclassifications and introduce margin errors.



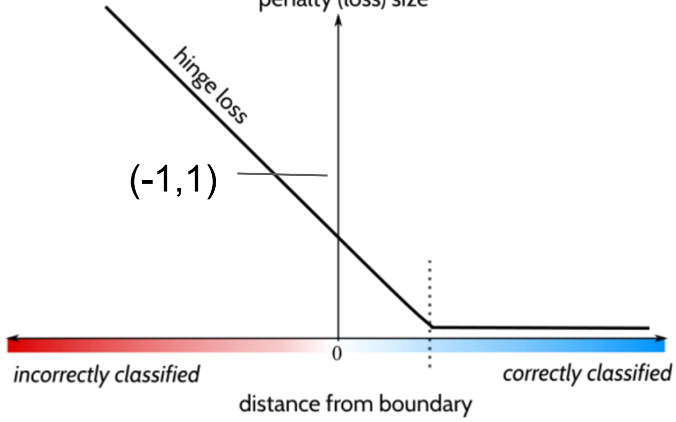
Hinge Loss

- The relaxed constraints $y_i\,(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i$ can be written more concisely with $\xi_i \geq 0$:

$$\xi_i = \max(0,1 - y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i - b))$$

- Rewrite the objective $\min\left[\frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_i \xi_i\right]$ in terms of data for optimization:

$$\min\left\{\underbrace{\frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w}}_{\text{regularization}} + C\sum_i \underbrace{\max\left(0,1 - y_i(\boldsymbol{w}\cdot x_i - b)\right)}_{\text{Hinge loss function}}\right\}$$



Non-linear Support Vector Machines

The data may not be linearly separable in **input space**

To get an alternative representation:

- mapping the data points into **feature space**, through a replacement

$$\mathbf{x}_i \cdot \mathbf{x}_j \rightarrow \Phi\left(\mathbf{x}_i\right) \cdot \Phi\left(\mathbf{x}_j\right)$$

where $\Phi(\cdot)$ is the mapping function

kernel: $K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \Phi\left(\mathbf{x}_i\right) \cdot \Phi\left(\mathbf{x}_j\right)$

- **Data which is not separable in input space can always be separated in a space (input space) of high enough dimensionality (feature space).**

Kernel Trick

- Assumption/Restriction:
  - must be a consistently defined inner product in feature space
  - an inner product space as a **Reproducing Kernel Hilbert Space (RKHS)**

- The functional form of the mapping **does not need to be known** since it is implicitly defined by the choice of kernel

  $$\textit{kernel:}\ K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

  or inner product in feature space.

$$\boxed{L(\mathbf{w}, b) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) - \sum_{i=1}^{m} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right)} \rightarrow \boxed{W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)} \rightarrow \boxed{W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)}$$

$$\min_{\{i|y_i=+1\}} [\mathbf{w} \cdot \mathbf{x}_i + b] = \min_{\{i|y_i=+1\}} \left[\sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)\right] + b = 1$$

Kernel Properties

- For any set of real-valued variables, a positive semi-definite (PSD) kernel satisfies

$$\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

- This type of kernel is symmetric, with positive components on the diagonal

$$K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i) \qquad K(\mathbf{x}, \mathbf{x}) \geq 0.$$

Mercer's Theorem: A symmetric function $k(.,.)$ is a kernel iff for any finite sample S the kernel matrix for S is positive semi-definite.

Simple Kernels

There is at least one kernel parameter

With many applications, simply defined kernels are sufficient.

Generalizing the linear kernel

- *homogeneous polynomial kernel* $$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$$
- inhomogeneous polynomial $$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d$$
- Gaussian kernel $$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x}_j)^2}{2\sigma^2}\right)$$
- hyperbolic tangent kernel $$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\kappa(\mathbf{x}_i \cdot \mathbf{x}_j) + c\right)$$

This kernel is not positive definite, in general, but has been used successfully in applications.

Build new kernels

$$K(\mathbf{x}_i, \mathbf{x}_j) = cK_1(\mathbf{x}_i, \mathbf{x}_j)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = cK_1(\mathbf{x}_i, \mathbf{x}_j) + dK_2(\mathbf{x}_i, \mathbf{x}_j)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = K_1(\mathbf{x}_i, \mathbf{x}_j)K_2(\mathbf{x}_i, \mathbf{x}_j)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{x}_i)K_1(\mathbf{x}_i, \mathbf{x}_j)f(\mathbf{x}_j)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(K_1(\mathbf{x}_i, \mathbf{x}_j))$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = p\left[K_1(\mathbf{x}_i, \mathbf{x}_j)\right]$$

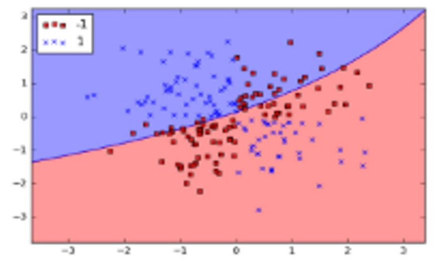
Kernel hyperparameters

- Cross validation: a validation study in which we train the learning machine with different, regularly spaced, choices of the kernel parameter and use that value which gives the lowest error on the validation data

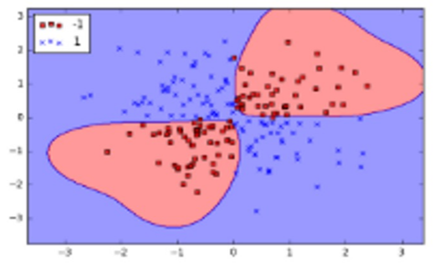
- Multiple kernel learning: Learning the coefficients of a linear combination of kernels with different pre-defined hyperparameters, and pick the ones with highest weights.

Overfitting

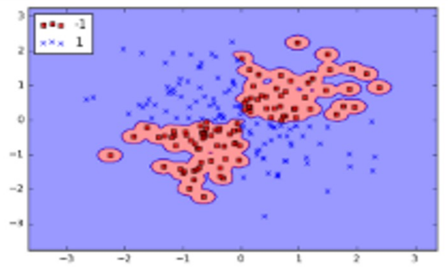
- Kernel trick is useful for finding many pattern types, but this gives a lot of freedom to bend the decision boundary to fit our training data.
- Tend to overfit and hard to generalize.
- To reduce overfitting:
  - Reduce kernel complexity (e.g. precision γ in Gaussian dist, and degree in polynomial kernel).
  - Higher *C* to increase penalization of the slack variables: $\min \left[\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_i \xi_i \right]$



$\gamma = 0.01$



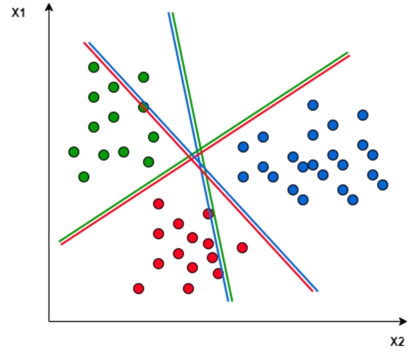
$\gamma = 1$



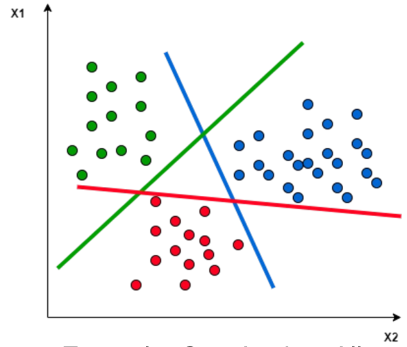
$\gamma = 100$

Multiclass Classification

- To this point, we have looked at binary classification - can we do more?
  - As long as we can break it down into multiple binary processes - yes!
- 3 Main approaches:
  - One-against-one: Fit $\binom{k}{2}$ classifications that separate each pair of groups
  - One-against-all: Fit *k* classifications that maximize distance between one group and points from all other groups (*Least efficient*)
  - DAGSVM: Fit $\binom{k}{2}$ classifications, new observations are fit by traversing a rooted-binary decision tree, using the classification decision at each step
- New predictions can be fit by max voting in one-against-one or one-against-all or by traversing tree in DAGSVM.



Example: One Against One



Example: One Against All

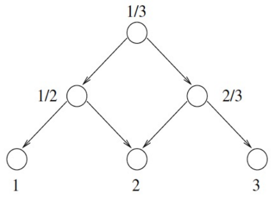


Figure 1.4: With DGSVM a multi-class classification problem is reduced to a series of binary classification tasks.

Feature Scaling and Outliers

- Recall the dual:

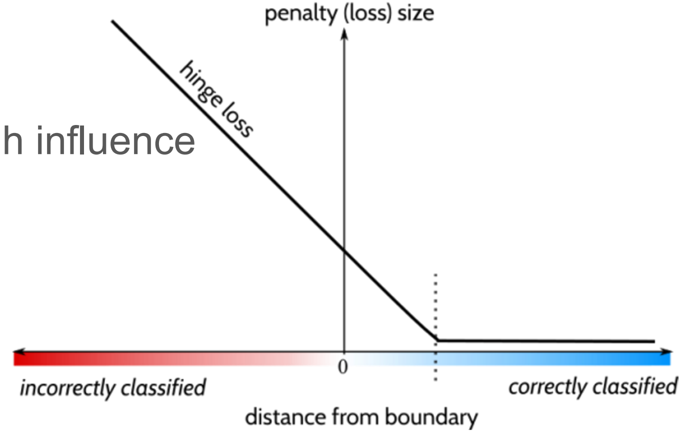
$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

- Model fitting is based on the inner product, which is highly sensitive to the scale of the features.
- Without proper scaling, features with larger scale would dominate the model fitting.
- Recall hinge loss:

$$\ell(y) = \max(0, 1 - t \cdot y)$$

$$t = \pm 1 \qquad y = \mathbf{w} \cdot \mathbf{x} + b$$

- Outliers will pose large penalty on the model, leading to high influence to the decision boundary.
- The data needs to be carefully preprocessed!



Computational Cost

- SVM is generally computationally expensive.
- SVM requires storing an n x n matrix and computing n^2 dot products, so memory requirements and training time may be considerable with high n
- For linear SVM, computational training complexity is $O(n^2)$.
- For SVM with a radial basis function, training complexity is more likely to be $O(dn^2)$, where d is the number of features
- Testing efficiency is also slow compared to other methods (e.g. logistic).
- If sample size is relatively small, than even with high number of features computation time is favorable.

Interpreting Predictions

- Predictions based on distance to separating hyperplane:

$$\phi(\mathbf{z}) = \sum_i y_i \alpha_i K(\mathbf{x}_i, \mathbf{z}) + b$$

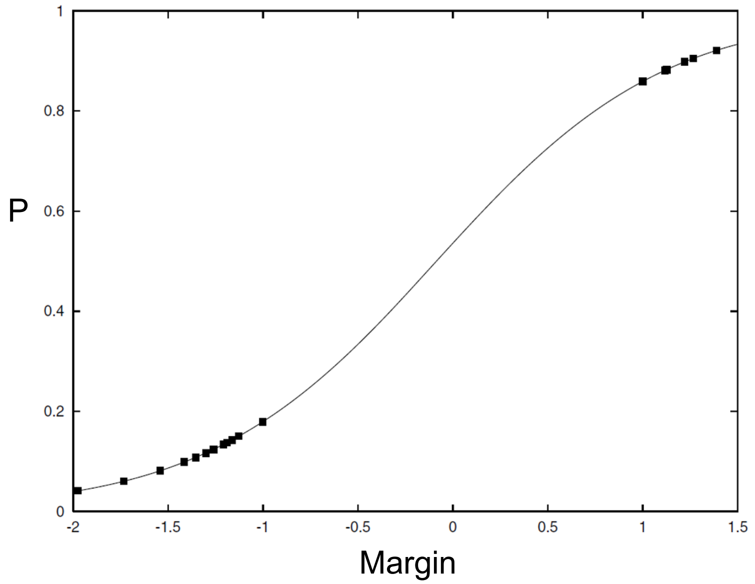
- Could be used for outlier detection.
- No confidence measure associated with the predicted label.
- Large distance might indicate higher confidence.

Platt’s scaling

- Platt’s scaling: logistic transformation producing probabilistic estimates.

$$p(y=+1|\phi)=\frac{1}{1+\exp(A\phi+B)}$$

- Not a real probability:
  - No statistical foundation.
  - Other transformation could also do the trick, no justification why using logit transformation.
  - Distance is sensitive to data scale, thus not robust.
- Bad performance when the ratio of the score distributions for both binary classes are not similar.

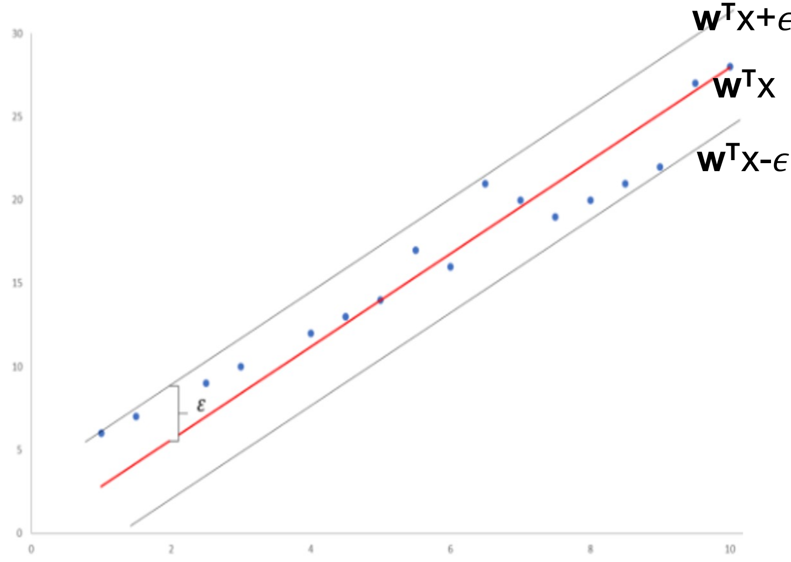


Support Vector Regression (SVR)

- Any SVM analogue to real-value labels?
- SVR aims to fit the error within a certain threshold.
- Considering the constraints:

$$\begin{aligned} y_i - \mathbf{w}^T\mathbf{x}_i - b &\le \epsilon \\ \mathbf{w}^T\mathbf{x}_i + b - y_i &\le \epsilon \end{aligned} \quad \forall i$$

- SVR finds a function that deviates from each datapoint by a value no greater than ϵ.
- At the same time, the function is as flat as possible.



Extension of ϵ-SVR

- What if data falls outside the ϵ-margin?
- Introduction of slack variable.

$$\min_{\mathbf{w},\xi_i,\widehat{\xi}_i} \left[\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\left(\xi_i + \widehat{\xi}_i\right) \right]$$

With constraints:

$$\begin{aligned} y_i - \mathbf{w}^T\mathbf{x}_i - b &\leq \epsilon + \xi_i \\ (\mathbf{w}^T\mathbf{x}_i + b) - y_i &\leq \epsilon + \widehat{\xi}_i \end{aligned}$$

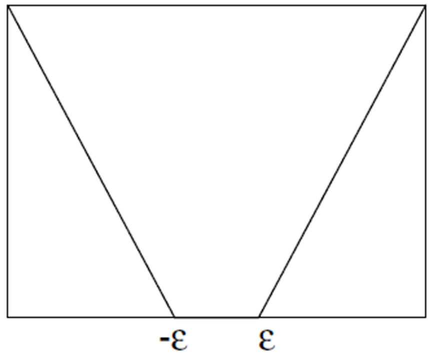
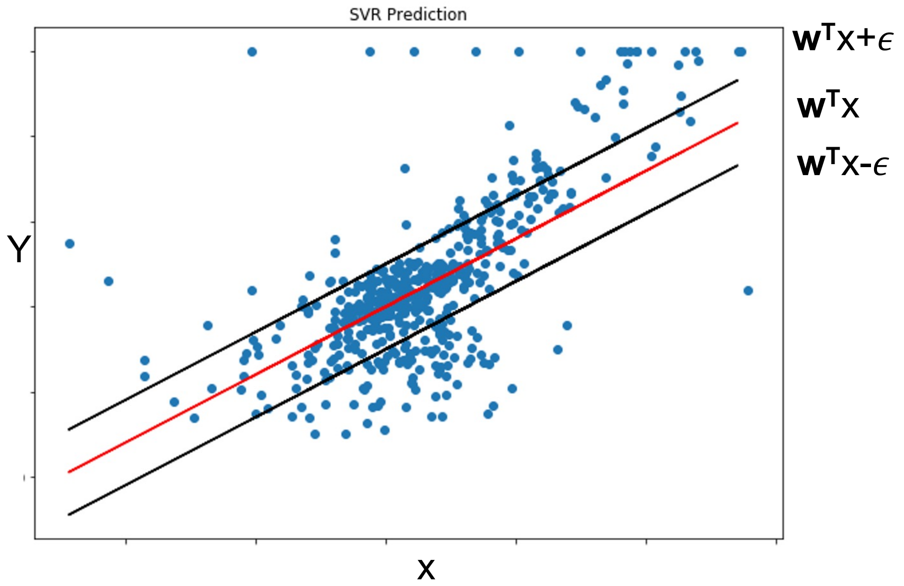


Fig. The loss function for SVR with maximum error tolerance ϵ.

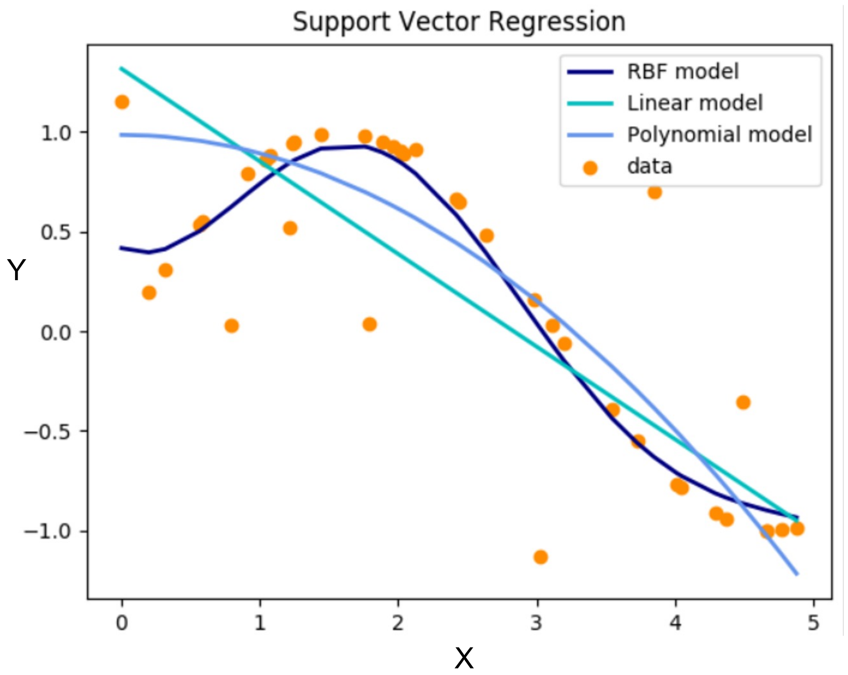


Extension of ϵ-SVR

- Non linear SVR using kernel trick.
- After kernel substitution:

$$
\begin{aligned}
W(\alpha,\widehat{\alpha}) &= \sum_{i=1}^{m} y_i(\alpha_i-\widehat{\alpha}_i)-\epsilon\sum_{i=1}^{m}(\alpha_i+\widehat{\alpha}_i)\\
&-\frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i-\widehat{\alpha}_i)(\alpha_j-\widehat{\alpha}_j)\boxed{K(x_i,x_j)}
\end{aligned}
$$

Subject to $\sum_{i=1}^{m}\widehat{\alpha}_i=\sum_{i=1}^{m}\alpha_i$, $0\le\alpha_i\le C$, and $0\le\widehat{\alpha}_i\le C$



Consideration of SVR

- SVR is an extension of SVM.
- SVR finds the best fit as the line containing the maximum number of points.
- Pros:
  - Can handle non i.i.d. observations.
  - Can model non-linear relationships.
- Cons:
  - Harder to understand.
  - Harder interpretation compared to least squared models.
  - Lack of uncertainty measurements.