ST720 Data Science

Binary Classification

Seung Jun Shin (sjshin@krea.ac.kr)

Department of Statistics, Korea University

Machine Learning

- Machine learning refers the class of methods (or algorithms) that uncovers informative and structured signals burried in the data, often with large scales.

Figure 1: Find needles in a haystack.

Machine Learning

- This is, in fact, what statistican have always been doing for data analysis.
- Traditional statsticians focus more on inference based on the model to understand the (random) data generating process.
- Modren applications focus more on prediction of the outcome, without much understanding about the data generating process (non-stochastic & numerical algorithm becomes more popular).

Supervised vs Unsupervised Learning

- Supervised learning seeks the signals of the relation between

  response y_i vs predictor $\mathbf{x}_i = (x_{i1}, \cdots, x_{ip})^T, \quad i = 1, \cdots, n.$

- Usupervised learning seeks the signlas of

  the intra-relation in $\mathbf{x}_1, \cdots, \mathbf{x}_n$.

  - Clustering: signals between observations.
  - Feature Extraction: signals between variables.

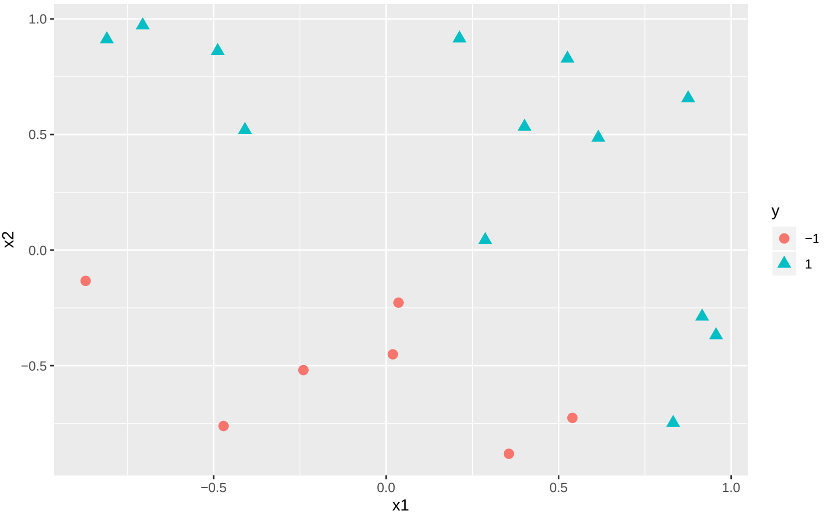
    (ex, PCA, Graphical Model)

Regression vs. Classification

- Supervised learning can be classified into two types:
  - Regression with quantatative/numerical responses.
  - Classification with qualitative/categorical (often binary) responses.
- In machine learning aplications, binary classsification is much more popular.

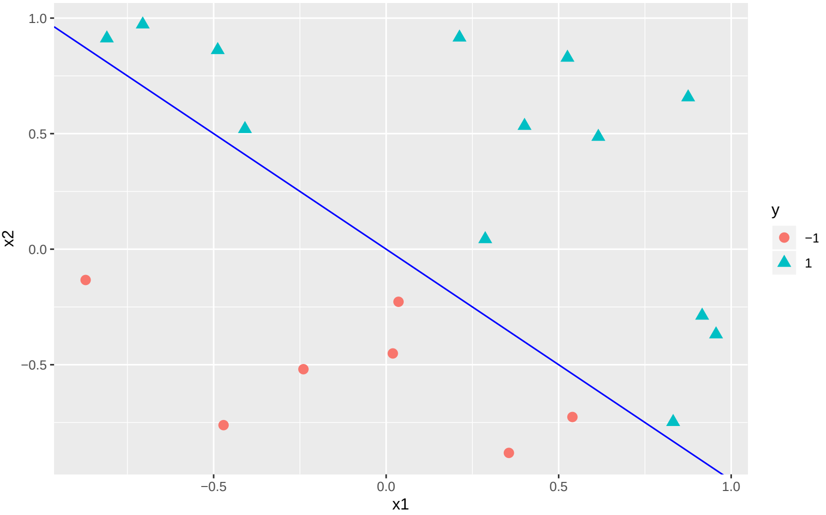
Toy Example

- Consider a simple (linearly separable) example.



Toy Example

- A simple and obvious solution!



Hard Classification

- The line is called the classification/decision boundary.

$$f(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T \mathbf{x} = 0$$

- Prediction of Y given $\mathbf{X} = \mathbf{x}$ is

$$\hat{Y} = \text{sign}\{f(\mathbf{x})\}$$

- In this lecture, we assume linear classification function, unless stated otherwise.

Perceptron

- Proposed by Rosenblatt (1956)
- Works when linearly separable.

Given a linearly separble training set S and learning rate $\eta \in \mathbb{R}^+$

$\mathbf{w}_0 \to 0$; $b_0 \to 0$; $k < -0$

$R \to \max_i \|\mathbf{x}_i\|$

repeat

For $i = 1, \cdots, n$

if $y_i(\beta_1^T \mathbf{x}_i) + \beta_0 \leq 0$ then

$$\beta^{(t+1)} = \beta^{(t)} + \eta y_i \mathbf{x}_i$$

$$\beta_0^{t+1} = \beta_0^{(t)} + \eta y_i R^2$$

$$t = t + 1$$

end if

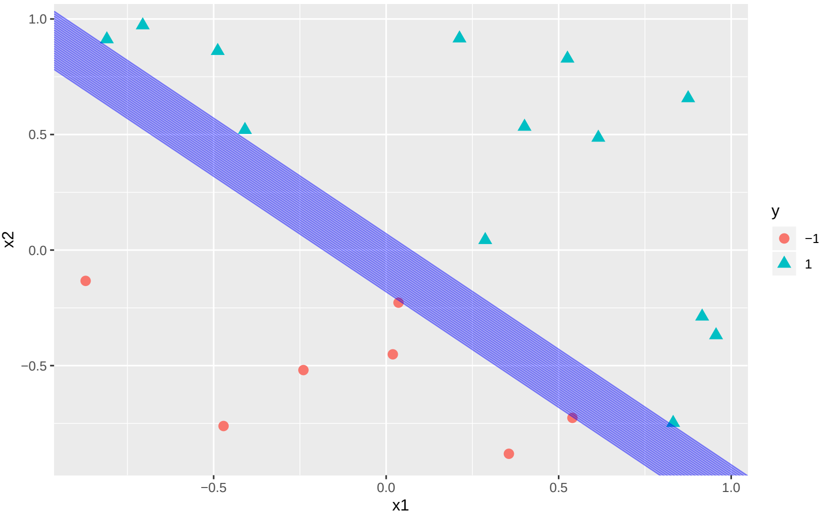
end for

until no mistakes made within the for loop.

retrun $(\beta_0^{(t)}, \beta_0^{(t)})$.

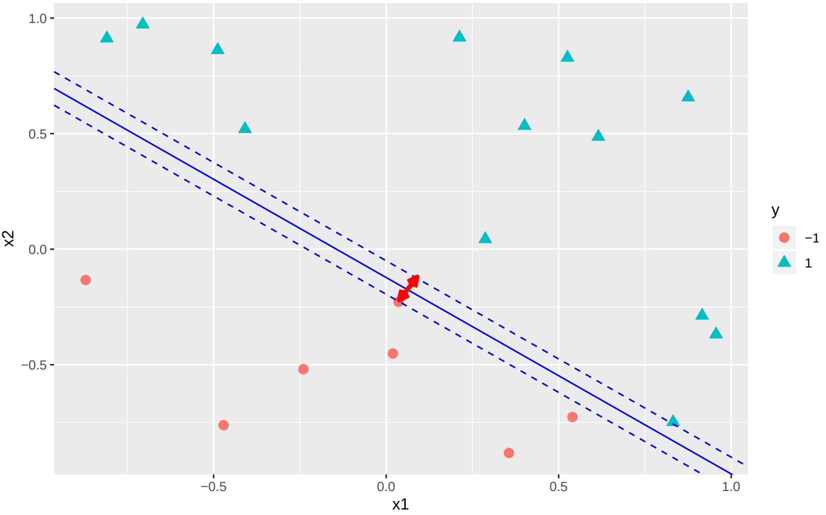
Perceptron

- Solution (i.e., separting hyperplane) may not be unique.



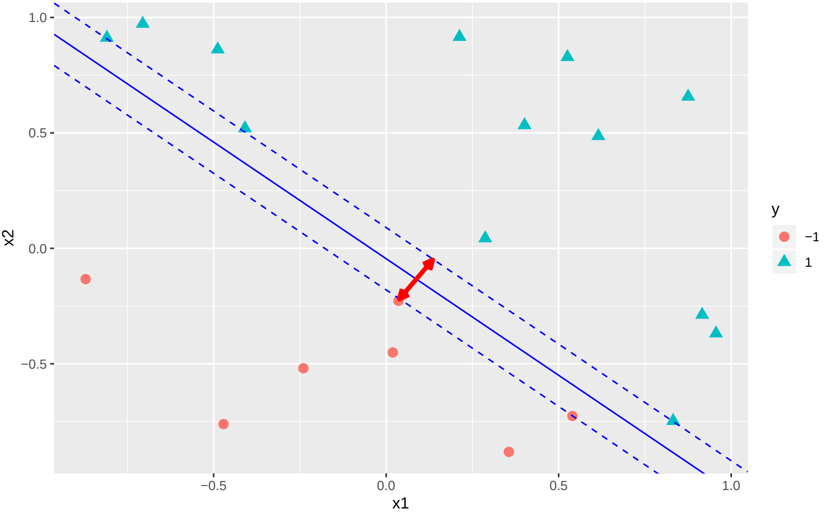
Optimal Separating Hyperplane

- One example.



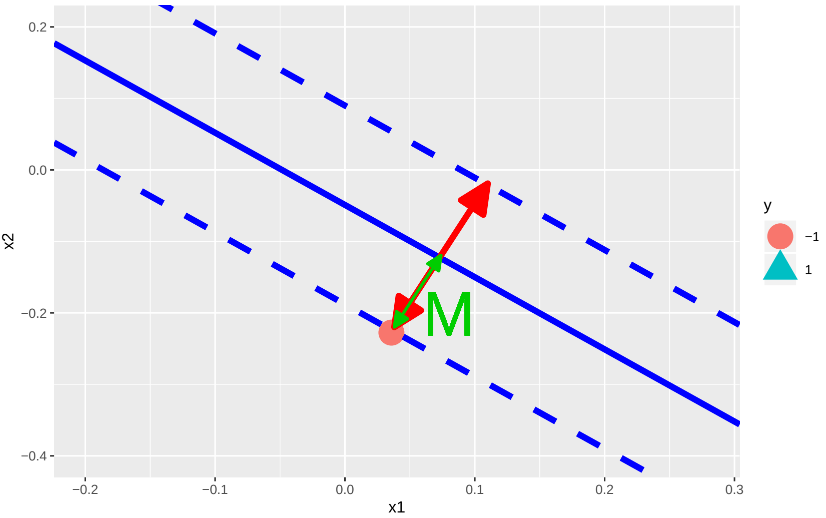
Optimal Separating Hyperplane

- better?



Optimal Separating Hyperplane

- Optimal Separating Hyperplane maximizes Gemetric Margin, M.



Geometric Margin

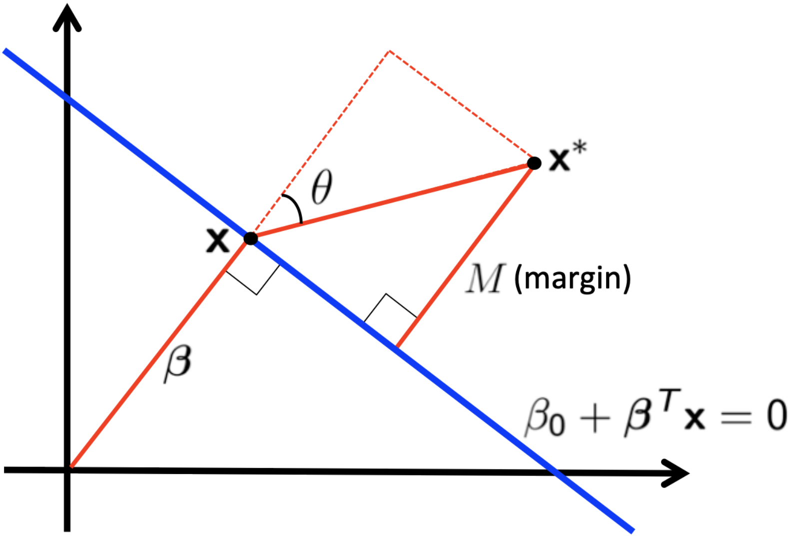


Figure 2: Geometric Margin

Geometric Margin

Let $\mathbf{x}^*$ be the closest point to $\beta_0 + \boldsymbol{\beta}^T\mathbf{x} = 0$, then

$$\cos(\theta) = \frac{\langle \mathbf{x}^* - \mathbf{x}, \boldsymbol{\beta} \rangle}{\|\mathbf{x}^* - \mathbf{x}\| \|\boldsymbol{\beta}\|}$$

Assuming $\|\boldsymbol{\beta}\| = 1$ WLOG,

$$\begin{aligned} M &= \cos(\theta)\|\mathbf{x}^* - \mathbf{x}\| \\ &= \beta_0 + \boldsymbol{\beta}^T\mathbf{x}^* (\text{ } \mathbf{x}^* \text{ is on right}) \end{aligned}$$

Geometric Margin of $\mathbf{x}^*$ is

$$M = \begin{cases} \beta_0 + \boldsymbol{\beta}^T\mathbf{x}^* \text{ ,} & y^* = \text{ } 1 \\ -(\beta_0 + \boldsymbol{\beta}^T\mathbf{x}^*), & y^* = -1 \end{cases}$$

Support Vector Machine

- In linearly separable case, SVM solves

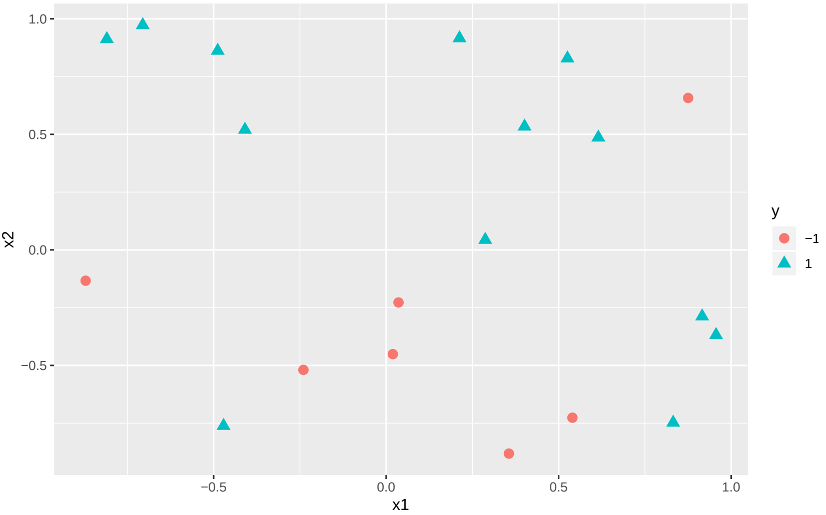
$$\begin{aligned} &\max_{\beta_0, \boldsymbol{\beta}} M \\ \text{subject to} \quad & y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i) \geq M, i = 1, \cdots, n; \\ & \|\boldsymbol{\beta}\| = 1 \end{aligned}$$

SVM (Maximal Margin Classifer)

- Let $\|\boldsymbol{\beta}\| = 1/M$, SVM is

$$\begin{aligned} &\min_{\beta_0, \boldsymbol{\beta}} \boldsymbol{\beta}^T \boldsymbol{\beta} \\ \text{subject to} \quad & y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i) \geq 1, \quad i = 1, \cdots, n. \end{aligned}$$

Binary Classification: Linearly Non-separable 1



Support Vector Machine (non-separable case)

- No solution that satisfies

$$y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i) \geq M \text{ and } \|\boldsymbol{\beta}\| = 1 \qquad \Leftrightarrow \qquad y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i) \geq 1$$

- Let's relax the constraints and add penalty C for the violations.

SVM (Soft Margin Classifier)

Introducing slack variables $\xi_i \geq 0$, SVM solves

$$\begin{aligned} &\min_{\beta_0, \boldsymbol{\beta}, \xi_i} \boldsymbol{\beta}^T \boldsymbol{\beta} + C \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i) \geq 1 - \xi_i, \quad i = 1, \cdots, n \\ & \xi_i \geq 0, \qquad\qquad\qquad\qquad i = 1, \cdots, n. \end{aligned}$$

Computation of SVM

- Lagrangian primal function of the linear SVM is

$$\boldsymbol{\beta}^T\boldsymbol{\beta} + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\{1 - y_i(\beta_0 - \boldsymbol{\beta}^T\mathbf{x}_i) - \xi_i\} - \gamma_i\sum_{i=1}^{n}\xi_i \quad (1)$$

- Taking derivative w.r.t primal variables $\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}$:

$$\frac{\partial}{\partial\boldsymbol{\beta}}L_p: \quad \boldsymbol{\beta} = \sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i \quad (2)$$

$$\frac{\partial}{\partial\beta_0}L_p: \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \quad (3)$$

$$\frac{\partial}{\partial\xi_i}L_p: \quad \alpha_i = C - \gamma_i \quad (4)$$

- KKT complementary conditions:

$$\begin{aligned}\alpha_i\{1 - y_i(\beta_0 + \boldsymbol{\beta}^T\mathbf{x}_i) - \xi_i\} &= 0\\ \gamma_i\xi_i &= 0\end{aligned}$$

Computation of SVM

- Plugging (2)– (4) into (1), dual problem is the following QP:

$$\max_{\alpha_1,\cdots,\alpha_n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

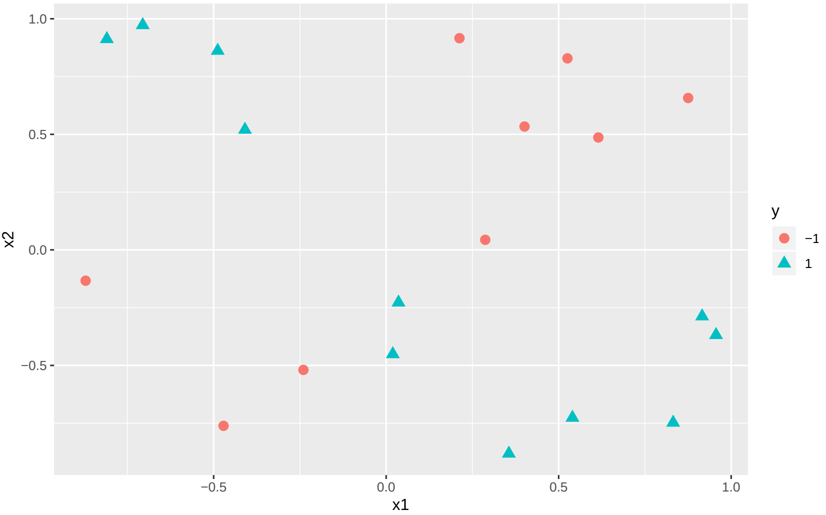
$$\text{subject to } 0 \le \alpha_i \le C, \quad i = 1, \cdots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

Computation of SVM

- By KKT conditions, we must have for all $k \in \{i : 0 < \alpha_i < 1\}$ (a.k.a Support Vectors)
$$1 - y_k(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_k) = 0$$
- The intercept is computed by
$$\beta_0 = y_i - \frac{1}{\lambda} \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_k$$
for any support vector $\mathbf{x}_k$.

Binary Classification: Linearly Non-separable 2



Kernel Trick

- Linear learning is often limited and more flexible learning methods are required.
- A common strategy is changing the representation of the data (transformation).
- Ex) Newton's law of gravitation:

$$f(m_1, m_2, r) = C\frac{m_1 m_2}{r}$$

- Linear machine cannot learning this law.
- However, a simple change of corrdinates

$$(m_1, m_2, r) \mapsto \mathbf{x} = (x_1, x_2, x_3) = (\ln m_1, \ln m_2, \ln r)$$

gives

$$g(\mathbf{x}) = c + x_1 + x_2 - 2x_3$$

Kernel Trick

- Need to select a set of non-linear features, and then learn the linear SVM on feature space.
- We now assume

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_i(\mathbf{x}) + b$$

where $\phi : X \mapsto F$ is a nonlinear map from the input sapce to a feature space.

- In linear SVM, we have

$$f(\mathbf{x}) = \beta_0 + \beta^T \mathbf{x} = \beta_0 + \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$

where $\langle \mathbf{x}, \mathbf{x}' \rangle = \mathbf{x}^T \mathbf{x}'$ denotes inner product of the input space.

Kernel trick

- The decision function exploits the inputs through their inner products.
- On the feature space, the decision function is given by

$$f(\mathbf{x}) = b + \sum_{i=1}^{n} \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle$$

- We define a function K as kernel iff

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

  where ϕ is a feature.

- Suppose $K(\mathbf{x}, \mathbf{x}')$ is a symmetric function on $\mathbf{x}$. K is a kernel function iff

$$\mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{ij} \in \mathbb{R}^{n \times n}$$

  is positive semi-definite matrix.

Kernel trick

- A kernel K defined on the input space uniquely determines the corresponding space:

$$\mathcal{H}_K = \left\{ \sum_{i=1}^{n} \theta_i K(\mathbf{x}_i, \mathbf{x}), \theta_i \in \mathbb{R}, \mathbf{x}_i \in \mathcal{X} \right\}$$

which we call the reproducing kenrel Hilbert space (RKHS).

Kernel SVM

- Let's get back to the SVM:

$$f(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T\mathbf{x} \qquad \Rightarrow \qquad f(\mathbf{x}) = \beta_0 + \sum_{i=1}^{n} \theta_i K(\mathbf{x}_i, \mathbf{x})$$

Kernel SVM

Introducing slack variables $\xi_i \geq 0$, SVM solves

$$\min_{\beta_0, \boldsymbol{\theta}, \xi_i} \boldsymbol{\theta}^T \mathbf{K} \boldsymbol{\theta} + C \sum_{i=1}^{n} \xi_i$$

$$\text{subject to} \quad y_i \left(\beta_0 + \sum_{j=1}^{n} \theta_i K(\mathbf{x}_j, \mathbf{x}_i) \right) \geq 1 - \xi_i, \quad i = 1, \cdots, n$$

$$\xi_i \geq 0, \qquad i = 1, \cdots, n.$$

Computation of Kernel SVM

- Notice that $\theta = y_i\alpha_i,\ i = 1, \cdots, n$.
- Kernel SVM solves

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha} \odot \mathbf{y})^T \mathbf{K}(\boldsymbol{\alpha} \odot \mathbf{y})$$

$$\text{subject to } \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}$$

$$\boldsymbol{\alpha}^T \mathbf{y} = 0.$$

- The decision function is

$$f(\mathbf{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})$$

Kernel Function

- Popular choice of the kernel includes:
  - Linear: $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$
  - Polynomial: $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T\mathbf{x}')$
  - Radial (Gaussian): $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|)$

Example: Spam data

- Road Libraries and Data

```r
library(e1071)
library(kernlab)
data(spam)

str(spam)
```

```
## 'data.frame':    4601 obs. of  58 variables:
##  $ make              : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ..
##  $ address           : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ all               : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77
##  $ num3d             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ our               : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92
##  $ over              : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
##  $ remove            : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.
##  $ internet          : num  0 0.07 0.12 0.63 0.63 1.85 0 1.88
##  $ order             : num  0 0 0.64 0.31 0.31 0 0 0 0.92 0.06
##  $ mail              : num  0 0.94 0.25 0.63 0.63 0 0.64 0 0.7
##  $ receive           : num  0 0.21 0.38 0.31 0.31 0 0.96 0 0.7
##  $ will              : num  0.64 0.79 0.45 0.31 0.31 0 1.28 0
```

Example: Spam data

- Test vs Training

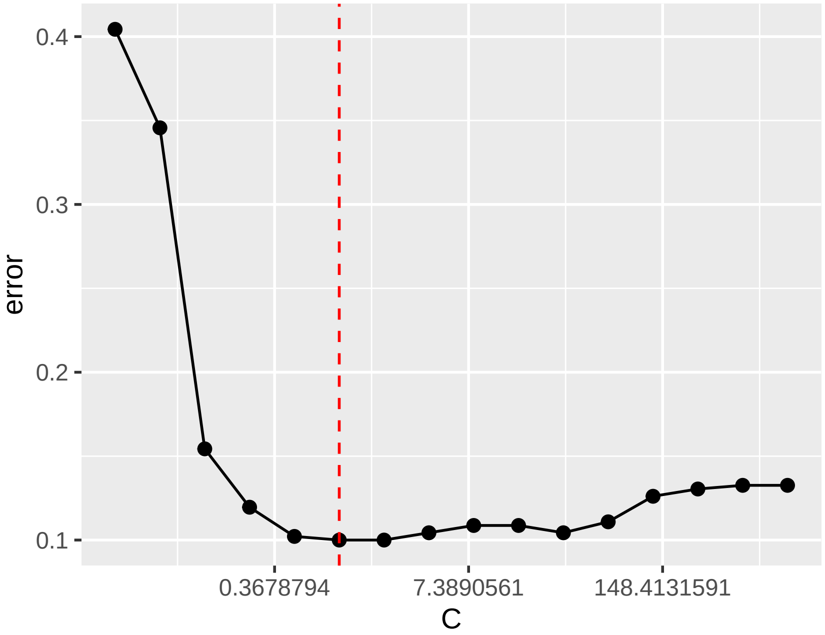
```r
set.seed(2)
index <- sample(n)
spamtrain <- spam[index[1:floor(n/10)], ]
spamtest <- spam[index[-(1:floor(n/10))], ]
```

- Tuning

```r
gamma <- 1/2*(median(dist(spamtrain[,-p])))^2
C.grid <- (2)^(-5:10)
tune.obj <- tune(svm, type~. , data = spamtrain,
                 type = "C-classification",
                 kernel = "radial", scaled = F,
                 ranges = list(cost = C.grid),
                 cross = 5)

best.C <- tune.obj$best.model$cost
```

Example: Spam data



Example: Spam data

- Final Model

```r
model <- svm(type~. , data = spamtrain,
             type = "C-classification",
             cost = best.C, scaled = F)

test.hat.y <- predict(model, spamtest[,-p])
test.table <- table(test.hat.y, spamtest[,p])
print(test.table)
```

```
##
## test.hat.y nonspam spam
##    nonspam    2396  279
##    spam        118 1348
```

```r
test.err <- mean(spamtest[,p] != test.hat.y)
cat("test.error  =", round(test.err, 4), "\n")
```

```
## test.error  = 0.0959
```

Statistical Learning

- In statical learning, data are regarded as (realizations of) random variables.
  - Handling Non-separable case is now natural.
- Fundamental target is the data generating process or simply the distribution of $(Y, \mathbf{X})$.

$$(Y, \mathbf{X}) \sim P(y, \mathbf{x}) = P(\mathbf{x}) \times P(y \mid \mathbf{x})$$

  - Regression with quantatative/numerical responses:

$$P(Y \leq y \mid \mathbf{X} = \mathbf{x}), \qquad -\infty < y < \infty$$

  - Classification with qualitative/categorical (often binary) responses.

$$P(Y = k \mid \mathbf{X} = \mathbf{x}), \qquad k \in \{1, \cdots, K\}$$

Regression

- Direct estimation of $P(y \mid \mathbf{x})$ is often difficult.
- We often focus on summary of the distribution of $Y \mid \mathbf{X} = \mathbf{x}$.
- Conditional expectation of Y given $\mathbf{X} = \mathbf{x}$ is a natrual choice.

$$f(\mathbf{x}) = E(Y \mid \mathbf{X} = \mathbf{x})$$

- We call $f(\mathbf{x})$ regression function, its estimation is of primal interest in regression.

Linear Regression

- Linear regression tackles

$$f(\mathbf{x}) = E(Y \mid \mathbf{X} = \mathbf{x})$$

- Given $(y_i, \mathbf{x}_i) \in \mathbb{R} \times \mathbb{R}^p$, $i = 1, \cdots, n$, linear regression assumes

$$y_i = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i + \varepsilon_i, \qquad \epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$$

  where $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_p)^T$.

- Estimates can be obtained by solving

$$(\hat{\beta}_0, \hat{\boldsymbol{\beta}})^T = \min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - \boldsymbol{\beta}^T \mathbf{x}_i)^2.$$

  (Least Square Estimation / Maxmum Likelihood Estimation)

Soft Classification

- In the binary classification, the fundamental target is

$$p(\mathbf{x}) = P(Y = 1 \mid \mathbf{X} = \mathbf{x}) = 1 - P(Y \neq 1 \mid \mathbf{X} = \mathbf{x})$$

  which we call $p(\mathbf{x})$ class probability,
- $p(\mathbf{x})$ contains a complete information for the classification.
- $p(\mathbf{x})$ provides the uncertainty of the prediction.
- Soft classification seeks $p(\mathbf{x})$, not $f(\mathbf{x}) = 0$ a target of hard classification.

Bayes Classification Rule

- Misclassification Probability is

$$P\{\hat{Y} \neq Y \mid \mathbf{X} = \mathbf{x}\} = P\{f(\mathbf{x})Y < 0 \mid \mathbf{X} = \mathbf{x}\}$$

- Bayes classification boundary is the theorical opitmal that minimizes the true misclassification probability:

$$f^*(\mathbf{x}) = \operatorname*{argmin}_{f(\mathbf{x}) \in \mathbb{R}} \; E\big\{\mathbb{1}\{Yf(\mathbf{x}) < 0\}\big],$$

- This yields

$$\text{sign}\{f^*(\mathbf{x})\} = \text{sign}\{p(\mathbf{x}) - 0.5\}$$

- This reveals the theoretical connection between $p(\mathbf{x})$ and $f(\mathbf{x})$.

Naive Appraches 1: Naive Bayes Classifier

- Bayes Theorem states

$$P(Y = 1 \mid \mathbf{X} = \mathbf{x})$$
$$= \frac{P(\mathbf{X} = \mathbf{x} \mid Y = 1)P(Y = 1)}{P(\mathbf{X} = \mathbf{x} \mid Y = 1)P(Y = 1) + P(\mathbf{X} = \mathbf{x} \mid Y = -1)P(Y = -1)}$$

- $P(Y = 1)$: Sample proportion of the positive class.
- Indpendence assumption of $\mathbf{X} = (X_1, \cdots, X_p)^T$ implies

$$P(\mathbf{X} = \mathbf{x} \mid Y = 1) = P(X_1 = x_1 \mid Y = 1) \times P(X_p = x_p \mid Y = 1)$$

each of which on the RHS can be estimated by the corresponding sample proportions.

Naive Appraches 1: Naive Bayes Classifier

```
library(e1071)
data(iris) # iris data
obj <- naiveBayes(Species ~ ., data = iris) # fit NB clssifier
fitted <- predict(obj, iris[,-5] )
print(table(fitted, iris[,5]))
```

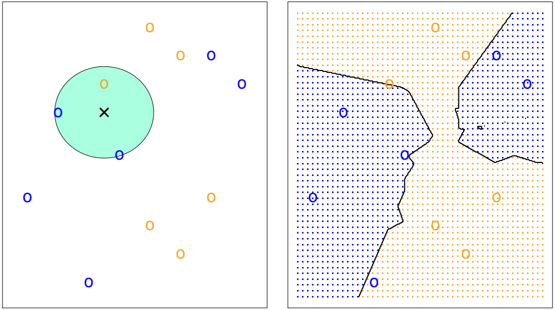
```
##
## fitted       setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         47         3
##   virginica       0          3        47
```

Naive Appraches 2: K Nearest Neighbor Classifier

- Compute sample proportion of the positive class based on k-nearest observations from $\mathbf{x}$

$$P(Y = 1|\mathbf{X} = \mathbf{x}) \approx \frac{1}{k} \sum_{i \in \mathcal{N}_k} \mathbb{1}(y_i = 1)$$

where $\mathcal{N}_k = \{\text{indices of k-nearest observations from } \mathbf{x}\}$



Naive Appraches 2: K Nearest Neighbor Classifier

```r
x <- iris[,-5] # predictors
cl <- factor(iris[,5]) # class labels
cv.error <- NULL
for (k in 1:50)  { # CV starts
   cl.hat <- knn.cv(x, cl, k = k) # CV fit
   cv.error[k] <- mean(cl.hat != cl) # CV error
}
opt.k <- max(which(cv.error == min(cv.error)))
cl.hat <- knn(train = x, test = x, cl = cl, k = opt.k)
print(table(cl, cl.hat))
```

```
##              cl.hat
## cl            setosa versicolor virginica
##   setosa          50          0         0
##   versicolor       0         48         2
##   virginica        0          1        49
```

Revisit: Bayes Classifier

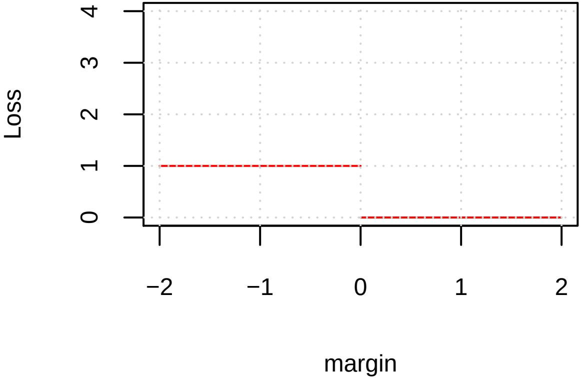
- Recall that Bayes Classifier minimzes the missclassification error rate:

$$\begin{aligned} f^* &= \underset{f}{\operatorname{argmin}} \; P\big[Y \neq \text{sign}\{f(\mathbf{x})\}\big] \\ &= \underset{f}{\operatorname{argmin}} \; E\big[\mathbb{1}\{Yf(\mathbf{x}) < 0\}\big] \end{aligned}$$

- Margin: $m = Yf(\mathbf{x})$ (the larger the better).
- Loss : $L(m) = \mathbb{1}\{m < 0\}$ (decreasing).
- Risk : $E\big\{L(m)\big]$ (Expected Loss).

0–1 loss.

- Bayes Classifier is a minimzer of the 0–1 Risk.



- In statistical learning, a lot of methods can be formulated as a risk minimzation problem.

Empirical Risk Minization (ERM)

- Given $(y_i, \mathbf{x}_i), i = 1, \cdots, n$, it is natural to solve

$$
\begin{aligned}
\hat{f} &= \operatorname*{argmin}_f \ \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{y_i f(\mathbf{x}_i) < 0\} \\
&= \operatorname*{argmin}_f \ \frac{1}{n} \sum_{i=1}^{n} L_{\text{0-1}}(m_i) \quad (\text{Empirical Risk}) \\
&\approx \operatorname*{argmin}_f \ E\{L_{\text{0-1}}(m)\}
\end{aligned}
$$

- Last line holds by the law of large numbers.

Linear Regression as an ERM Problem

- In regression we define margin (a.k.a residual) as

$$m_i = y_i - f(\mathbf{x}_i)$$

- The cloeser m_i to 0 the better f.
- Linear regression solves

$$\begin{aligned}
\hat{f} &= \underset{f}{\operatorname{argmin}} \ \frac{1}{n}\sum_{i=1}^{n}\{y_i - f(\mathbf{x}_i)\}^2 \\
&= \underset{f}{\operatorname{argmin}} \ \frac{1}{n}\sum_{i=1}^{n} L_2(m_i) \quad (\text{Empirical Risk}) \\
&\approx \underset{f}{\operatorname{argmin}} \ E\{L_2(m)\}.
\end{aligned}$$

where

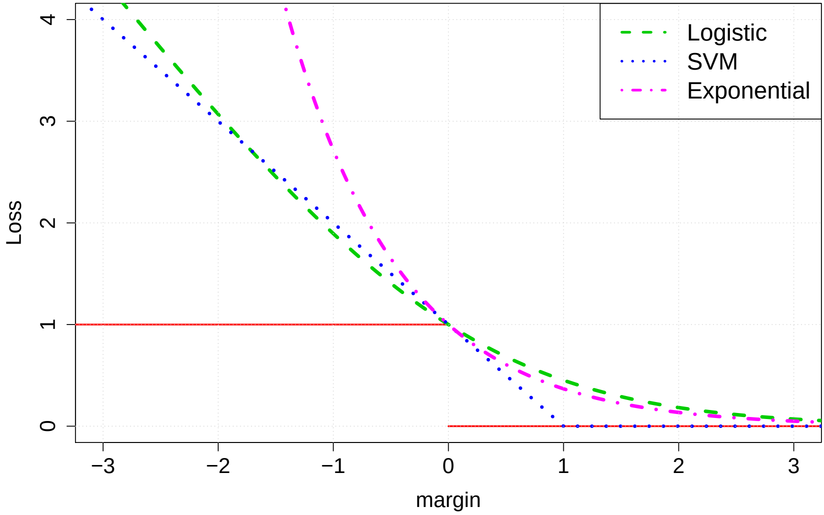
$$L_2(m) = m^2. \qquad (L_2 \text{ loss function})$$

Convex Surrogate of 0–1 loss.

- $L_{\text{0-1}}(m) = \mathbb{1}(m < 0)$ is difficult to handle.
- Relace it with convex functions.
- Examples:

$$
\begin{array}{llcl}
\text{(Logistic)} & L_{\text{logit}}(m) & = & \log\{1 + \exp(-m)\} \\
\text{(Hinge)} & L_{\text{hinge}}(m) & = & [1 - m]_+ \\
\text{(Exponential)} & L_{\text{exp}}(m) & = & \exp(-m)
\end{array}
$$

Convex Surrogate of 0–1 loss.



Revisit: Linear Regression

- Linear Regression Model

$$y_i = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i + \epsilon_i, \qquad \epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$$

which is equivalent

$$Y \mid \mathbf{X} = \mathbf{x}_i \sim N(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i, \sigma^2)$$

and implies

$$E(Y \mid \mathbf{X} = \mathbf{x}_i) = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i$$

Logistic Regression: Statistical Formulation

- For binary $Y \in \{0, 1\}$, the normal distribution is inadequate.
- Bernoulli distribution is a natural choice.

$$Y \mid \mathbf{X} = \mathbf{x}_i \sim \text{Bernoulli}(p(\mathbf{x}_i))$$

where

$$\begin{aligned} p(\mathbf{x}) &= P(Y = 1 \mid \mathbf{X} = \mathbf{x}) \\ &= E(Y \mid \mathbf{X} = \mathbf{x}). \end{aligned}$$

Logistic Regression: Statistical Formulation

- A linear model:
$$p(\mathbf{x}_i) = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i$$
yet, not accepable since $0 \leq p(\mathbf{x}_i) \leq 1$.
- LR emplys the logit transformation:
$$\text{logit}\{p(\mathbf{x}_i)\} = \log\left\{\frac{p(\mathbf{x}_i)}{1 - p(\mathbf{x}_i)}\right\} = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i.$$

Logistic Regression: Statistical Formulation

- LR regression

$$y_i \mid \mathbf{x}_i \sim \text{Bernoulli}(p_\beta(\mathbf{x}_i))$$

  where $y_i \in \{0, 1\}$ and

$$\log\left\{\frac{p_\beta(\mathbf{x}_i)}{1 - p_\beta(\mathbf{x}_i; \boldsymbol{\beta})}\right\} = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i$$

  or equivalently

$$p_\beta(\mathbf{x}_i) = \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}_i)}{1 + \exp(\boldsymbol{\beta}^T \mathbf{x}_i)}.$$

- Subscript β is used to emphasize that p_β is a function of parameters.

Logistic Regression: Application to Spam Data

```r
logit <- glm(type ~., data = spamtrain, family = "binomial")
fit.logit <- predict(logit, newdata = spamtest[,-p])
test.logit.y <- ifelse(fit.logit> 0.5, "spam", "nonspam")
table(spamtest$type, test.logit.y)
```

```
##          test.logit.y
##           nonspam spam
##   nonspam    2253  261
##   spam        259 1368
```

```r
lg.test.err <- mean(spamtest$type != test.logit.y)
cat("test.error  =", round(lg.test.err, 4), "\n")
```

```
## test.error  = 0.1256
```

Logistic Regression: Statistical Formulation

- To estimate β, we employ the maximum likelihood estimator (MLE) that solves

$$\hat{\beta} = \operatorname*{argmax}_{\beta} \prod_{i=1}^{n} L(\beta; \mathbf{x}_i)$$

where the likelihood is given by

$$L(\beta; \mathbf{x}_i) = \{p_\beta(\mathbf{x}_i)\}^{y_i} \cdot \{1 - p_\beta(\mathbf{x}_i)\}^{1-y_i}$$

- It is equivalent to minimze its negative of the logarithm:

$$-\sum_{i=1}^{n} [y_i \log\{p_\beta(\mathbf{x}_i)\} + (1 - y_i) \log\{1 - p_\beta(\mathbf{x}_i)\}]$$
$$= \sum_{i=1}^{n} \left[\log(1 + \exp(\beta^T \mathbf{x}_i)) - y_i \beta^T \mathbf{x}_i\right]$$

Logistic Regression: ERM formulation

- Inner part is

$$\begin{cases} \log\left\{1+\exp(-\boldsymbol{\beta}^T\mathbf{x}_i)\right\}, & \text{if } y_i = 1 \\ \log\left\{1+\exp(\boldsymbol{\beta}^T\mathbf{x}_i)\right\}, & \text{if } y_i = 0 \end{cases}$$

- For $y_i \in \{-1, 1\}$, it is equivalent to

$$\log\{1+\exp(-y_i f(\mathbf{x}_i))\}$$

and thus LR solves

$$\min_f \frac{1}{n}\sum_{i=1}^{n} \log\{1+\exp(-y_i f(\mathbf{x}_i))\} \approx E\{L_{\text{logit}}(m)\},$$

where $f(\mathbf{x}_i) = \beta_0 + \boldsymbol{\beta}^T\mathbf{x}_i$ and $m_i = y_i f(\mathbf{x}_i)$.

Fisher Consistency

- Logistic loss is one example of convex surrogate of the zero-one loss function.
- Can we use any convex function? Certainly not.
- It is desired for a convex loss function $L(m)$ to satisfy

$$\text{sign}\{f^*(\mathbf{x})\} = \text{sign}\{p(\mathbf{x}) - 0.5\}$$

  where

$$f^* = \underset{f}{\text{argmin}}\, E[L\{Yf(\mathbf{x})\}]$$

- If this holds, we say the loss function L is Fisher consistent.

Fisher Consistency

- Fisher Consistency is a minimal condition for any convex surrogate loss function should possess.

Lemma (Sufficient Condition for Fisher Consistency)

If $L'(m)$ exists at $m = 0$ and negiative, then L is Fisher consistent.

- Check that all the aforementioned loss functions are Fisher consistent.

ERM formulation for Binary Classifier

- Any resonable binary classifier solves

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(m_i) \approx E\{L(m)\}$$

  for any Fisher consistent loss function L.

- The optimization is not difficult when $f(\mathbf{x})$ is linear, i.e.,

$$f(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\beta}$$

- What if the parameter space of f is too rich?
  - f is finite dimensional but too high (i.e., too many covariates)?
  - f is an arbitrary function?

Regularization

- The idea of regularzation is simple.
- Restrict the parameter space, and focus of functions satisfying

  $$J(f) < C$$

  for a given constant $C > 0$ and $J(f)$ being the functional of f that maeasues the complexity of f.
- For linear $f(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T\mathbf{x}$, a common choice is

  $$J(f) = \|f(\mathbf{x})\|_2^2 \propto \boldsymbol{\beta}^T\boldsymbol{\beta}(= \|\boldsymbol{\beta}\|_2^2).$$

Regularization: Ridge Regression

- Linear regression solves

$$\min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - \boldsymbol{\beta}^T \mathbf{x}_i)^2.$$

- When $p > n$, a regularization can be considered:

$$\min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} \{y_i - \beta_0 - \boldsymbol{\beta}^T \mathbf{x}_i\}^2, \qquad \text{s.t. } \boldsymbol{\beta}^T \boldsymbol{\beta} < C.$$

- Equivalently written as

$$\min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - \boldsymbol{\beta}^T \mathbf{x}_i)^2 + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta}.$$

where $\lambda > 0 \;\; (\propto C^{-1})$.

Regularization: LR

- LR solves

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n} \log\left\{1+\exp(-y_i f(\mathbf{x}_i))\right\}$$

- Ridge LR:

$$\min_{f} \frac{1}{n}\sum_{i=1}^{n} \log\left\{1+\exp(-y_i f(\mathbf{x}_i))\right\} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

Kernel Logistic Regression

- Emplying the kernel trick, we have

$$f(\mathbf{x}) = \beta_0 + \sum_{i=1}^{n} \theta_i K(\mathbf{x}_i, \mathbf{x})$$

where K is a kerel function.

$$\min_{f} \ \frac{1}{n} \sum_{i=1}^{n} \log\left\{1 + \exp(-y_i f(\mathbf{x}_i))\right\} + \lambda \boldsymbol{\theta}^T \mathbf{K} \boldsymbol{\theta}$$

Revisit: SVM

- Original formulation:

$$\min_{\beta_0, \boldsymbol{\beta}, \xi_i} \boldsymbol{\beta}^T \boldsymbol{\beta} + C \sum_{i=1}^{n} \xi_i$$

$$\begin{aligned} \text{subject to} \quad & y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i) \geq 1 - \xi_i, \quad && i = 1, \cdots, n \\ & \xi_i \geq 0, && i = 1, \cdots, n. \end{aligned}$$

- Statistical/ERM formulation:

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_i)]_+ + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta}$$

where $\lambda \propto C^{-1}$.

SVM vs Logistic

- SVM

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n}[1 - y_i(\beta_0 + \boldsymbol{\beta}^T\mathbf{x}_i)]_+ + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

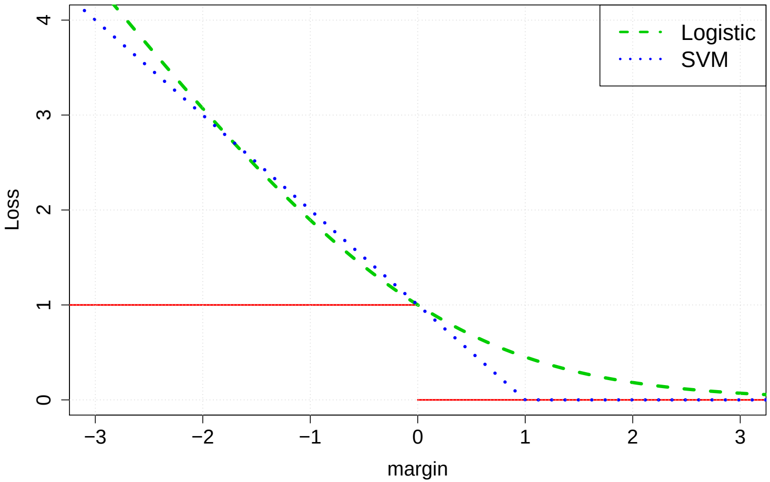
- Ridge LR

$$\min_{\beta_0,\boldsymbol{\beta}} \ \frac{1}{n}\sum_{i=1}^{n}\log\left\{1 + \exp(-y_i(\beta_0 + \boldsymbol{\beta}^T\mathbf{x}_i))\right\} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

SVM vs Logistic as ERM problems

- Both are ERM problems (Ridge-Penalized)

$$\min_{\beta_0, \boldsymbol{\beta}} \; \frac{1}{n} \sum_{i=1}^{n} \{L(m_i)\} + \lambda J(f)$$



ERM in ML

- Most, if not all, existing ML algorithms can be viewed as ERM problem (with regularzation if needed).
- ERM can be uniquely determined by combination of
  - Loss function L
    - Logistic / Hinge / Exponental
    - Large margine Unified Machine
    - ψ-loss / Truncated Hinge
    - ....
  - Model f: Beyond Linear
    - Generalized Additive Model
    - Piecewise Constant: Tree
    - Composite function: NN
    - ....