Notes

- No class this Friday or next Friday
- No office hours next Thursday or Friday
- HW 3 coming on Mar 16, due last day of classes
- project due 1 week after last day of classes
Tree-based methods (§9.2)

- §9.2.1, 9.2.2 regression trees ($y$ is continuous)
- formalism: $\hat{f}(x) = \sum_{m=1}^{M} c_m 1\{x \in R_m\}$
- $R_m$ is a subspace of $R^p$ obtained by partitioning the feature space using binary splits
- if $R_m$ is fixed, then the optimal choice of $c_m$ to minimize $\sum (y_i - f(x_i))^2$ is just $\text{ave}(y_i \mid x_i \in R_m)$
- trees are ‘grown’ in a greedy fashion, starting with any node and finding the variable to split on $X_j, \ j = 1, \ldots, p$ and the split point $s$
- to minimize squared error after splitting

$$\min_{j,s} \left[ \min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

- $R_1(j, s) = \{X \mid X_j \leq s\}, \ R_2(j, s) = \{X \mid X_j > s\}$
- $\hat{c}_1 = \text{ave}\{y_i \mid x_i \in R_1(j, s)\}, \ \hat{c}_2 = \text{ave}\{y_i \mid x_i \in R_2(j, s)\}$
Tree-based methods (§9.2)

- Trees are grown to be quite large and then pruned, using a cost-complexity criterion.

\[ C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T| \] (9.16)

\[ Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2, \quad \hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i \]

- \( |T| \) is the number of terminal nodes.
- \( C_\alpha(T) \) trades off fit to data \( Q_m \) and tree size \( T \).
- For each \( \alpha \) there is a pruning strategy.
- Choose \( \alpha \) by 5 or 10 fold CV.
- See Figure 9.5 for a classification tree.
- More on trees and MARS on March 16.
Projection pursuit regression (§11.2)

- response $Y$, inputs $X = (X_1, \ldots, X_p)$
- model $f(X) = E(Y \mid X)$ or $f(X) = \text{pr}(Y = 1 \mid X)$ or $f_k(X) = \text{pr}(Y = k \mid X)$
- PPR model $f(X) = \sum_{m=1}^{M} g_m(\omega_m^T X) = \sum g_m(V_m)$, say
- $g_m$ are 'smooth' functions, as in generalized additive models
- $V_m = \omega_m^T X$ are derived variables: the projection of $X$ onto $\omega_m = (\omega_{m1}, \ldots, \omega_{mp})$, with $\|\omega_m\| = 1$
- see Figure 11.1
- as $g_m$ are nonlinear (in general), we are forming nonlinear functions of linear combinations
- as $M \to \infty$, $\sum g_m(\omega_m^T X)$ can get arbitrarily close to any continuous function on $\mathbb{R}^p$
- if $M = 1$ a generalization of linear regression
PPR fitting

- training data \((x_i, y_i), \quad i = 1, \ldots, N\)

\[
\min_{\{g_m, \omega_m\}} \sum_{i=1}^{N} \left\{ y_i - \sum_{m=1}^{M} g(\omega_m^T x_i) \right\}^2
\]

- \(M = 1\): fix \(\omega\), form \(v_i = \omega^T x_i, i = 1, \ldots, N\)
- solve for \(g\) using a regression smoother – kernel, spline, loess, etc.
- given \(g\), estimate \(\omega\) by weighted least squares of a derived variable \(z_i\) on \(x_i\) with weights \(g_0^2(\omega_0^T x_i)\) and no constant term
- uses a simple linear approximation to \(g(\cdot)\) (see note)
- if \(M > 1\) add in each derived input one at a time
training data \((x_i, y_i), \ i = 1, \ldots, N\)

\[
\min_{\omega} \sum_{i=1}^{N} (y_i - \sum_{m=1}^{M} g^{(m)}(x_i))^{2}
\]

- \(M = 1\): fix \(\omega\), form \(v_i = \omega^T x_i\), \(i = 1, \ldots, N\)
- solve for \(g\) using a regression smoother – kernel, spline, \(\text{loess}\), etc.
- given \(g\), estimate \(\omega\) by weighted least squares of a derived variable \(z_i\) on \(x_i\) with weights \(g^2(\omega^T x_i)\) and no constant term
- uses a simple linear approximation to \(g(\cdot)\) (see note)
- if \(M > 1\) add in each derived input one at a time

PPR fitting

\[
g(\omega^T x_i) \approx g(\omega_0^T x_i) + g'(\omega_0^T x_i)(\omega - \omega_0)^T x_i
\]

\[
\{y_i - g(\omega^T x_i)\}^2 = \{y_i - g_0 - g'(\omega - \omega_0)^T x_i\}^2
\]

\[
= (g')^2 \left\{ \frac{y_i}{g'} - \frac{g_0}{g'} - (\omega - \omega_0)^T x_i \right\}^2
\]

\[
= (g')^2 \left\{ \omega_0^T x_i + \left( \frac{y_i - g_0}{g'} \right) - \omega^T x_i \right\}^2
\]

weight derived response (target)
PPR implementation

- A smoothing method that provides derivatives is convenient.
- Possible to put in a backfitting step to improve $g_m$'s after all $M$ are included; possible as well to refit the $\omega_m$.
- $M$ is usually estimated as part of the fitting.
- Provided in MASS library as `ppr`: fits $M_{\text{max}}$ terms and drops least effective term and refits, continues down to $M$ terms: both $M$ and $M_{\text{max}}$ provided by the user.
- `ppr` also accommodates more than a single response $Y$; see help file and VR p.280.
- Difficult to interpret results of model fit, but may give good predictions on test data.
- PPR is more general than GAM, because it can accommodate interactions between features: eg. $X_1X_2 = \{(X_1 + X_2)^2 - (X_1 - X_2)^2\}/4$.
- The idea of ‘important’ or ‘interesting’ projections can be used in other contexts to reduce the number of features, in classification and in unsupervised learning, for example.
Neural networks (§11.3)

- inputs $X_1, \ldots, X_p$
- derived inputs $Z_1, \ldots, Z_M$ (hidden layer)
- output (response) $Y_1, \ldots, Y_K$
- usual regression has $K = 1$; classification has $(Y_1, \ldots, Y_K) = (0, \ldots, 1, 0, \ldots)$
- also can accommodate multivariate regression with several outputs
- derived inputs $Z_m = \sigma(\alpha_{0m} + \alpha_m^T X)$ for some choice $\sigma(\cdot)$
- output $Y_k = f_k(X) = g_k(\beta_{0k} + \beta_k^T Z)$ for some choice $g_k(\cdot)$
- $\beta_{0k} + \beta_k^T Z$ called the $k$th target, $T_k$
- $\sigma(v)$ called an activation function, usually chosen to be logistic $1/(1 + e^{-v})$ (sigmoid)
- in regression $g_k$ would usually be the identity function, in classification logistic
- in $K$-class classification usually use $g_k(T) = \frac{e^{T_k}}{\sum_{\ell=1}^{K} e^{T_k}}$
Neural networks (§11.3)

- connection to PPR: $\sum_{m=1}^{M} g_m(\omega_m^T X)$
- $V_m \rightarrow Z_m = \sigma(\alpha_{0m} + \alpha_m^T X)$
- $g_m \rightarrow \sum_{m=1}^{M} \beta_{km} Z_m$
- i.e. $g_m(V_m)$ replaced by $\beta_m\sigma(\alpha_{0m} + \alpha_m^T X)$
- smooth functions are less flexible, but may have many derived $Z$’s
- note that the intercept terms $\alpha_{0m}$ and $\beta_{0k}$ could be absorbed into the general expression by including an input of 1, and a hidden layer input of 1; these are called ‘bias units’
need to estimate \((\alpha_{0m}, \alpha_m), m = 1, \ldots, M M(p + 1)\) and \((\beta_{0k}, \beta_k), k = 1, \ldots K K(M + 1)\).

loss function \(R(\theta); \theta = (\alpha_{0m}, \alpha_m, \beta_{0k}, \beta_k)\) to be minimized; regularization needed to avoid overfitting.

loss function would be least squares in regression setting, e.g.

\[
\sum_{k=1}^{K} \sum_{i=1}^{N} \{y_{ij} - f_k(x_i)\}^2
\]

for classification could use cross-entropy

\[
\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i)
\]

the parameters \(\alpha\) and \(\beta\) called (confusingly) weights, and regularization is called weight decay.
data \((y_{ik}, x_i), i = 1, \ldots, N, k = 1, \ldots, K\): let 
\[ z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i) \text{ and } z_i = (z_{1i}, \ldots, z_{mi}) \]

\[ f_k(X) = g_k(\beta_{0k} + \beta_k^T Z) : Z_m = \sigma(\alpha_{0m} + \alpha_m^T X) \]

\[ R(\theta) = \sum_{i=1}^{N} \sum_{k=1}^{K} \{y_{ik} - f_k(x_i)\}^2 = \sum R_i(\theta), \text{ say} \]

\[ \frac{\partial R_i}{\partial \beta_{km}} = -2\{y_{ik} - f_k(x_i)\} g_k'(\beta_k^T z_i) z_{mi} \]

\[ \frac{\partial R_i}{\partial \alpha_{ml}} = -2 \sum_{k=1}^{K} \{y_{ik} - f_k(x_i)\} g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) x_{i\ell} \]

at each iteration use \(\partial R/\partial \theta\) to guide choice to next point
Back propagation

\[ \beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}} \]

\[ \alpha_{m\ell}^{(r+1)} = \alpha_{m\ell}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{m\ell}^{(r)}} \]

\[ \delta_{ki} = -2(y_{ik} - f_k(x_i))g'_k(\beta_k^T z_i) \]

\[ s_{mi} = -2 \sum_{k=1}^{K} \{y_{ik} - f_k(x_i)\} g'_k(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) \]

\[ s_{mi} = \sigma'(\alpha_m^T x_i) \sum_{i=1}^{K} \beta_{km} \delta_{ki} \quad (11.15) \]

use current estimates to get \( \hat{f}_k(x_i) \)
compute \( \delta_{ki} \) and hence \( s_{mi} \) from (11.15)
put these into (11.13)
the coefficients \((\alpha_{m\ell}, \beta_{km})\) are usually called weights

the algorithm is called back propagation or the \(\delta\)-rule

can be computed in time linear in the number of hidden units

can be processed one instance (case) at a time

any continuous function can be represented this way (with enough \(Z\)'s)
Training NNs (§11.5)

- with small $\alpha_{m\ell}$, $\sigma(v) \approx v$; large linear regression
- if algorithm stops early, $\alpha_{m\ell}$ still small; fit ‘nearly’ linear or shrunk towards a linear fit
- use penalty as in ridge regression to avoid overfitting
- $\min R(\theta) + \lambda J(\theta)$
- $J(\theta) = \sum \beta_{km}^2 + \sum \alpha_{m\ell}^2$
- as in ridge regression need to scale inputs to mean 0, var 1 (at least approx.)
- $\lambda$ called weight decay parameter; seems to be more crucial than the number of hidden units
- nnet in MASS library
- regression examples: §11.6, simulated, also cpus data from MASS
- classification examples: Figure 11.4 and Figures 2.1-4