Support Vector Machines (2)
Learning from Data

Overview
- Separable Case
- Kernel Functions
- Allowing Errors (Soft Margin, Regularization)
- SVM’s in R: Computing the entire regularization path with the `svmpath` package.

Allowing Errors
- So far we assumed that the training data points are linearly separable in feature space $\phi(x)$.
- Resulting SVM gives exact separation of training data in original input space $x$, with non-linear decision boundary.
- Class distributions typically overlap, in which case exact separation of the training data leads to poor generalization (overfitting).

Error function
We can express the maximum margin classifier in terms of the minimization of an error function, with a simple quadratic regularizer, in the form

$$
\sum_{n=1}^{N} E_\infty(y(x_n)t_n - 1) + \lambda \|w\|^2
$$

(7.19)

where $\lambda > 0$, and $E_\infty(z)$ is zero if $z \geq 0$ and $\infty$ otherwise. $E_\infty$ ensures that the constraints

$$
t_n(w^T\phi(x_n) + b) \geq 1 \quad n = 1, \ldots, N
$$

(7.5)

are satisfied.

Definition of Slack Variables
We define $\xi_n = 0$ for data points that are on the inside of the correct margin boundary and $\xi_n = |t_n - y(x_n)|$ for all other data points.

Data points are allowed to be on the “wrong” side of the margin boundary, but with a penalty that increases with the distance from that boundary.

For convenience we make this penalty a linear function of the distance to the margin boundary.

Introduce slack variables $\xi_n \geq 0$ with one slack variable for each training data point.
New Constraints

The exact classification constraints
\[ t_n y(x_n) \geq 1 \quad n = 1, \ldots, N \] (7.5)
are replaced by
\[ t_n y(x_n) \geq 1 - \xi_n \quad n = 1, \ldots, N \] (7.20)

Optimization Problem

The Lagrangian is given by
\[ L(w, b, \alpha) = \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} \alpha_n (t_n y(x_n) - 1 + \xi_n) - \sum_{n=1}^{N} \alpha_n \xi_n \] (7.22)
where \( \alpha_n \geq 0 \) and \( \mu_n \geq 0 \) are Lagrange multipliers.

The KKT conditions are given by:
\[
\begin{align*}
\alpha_n &\geq 0 \quad (7.23) \\
t_n y(x_n) - 1 + \xi_n &\geq 0 \quad (7.24) \\
a_n (t_n y(x_n) - 1 + \xi_n) &\geq 0 \quad (7.25) \\
\mu_n &\geq 0 \quad (7.26) \\
\xi_n &\geq 0 \quad (7.27) \\
\mu_n \xi_n &\geq 0 \quad (7.28)
\end{align*}
\]

Dual

Take derivative with respect to \( w \), \( b \) and \( \xi_n \) and equate to zero:
\[
\begin{align*}
\frac{\partial L}{\partial w} = 0 \Rightarrow w &= \sum_{n=1}^{N} a_n t_n \phi(x_n) \quad (7.29) \\
\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} a_n t_n &= 0 \quad (7.30) \\
\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n &= C - \mu_n \quad (7.31)
\end{align*}
\]

Prediction

Recall that
\[ y(x) = w^T \phi(x) + b \] (7.1)
Substituting
\[ w = \sum_{n=1}^{N} a_n t_n \phi(x_n) \] (7.8)
into (7.1), we get
\[ y(x) = \sum_{n=1}^{N} a_n t_n k(x, x_n) + b \] (7.13)
with \( k(x, x_n) = \phi(x)^T \phi(x_n) \).
Interpretation of Solution

- Points with $a_n = 0$ do not play a role in making predictions.
- Points with $a_n > 0$ are called support vectors:
  - If $a_n < C$ then $\xi_n = 0$ and hence such points lie on the margin.
  - Points with $a_n = C$ can be inside the margin and can either be correctly classified if $\xi_n \leq 1$ or misclassified if $\xi_n > 1$.

Relation to logistic regression

Recast the SVM in terms of minimization of a regularized error function. The objective was to minimize

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|w\|^2$$  \hspace{1cm} (7.21)

with $\xi_n = 0$ for data points on the correct side of the margin boundary (i.e. $y_nt \geq 1$) and $\xi_n = 1 - y_nt$ for all other points. Hence we can write (7.21) as

$$C \sum_{n=1}^{N} ESV(y_nt) + \frac{1}{2} \|w\|^2$$

where $ESV(\cdot)$ is the hinge error function defined by

$$ESV(y_n) = \max(0, 1 - y_n)$$

Relation to logistic regression

Dividing by $C$ we get objective function

$$\sum_{n=1}^{N} ESV(y_nt) + \frac{1}{2C} \|w\|^2$$

which we can write as

$$\sum_{n=1}^{N} ESV(y_nt) + \lambda \|w\|^2$$  \hspace{1cm} (7.44)

with $\lambda = 1/(2C)$.

Computing the intercept

To compute the value of $b$, we use the fact that those support vectors with $0 < a_n < C$ have $\xi_n = 0$ so that $t_n y_n(x_n) = 1$, and hence (using 7.13) satisfy

$$t_n \left( \sum_{m \in S} a_m t_m k(x_n, x_m) + b \right) = 1$$  \hspace{1cm} (7.36)

Hence we have

$$t_n \sum_{m \in S} a_m t_m k(x_n, x_m) + t_n b = 1$$

$$t_n b = 1 - t_n \sum_{m \in S} a_m t_m k(x_n, x_m)$$

$$b = t_n - \sum_{m \in S} a_m t_m k(x_n, x_m)$$

since $t_n \in \{-1, +1\}$ and so $1/t_n = t_n$.

Hinge loss function

$$\max(0, 1 - y_nt)$$

Logistic Regression Error

Logistic regression assumption

$$p(t = 1 | y) = \sigma(y) = (1 + \exp(-y))^{-1}$$

Hence

$$p(t = -1 | y) = 1 - \sigma(y) = \sigma(-y)$$

so we can write for short

$$p(t | y) = \sigma(yt) = (1 + \exp(-yt))^{-1}$$  \hspace{1cm} (7.46)
Logistic Regression Error

Hence the negative log-likelihood of the \( n \) th observation is given by
\[
- \ln(1 + \exp(-y_n t_n))^{-1} = \ln(1 + \exp(-y_n t_n))
\]
so logistic regression with a quadratic regularizer minimizes the error function
\[
\sum_{n=1}^{N} E_{LR}(y_n t_n) + \lambda \| w \|^2 \tag{7.47}
\]
where
\[
E_{LR}(yt) = \ln(1 + \exp(-yt)) \tag{7.48}
\]

Model Selection

- As usual we are confronted with the problem of selecting the appropriate model complexity.
- The relevant parameters are \( C \) and any parameters of the chosen kernel function.
- Trying a range of values for \( C \) is expensive if we have to solve a potentially complex optimization problem each time.
- Hastie et al. show that this is not necessary.

svmpath

Algorithm keeps track of the following sets of points:
1. \( E = \{ n : t_n y(x_n) = 1, 0 \leq a_n \leq C \} \)
2. \( L = \{ n : t_n y(x_n) < 1, a_n = C \} \)
3. \( R = \{ n : t_n y(x_n) > 1, a_n = 0 \} \)
   - \( E \): on the “elbow” of hinge loss.
   - \( L \): left of the elbow.
   - \( R \): right of the elbow.

Breakpoints occur when set \( E \) changes due to one of the following events.
1. Initialization: two or more points start at the elbow.
2. A point from \( L \) enters \( E \).
3. A point from \( R \) reenters \( E \).
4. One or more points in \( E \) leave for \( L \) or \( R \).
These events establish breakpoints for \( a_n \) values. Values in-between breakpoints are obtained by linear interpolation.
The path for Conn’s syndrome with linear kernel

```
 Event Lambda Margin Elbow Error
 1: Obs 18 ->E 57.770000 14.38 2  6
 2: Obs 14 L->E 38.052095 14.27 3  8
 3: Obs 23 E->R 36.494989 13.69 3  5
 4: Obs 21 L->E 31.617498 13.69 3  5
 6: Obs 6 L->E 29.864894 13.68 3  5
 7: Obs 21 E->R 16.943596 13.68 2  5
 8: Obs 26 L->E 11.405999 12.55 3  5
 9: Obs 6 E->R  4.822507 12.56 2  5
10: Obs 14 E->R  4.765000 12.55 2  5
11: Obs 30 ->E  4.130000 12.33 2  5
12: Obs 15 ->E  4.130000 12.33 2  5
```

Note: \( \lambda = 1/C \), so lower values of \( \lambda \) correspond to more complex models. As \( \lambda \) becomes smaller, \( \|w\|^2 \) becomes bigger, and so the margin becomes smaller.

---

### Polynomial kernel with \( M = 2 \)

```
 Event Lambda Margin Elbow Error
 7: Obs 21 E->R 30.342646 13.69 2  5
10: Obs 18 E->R 31.617498 13.69 3  5
11: Obs 23 E->R 36.494989 13.68 2  5
12: Obs 7 L->E  3.967408 12.29 3  5
```

### Assignment Data

```
# replace 0 labels by -1
kc1.dat$ERROR[kc1.dat$ERROR==0] <- -1
# indices of positive examples
pos.index <- c(1:2107)[kc1.dat$ERROR==1]
# indices of negative examples
neg.index <- c(1:2107)[kc1.dat$ERROR==-1]
# draw sample of size 40 from positive and negative examples
pos.sample <- sample(pos.index,40)
neg.sample <- sample(neg.index,40)
kc1.sample <- c(pos.sample,neg.sample)

# Fit path for linear SVM. kc1.svmdat contains halstead difficulty, halstead content and target.
kc1.svm1 <- svmpath(as.matrix(kc1.svmdat[kc1.sample, 1:2]),
kc1.svmdat[kc1.sample,3],plot.it = TRUE)
```

---

### SVM with Linear Kernel

```
 Step: 12 Error:  5 Elbow Size:  3 Margin: 12.29
```

---

### Assignment Data

```
Step:  6 Error:  5 Elbow Size:  4 Margin: 13.61
```

---

### SVM with Linear Kernel

```
Step:  36 Error: 29 Elbow Size:  3 Margin: 56.4
```

---

```
svm2007-2.pdf — March 26, 2009 — 5
```
SVM with Radial Kernel

```
Making Predictions

# Linear Kernel, lambda=600
> kc1.svm1.pred <- predict(kc1.svm1, kc1.svmdat[kc1.sample,1:2], type="class", lambda=600)
> table(kc1.svmdat[kc1.sample,3],kc1.svm1.pred)
   kc1.svm1.pred
     -1  1
     -1 30 10
      1 18 22

# Linear Kernel, lambda=5000
> kc1.svm1.pred <- predict(kc1.svm1, kc1.svmdat[kc1.sample,1:2], type="class", lambda=5000)
> table(kc1.svmdat[kc1.sample,3],kc1.svm1.pred)
   kc1.svm1.pred
     -1  1
     -1 32  8
      1 22 18
```

SVM in R

```
# Radial Kernel, lambda=0.8
> kc1.svm2.pred <- predict(kc1.svm2, kc1.svmdat[kc1.sample,1:2], type="class", lambda=0.8)
> table(kc1.svmdat[kc1.sample,3],kc1.svm2.pred)
   kc1.svm2.pred
     -1  1
     -1 37  3
      1  1 39
```

```