Install Packages for Snowfall

- Check out the HighPerformanceComputing link at http://cran.r-project.org/web/views
- A good guide for using snowfall is Tutorial: Parallel Computing using R package snowfall
- You need snow and snowfall
  
  ```r
  install.packages(c("snow","snowfall"))
  ```

- We will set up a cluster that communicates with TCP/IP sockets, because this works on Linux and MS Windows without installing any additional software.

Initialize a Cluster

- To initialize a cluster
  
  ```r
  sfinit ( parallel = TRUE, cpus = 4, type = "SOCK")
  ```

  `parallel` may be set to FALSE to run on a single

- You may also specify which machines to use
  
  ```r
  sfinit ( parallel = TRUE, type="SOCK", 
  socketHosts=c( "corn", "cucumber", "cucumber", "radish"))
  ```

  Without the `socketHosts` argument, you will be running on just your local host.

- Or, you may just call
  
  ```r
  sfinit()
  ```

  in your code and set the argument values in the R command line

  ```r
  R --no-save --no-restore --args --parallel --cpus=4 
  --type=SOCK
  ```

  The items that follow --args are parsed by sfInit using the R function commandArgs.
Which Hosts?

- To simplify the creation of the socketHost host list, Andrew Sutton has written a clever R function, which I call `snowfallSelectHosts` that
  - consults a file of host names and maximum number of CPUs to use for each,
  - uses the unix command `rup` to determine the current load on each host,
  - calculates the number of CPUs to use as the given maximum minus the current load,
  - duplicates the host name that many times, and returns the list of all host names.

Using `snowfallSelectHosts`

- You can also see the processing of each host. Remember to stop the cluster we just created first.

```r
sfStop()
hosts <- snowfallSelectHosts("machines", localhost=TRUE, print=TRUE)
```

which produces this output

```
Read 5 hosts from file "machines"
Using 0 of 5 slot(s) on brussels - sprout
Using 0 of 5 slot(s) on cauliflower
Using 0 of 5 slot(s) on horseradish
Using 3 of 5 slot(s) on kelp
Using 0 of 5 slot(s) on romanesco
Using 7 of 8 slot(s) on thoumire
for total of 10 slots
```

after which you continue with

```r
sfInit ( parallel = TRUE, type = "SOCK", socketHosts = hosts)
snowfall 1.70 initialized : parallel execution on 10 CPUs.
```

Using new distributed apply functions

- Say we want to square each value of a list named `data`. Can use `sfLapply`.

```r
data <- 1:5
result <- sfLapply(data, function(x) {x * x})
print(result)
```

```
[[1]]
[1] 1
[[2]]
[1] 4
[[3]]
[1] 9
[[4]]
[1] 16
[[5]]
[1] 25
```
Can use the automatic load balancing provided by snowfall by using sfClusterApplyLB.

```r
data <- 1:5
result <- sfClusterApplyLB(data, function(x) {x * x})
print(result)
```

### Distributed Training of Multiple Neural Networks

```r
### Function and Data to be approximated by neural network.

f <- function(x) -1 + 0.05 * x + 0.4 * sin(x) + 0.1 * nrm(abs(length(x)))
N <- 40
xmax <- 40
Xtrain <- matrix(seq(0,xmax,length=N),N,1)
Xtest <- Xtrain + xmax/N/2
Ttest <- f(Xtest)

### Helpful function

```r
rmse <- function(y,t) {
  sqrt(mean((y-t)^2))
}
```

### Set up cluster

```r
library(snowfall)
sfSource("snowfallUtilities.R")
sfinit(parallel = TRUE, cpus = 40, type = "SOCK")
sfExport("Xtrain", "Xtest", "Ttest", "rmse", "f", "N", "xmax")
```

### Initialize a Cluster

```r
sfLibrary(myneuralnet)
sfSource("/s/parsons/e/fac/anderson/tmp/nn.R")
```

### Example

```r
Example

sfExport("Xtrain", "Xtest", "Ttest", "rmse", "f", "N", "xmax")
```

### Must also make sure each process loads the needed libraries and sources the needed R source files.

```r
sfLibrary(myneuralnet)
sfSource("/s/parsons/e/fac/anderson/tmp/nn.R")
```

### Also must move all needed R objects, including data and functions, to each node.

```r
sfExport("Xtrain", "Xtest", "Ttest", "rmse", "f", "N", "xmax")
```
R Version: R version 2.9.1 (2009-06-26)
snowfall 1.70 initialized: parallel execution on 32 CPUs.

Stopping cluster

nNodes   RMSE
[1,] 1 0.14635357
[2,] 2 0.12422137

Argument 1 ran on lang process 31507
Argument 2 ran on horseradish process 3352

Argument 19 ran on lang process 31507
Argument 20 ran on horseradish process 3352

Argument 19 ran on horseradish process 3352
Argument 20 ran on horseradish process 3357
Large Margin Classifiers

- For a two-class classification problem, using target values of -1 and 1, a sample $x_n$ is classified correctly by linear classifier if $t_n(w^T \phi(x_n) + b) > 0$.

![Linear Classifier](image)

- Multiple lines work. Prefer the one for which the smallest perpendicular distance to a sample is maximized.

- For a correctly classified sample, $x_n$, $t_n y(x_n) > 0$, so the distance of the sample to the boundary is $\frac{t_n y(x_n)}{||w||}$.

- So, what we want is

$$\arg\max_{w,b} \left( \min_n \frac{t_n y(x_n)}{||w||} \right)$$

$$= \arg\max_{w,b} \left( \frac{1}{||w||} \min_n t_n y(x_n) \right)$$

- This is difficult. Must simplify. Notice that

$$\frac{t_n y(x_n)}{||w||} = \frac{t_n (w^T \phi(x_n) + b)}{||w||}$$

$$= \frac{t_n (cw^T \phi(x_n) + cb)}{||cw||}$$

for any $c$.

- Let's choose a $c$ for which $t(w^T \phi(x) + b) = 1$ (once $c$ is absorbed into $w$ and $b$) for the sample $x$ that is closest to the boundary. So $t_n(w^T \phi(x_n) + b) \geq 1$ for all $n$.

Perpendicular Distance

- What is perpendicular distance, $r$, from the line to a sample, $x$?

![Perpendicular Distance](image)

- Now our optimization problem is

$$\arg\max_{w,b} \left( \frac{1}{||w||} \min_n t_n y(x_n) \right)$$

$$= \arg\max_{w,b} \left( \frac{1}{||w||} \cdot 1 \right)$$

$$= \arg\min_{w,b} \frac{1}{2} ||w||^2$$

with the constraint that $t_n(w^T \phi(x_n) + b) \geq 1$.

- Can use algorithms designed for quadratic optimization subject to linear constraints to find optimum $w$, but the following steps usually result in faster solutions.
Using Lagrange Multipliers

- Use Lagrange multipliers, \( \mathbf{a} = \{ \alpha_1, \ldots, \alpha_N \} \), to include the constraints in the optimization problem.

\[
L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} \alpha_n(t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) - 1)
\]

We want \( \text{argmax}_a \text{argmin}_w b L(\mathbf{w}, b, \mathbf{a}) \)

- How can we optimize this?

Substituting these results into \( L \) replaces \( \mathbf{w} \) and \( b \) with expressions involving \( \alpha_n \):

\[
L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{n=1}^{N} \alpha_n t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) + \sum_{n=1}^{N} \alpha_n
\]

\[
= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{n=1}^{N} \alpha_n t_n (\mathbf{w}^T \phi(\mathbf{x}_n)) - b \sum_{n=1}^{N} \alpha_n t_n + \sum_{n=1}^{N} \alpha_n
\]

\[
= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \mathbf{w}^T \mathbf{t} \sum_{n=1}^{N} \alpha_n t_n \phi(\mathbf{x}_n) - b \sum_{n=1}^{N} \alpha_n t_n + \sum_{n=1}^{N} \alpha_n
\]

\[
= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \mathbf{w}^T \mathbf{t} \sum_{n=1}^{N} \alpha_n t_n \phi(\mathbf{x}_n) - b \sum_{n=1}^{N} \alpha_n t_n + \sum_{n=1}^{N} \alpha_n
\]

\[
= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \mathbf{w}^T \mathbf{t} \sum_{n=1}^{N} \alpha_n t_n \phi(\mathbf{x}_n) - b \sum_{n=1}^{N} \alpha_n t_n + \sum_{n=1}^{N} \alpha_n
\]

such that \( \alpha_n \geq 0 \) and \( \sum_{n=1}^{N} \alpha_n t_n = 0 \).

Gradients!

- First work on inner part (argmin)

\[
\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^{N} \alpha_n t_n \phi(\mathbf{x}_n) = 0
\]

\[
\mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \phi(\mathbf{x}_n)
\]

and

\[
\frac{\partial L}{\partial b} = -\sum_{n=1}^{N} \alpha_n t_n = 0
\]

\[
\sum_{n=1}^{N} \alpha_n t_n = 0
\]

- Can optimize (incorrectly) by simply climbing the gradient with respect to \( \mathbf{a} \) and force all \( \alpha_n \geq 0 \).

\[
\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial \alpha_k} = 1 - \sum_{n=1}^{N} \alpha_n t_n t_k \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_k)
\]

- After climbing the gradient, can calculate

\[
\mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \phi(\mathbf{x}_n),
\]

and make predictions with

\[
y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b
\]

- But, what is \( b \)? Can show (Appendix E) that solution \( \mathbf{a} \) guarantees that

\[
\alpha_n \geq 0
\]

\[
t_n y(\mathbf{x}_n) - 1 \geq 0
\]

\[
\alpha_n (t_n y(\mathbf{x}_n) - 1) = 0
\]

so, for every sample, either \( \alpha_n = 0 \) or \( t_n y(\mathbf{x}_n) = 1 \).

- All samples for which \( \alpha_n > 0 \) are called support vectors.
So for support vector \( x_n \),
\[
t_n y(x_n) = 1 \\
t_n (w^T \phi(x_n) + b) = 1 \\
b = \frac{1}{t_n} - w^T \phi(x_n)
\]

So, \( \phi(x) \) only appears as a dot product with another \( \phi(x) \).

Key idea: Never have to explicitly calculate the feature vector \( \phi(x) \). Why is this a good idea?

If \( \phi(x) \) is high dimensional, would be more efficient if we can calculate \( \phi(x)^T \phi(v) \) in some way that doesn’t require calculating \( \phi(x) \).

Let \( k(x,v) = \phi(x)^T \phi(v) \). Can we just calculate \( k(x,v) \)?

Example (from Section 6.2). Let \( x \) and \( v \) be two-dimensional samples.
\[
k(x,v) = (x^Tv)^2 \\
= (x_1v_1 + x_2v_2)^2 \\
= x_1^2v_1^2 + 2x_1v_1x_2v_2 + x_2^2v_2^2 \\
= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(v_1^2, \sqrt{2}v_1v_2, v_2^2)^T \\
= \phi(x)^T \phi(v)
\]

What if \( x \) and \( v \) are 100-dimensional?

But, what about kernels?

- We found \( a = \{ \alpha_1, \ldots, \alpha_N \} \) that maximized \( L \). Many of the \( \alpha_i \)'s are zero. Let \( S \) be the set of sample indices for support vectors (the samples with \( \alpha_i > 0 \)). Rather than calculating the weight vector \( w \), we can leave the summation in place.

- Since
  \[
  w = \sum_{s \in S} \alpha_s t_s \phi(x_s)
  \]
  and
  \[
  y(x) = w^T \phi(x) + b
  \]
we can write
\[
y(x) = \sum_{s \in S} \alpha_s t_s \phi(x_s)^T \phi(x) + b
\]

- And, for \( b \), for support vector \( n \),
  \[
  b = \frac{1}{t_n} - w^T \phi(x_n)
  \]
  \[
  = \frac{1}{t_n} - \sum_{s \in S} \alpha_s t_s \phi(x_s)^T \phi(x_n)
  \]

- The matrix composed of all \( k(x_n, x_m) \) is called the kernel matrix, or the Gram matrix. It must satisfy certain properties to be a valid kernel matrix, meaning one that can be formed by the dot product of feature vectors. (symmetric, positive semidefinite)

- Can combine kernel matrices to form new ones.

- Another common example is the “Gaussian” kernel
  \[
  k(x,v) = e^{-||x-v||^2/2\sigma^2}
  \]

The feature vector that corresponds to this kernel has infinite dimensionality!

- Can construct kernel matrices from samples with symbolic attributes. If \( A_1 \) and \( A_2 \) are two subsets of a given set, then the following is a valid kernel function.
  \[
  k(A_1, A_2) = 2|A_1 \cap A_2|
  \]
Overlapping Class Distributions

- Above derivation assumed samples can be separated, so that
  \[ t_n y(x_n) \geq 1 \]
  - This assumption can be relaxed by allowing some “slack”
  \[ t_n y(x_n) \geq 1 - \psi_n \]
  - The goal is now to maximize the margin while softly penalizing samples that lie on the wrong side of the boundary. So, we want to minimize
  \[
  C \sum_{n=1}^{N} \psi_n + \frac{1}{2}||w||^2
  \]
  - Doing the Lagrangian thing again and taking derivatives, we end up with the same optimization problem for the \( \alpha \)'s but different constraints
    \[
    0 \leq \alpha_n \leq C \\
    \sum_{n=1}^{N} \alpha_n t_n = 0
    \]

Example

- Here is an example of following the gradient to optimize \( \alpha \)'s (in svmGradient.R)

Comparisons

- Example code from e1071 and our nnLogReg (in svmExample.R)

\[
\begin{align*}
\text{svm.model} & \leftarrow \text{svm}(\text{Type } \sim ., \text{ data } = \text{trainset}, \text{ cost } = 100, \text{ gamma } = 1) \\
\text{svm.pred} & \leftarrow \text{predict(svm.model, testset } [. , -10]) \\
\text{rpart.model} & \leftarrow \text{rpart(} \text{Type } \sim ., \text{ data } = \text{trainset}) \\
\text{rpart.pred} & \leftarrow \text{predict(rpart.model, testset } [. , -10], \text{ type } = \text{"class"}) \\
\text{nn.model} & \leftarrow \text{makeNNLogReg(trainset[,1:9], trainset[,10, drop=FALSE], } \\
& \quad \text{nn} = 20, \text{ lambda} = 0.1, \\
& \quad \text{fPrec=1e-6,xPrec=1e-8,nter=10000}) \\
\text{nn.pred} & \leftarrow \text{useNNLogReg(nn.model, testset[,1:9])}
\end{align*}
\]

SVMs from the e1071 Package

- “e1071: Misc Functions of the Department of Statistics (e1071), TU Wien”
- Support Vector Machines: The Interface to libsvm in package e1071, by David Meyer, Technische Universität Wien, Austria
- libsvm: the award-winning (IJCNN 2001) C++ implementation by Chih-Chung Chang and Chih-Jen Lin

libsvm: the award-winning (IJCNN 2001) C++ implementation by Chih-Chung Chang and Chih-Jen Lin