Lecture 11: Support Vector Machines

Statistical Learning and Data Mining

Xingye Qiao

Department of Mathematical Sciences

Binghamton University

E-mail: qiao@math.binghamton.edu

Read: ELSII Chs. 4.5 and 12, ISLR Ch. 9, and SLS Ch. 3.6

Outline

1. Separable Data: Maximum Margin Classifiers
2. Linear SVM
3. Hinge Loss + Penalty
4. SVM Dual Problem
5. Non-Linear SVMs and the Kernel Trick

What's the Problem?

- Training Data: $\left\{(y_i, \boldsymbol{x}_i),\ i = 1 \cdots n,\ y_i \in \mathcal{Y},\ \boldsymbol{x}_i \in \mathcal{S} \subset \mathbb{R}^d\right\}$.
- Goal: a mapping $\phi:\ \mathcal{S} \mapsto \mathcal{Y}$
  - Inputs: $\boldsymbol{x} \in \mathcal{S}$.
  - Outputs: $y \in \mathcal{Y}$.

Regression Setting

- $\mathcal{Y} \subset \mathbb{R}$.
- Predict y_i as $\widehat{y_i} = \phi(\boldsymbol{x}_i)$, so that $\sum_i (y_i - \widehat{y_i})^2$ is as small as possible.

(Binary) Classification Setting

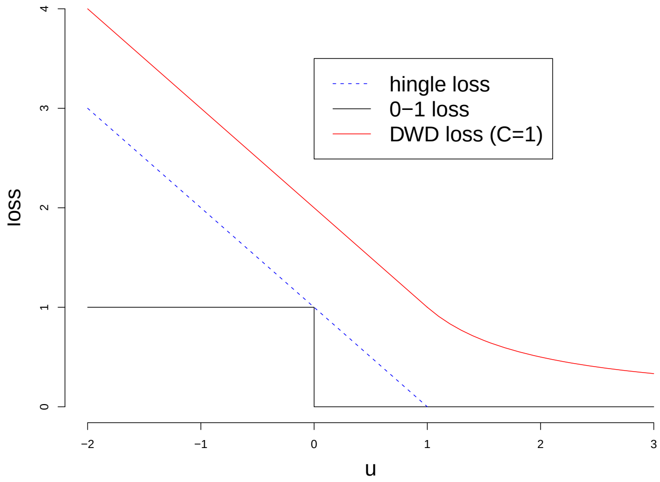
- $\mathcal{Y} = \{-1, +1\}$.
- Predict y_i as $\widehat{y_i} = \phi(\boldsymbol{x}_i)$, so that $\sum_i \mathbb{1}\{y_i \neq \widehat{y_i}\}$ is as small as possible.

0-1 Loss or other Loss

- Goal: a mapping $\phi: \mathcal{S} \mapsto \mathcal{Y}$, which predicts y_i as $\widehat{y_i} = \phi(\boldsymbol{x}_i)$, so that $y_i \neq \widehat{y_i}$ as few as possible.
- Introduce a function of $\boldsymbol{x}$, $f(\boldsymbol{x})$: let $\phi(\boldsymbol{x}) = \mathrm{sign}(f(\boldsymbol{x}))$.
  - $f(\boldsymbol{x}) > 0 \Rightarrow +1$ (positive class; case),
  - $f(\boldsymbol{x}) < 0 \Rightarrow -1$ (negative class; control).
- $\min_f \sum_i \mathbb{1}\{y_i \neq \phi(\boldsymbol{x}_i)\}$, or equivalently

$$\min_f \sum_i \mathbb{1}\{y_i f(\boldsymbol{x}_i) \leq 0\}$$

- 0-1 loss function: $\mathbb{1}\{u \leq 0\}$, where $u = y_i f(\boldsymbol{x}_i)$.
  - may not work: not convex !!!
- What function looks similar to 0-1 loss, but is convex?
  - One answer: Hinge loss, $[1-u]_+$.


hingle loss
0−1 loss
DWD loss (C=1)
loss
u
0
1
2
3
4
−2
−1
0
1
2
3

Linear Discrimination

- If $f(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{\omega} + \beta$ is linear, then $f(\boldsymbol{x}) = 0$ is a hyperplane.
- Want: a hyperplane in the middle of two classes.

The next section would be . . . . . .

1 Separable Data: Maximum Margin Classifiers

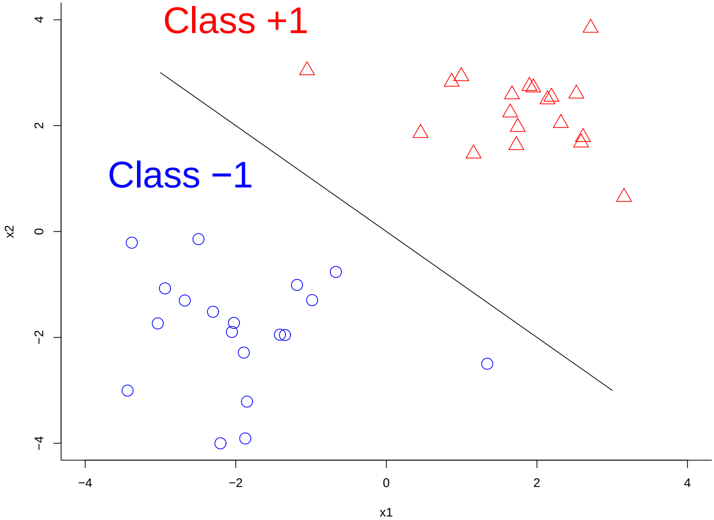
2 Linear SVM

3 Hinge Loss + Penalty

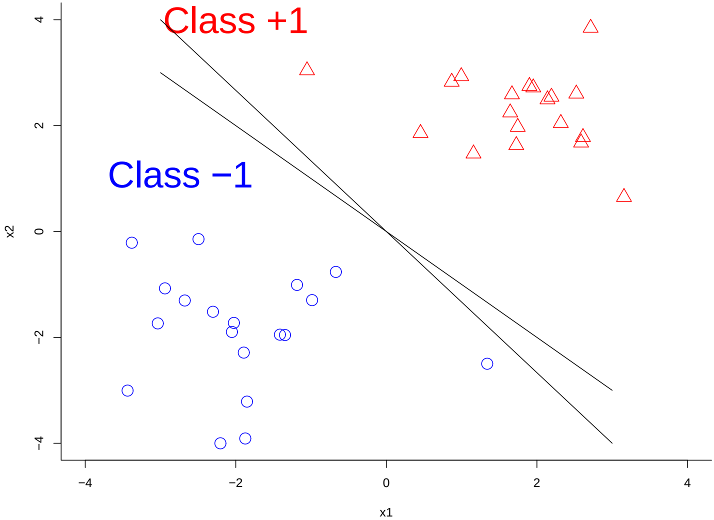
4 SVM Dual Problem

5 Non-Linear SVMs and the Kernel Trick

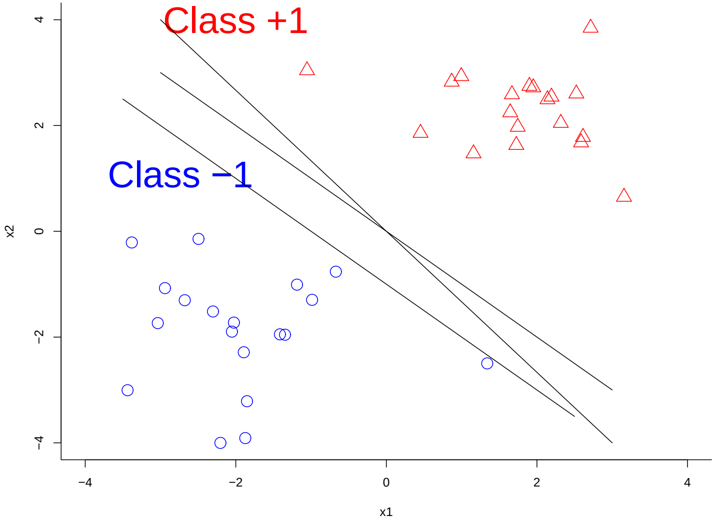
Separable Case



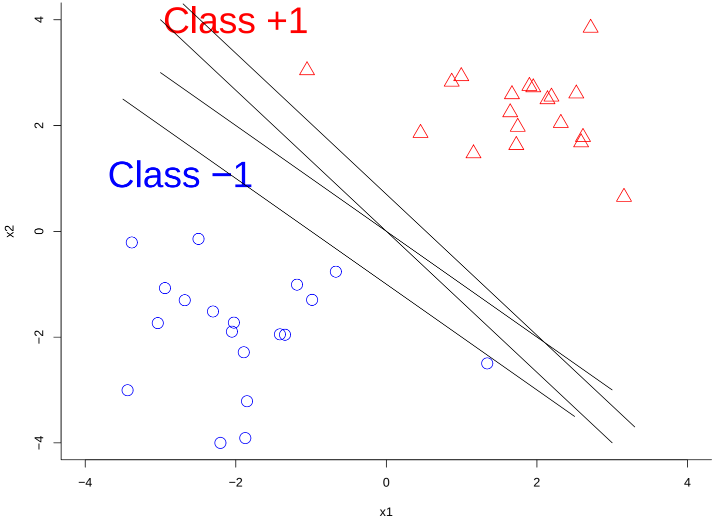
Separable Case



Separable Case



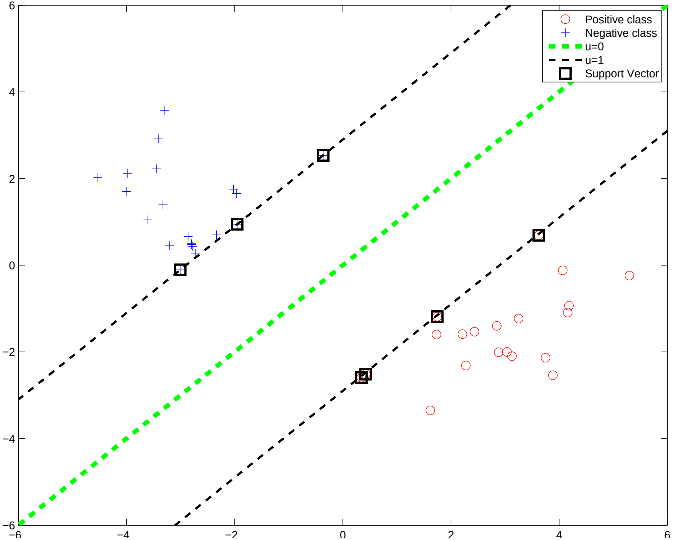
Separable Case



Linear Discrimination

- If $f(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{\omega} + \beta$ is linear, then $f(\boldsymbol{x}) = 0$ is a hyperplane.
- Want: a hyperplane in the middle of two classes.
- Want: the margin induced by the hyperplane between these two classes to be large.

Illustration Plot of SVM



Three hyperplanes

- The separating hyperplane: $\{\boldsymbol{x} : f(\boldsymbol{x}) = 0\}$. It has dimension $(p-1)$.
- The + plane: $\{\boldsymbol{x} : f(\boldsymbol{x}) = A\}$ for some $A > 0$
- The − plane: $\{\boldsymbol{x} : f(\boldsymbol{x}) = -A\}$
- The support vectors: those points which the $\pm$ planes go through. Denoted as $\boldsymbol{x}_{SV}$. Clearly

$$A = |f(x_{SV})|$$

- **The margin** := the distance between the $\pm$ planes

$$= \frac{2A}{\|\boldsymbol{\omega}\|}$$

We want to maximize it.

- By the definition of support vectors, also need to make sure that non-SVs are farther away:

$$f(\boldsymbol{x}_i) \geq A \text{ for all } y_i = 1 \text{ and } f(\boldsymbol{x}_i) \leq -A \text{ for all } y_i = -1$$

$$\forall i,\ y_i f(\boldsymbol{x}_i) \geq A$$

- $\|\boldsymbol{\omega}\|$ can be arbitrary, making $y_i f(\boldsymbol{x}_i)$ hard to control.
  - Rescale $\|\omega\|$, so that $A = |f(\boldsymbol{x}_{SV})| = 1$.
- So we try to

$$\max_{\boldsymbol{\omega},\beta} \frac{2}{\|\boldsymbol{\omega}\|},$$

$$\text{s.t. } y_i f(\boldsymbol{x}_i) \geq 1.$$

- Equivalently,

$$\min_{\boldsymbol{\omega},\beta} \frac{\|\boldsymbol{\omega}\|^2}{2},$$

$$\text{s.t. } y_i f(\boldsymbol{x}_i) \geq 1.$$

Maximum Margin Classifiers

$$\min_{\boldsymbol{\omega},\beta} \frac{\|\boldsymbol{\omega}\|^2}{2}, \text{ s.t. } y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b) \geq 1.$$

Can be convert to primal form optimization problem

$$\min_{\boldsymbol{\omega},\beta} \frac{\|\boldsymbol{\omega}\|^2}{2} - \sum_{i=1}^{n} \alpha_i y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b) + \sum_{i=1}^{n} \alpha_i,$$

- Gradient equation: $-\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i + \boldsymbol{\omega} = 0$
- That is the solution is $\boldsymbol{\omega} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$
- As will be shown later for the SVM, $\alpha > 0$ when $\boldsymbol{x}_i$ is a support vector, or $= 0$ otherwise.
- In other words, $\boldsymbol{\omega}$ is defined by support vectors only.

The next section would be . . . . . .

1 Separable Data: Maximum Margin Classifiers

2 Linear SVM

3 Hinge Loss + Penalty

4 SVM Dual Problem

5 Non-Linear SVMs and the Kernel Trick

Non-separable Case

$$\min_{\boldsymbol{\omega},\beta} \frac{\|\boldsymbol{\omega}\|^2}{2}, \text{ s.t. } y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b) \geq 1.$$

- If "$\forall i,\ y_i f(\boldsymbol{x}_i) \geq 1$" is absolutely impossible, introduce a slack variable $\xi_i \geq 0$ for each data vector $\boldsymbol{x}_i$.
- $y_i f(\boldsymbol{x}_i) \geq 1$ is relaxed to $y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i$.
- ξ_i represents the violation of 'correct classification'.
- We want $\sum_i \xi_i$ to be small. Add it to the objective function, rescaled by a constant C.

$$\min_{\boldsymbol{\omega},\beta,\boldsymbol{\xi}} \left(\frac{\|\boldsymbol{\omega}\|^2}{2} + C \sum_i \xi_i \right),$$

$$\text{s.t. } y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i,$$

$$\xi_i \geq 0.$$

Regularization Framework

- Equivalent to,

$$\min_{\boldsymbol{\omega},\beta} \left(\frac{\|\boldsymbol{\omega}\|^2}{2} + C \sum_i [1 - y_i f(\boldsymbol{x}_i)]_+ \right).$$

- Or,

$$\min_{\boldsymbol{\omega},\beta} \left(\frac{1}{n} \sum_i [1 - y_i f(\boldsymbol{x}_i)]_+ + \lambda \frac{\|\boldsymbol{\omega}\|^2}{2} \right).$$

- Or,

$$\min_{\boldsymbol{\omega},\beta} \frac{1}{n} \sum_i [1 - y_i f(\boldsymbol{x}_i)]_+,$$

$$\text{s.t. } \|\boldsymbol{\omega}\|^2 \leq L.$$

Interpretations

- Solved by Quadratic Programming (QP) of the duality problem. Details to come.
- Final solution is

$$\boldsymbol{\omega} = \sum_i \alpha_i y_i \boldsymbol{x}_i.$$

- For support vectors, $\alpha_i > 0$; otherwise $\alpha_i = 0$
- Important observation:
  determined / influenced by the support vectors only.
- Points in the margin? $|f(\boldsymbol{x}_i)| < 1$
- Misclassified points? $f(\boldsymbol{x}_i) < 0$

The next section would be . . . . . .

1 Separable Data: Maximum Margin Classifiers

2 Linear SVM

3 Hinge Loss + Penalty

4 SVM Dual Problem

5 Non-Linear SVMs and the Kernel Trick

Hinge Loss + Penalty

1. The maximum margin classifier for the linearly separable case is called hard-margin SVM.
2. The non-(linearly separable) case SVM with slack variables is called (1-norm) soft-margin SVM, which can be expressed as

$$\min_{\boldsymbol{\omega},\beta} \sum_{i=1}^{n} [1 - y_i(\beta + \boldsymbol{\omega}^T \boldsymbol{x}_i)]_+ + \lambda \|\boldsymbol{\omega}\|_2$$

   which is the sum of the Hinge loss functions over the sample plus a ℓ_2 penalty term.

Hinge Loss:

$$H(u) = [1 - u]_+, \text{ where } u = y_i f(\boldsymbol{x}_i) = y_i(\beta + \boldsymbol{\omega}^T \boldsymbol{x}_i)$$

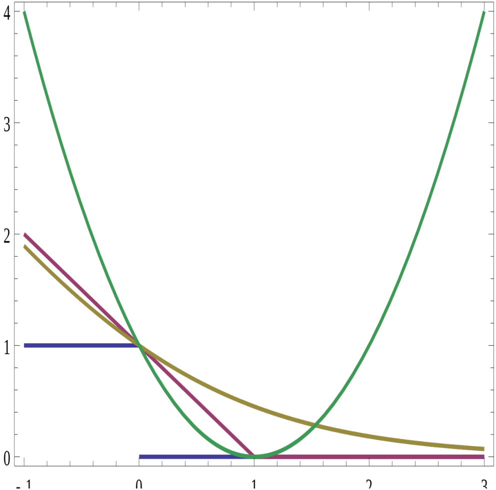
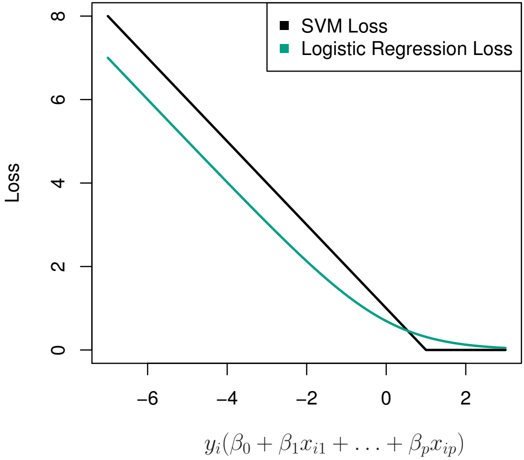


Figure: Blue is the 0-1 indicator function. Green is the square loss function. Purple is the hinge loss function. Yellow is the logistic loss function.



Hinge Loss:

$$H(u) = [1 - u]_+$$

Logistic Loss:

$$L(u) = \log(\exp(-u) + 1)$$

- Logistic loss is almost linear for very negatively large u, like the Hinge loss.
- Logistic loss is never exactly zero, while Hinge loss is 0 for large u.
- Logistic regression and the support vector classifier often give very similar results.
- When the classes are well separated, SVMs tend to behave better than logistic regression; in more overlapping regimes, logistic regression is often preferred.

Some extensions of SVM

Recall: (1-norm) soft-margin SVM

$$\min_{\boldsymbol{\omega},\beta} \sum_{i=1}^{n} [1 - y_i(\beta + \boldsymbol{\omega}^T \boldsymbol{x}_i)]_+ + \lambda \|\boldsymbol{\omega}\|_2$$

2-norm soft-margin SVM

$$\min_{\boldsymbol{\omega},\beta} \sum_{i=1}^{n} \left\{ [1 - y_i(\beta + \boldsymbol{\omega}^T \boldsymbol{x}_i)]_+ \right\}^2 + \lambda \|\boldsymbol{\omega}\|_2$$

ℓ_1 (Sparse) SVM

$$\min_{\boldsymbol{\omega},\beta} \sum_{i=1}^{n} [1 - y_i(\beta + \boldsymbol{\omega}^T \boldsymbol{x}_i)]_+ + \lambda \|\boldsymbol{\omega}\|_1$$

Support vector regression (omitted).

Multiclass SVM

For multiclass-classification with k levels, $k > 2$, `libsvm` uses the one-against-one-approach, in which $k(k-1)/2$ binary classifiers are trained; the appropriate class is found by a voting scheme.

There are proposals for multiclass SVM using a single optimization. But this is beyond the scope of this class.

The next section would be . . . . . .

1 Separable Data: Maximum Margin Classifiers

2 Linear SVM

3 Hinge Loss + Penalty

4 SVM Dual Problem

5 Non-Linear SVMs and the Kernel Trick

Solution

- How to solve SVM?
- Take the following formulation as an example:

$$\text{Primal:} \quad \min_{\boldsymbol{\omega},\beta,\boldsymbol{\xi}} \left(\frac{\|\boldsymbol{\omega}\|^2}{2} + C \sum_i \xi_i \right),$$

$$\text{s.t. } y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i,$$

$$\xi_i \geq 0.$$

- The primal problem is Quadratic Programming (QP) with $d + 1 + n$ variables.
- However, often solve the duality problem, which is a QP problem with fewer variables.
- Can use a standard QP package to solve it.

Duality of SVM

- Lagrangian:

$$L_S = \frac{\boldsymbol{\omega}'\boldsymbol{\omega}}{2} + \sum_{i=1}^{n} \left\{ C\xi_i - \alpha_i[y_i(\boldsymbol{x}_i'\boldsymbol{\omega} + \beta) + \xi_i - 1] - \mu_i\xi_i \right\}.$$

Here $\alpha_i \geq 0, \ \mu_i \geq 0$ are KKT multipliers.

KKT Condition for Optimality

- KKT **stationary conditions** say $\frac{\partial L_S}{\partial \boldsymbol{\omega}}$, $\frac{\partial L_S}{\partial \beta}$ and $\frac{\partial L_S}{\partial \xi_i}$ all need to equal to 0.

$$\boldsymbol{\omega} - \sum_i \alpha_i y_i \boldsymbol{x}_i = \mathbf{0},$$

$$\sum_i \alpha_i y_i = 0,$$

$$C - \alpha_i - \mu_i = 0.$$

- KKT **complementary slackness condition** says that,

$$\alpha_i [y_i(\boldsymbol{x}_i' \boldsymbol{\omega} + \beta) + \xi_i - 1] \equiv 0 \text{ and}$$

$$\mu_i \xi_i \equiv 0$$

Solve the dual form

- KKT conditions combined with $\alpha_i \geq 0$, $\mu_i \geq 0$, lead to the dual problem:

$$\text{Dual:} \quad \max_{\alpha_i} \left(-\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i' \boldsymbol{x}_j + \sum_i \alpha_i \right)$$

$$\text{s.t.} \ \sum_i \alpha_i y_i = 0, \ 0 \leq \alpha_i \leq C.$$

- This is a standard QP problem with α_i's as the unknowns.
- There are n unknown variables instead of $d + 1 + n$ as in the primal problem.

From Dual Solution to Primal Solution

- The $\boldsymbol{\omega}$ can be found by the first KKT condition:

$$\boldsymbol{\omega} = \sum_i \alpha_i y_i \boldsymbol{x}_i.$$

- Also note that $\alpha_i[y_i(\boldsymbol{x}_i'\boldsymbol{\omega} + \beta) + \xi_i - 1] \equiv 0$ and $\mu_i \xi_i \equiv 0$, and $C - \alpha_i - \mu_i = 0$.
- Thus

$$\alpha_i > 0 \Rightarrow y_i(\boldsymbol{x}_i'\boldsymbol{\omega} + \beta) + \xi_i - 1 = 0,$$

$$\alpha_i < C \Rightarrow \mu_i \neq 0 \Rightarrow \xi_i = 0.$$

- In order to find out β, we find a data vector $(\boldsymbol{x}_i, y_i)$ where $0 < \alpha_i < C$, and calculate $\beta = -(\boldsymbol{x}_i'\boldsymbol{\omega}) + 1/y_i$. (Or take an average of such estimators.)

Support vectors and α_i

We call the data vectors with $0 < \alpha_i \leq C$ the support vectors

- Since $\alpha > 0$, $y_i(\boldsymbol{x}_i'\boldsymbol{\omega} + \beta) + \xi_i - 1 = 0$ holds exactly. Support vectors are either on one of the $\pm$ planes, or fall into the middle, or are completely misclassified.
- Among those which are on the planes ($\xi_i = 0$), we have points with $0 < \alpha_i < C$. This is because if $\alpha_i < C$, then $\mu_i \neq 0$, then ξ_i must $= 0$.
- The reminders of the support vectors ($\xi_i > 0$) have $\alpha_i = C$

If $\alpha_i = 0$ (not a support vectors), then the data i corresponds to $y_i f(\boldsymbol{x}_i) > 1$.

The next section would be ......

1 Separable Data: Maximum Margin Classifiers

2 Linear SVM

3 Hinge Loss + Penalty

4 SVM Dual Problem

5 Non-Linear SVMs and the Kernel Trick

Dual Problem of SVM

For training:

$$\max_{\alpha_i} \left(-\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i' \boldsymbol{x}_j + \sum_i \alpha_i \right)$$

$$\text{s.t. } \sum_i \alpha_i y_i = 0, \ 0 \leq \alpha_i \leq C.$$

For prediction:

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{\omega} + b = \sum_i \alpha_i y_i \boldsymbol{x}^T \boldsymbol{x}_i + b.$$

since $\boldsymbol{\omega} = \sum_i \alpha_i y_i \boldsymbol{x}_i$

- training depends on inner products $\boldsymbol{x}_i^T \boldsymbol{x}_j$, for $i, j = 1, \ldots, n$ only.
- prediction depends on inner products $\boldsymbol{x}^T \boldsymbol{x}_i$, for $i = 1, \ldots, n$ only.

Kernel Trick

Replace ${x_i}^T x_j$ with $k(x_i, x_j)$ for a pre-chosen kernel function $k : \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$.

- ${x_i}^T x_j$ is the dot product between x_i and x_j in the Euclidean space.
- What really happens in kernel trick:
  1. each input vector $\boldsymbol{x}_i$ is mapped to a feature first before applying SVM:
  $$\boldsymbol{x}_i \mapsto \phi(\boldsymbol{x}_i) \in \mathbb{R}^Q,$$
  2. only the inner products of the mapped features $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ are needed in computation.
  3. It so happens that $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Training

$$\max_{\alpha_i} \left(-\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i' \boldsymbol{x}_j + \sum_i \alpha_i \right)$$

$$\text{s.t. } \sum_i \alpha_i y_i = 0,\ 0 \le \alpha_i \le C$$

becomes

$$\max_{\alpha_i} \left(-\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_i \alpha_i \right)$$

$$\text{s.t. } \sum_i \alpha_i y_i = 0,\ 0 \le \alpha_i \le C$$

Prediction

$$f(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{\omega} + \beta = \boldsymbol{x}' \sum_i \alpha_i y_i \boldsymbol{x}_i + \beta = \sum_i \alpha_i y_i \boldsymbol{x}' \boldsymbol{x}_i + \beta$$

where $\boldsymbol{\omega} = \sum_i \alpha_i y_i \boldsymbol{x}_i$. has become

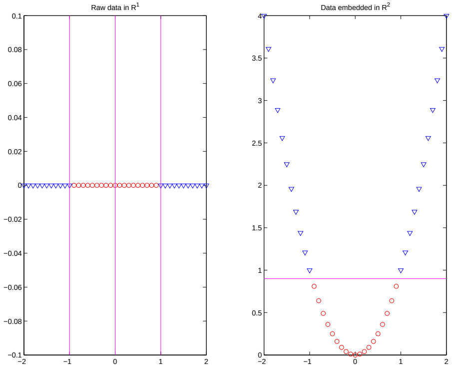
$$f(\boldsymbol{x}) = \sum_i \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x}_i) + \beta$$

Note that for fixed $\boldsymbol{x}_i$, $k(\boldsymbol{x}, \boldsymbol{x}_i)$ is usually not a linear function of $\boldsymbol{x}$. Hence instead of a flat separating hyperplane, the kernel SVM has a curved separating boundary

$$\{\boldsymbol{x} : \sum_i \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x}_i) + \beta = 0\}$$

Why kernel trick works?

- Replacing $\boldsymbol{x}_i'\boldsymbol{x}_j$ by $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a wonderful idea that can extend linear SVM to nonlinear situations.
- **Kernel SVM = "linear" SVM conducted in a feature space with transformed training data** $(\phi(\boldsymbol{x}_i), y_i)$
- Example of a more general methods called *embedding*.
- Example: $\phi(x) : x \mapsto (x, x^2)^T$. Then instead of a linear SVM in the $\mathbb{R}$ space, do it in the $\mathbb{R}^2$ space, with the transformed $((x_i, x_i^2)^T, y_i)$ as training data.



- It was hopeless to find a 'good' cut-off point in $\mathbb{R}$; it is easy to find a separating line in $\mathbb{R}^2$. When the data is 'bend' back to $\mathbb{R}^1$, the separating line becomes two cut-off points.
- Imagine that now ϕ is much more sophisticated and it can map to a richer feature space.

A more advanced example

Consider a less trivial example

- Let $\boldsymbol{x} = (x_1, x_2)' \in \mathbb{R}^2$, and $\phi(\boldsymbol{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)'$.
- Then the new features $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{y})$ has inner product

$$\begin{aligned}\phi(\boldsymbol{x})'\phi(\boldsymbol{y}) &= (x_1^2, x_2^2, \sqrt{2}x_1x_2)(y_1^2, y_2^2, \sqrt{2}y_1y_2)' \\ &= (x_1y_1 + x_2y_2)^2 = (\boldsymbol{x}'\boldsymbol{y})^2,\end{aligned}$$

  which corresponds to the polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (c + \boldsymbol{x}'\boldsymbol{y})^d$ with $c = 0$ and $d = 2$

Remember in this case:

$$\phi(\boldsymbol{x})'\phi(\boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y})$$

- When we conduct linear SVM in the new feature space, with training data $(\phi(\boldsymbol{x}_i), y_i)_{i=1,\ldots,n}$, the resulting classifier would be

$$f(\boldsymbol{x}) = \phi(\boldsymbol{x})' \sum_i \alpha_i y_i \phi(\boldsymbol{x}_i) + \beta = \sum_i \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x}_i) + \beta$$

- **In both training and in prediction, the precise form of $\phi(\boldsymbol{x})$ is not needed, only that of $k(\cdot, \cdot)$ is.**

ϕ can map to infinite dimensional space

For example, for Gaussian kernel, and assume $x \in \mathbb{R}^1$.

$$
\begin{aligned}
k(x_i, x_j) &= e^{-\gamma\|x_i - x_j\|^2} = e^{-\gamma(x_i - x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2} \\
&= e^{-\gamma x_i^2 - \gamma x_j^2}\left(1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \dots\right) \\
&= e^{-\gamma x_i^2 - \gamma x_j^2}\left(1 + \sqrt{\frac{2\gamma}{1!}}x_i \cdot \sqrt{\frac{2\gamma}{1!}}x_j + \sqrt{\frac{(2\gamma)^2}{2!}}x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}}x_j^2 + \dots\right) \\
&= \phi(x_i)^T \phi(x_j)
\end{aligned}
$$

where

$$
\phi(x) = e^{-\gamma x^2}\left(1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \dots\right) \in \mathbb{R}^\infty
$$

Again, don't torture yourself. Only need to know the kernel!

Examples of popular kernels

- The linear kernel:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}'\boldsymbol{y}.$$

  This leads to the original, linear SVM.

- The polynomial kernel:

$$k(\boldsymbol{x}, \boldsymbol{y}) = (c + \boldsymbol{x}'\boldsymbol{y})^d.$$

  We can write down the expansion explicitly, so the mapping ϕ can be explicitly expressed.

- The Gaussian (radial basis function) kernel:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{1}{\sigma^2}\|\boldsymbol{x} - \boldsymbol{y}\|^2\right).$$

  The feature space where $\phi(\boldsymbol{x})$ lies is infinite dimensional. (The mapping ϕ is only implicitly defined)