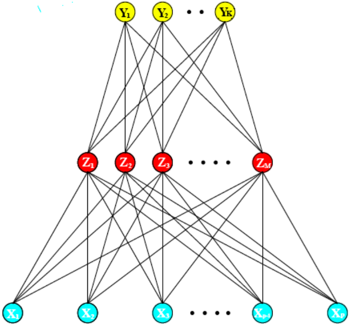
Notes

- Sample test questions posted
- Review and/or questions on Thursday this week
- Test will have 3 questions: one from Sample test, one specific to 414/2104
- Extra Office Hour Monday, March 15, 3-4
- Watch web site for late breaking announcement re MidTerm

Neural Networks

- "feed forward single layer neural network"



- $$Y_k = g_k\{\beta_{0k} + \sum_{m=1}^{M} \beta_{km}\sigma(\alpha_{0m} + \sum_{\ell=1}^{p} \alpha_{\ell m} X_\ell)\} = f_k(X_\ell)$$

- $\sigma(x) = \frac{1}{1+e^{-x}}$ $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, maps to $(-1, +1)$

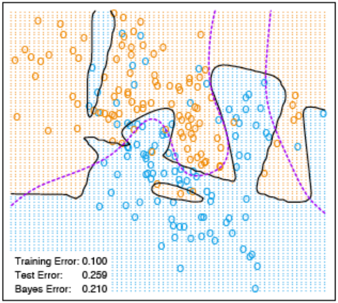
... neural networks

- $$Y_k = g_k\{\beta_{0k} + \sum_{m=1}^{M} \beta_{km}\sigma(\alpha_{0m} + \sum_{\ell=1}^{p} \alpha_{\ell m}X_\ell)\} = f_k(X_\ell)$$
- $\theta = (\alpha_{0m}, \alpha_m, \beta_{0k}, \beta_k)$
- $R(\theta) = \sum_{i=1}^{N}\sum_{k=1}^{K}\{y_{ik} - f_k(x_i)\}^2$, or
- $R(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log f_k(x_i)$
- $\dim(\theta) = M(p+1) + K(M+1) \longrightarrow$ regularization/shrinkage, also called weight decay
- minimize
$$R(\theta) + \lambda J(\theta) = R(\theta) + \lambda \left(\sum_{km} \beta_{km}^2 + \sum_{m\ell} \alpha_{m\ell}^2 \right)$$
- standardize inputs to mean 0, variance 1 for regularization
- backfitting algorithm for minimizing $R(\theta)$ described in §11.4; extension to $R(\theta) + \lambda J(\theta)$ in §11.5.2

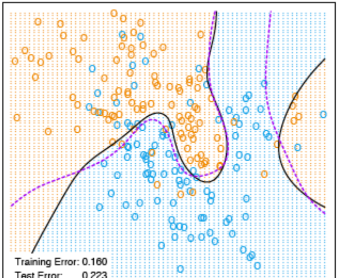
... neural networks

- `nnet` in `MASS` library: recommend
  $\lambda \in (10^{-4}, 10^{-2})$ for squared error loss;
  $\lambda \in (.01, .1)$ for log-likelihood
- compare Figure 11.4 top/bottom
- results very sensitive to starting values: $R(\theta)$ has many local maxima
- recommendation (Ripley): take average predictions over several `nnet` fits
- weight decay seems to be more important than number of hidden units
- See §11.7, 8, 9 for interesting examples where neural nets work well

Neural Network - 10 Units, No Weight Decay



Neural Network - 10 Units, Weight Decay=0.02



Aside: “Bayes error rate”

- $$pr(G = k \mid x) = \frac{f_k(x)\pi_k}{\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell}$$
- In Figure 11.4 (and many others) $x = (x_1, x_2)$
- data is simulated from known f_k with known probability π_k
- $pr(G = k \mid x_0)$ can be calculated for any x_0 in R^2
- x_0 assigned to, e.g., class 2 if

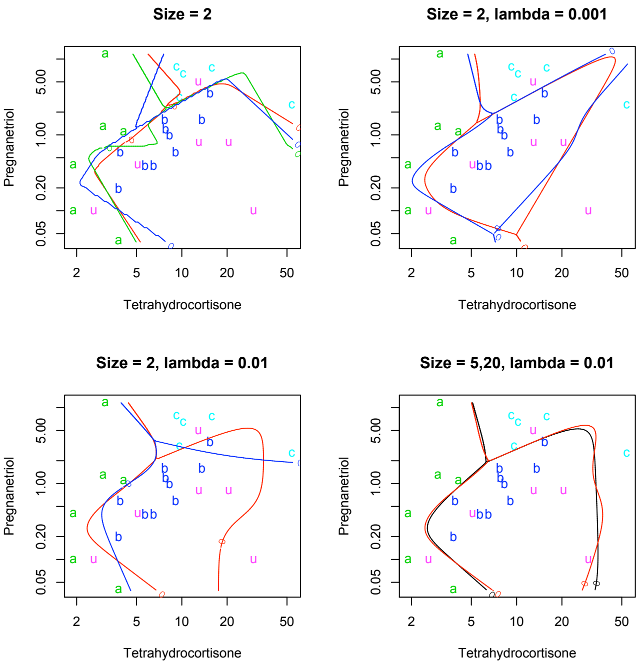
$$\begin{aligned} \Pr(G = 2 \mid x_0) &> \Pr(G = 1 \mid x_0) \\ &> \Pr(G = 3 \mid x_0), \text{etc.} \end{aligned}$$ (2.23) and Figure 2.5

- this gives the Bayes boundary (best possible)

Example from Venables and Ripley, Ch. 11

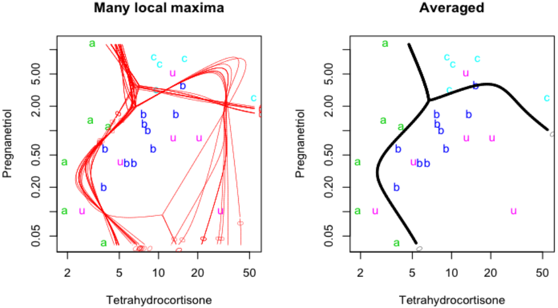
Handout

```
> library(MASS); library(nnet)
> data(Cushings)
> dim(Cushings)
[1] 27  3
> Cushings # result omitted
> cush = log(as.matrix(Cushings[,-3])) [1:21,]  ## use log scale for inputs, use known classes
> tp = Cushings$Type[1:21,drop=T]  ## record type when it is known
> par(mfrow=c(2,2))
> pltnn("Size = 2")
set.seed(1); plt.bndry(size = 2, col = 2)
set.seed(3); plt.bndry(size = 2, col = 3)
plt.bndry(size = 2, col = 4)

pltnn("Size = 2, lambda = 0.001")
set.seed(1); plt.bndry(size = 2, decay = 0.001, col = 2)
set.seed(2); plt.bndry(size = 2, decay = 0.001, col = 4)

pltnn("Size = 2, lambda = 0.01")
set.seed(1); plt.bndry(size = 2, decay = 0.01, col = 2)
set.seed(2); plt.bndry(size = 2, decay = 0.01, col = 4)

pltnn("Size = 5, 20  lambda = 0.01")
set.seed(2); plt.bndry(size = 5, decay = 0.01, col = 1)
set.seed(2); plt.bndry(size = 20, decay = 0.01, col = 2)
```


Size = 2
Size = 2, lambda = 0.001
Size = 2, lambda = 0.01
Size = 5,20, lambda = 0.01
Pregnanetriol
0.05
0.20
1.00
5.00
Tetrahydrocortisone
2
5
10
20
50

Average predictions over several fits

```
pltnn("Many local maxima")
Z <- matrix(0, nrow(cushT), ncol(tpi))
for(iter in 1:20) {
   set.seed(iter)
   cush.nn <- nnet(cush, tpi, skip = T, softmax = T, size = 3,
       decay = 0.01, maxit = 1000, trace = F)
   Z <- Z + predict(cush.nn, cushT)
# In R replace @ by $ in next line.
   cat("final value", format(round(cush.nn$value,3)), "\n")
   b1(predict(cush.nn, cushT), col = 2, lwd = 0.5)
}
pltnn("Averaged")
b1(Z, lwd = 3)
```



Support Vector Machines §12.2, 12.3
Zhu, M. *Amer. Statist.*

- not on test
- two class classification
- change notation so that $y = \pm 1$
- use *linear* combinations of p inputs to predict y

$$y = \begin{cases} -1 \\ +1 \end{cases} as \begin{array}{l} \beta_0 + x^T\beta < 0 \\ \beta_0 + x^T\beta > 0 \end{array}$$

- $f(x) = \beta_0 + x^T\beta = 0$ defines a hyperplane in $\mathbb{R}^p$
- this is a separating hyperplane if there exists $c > 0$ s.t.

$$y_i(\beta_0 + x_i^T\beta) > c, i = 1, \ldots, N$$

- by rescaling we can take $c = 1$ w.l.o.g.
- margin $= 2 \times \min\{y_i d_i, i = 1, \ldots, N\}$; d_i signed distance from x_i to hyperplane

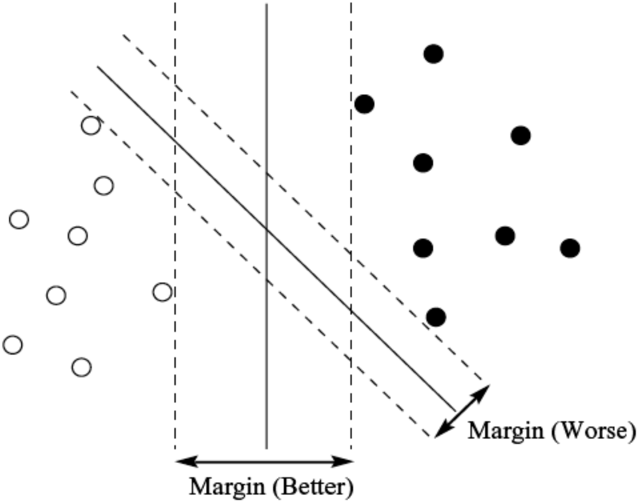


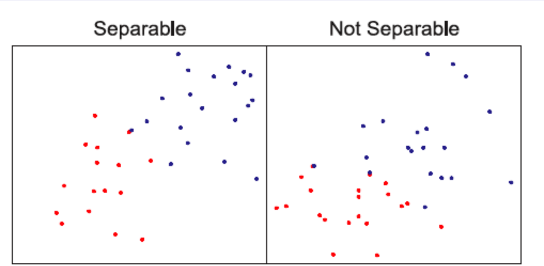
Figure 1. Two separating hyperplanes, one with a larger margin than the other.

... support vector machines

- $$\text{margin} = \frac{2}{||\beta||}$$
- maximizing margin means small β
- optimization problem becomes

$$\begin{aligned} &\min_{\beta_0,\beta} \frac{1}{2}||\beta||^2 \\ \text{s.t.} \quad & y_i(\beta_0 + x_i^T\beta) \geq 1, \quad i = 1, \dots N \end{aligned} \tag{4.48}$$

- Note: text has min $||\beta||$ (12.4)
  later changes to min $\frac{1}{2}||\beta||^2$ (12.8)



-
- Allowing overlap

$$\min_{\beta_0,\beta} \frac{1}{2}||\beta||^2 + \gamma \sum_{i=1}^{N} \xi_i \quad s.t. \quad \xi_i \geq 0$$

- subject to $\quad y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i$
- ξ_i called slack variables
- Book uses C; Zhu uses γ for tuning parameter (user specified)

- some points allowed to cross into the margin
- some points allowed to cross to the wrong side of the margin Figure 12.1
- the number of $\xi_i > 1$ is the number of misclassified points
- $\sum \xi_i$ is the total proportional amount by which predictions are on the wrong side

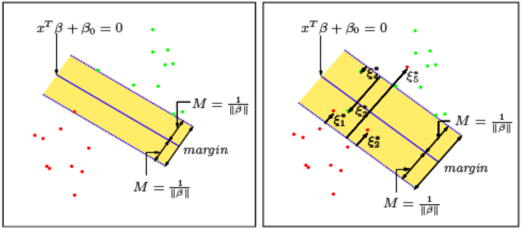


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.*

Constrained optimization

- $$\min_{\beta_0,\beta} \frac{1}{2}||\beta||^2 + \gamma \sum_{i=1}^{N} \xi_i \quad s.t. \quad \xi_i \geq 0$$
- subject to $\quad y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i$
- equivalent to
- $$\min \sum_{i=1}^{N} \{1 - y_i(x_i^T \beta + \beta_0)\}_+ + \lambda ||\beta||^2$$
- loss function with penalty
- solution is
$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$$

... optimization

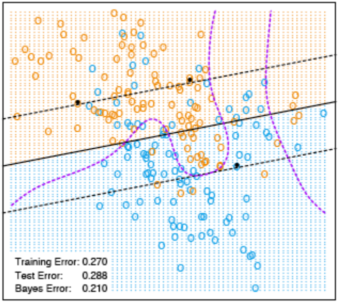
- $$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$$
- $\hat{\alpha}_i$ are solutions to
- $$\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$
- $$\text{s.t.} \ \sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \geq 0$$

... optimization

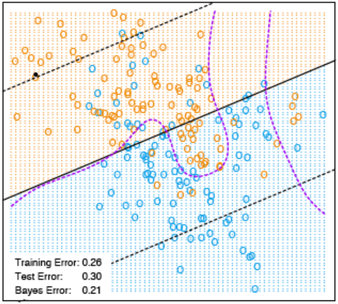
- the solution for β has the form

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i$$

- i.e. linear combinations of x_i ($y_i = \pm 1$)
- only some of the $\hat{\alpha}_i$ are nonzero: those where the lower bound is exact (12.14)
- these observations are called the support vectors
- $\hat{\alpha}_i$ are solutions to
- $$\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$
- $$\text{s.t.} \sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \geq 0$$
- Figure 12.2


Training Error: 0.270
Test Error: 0.288
Bayes Error: 0.210

$C = 10000$


Training Error: 0.26
Test Error: 0.30
Bayes Error: 0.21

$C = 0.01$

Beyond linear (§12.3)

- $$\hat{f}(x) = \hat{\beta}_0 + x^T\hat{\beta} = \hat{\beta}_0 + \sum_{i \in \mathrm{SV}} \hat{\alpha}_i y_i x_i^T x = 0$$
- depends only on inner products $x_i^T x$
- use basis function expansions to create more flexible boundaries
- $f(x) = h(x)^T\beta + \beta_0$
- new $L_D = \sum \alpha_i - \frac{1}{2}\sum\sum \alpha_i \alpha_{i'} y_i y_{i'} h(x_i)^T h(x_{i'})$
- solution depends only on inner products
- Iternatively depends on $h(\cdot)$ only through its
- Kernel function $K(x, x') =< h(x), h(x') >$
  - polynomial: $(1+ < x, x' >)^d$
  - radial basis: $\exp(-||x - x'||^2/c)$
  - neural network $\tanh(\kappa_1 < x, x' > + \kappa_2)$

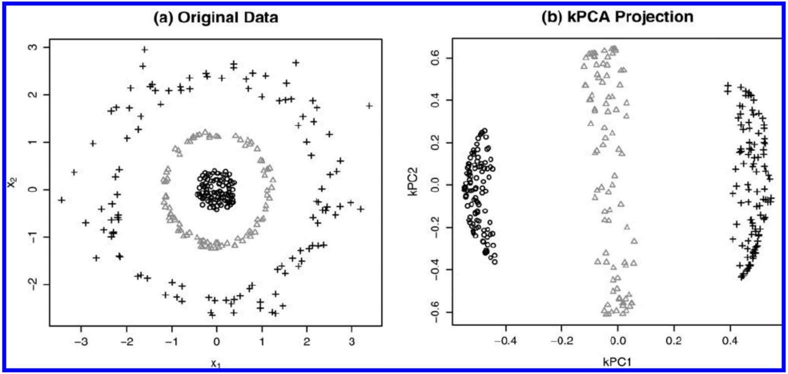


Figure 2. Kernel PCA, toy example. (a) Original data. (b) Projection onto the first two kernel principal components.

JFP slides