Lecture 10: Support Vector Machine and Large Margin Classifier

Applied Multivariate Analysis

Math 570, Fall 2014

Xingye Qiao
Department of Mathematical Sciences
Binghamton University

E-mail: qiao@math.binghamton.edu

Outline

1 Support Vector Machine

2 Kernel methods

3 Large Margin Classifiers

The next section would be . . . . . .

1. Support Vector Machine
2. Kernel methods
3. Large Margin Classifiers

What's the Problem?

- Training Data: $\{(y_i, \boldsymbol{x}_i),\ i = 1 \cdots n,\ y_i \in \mathcal{Y},\ \boldsymbol{x}_i \in \mathcal{S} \subset \mathbb{R}^d\}$.
- 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

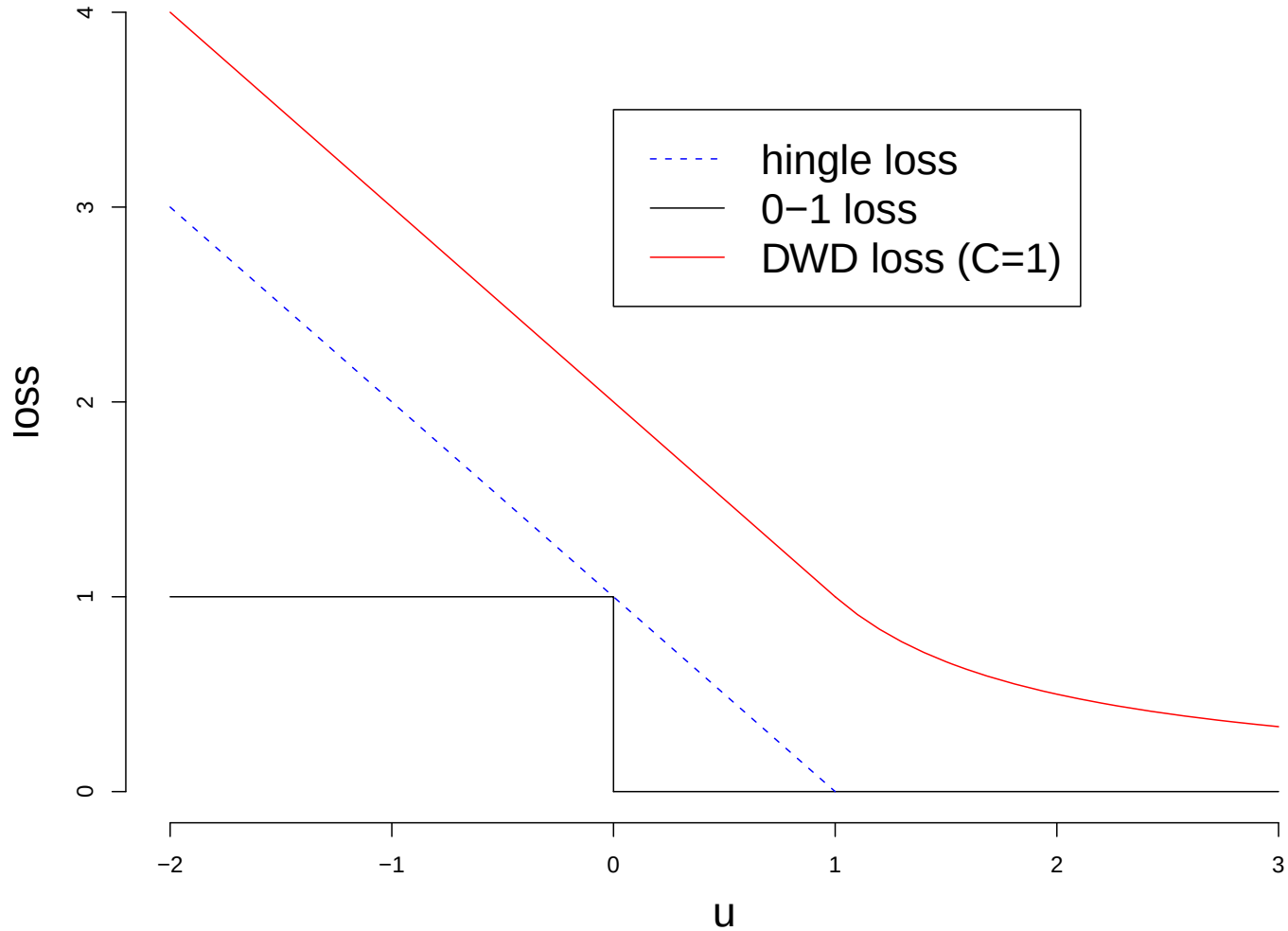
- $\mathcal{Y} = \{-1, +1\}$.
- Predict y_i as $\widehat{y}_i = \phi(\boldsymbol{x}_i)$, so that $\sum_i \mathbb{1}_{\{y_i \neq \widehat{y}_i\}}$ is as small as possible.

$0-1$ 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 $\phi(\boldsymbol{x}) = \text{sign}(f(\boldsymbol{x}))$.
  - $f(\boldsymbol{x}) > 0 \Rightarrow +1$ (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

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

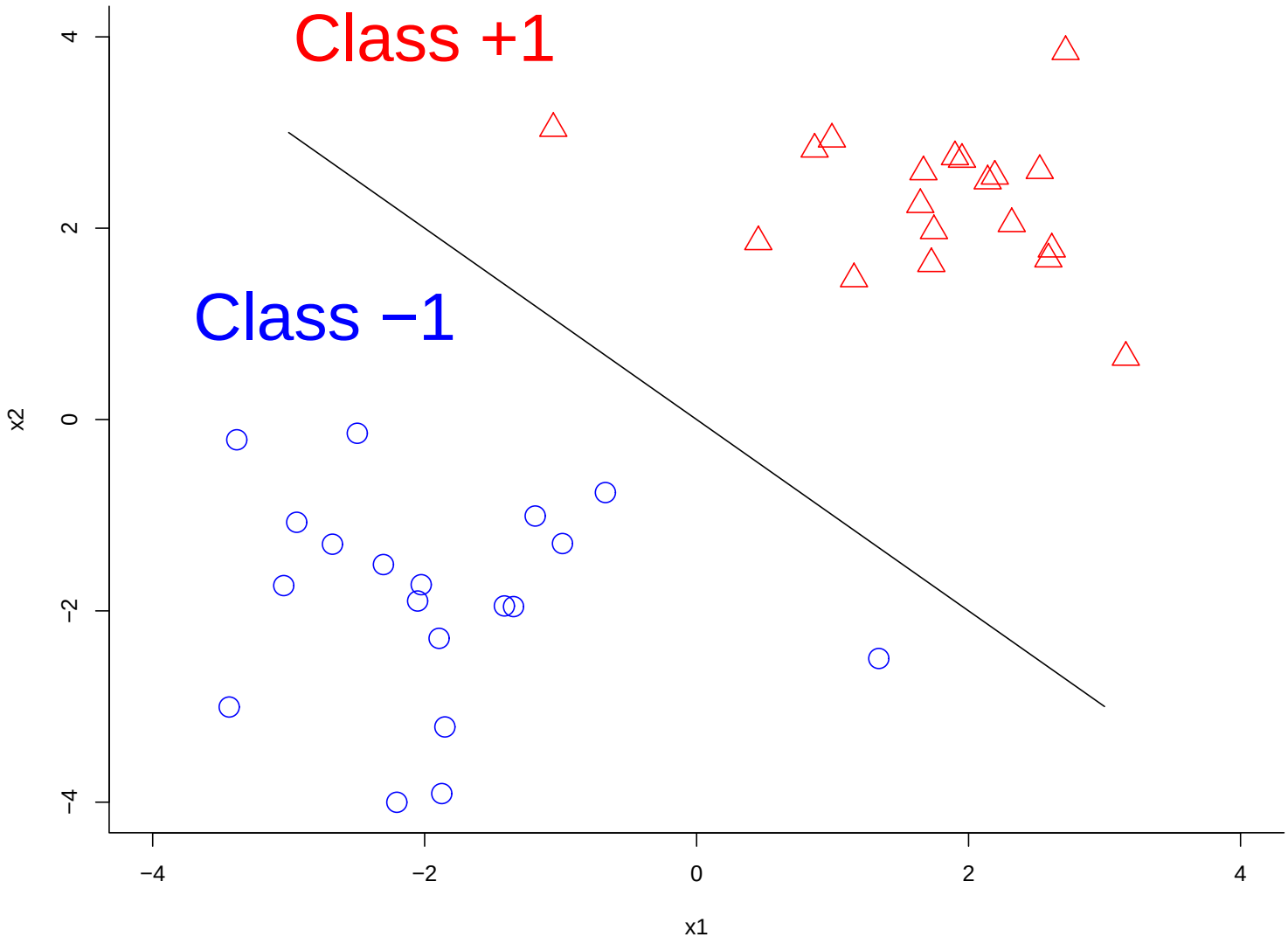
- $0-1$ loss function: $\mathbb{1}_{\{u \leq 0\}}$, where $u = y_i f(\boldsymbol{x}_i)$.
  - may not work: not convex !!!
- What function looks similar to $0-1$ loss, but is convex?
  - One answer: Hinge loss, $[1-u]_+$.


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

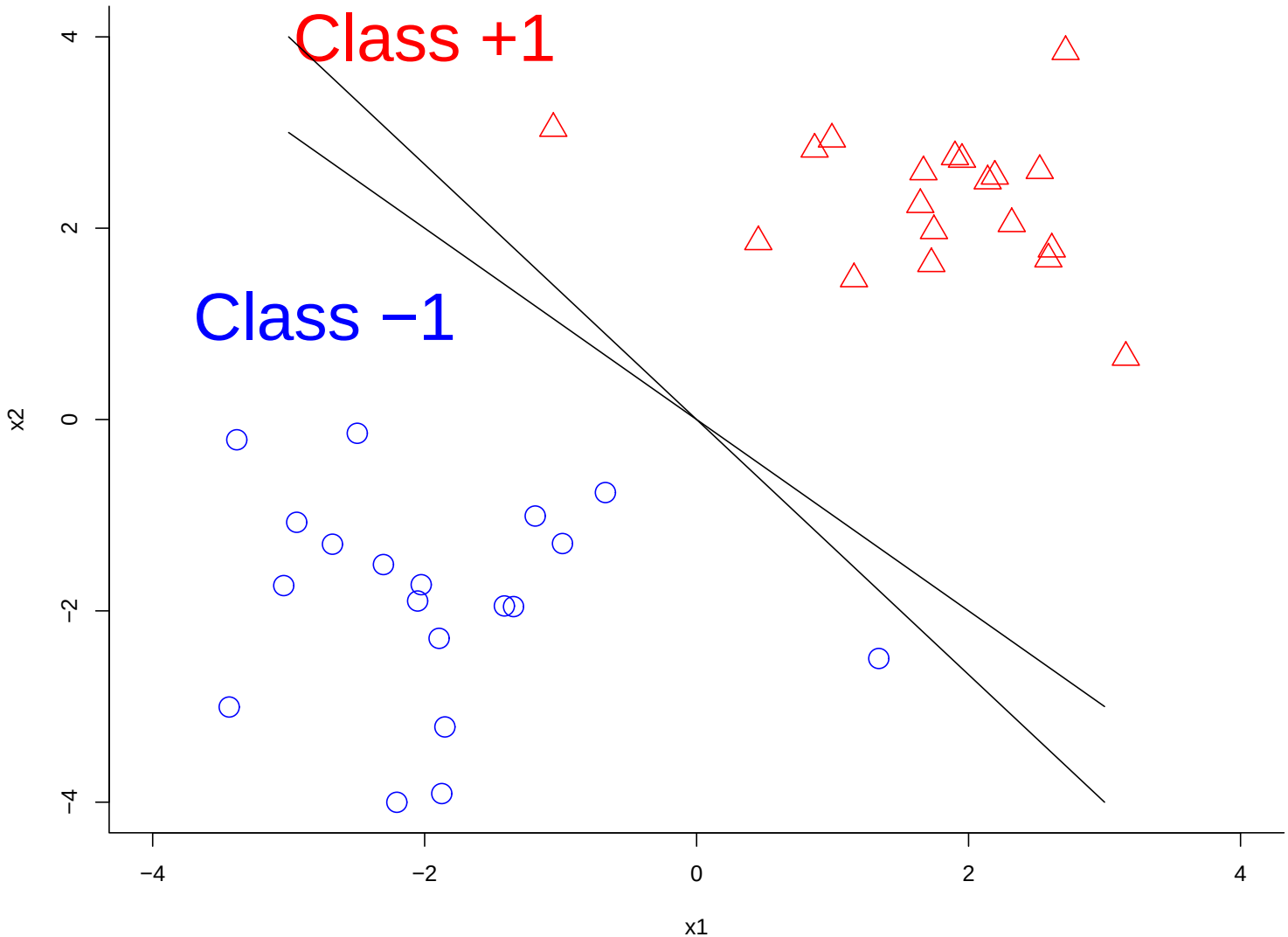
Linear Discrimination

- 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.

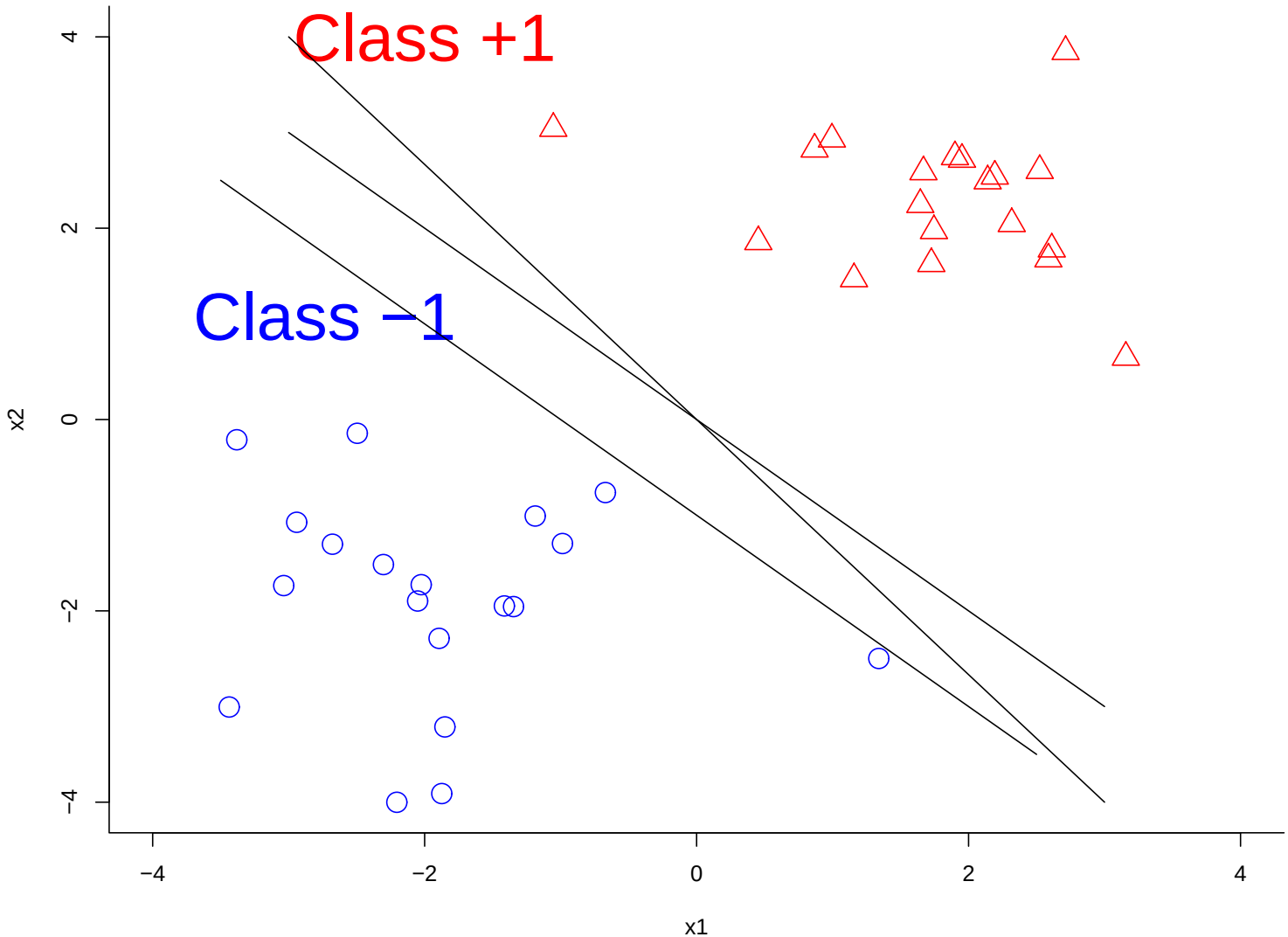
Separable Case



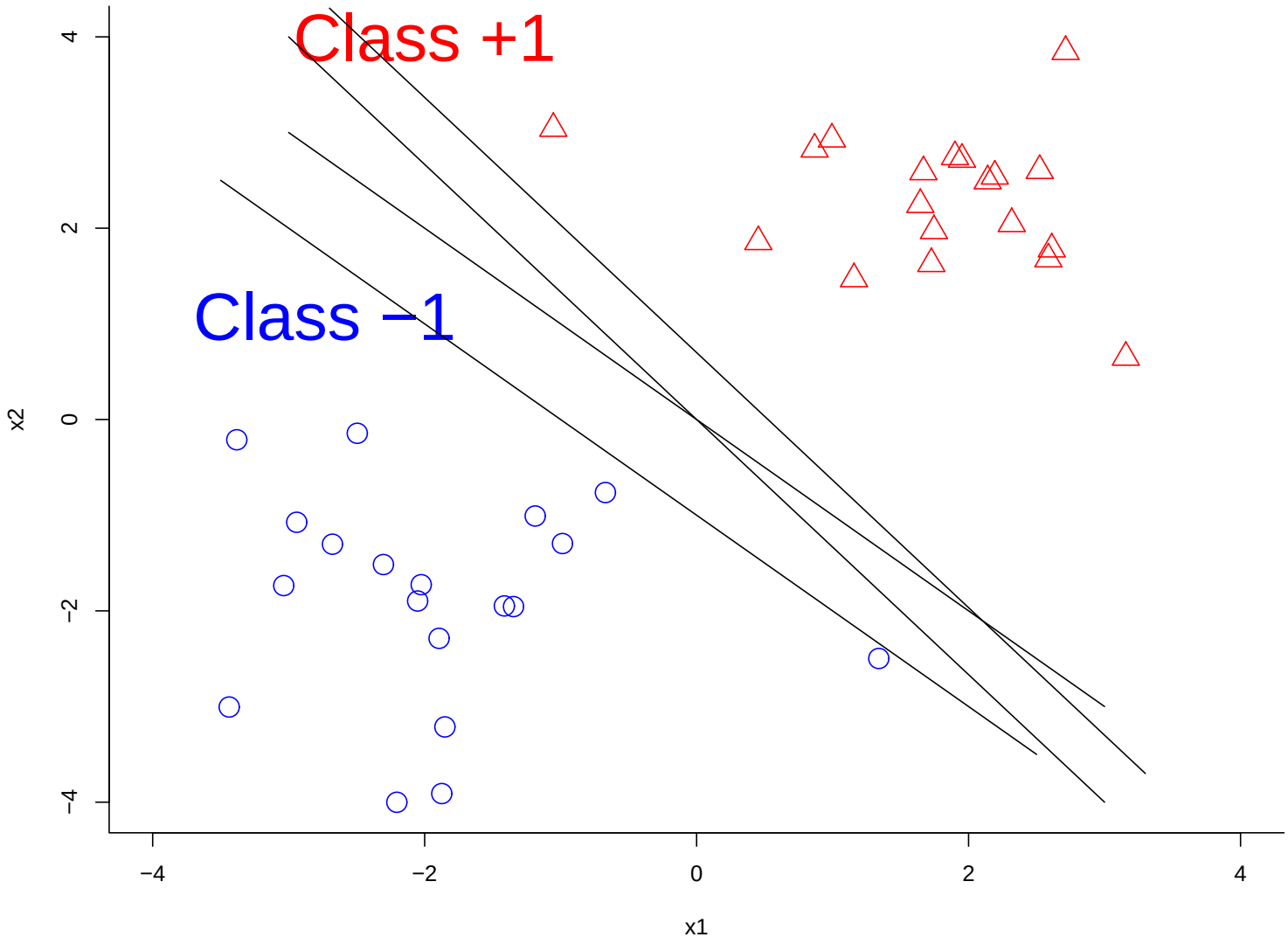
Separable Case



Separable Case



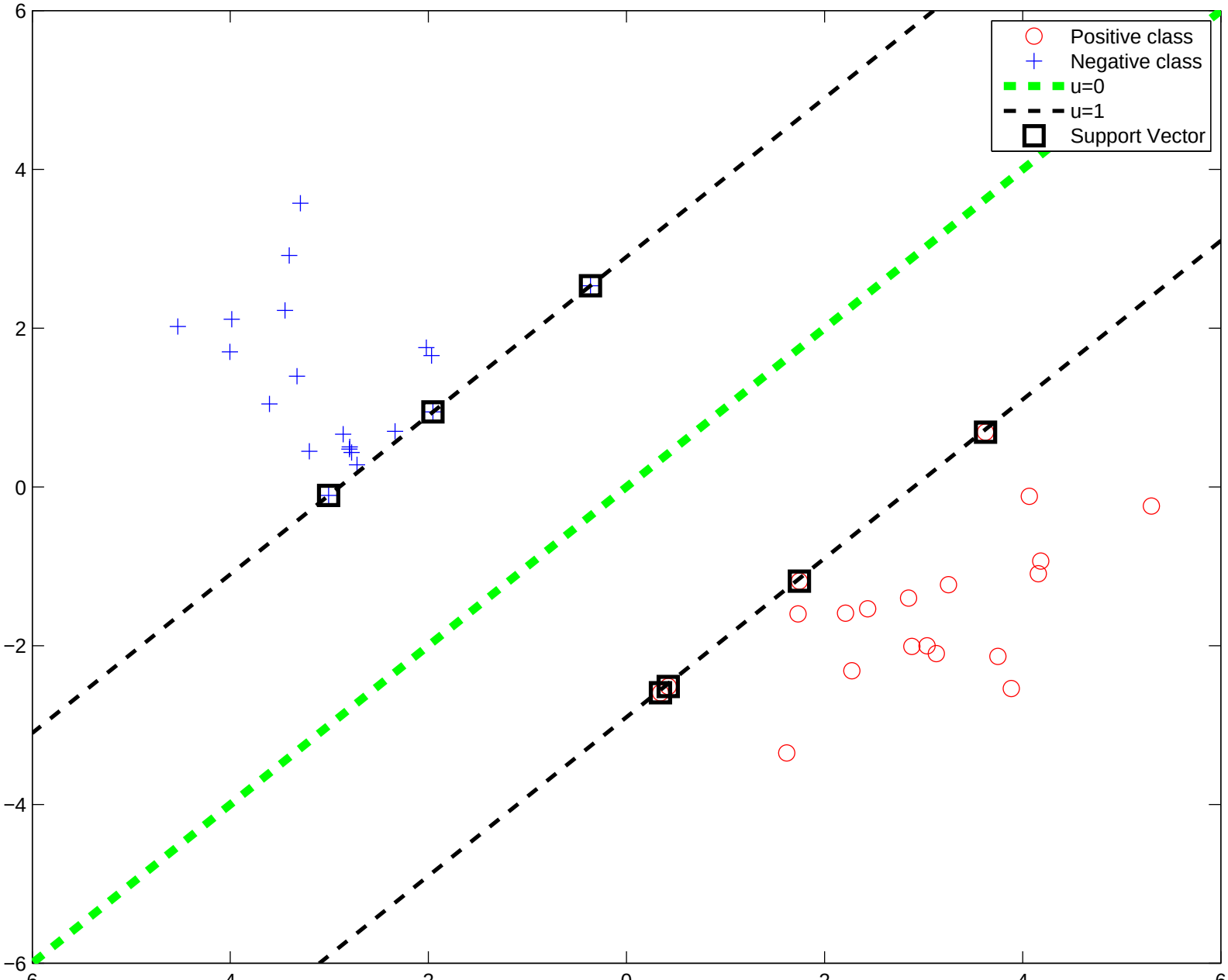
Separable Case



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



Formulation

- The margin will be $\frac{2|f(\boldsymbol{x}_{SV})|}{\|\boldsymbol{\omega}\|}$. We want to maximize it.
- By the definition of support vectors, also need to make sure that:

$$\forall i,\ y_i f(\boldsymbol{x}_i) \geq y_{SV} f(\boldsymbol{x}_{SV}) = |f(\boldsymbol{x}_{SV})|$$

- But, the norm of $\boldsymbol{\omega}$ can be arbitrary.
  - Rescale the norm of $\boldsymbol{\omega}$, so that $|f(\boldsymbol{x}_{SV})| = 1$.
- So we will try to

$$\max_{\boldsymbol{\omega},\beta} \frac{2}{\|\boldsymbol{\omega}\|},$$
$$\text{s.t. } y_i f(\boldsymbol{x}_i) \geq 1.$$

- Equivalently,

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

Non-separable Case

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

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

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

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

Remarks about SVM

- Solved by Quadratic Programming (QP) of the duality problem.
- 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.

Solution

- How to solve SVM?
- Take the first formulation for non-separable case as an example:

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

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

$$\xi_i \geq 0.$$

- We will derive the duality problem, which is a standard Quadratic Programming (QP) problem.
- Then use a standard QP package to solve it.

Duality of SVM

- Lagrangian:

$$L_S = \frac{\boldsymbol{\omega}'\boldsymbol{\omega}}{2} + \sum_i \{C\xi_i - \alpha_i[y_i(\boldsymbol{x}_i'\boldsymbol{\omega} + \beta) + \xi_i - 1] - \mu_i\xi_i\}.$$

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

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

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

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

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

Duality of SVM

- 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. Solvable. Very fast.
- 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 $\beta = -(\boldsymbol{x}_i'\boldsymbol{\omega}) - \xi_i/y_i + 1/y_i$.

The next section would be . . . . . .

1 Support Vector Machine

2 Kernel methods

3 Large Margin Classifiers

Nonlinear SVM

Why do we learn the dual solution?

- To implement the SVM
- To extend SVM for nonlinear classification by the *kernel trick*
  - In computation of linear SVM, the inner products $\boldsymbol{x}_i'\boldsymbol{x}_j = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$ (or the dot product), for $i, j = 1, \ldots, n$ are sufficient.
  - For nonlinear SVM, each input vector $\boldsymbol{x}_i$ is mapped to a higher-dimensional feature space:

    $$\boldsymbol{x}_i \mapsto \phi(\boldsymbol{x}_i) \in \mathbb{R}^Q,$$

    and only the inner products $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ are needed in computation. Here the map ϕ is induced by a kernel function so that $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
  - Overall, the kernel trick replaces $\boldsymbol{x}_i'\boldsymbol{x}_j$ with a kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$.

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)

Prediction function

- Recall in linear SVM:

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

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

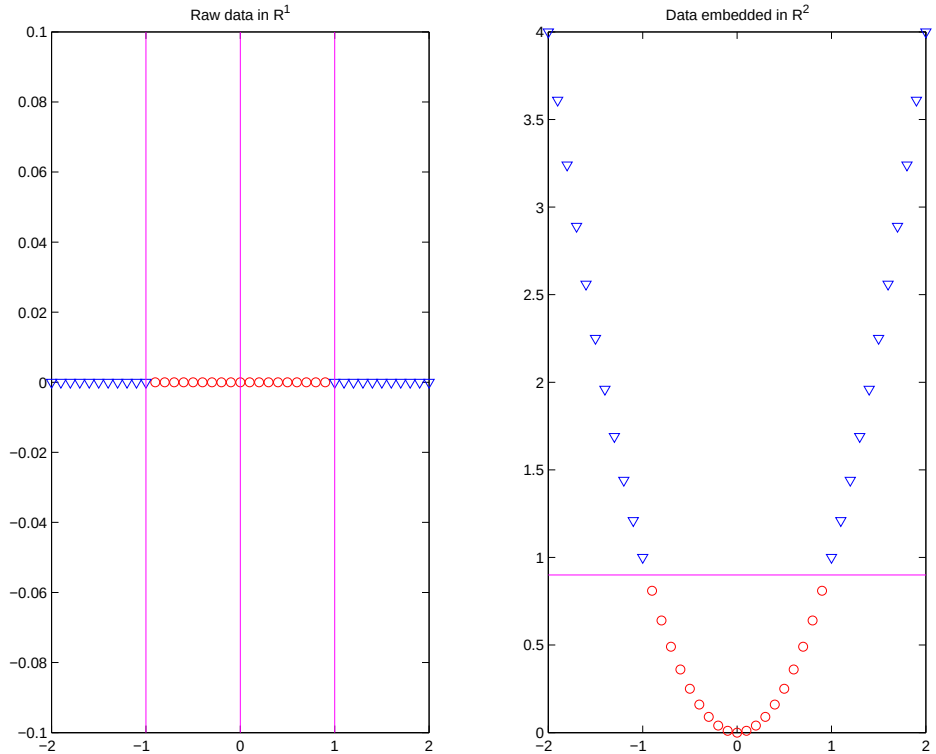
- In kernel SVM:

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

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

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. It is called the "kernel trick".
- Kernel SVM is indeed "linear" SVM conducted in a kernel space with training data $(\phi(\boldsymbol{x}_i), y_i)$
- Kernel trick is an application of a more general methods called embedding.
- Think a simple transformation: $\phi(x) : x \mapsto (x, x^2)^T$. Then instead of a linear SVM in the $\mathbb{R}$ space, do it in the $\mathbb{R}^2$ space, with $((x_i, x_i^2)^T, y_i)$ as training data.



- It was hopeless to find a 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 ϕ is much much more sophisticated and maps to richer feature space.

A better example

- Consider a less naive 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 $\phi(\boldsymbol{x})'\phi(\boldsymbol{y}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)(y_1^2, y_2^2, \sqrt{2}y_1y_2)' = (x_1y_1 + x_2y_2)^2 = (\boldsymbol{x}'\boldsymbol{y})^2$, which corresponds to the polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (c + \boldsymbol{x}'\boldsymbol{y})^d$ with $c = 0$ and $d = 2$
- When we conduct linear SVM in the new feature space, 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.

Kernel function

- The only condition for a kernel function $k(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ to be valid is that it is non-negative definite, that is for any $N \in \mathbb{N}$, $\boldsymbol{a} \in \mathbb{R}^N$ and $\boldsymbol{z}_i \in \mathbb{R}^d$ $(i = 1, \ldots, N)$, we have $\boldsymbol{a}'\mathbf{K}\boldsymbol{a} = \sum_i \sum_{i'} k(\boldsymbol{z}_i, \boldsymbol{z}_{i'}) a_i a_{i'} \geq 0$ where $\mathbf{K} := [k(\boldsymbol{z}_i, \boldsymbol{z}_j)]_{i,j=1,\ldots,N}$, that is, $\mathbf{K}$ is a non-negative definite matrix.
- In the case of linear kernel (ϕ is identity function) and polynomial kernel (such as the example from last page), $\phi(\boldsymbol{x})$ is a vector in $\mathbb{R}^Q$, where Q is finite.
- More generally, we should open our minds and view $\phi(\boldsymbol{x})$ itself as a function on the input space $\mathbb{R}^d$. To facilitate this view, it may be better to refer to $\phi(\boldsymbol{x})$ as $\phi_{\boldsymbol{x}}(\cdot)$ from now on.
  - In linear kernel case, for example, $\phi_{\boldsymbol{x}}(\boldsymbol{y}) = \boldsymbol{x}'\boldsymbol{y}$
  - In the example from last page, for example, $\phi_{\boldsymbol{x}}(\boldsymbol{y}) = (\boldsymbol{x}'\boldsymbol{y})^2$
- The next page formally discusses such a view.

Reproducing Kernel Hilbert Space

- Consider a Hilbert space $\mathcal{H}$ of real-valued functions defined on a domain $\mathcal{X}$, $\mathcal{H} = \{f | f : \mathcal{X} \mapsto \mathbb{R}\}$, equipped with an inner product $\langle f, g \rangle_{\mathcal{H}}$ for $f, g \in \mathcal{H}$.
- It is a reproducing kernel Hilbert space (RKHS) if there is a kernel function (called the reproducing kernel function) $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ such that
  1. $k(\boldsymbol{x}, \cdot) \in \mathcal{H}$ for each $\boldsymbol{x} \in \mathcal{X}$, and
  2. $\langle k(\boldsymbol{x}, \cdot), f \rangle_{\mathcal{H}} = f(\boldsymbol{x})$ for every $f \in \mathcal{H}$ and $\boldsymbol{x} \in \mathcal{X}$ (called *reproducing* property)
- If take f in the *reproducing* property above to be $k(\boldsymbol{y}, \cdot)$, then $\langle k(\boldsymbol{x}, \cdot), k(\boldsymbol{y}, \cdot) \rangle_{\mathcal{H}} = k(\boldsymbol{x}, \boldsymbol{y})$

Kernel SVM

- We are now at a position to understand kernel SVM better.
- Think $\phi(\boldsymbol{x}) : \boldsymbol{x} \mapsto \phi_{\boldsymbol{x}}(\cdot) = k(\boldsymbol{x}, \cdot)$ of that a point $\boldsymbol{x} \in \mathbb{R}^d$ is mapped to a function which sits in a space of many functions. *As a vector, $\phi(\boldsymbol{x})$ is now infinite-dimensional (because it is really a function now).*
- This space of functions (RKHS) has its own inner product, norm, and distance, etc. (all related to kernel function k).
- We pretend that we do not know that it is a space of functions and just conduct linear SVM in this functional space (RKHS).
- This causes no trouble because in the calculation, we only deal with $\tilde{\boldsymbol{x}}_i' \tilde{\boldsymbol{x}}_j$ in the new space (or more precisely $\langle \phi_{\boldsymbol{x}_i}(\cdot), \phi_{\boldsymbol{x}_j}(\cdot) \rangle_{\mathcal{H}}$, which equals $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$). So we do not need to touch the mysterious $\phi_{\boldsymbol{x}}(\cdot)$. Nor would that be any helpful.

Which kernel to use?

- No kernel is better than the others uniformly
- Personal favorite (other than linear kernel) is Gaussian RBF kernel
- Hyper-parameter tuning is also important. Use the overall scale as a benchmark. Try the 25%, 50% and 75% quantiles of pairwise distances.
- In low-dimensional data, if the separating boundary seems to be nonlinear, then kernel SVM usually works better (for prediction purpose.)
- However, for high-dimensional data ($d \gg n$), linear SVM is sufficient. Use the simpler linear SVM and avoid the complicated nonlinear SVM. The latter over-complicates the problem unnecessarily.
- Linear SVM gives better interpretations.
- Helpful reading: *A tutorial on support vector machines for pattern recognition* by Burges (Google it).

Software

```
library(kernlab)
  # train the linear SVM
svp <- ksvm(xtrain,ytrain,kernel='vanilladot')
  # Nonlinear SVM with Gaussian kernel
svp <- ksvm(xtrain,ytrain,kernel='rbfdot')
  # Nonlinear SVM with polynomial kernel
svp <- ksvm(xtrain,ytrain,kernel='polydot')
  # Predict labels on test
ypred = predict(svp,xtest)
table(ytest,ypred)
```

Another library is `e1071`, in addition to `kernlab`.

The next section would be . . . . . .

1 Support Vector Machine

2 Kernel methods

3 Large Margin Classifiers

Large margin

- SVM encourages large distances from data vectors to the separating hyperplane.
- For each data vector, the signed distance is $u_i := y_i(\boldsymbol{x}_i\boldsymbol{\omega} + \beta)$ (functional margin) [confine to $K = 2$ case and $y_i = \pm 1$]
  - If $\|\boldsymbol{\omega}\| = 1$, then $|\boldsymbol{x}_i\boldsymbol{\omega} + \beta|$ is the exact distance.
- Member of a family of classifiers called large-margin based classifier. All encourage large functional margin u_i's one way or another.
- SVM only chooses a subset of the training data (the support vectors) and encourages $u_i = y_i f(\boldsymbol{x}_i)$ of them to be large. Recall that the solution $\widehat{\boldsymbol{\omega}}$ depends on the support vectors only.
  - This leads to issue with high-dimensional. Vulnerable to overfitting. Less stability.

DWD

- Distance-weighted Discrimination (Marron, Todd and Ahn 2007), instead, minimizes

$$\sum_{i=1}^{n} \left\{ (u_i + \xi_i)^{-1} + C\xi_i \right\}$$

  with $\xi_i \geq 0$ and $u_i + \xi_i \geq 0$
- All the data points contribute!
- Small inverse distance $\Leftrightarrow$ large distance

Loss functions

- SVM: Hinge loss

$$L(u) = [1-u]_+ = \begin{cases} 1-u, & \text{if } u < 1 \\ 0, & \text{otherwise} \end{cases}$$

- DWD: DWD loss (in the case of $C = 1$)

$$L(u) = \begin{cases} 2-u, & \text{if } u < 1 \\ 1/u, & \text{otherwise} \end{cases}$$

FLAME

Flexible Assortment Machine (Qiao and Zhang 2014): a trade-off between overfitting and imbalanced data

$$L(u) = \begin{cases} (2-\nu) - u, & \text{if } u < 1 \\ (1/u - \nu)_+, & \text{otherwise} \end{cases}$$

Logistic regression

Recall that for $\tilde{y} = 0, 1$

$$\ell = \sum_{i=1}^{n} \left\{ \tilde{y}_i \log \left(\frac{\exp(f(\boldsymbol{x}_i))}{1 + \exp(f(\boldsymbol{x}_i))} \right) + (1 - \tilde{y}_i) \log \left[\frac{1}{1 + \exp(f(\boldsymbol{x}_i))} \right] \right\}$$

Let $y = 2 * \tilde{y} - 1$ or $\tilde{y} = \frac{1}{2}(y + 1)$

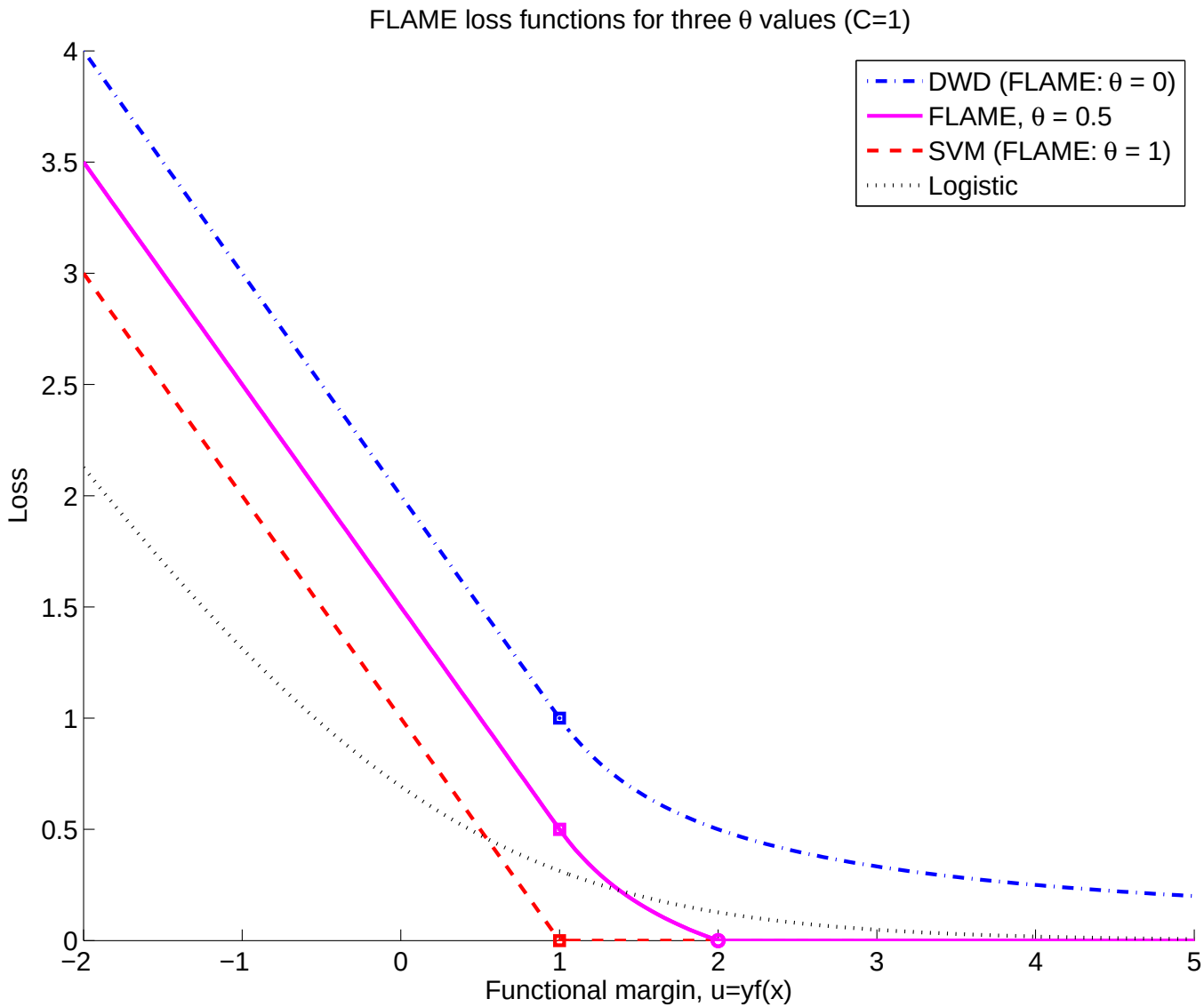
$$\begin{aligned} \ell(\beta, \boldsymbol{\omega}) = \sum_{i=1}^{n} \Bigg\{ & \frac{1}{2}(y_i + 1) \log \left(\frac{1}{\exp(-f(\boldsymbol{x}_i)) + 1} \right) \\ & + \frac{1}{2}(1 - y_i) \log \left[\frac{1}{1 + \exp(f(\boldsymbol{x}_i))} \right] \Bigg\} \\ = & \sum_{i=1}^{n} \log \left(\frac{1}{\exp(-y_i f(\boldsymbol{x}_i)) + 1} \right) \end{aligned}$$

Logistic Loss

So the loss function corresponding to logistic regression can be written as

$$L(u) = -\log\left(\frac{1}{\exp(-u)+1}\right) = \log\left(\exp(-u)+1\right)$$

Different loss functions



Large-margin based loss

- All loss are non-increasing functions of the functional margin $yf(\boldsymbol{x})$, encouraging small loss and large $yf(\boldsymbol{x})$
- $L(-a) > L(a)$ for all $a > 0$; $L'(0) < 0$
- $\Uparrow$ important for theoretical concern (Fisher consistency)
- The hockey stick shape of the Hinge loss causes data-piling issue. FLAME delays this issue. DWD and Logistic regression do not have such an issue.

Liu, Y., Zhang, H. H., and Wu, Y. (2011). Soft or hard classification? Large margin unified machines. Journal of the American Statistical Association, 106, 166-177.
A unified family of loss which encompasses most of these, linking soft and hard classification.