Adaptive Basis Function Models

ddebarr@uw.edu

2016-06-02

$$f(\mathbf{x}) = w_0 + \sum_{m=1}^{M} w_m \phi_m(\mathbf{x})$$

Agenda

- Classification And Regression Trees (CART)
- Generalized Additive Models (GAM)
- Boosting
- Multi-Layer Perceptron (MLP) Neural Networks
- Ensemble Learning
- Experimental Comparison
- Interpreting Black-Box Models

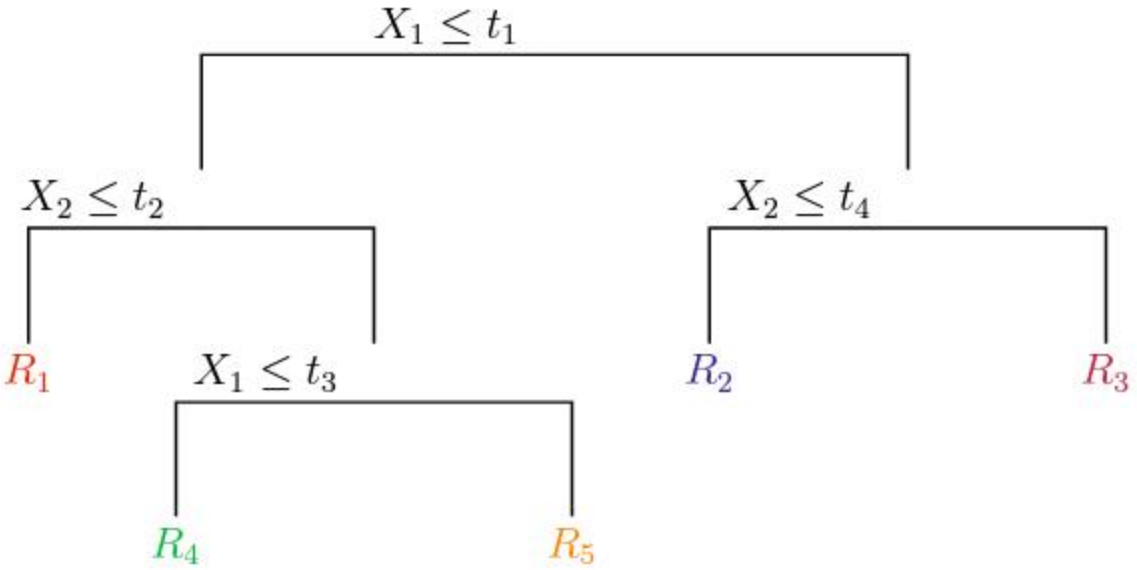
Recursive Partitioning

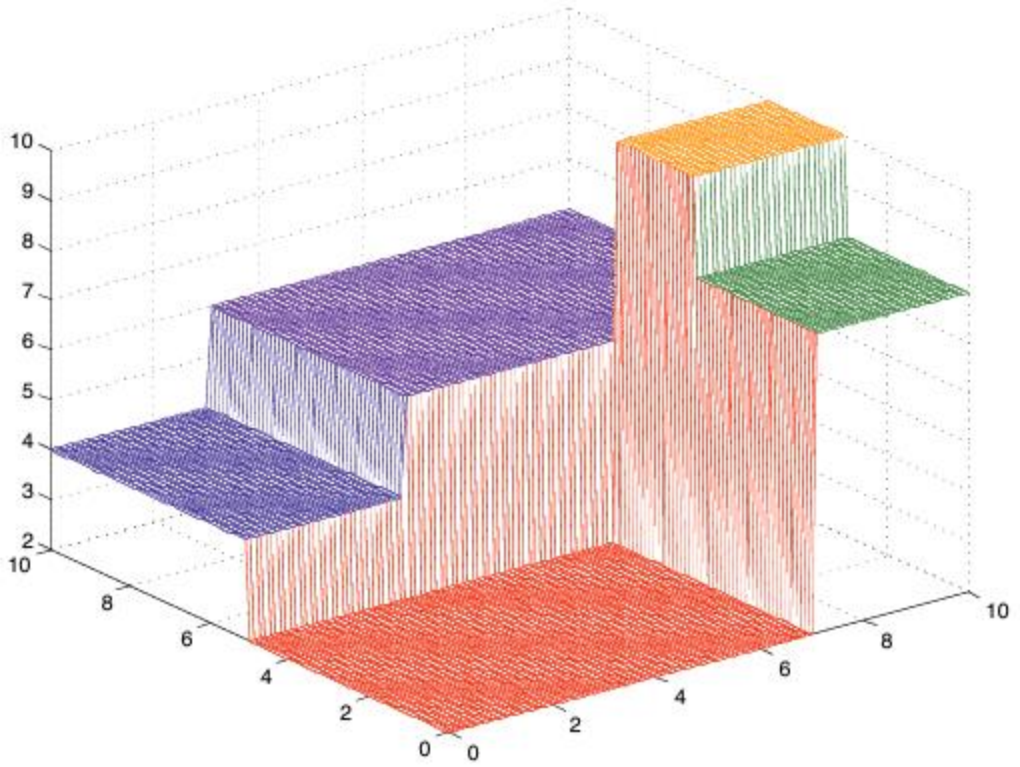
Algorithm 16.1: Recursive procedure to grow a classification/ regression tree

```
1  function fitTree(node, 𝒟, depth) ;
2  node.prediction = mean(y_i : i ∈ 𝒟) // or class label distribution ;
3  (j*, t*, 𝒟_L, 𝒟_R) = split(𝒟);
4  if not worthSplitting(depth, cost, 𝒟_L, 𝒟_R) then
5      return node
6  else
7      node.test = λx.x_{j*} < t* // anonymous function;
8      node.left = fitTree(node, 𝒟_L, depth+1);
9      node.right = fitTree(node, 𝒟_R, depth+1);
10     return node;
```

$$(j^*, t^*) = \arg \min_{j \in \{1,\ldots,D\}} \min_{t \in \mathcal{T}_j} \text{cost}(\{\mathbf{x}_i, y_i : x_{ij} \leq t\}) + \text{cost}(\{\mathbf{x}_i, y_i : x_{ij} > t\})$$

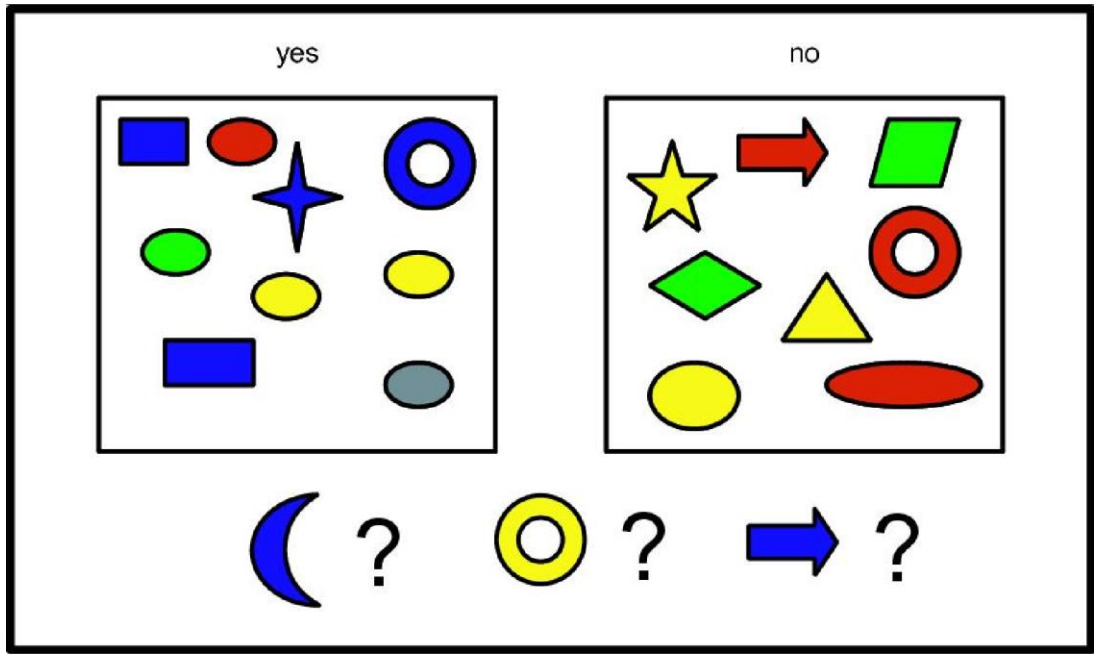
Regression Tree Example

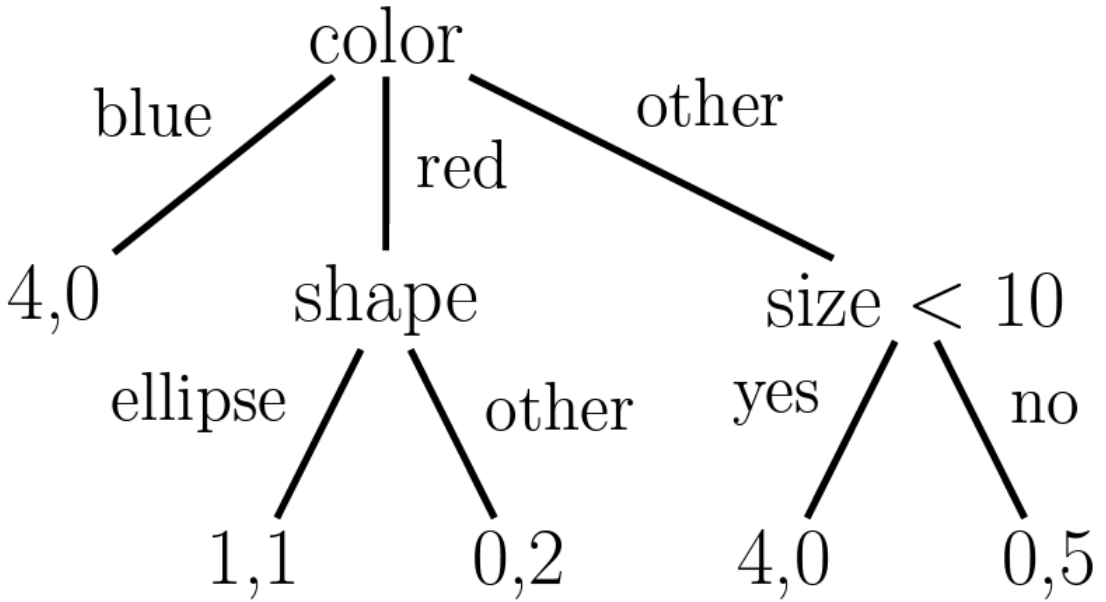




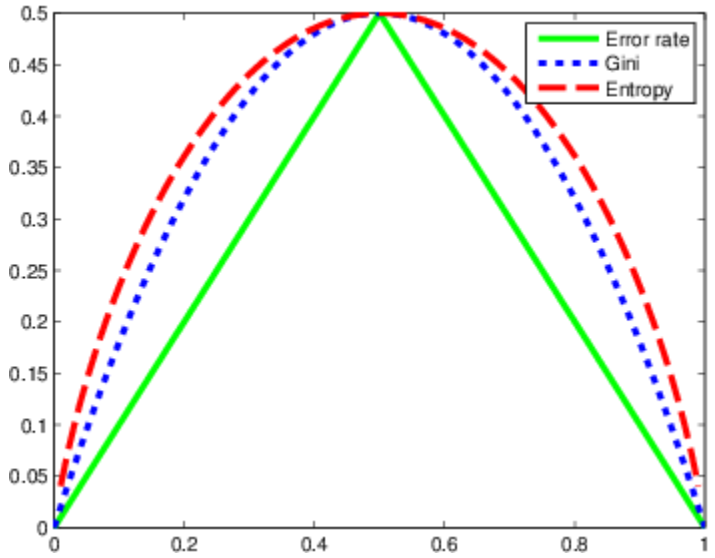
$$f(\mathbf{x}) = \mathbb{E}\left[y|\mathbf{x}\right] = \sum_{m=1}^{M} w_m \mathbb{I}(\mathbf{x} \in R_m) = \sum_{m=1}^{M} w_m \phi(\mathbf{x}; \mathbf{v}_m)$$

Classification Tree Example



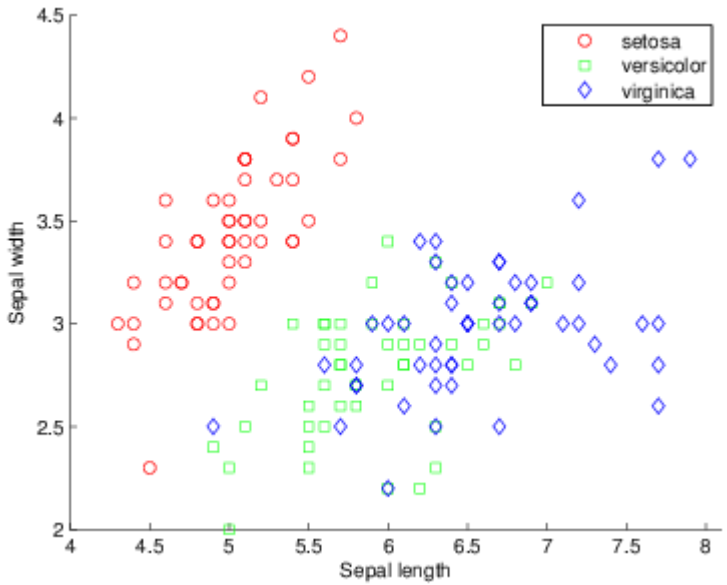


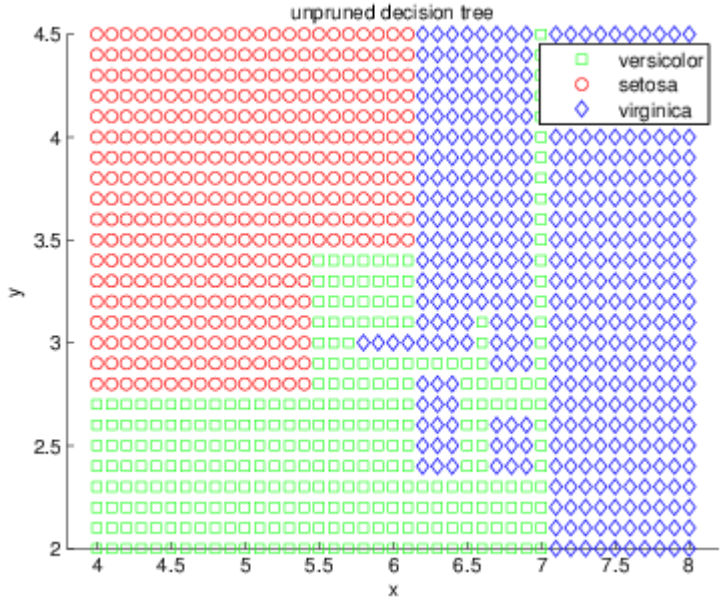
Impurity Measures for Binary Classification



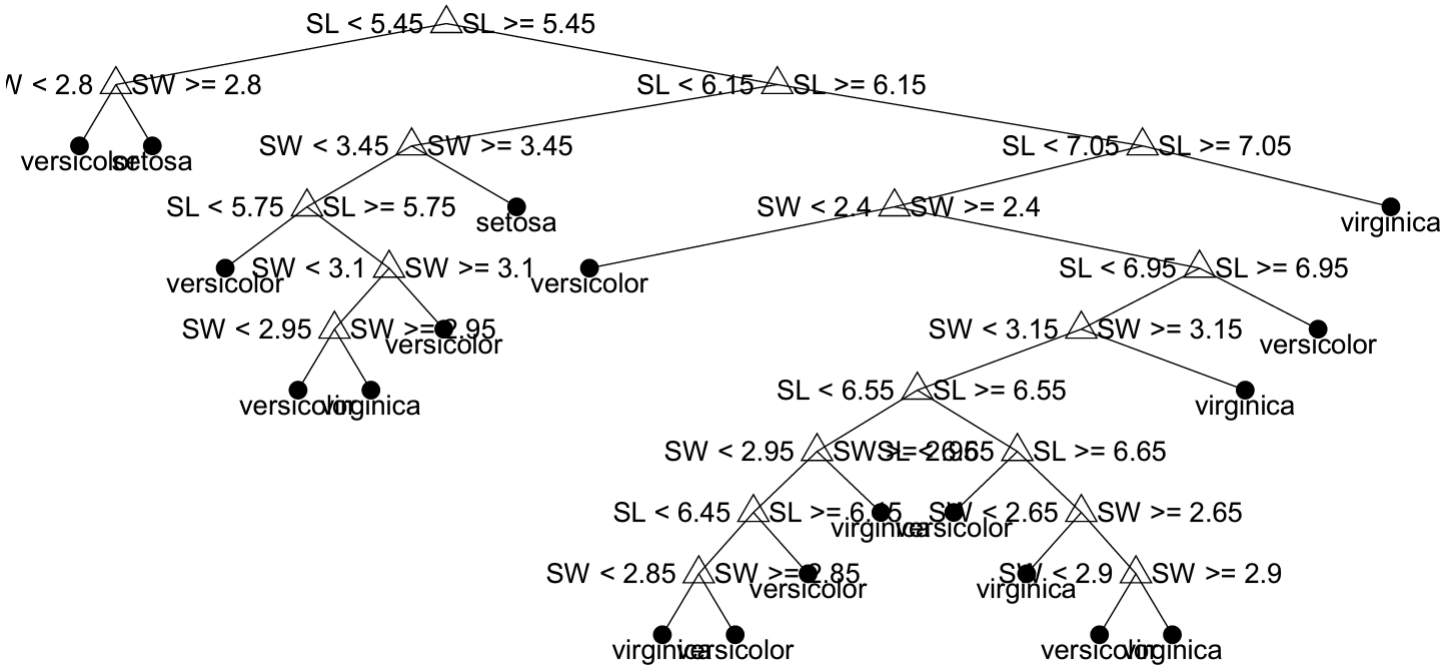
$$\sum_{c=1}^{C} \hat{\pi}_c(1-\hat{\pi}_c) = \sum_c \hat{\pi}_c - \sum_c \hat{\pi}_c^2 = 1 - \sum_c \hat{\pi}_c^2$$

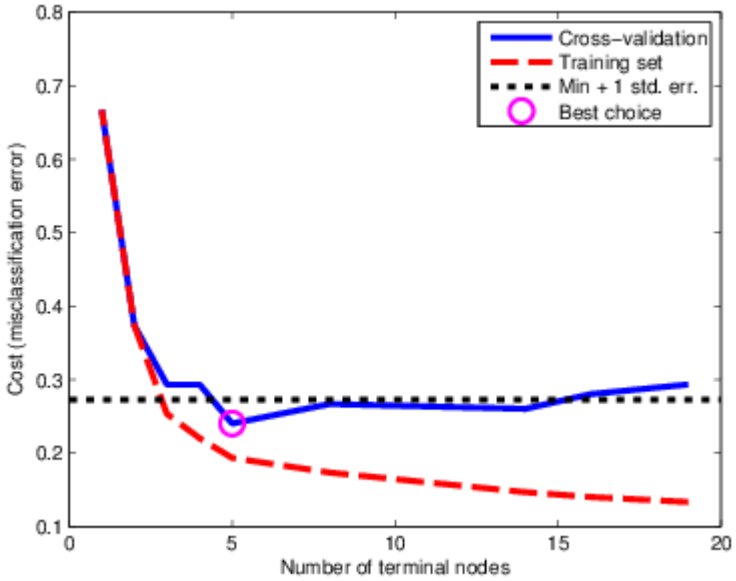
Iris Classification Example



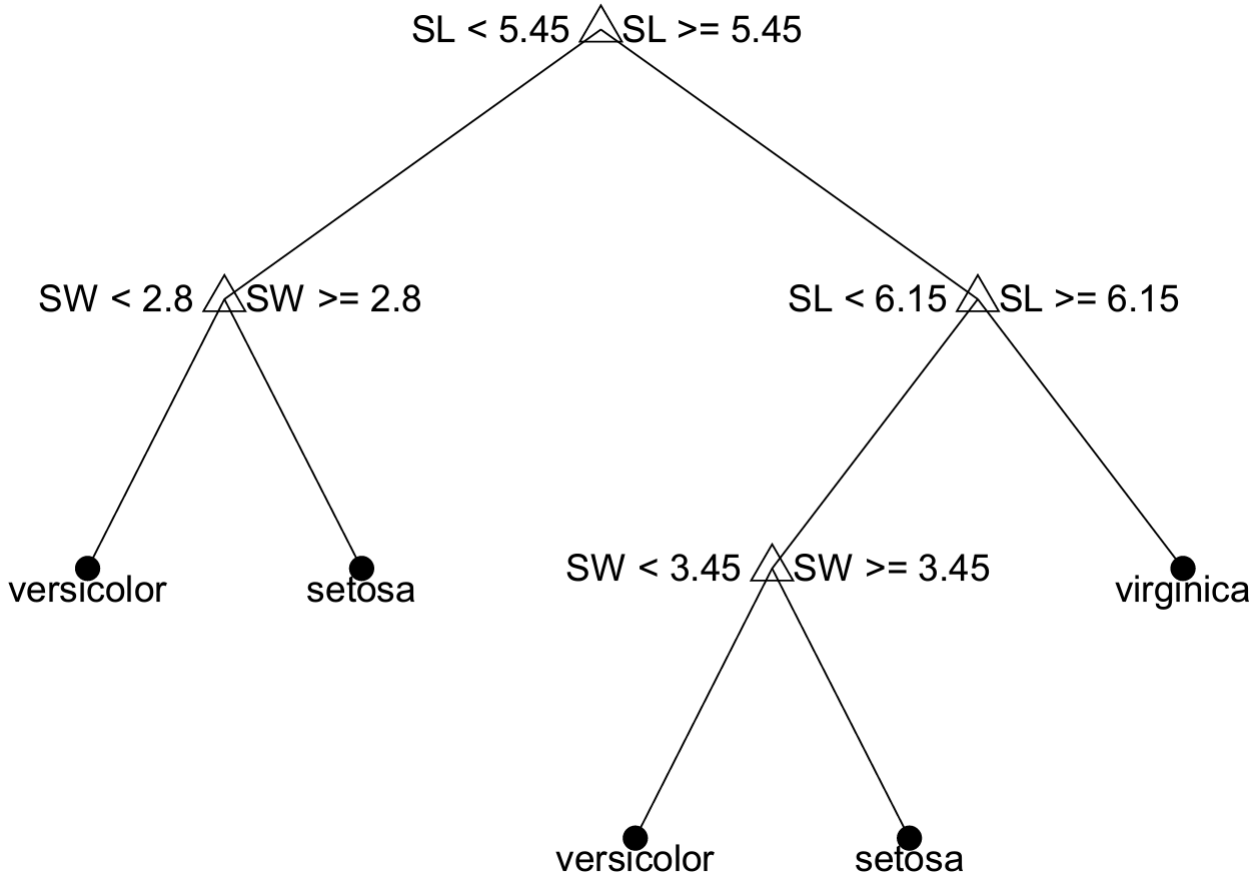


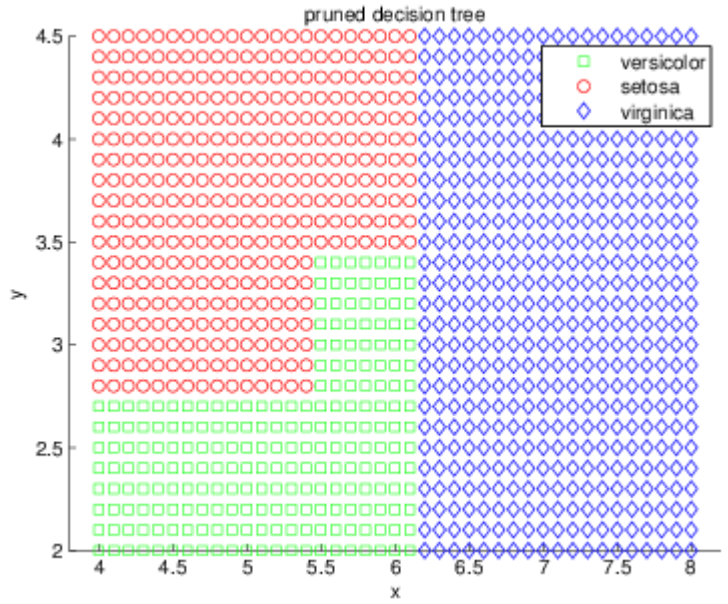
Iris Classification Example: Without Pruning





Iris Classification Example: With Pruning





Random Forests

1. If the number of cases in the training set is N, sample N cases at random - but *with replacement*, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

$$f(\mathbf{x}) = \sum_{m=1}^{M} \frac{1}{M} f_m(\mathbf{x})$$

Multiple Adaptive Regression Splines (MARS)

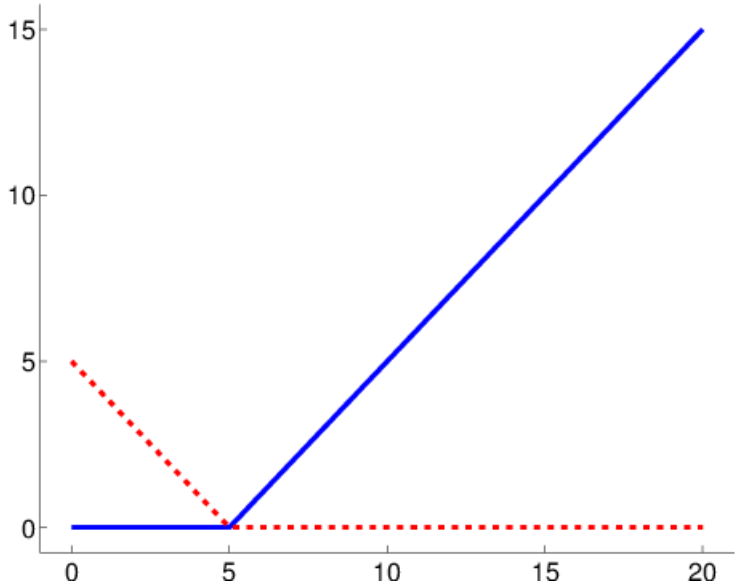
Each basis function takes one of the following three forms:

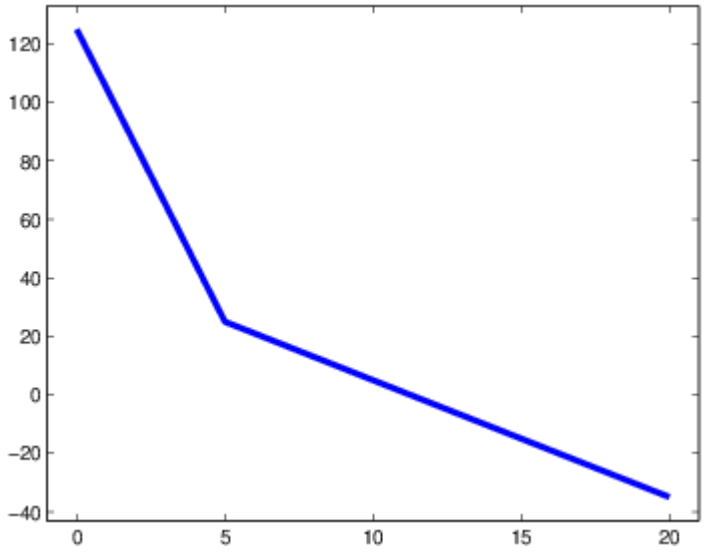
1. a constant 1. There is just one such term, the intercept. In the ozone formula above, the intercept term is 5.2.
2. a hinge function. A hinge function has the form max(0, x − const) or max (0, const − x). MARS automatically selects variables and values of those variables for knots of the hinge functions. Examples of such basis functions can be seen in the middle three lines of the ozone formula.
3. a product of two or more hinge functions. These basis functions can model interaction between two or more variables. An example is the last line of the ozone formula.

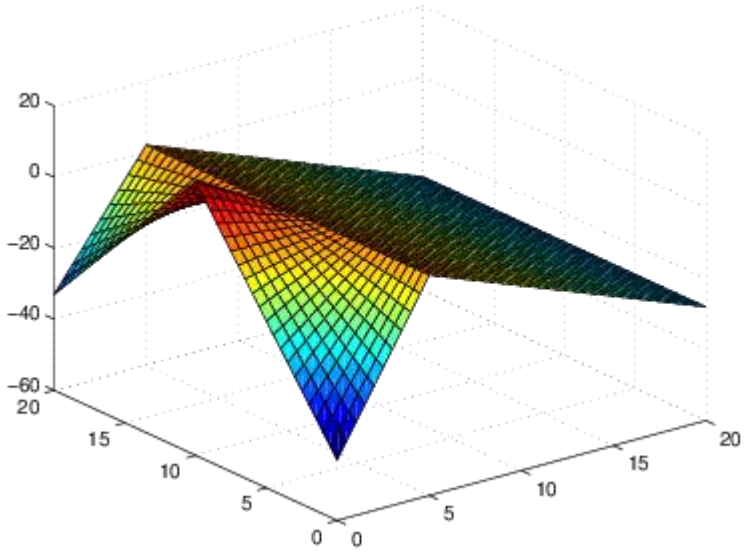
ozone = 5.2 + 0.93 * max(0, temp - 58) - 0.64 * max(0, temp - 68) - 0.046 * max(0, 234 - ibt) - 0.016 * max(0, wind - 7) * max(0, 200 - vis)

Multivariate Adaptive Regression Splines (MARS)

$$f(\mathbf{x}) = 25 - 4(x_1 - 5)_+ + 20(5 - x_1)_+$$







$$\begin{aligned} f(\mathbf{x}) \quad &= \quad = 2 - 2(x_1 - 5)_+ + 3(5 - x_1)_+ \\ &\quad -(x_2 - 10)_+ \times (5 - x_1) + -1.2(10 - x_2)_+ \times (5 - x_1)_+ \end{aligned}$$

Boosting

Name	Loss	Derivative	f^*	Algorithm
Squared error	$\frac{1}{2}(y_i - f(\mathbf{x}_i))^2$	$y_i - f(\mathbf{x}_i)$	$\mathbb{E}[y\|\mathbf{x}_i]$	L2Boosting
Absolute error	$\|y_i - f(\mathbf{x}_i)\|$	$\text{sgn}(y_i - f(\mathbf{x}_i))$	$\text{median}(y\|\mathbf{x}_i)$	Gradient boosting
Exponential loss	$\exp(-\tilde{y}_i f(\mathbf{x}_i))$	$-\tilde{y}_i \exp(-\tilde{y}_i f(\mathbf{x}_i))$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	AdaBoost
Logloss	$\log(1 + e^{-\tilde{y}_i f_i})$	$y_i - \pi_i$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	LogitBoost

Adaptive Boosting (AdaBoost)

Algorithm 16.2: Adaboost.M1, for binary classification with exponential loss

1 $w_i = 1/N$;

2 **for** $m = 1 : M$ **do**

3 Fit a classifier $\phi_m(\mathbf{x})$ to the training set using weights $\mathbf{w}$;

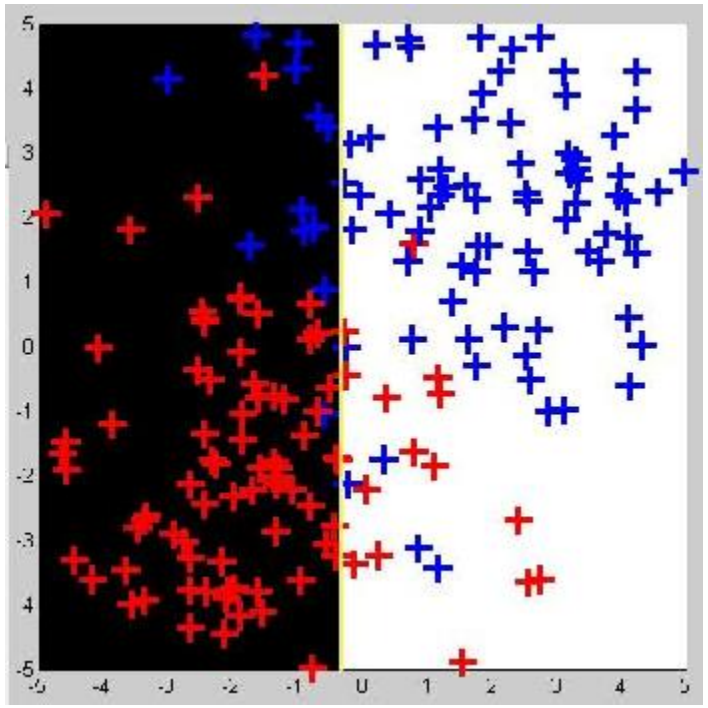
4 Compute $\text{err}_m = \frac{\sum_{i=1}^{N} w_{i,m} \mathbb{I}(\tilde{y}_i \neq \phi_m(\mathbf{x}_i))}{\sum_{i=1}^{N} w_{i,m})}$;

5 Compute $\alpha_m = \log[(1 - \text{err}_m)/\text{err}_m]$;

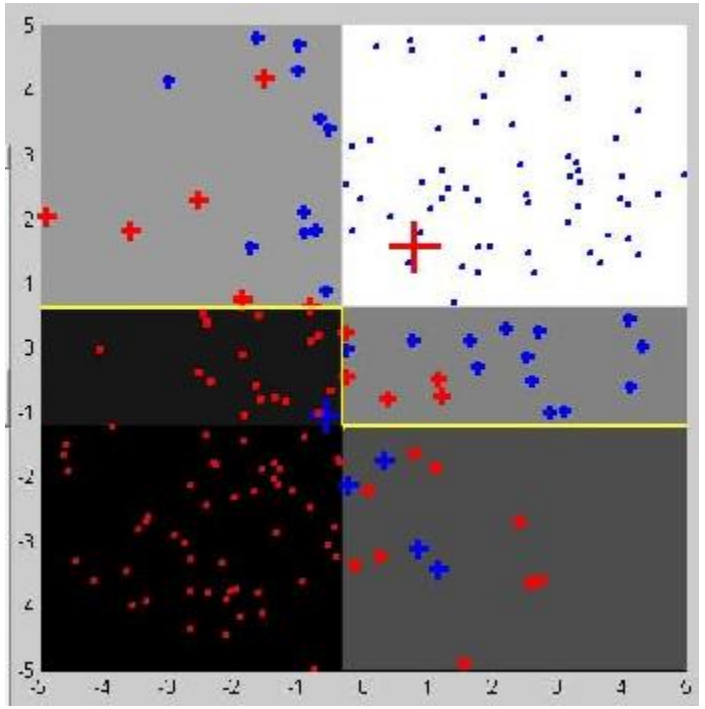
6 Set $w_i \leftarrow w_i \exp[\alpha_m \mathbb{I}(\tilde{y}_i \neq \phi_m(\mathbf{x}_i))]$;

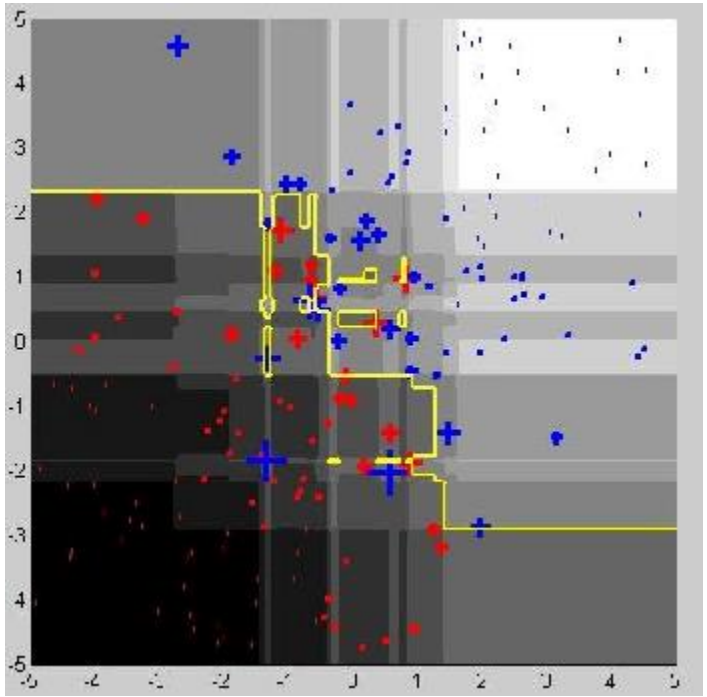
7 Return $f(\mathbf{x}) = \text{sgn}\left[\sum_{m=1}^{M} \alpha_m \phi_m(\mathbf{x})\right]$;

AdaBoosting a Diagonal Decision Boundary



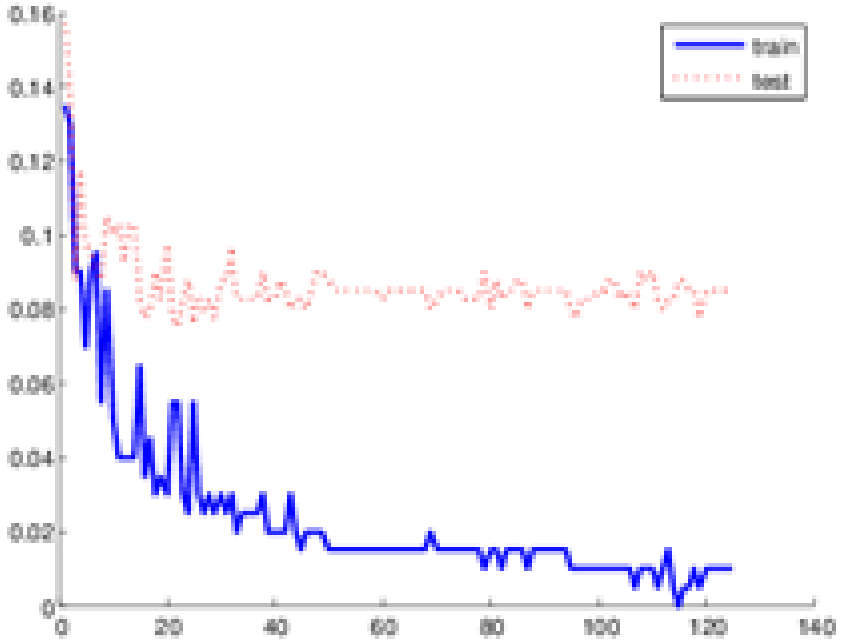
Decision stump
[size reflects weight]





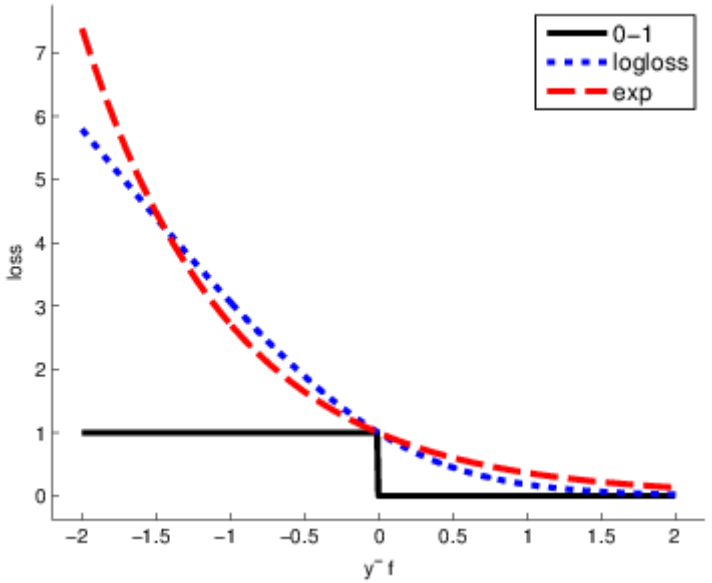
A more complex boundary

AdaBoost Performance



AdaBoost helps to resist overfitting,
where overfitting is defined as an increase in test error
while train error continues to decrease

Loss Functions for Binary Classification



Notice that the exponential loss function punishes more severely as we move to the left [it's a bit more sensitive to outliers]

LogitBoost

Algorithm 16.3: LogitBoost, for binary classification with log-loss

$w_i = 1/N$, $\pi_i = 1/2$;

for $m = 1 : M$ **do**

  Compute the working response $z_i = \frac{y_i^* - \pi_i}{\pi_i(1-\pi_i)}$;

  Compute the weights $w_i = \pi_i(1 - \pi_i)$;

  $\phi_m = \text{argmin}_\phi \sum_{i=1}^N w_i(z_i - \phi(\mathbf{x}_i))^2$;

  Update $f(\mathbf{x}) \leftarrow f(\mathbf{x}) + \frac{1}{2}\phi_m(\mathbf{x})$;

  Compute $\pi_i = 1/(1 + \exp(-2f(\mathbf{x}_i)))$;

Return $f(\mathbf{x}) = \text{sgn}\left[\sum_{m=1}^M \phi_m(\mathbf{x})\right]$;

Taylor Series Approximation

Taylor Series approximation

$$g(x+\delta) \approx g(x) + g'(x)\big((x+\delta) - x\big)$$

We want to add δ so the gradient of the negative log likelihood function will be zero, which gives us:

$$\begin{aligned} g(x) + g'(x)\big((x+\delta) - x\big) &\approx 0 \\ g'(x)\big((x+\delta) - x\big) &\approx -g(x) \\ \big((x+\delta) - x\big) &\approx \frac{-g(x)}{g'(x)} \\ \delta &\approx \frac{-g(x)}{g'(x)} \end{aligned}$$

First Derivative (Gradient) of Log Loss

$$-\frac{\partial}{\partial f(\mathbf{x})}\ln\left(1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)\right)$$

$$=-\frac{1}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\frac{\partial}{\partial f(\mathbf{x})}\left(1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)\right)=-\frac{1}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\left(\frac{\partial}{\partial f(\mathbf{x})}(1)+\frac{\partial}{\partial f(\mathbf{x})}\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)\right)$$

$$=-\frac{1}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\left(0+\frac{\partial}{\partial f(\mathbf{x})}\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)\right)=-\frac{1}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\left(\frac{\partial}{\partial f(\mathbf{x})}\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)\right)$$

$$=-\frac{1}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\left(\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)\right)\left(\frac{\partial}{\partial f(\mathbf{x})}\left(-\tilde{y}_i f(\mathbf{x}_i)\right)\right)=-\frac{\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\left(\frac{\partial}{\partial f(\mathbf{x})}\left(-\tilde{y}_i f(\mathbf{x}_i)\right)\right)$$

$$=-\frac{\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\left(-\tilde{y}_i\right)=\frac{\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\left(\tilde{y}_i\right)$$

$$=\frac{1}{\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1}\left(\tilde{y}_i\right)=\left(1-\frac{1}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\right)\left(\tilde{y}_i\right)$$

$$=y_i-\frac{1}{1+\exp\left(-f(\mathbf{x}_i)\right)}$$

$$\tilde{y}_i\in\{-1,1\};\quad y_i=\frac{\tilde{y}_i+1}{2}$$

Second Derivative of Log Loss

$$\frac{\partial^2}{\partial f(\mathbf{x})^2}\ln\left(1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)\right)$$

$$=\frac{\partial}{\partial f(\mathbf{x})}\left(-\frac{\tilde{y}_i}{\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1}\right)=-\tilde{y}_i\frac{\partial}{\partial f(\mathbf{x})}\left(\frac{1}{\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1}\right)$$

$$=\frac{\tilde{y}_i}{\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)^2}\frac{\partial}{\partial f(\mathbf{x})}\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)=\frac{\tilde{y}_i}{\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)^2}\left(\frac{\partial}{\partial f(\mathbf{x})}\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+\frac{\partial}{\partial f(\mathbf{x})}(1)\right)$$

$$=\frac{\tilde{y}_i}{\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)^2}\left(\frac{\partial}{\partial f(\mathbf{x})}\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+0\right)=\frac{\tilde{y}_i}{\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)^2}\left(\frac{\partial}{\partial f(\mathbf{x})}\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)\right)$$

$$=\frac{\tilde{y}_i}{\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)^2}\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)\frac{\partial}{\partial f(\mathbf{x})}\left(\tilde{y}_i f(\mathbf{x}_i)\right)\right)=\frac{\tilde{y}_i}{\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)^2}\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)\tilde{y}_i\right)$$

$$=\frac{\tilde{y}_i^2\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)}{\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)^2}=\frac{\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)}{\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)^2}=\frac{\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)}{\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)}\left(\frac{1}{\left(\exp\left(\tilde{y}_i f(\mathbf{x}_i)\right)+1\right)}\right)$$

$$=\frac{1}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\left(1-\frac{1}{1+\exp\left(-\tilde{y}_i f(\mathbf{x}_i)\right)}\right)=\frac{1}{1+\exp\left(-f(\mathbf{x}_i)\right)}\left(1-\frac{1}{1+\exp\left(-f(\mathbf{x}_i)\right)}\right)$$

$$=\hat{p}_i\left(1-\hat{p}_i\right)$$

Working Response for LogitBoost

Compute the working response $z_i = \frac{y_i^* - \pi_i}{\pi_i(1-\pi_i)}$

This is simply the ratio of the negative gradient to the second derivative

$$\frac{-g(x)}{g'(x)}$$

We're using a Taylor Series approximation to estimate the value needed to optimally "boost" (improve) log loss

Gradient Boosting

Algorithm 16.4: Gradient boosting

Initialize $f_0(\mathbf{x}) = \text{argmin}_{\boldsymbol{\gamma}} \sum_{i=1}^{N} L(y_i, \phi(\mathbf{x}_i; \boldsymbol{\gamma}))$;

for $m = 1 : M$ **do**

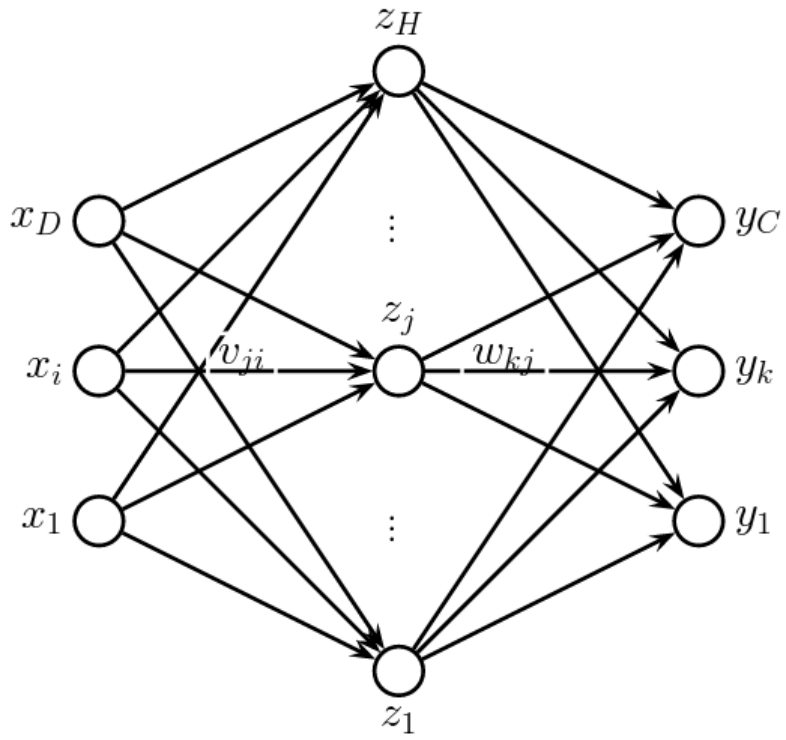
  Compute the gradient residual using $r_{im} = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f(\mathbf{x}_i)=f_{m-1}(\mathbf{x}_i)}$;

  Use the weak learner to compute $\boldsymbol{\gamma}_m$ which minimizes $\sum_{i=1}^{N}(r_{im} - \phi(\mathbf{x}_i; \boldsymbol{\gamma}_m))^2$;

  Update $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu\phi(\mathbf{x}; \boldsymbol{\gamma}_m)$;

Return $f(\mathbf{x}) = f_M(\mathbf{x})$

Neural Network with 1 Hidden Layer

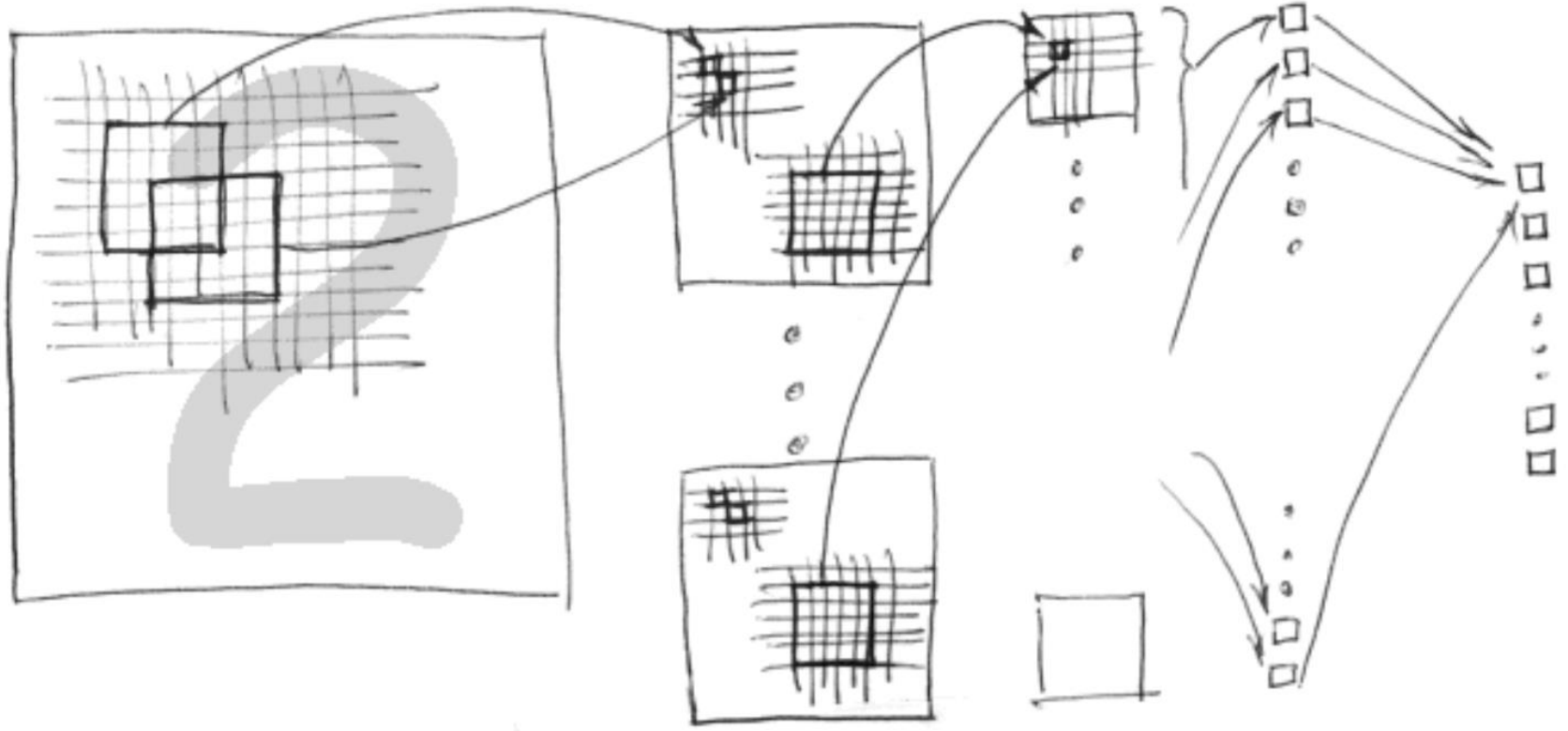


$$\delta_{nj}^{v} = \sum_{k=1}^{K} \delta_{nk}^{w} w_{kj} g'(a_{nj})$$

$$\delta_{nk}^{w} = (\hat{y}_{nk} - y_{nk})$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = [\sum_{n} \boldsymbol{\delta}_{n}^{v} \mathbf{x}_{n}; \sum_{n} \boldsymbol{\delta}_{n}^{w} \mathbf{z}_{n}]$$

Convolutional Neural Network for MNIST Task



Input Layer
29x29

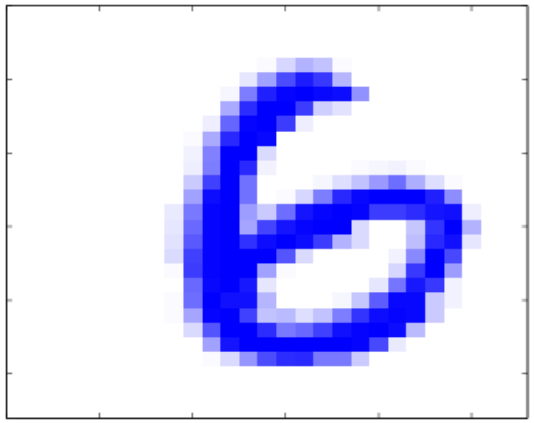
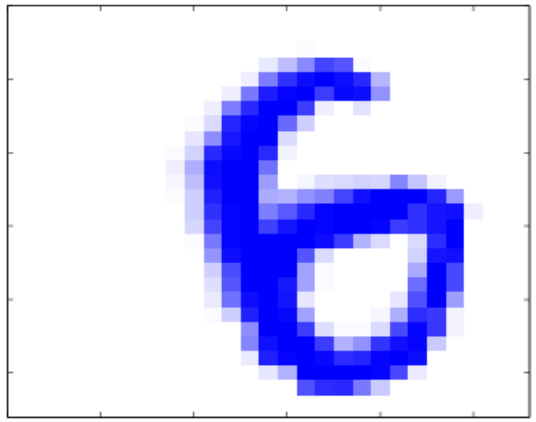
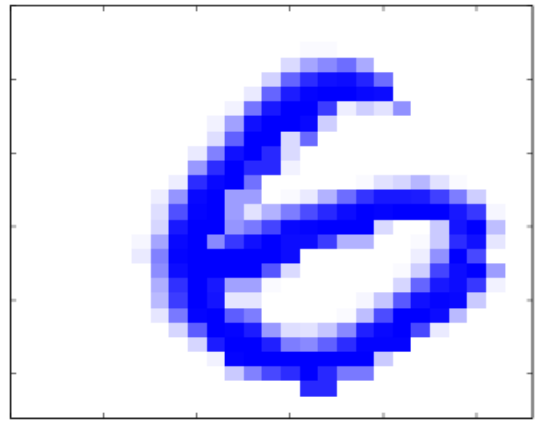
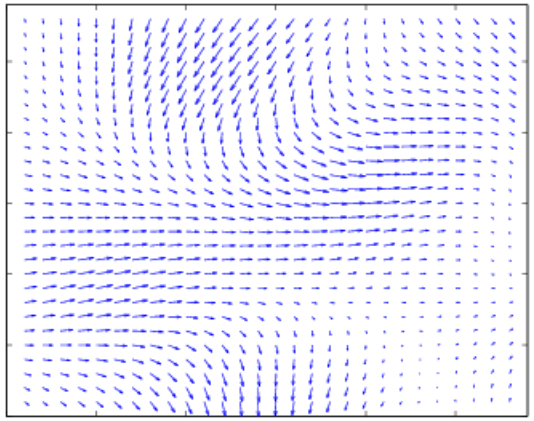
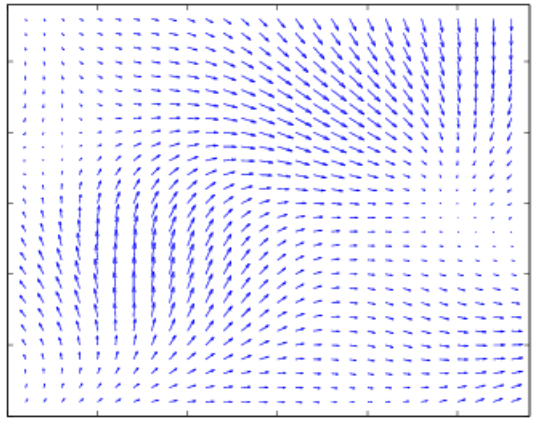
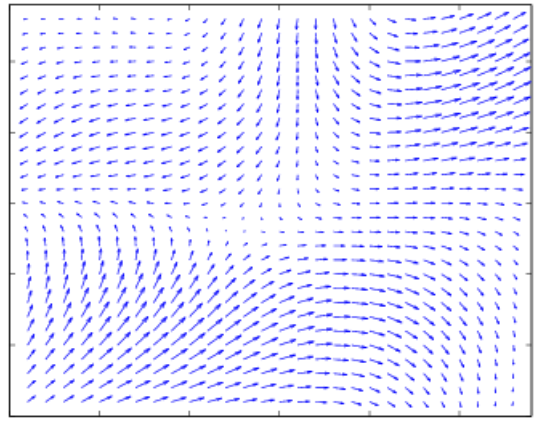
Layer #1
6 Feature Maps
Each 13x13

Layer #2
50 Feature
Maps
Each 5x5

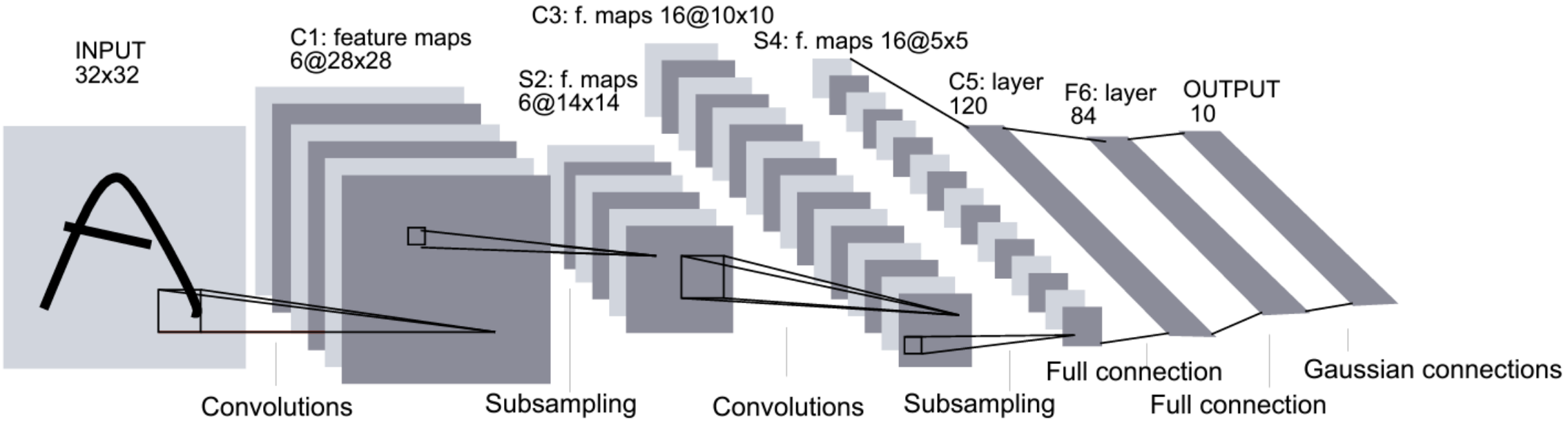
Layer #3
Fully
Connected
100
Neurons

Layer #4
Fully
Connected
10 Neurons

Warpings of a Handwritten Digit

LeNet 5 Convolutional Neural Net

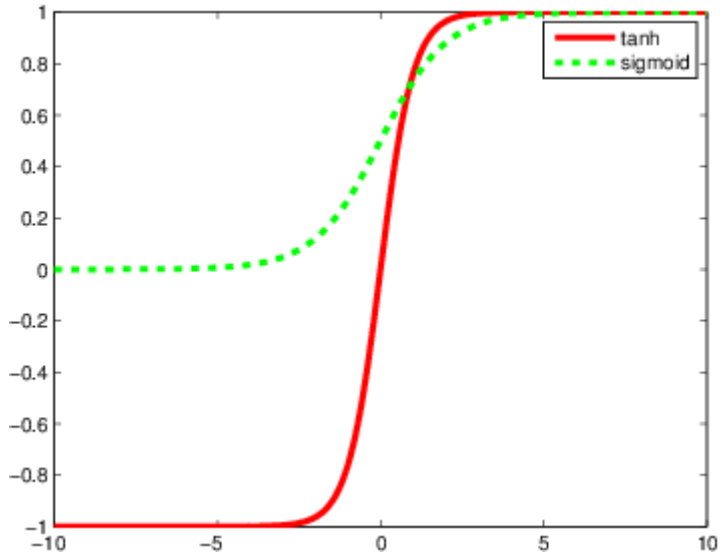


Web-based demo: http://yann.lecun.com/exdb/lenet/index.html
Convolutional Neural Networks are a variant of the Mutli-Layer Perceptron.
Created by Yann LeCun [yawn le cuh] et al: previously at AT&T.
Their system was eventually deployed by the United States Postal Service

Errors Made by LeNet 5

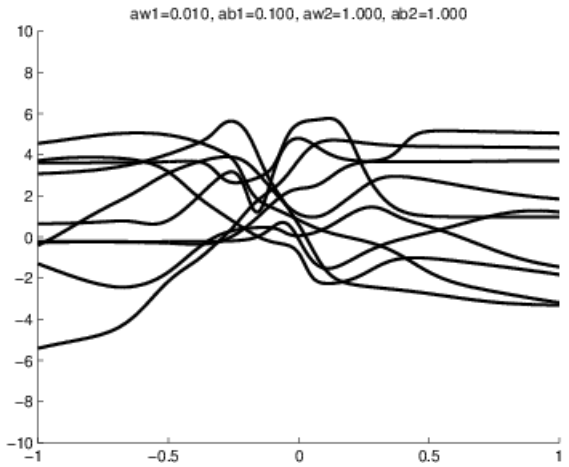
82 errors out of 10,000 test images

Activation Function Examples

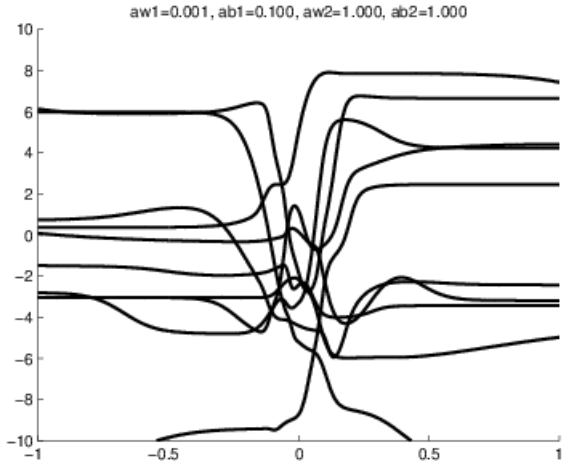


The logistic and hyperbolic tangent functions have a similar shape; but different output ranges: (-1, 1) versus (0, 1)

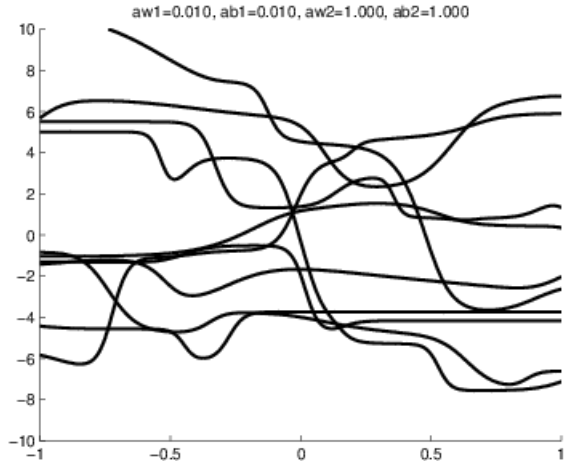
Effect of Changing Regularization Parameters



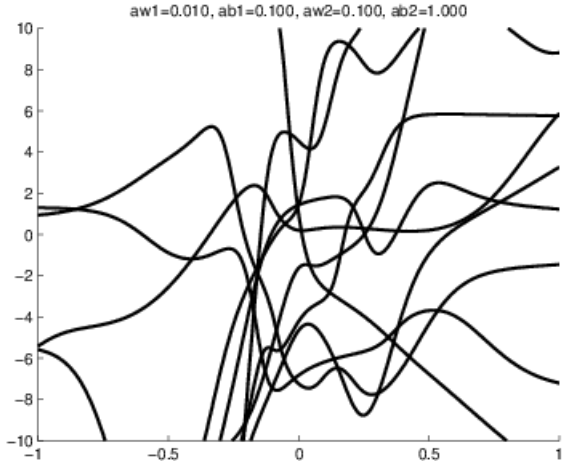
Baseline for regularization of hidden layer weights, hidden layer bias, output layer weights, and output layer bias



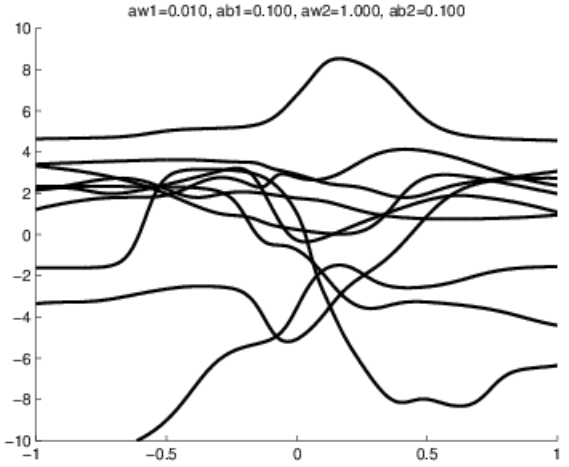
Reduced regularization of hidden layer weights [steeper curves]



Reduced regularization of hidden layer bias [curves shifting horizontally]

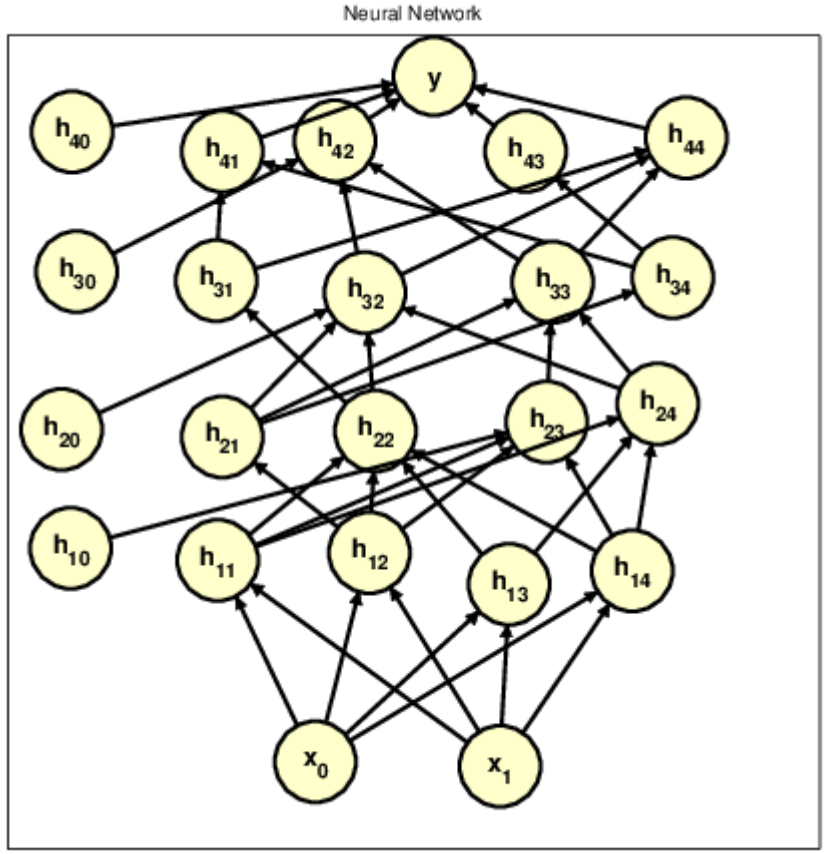


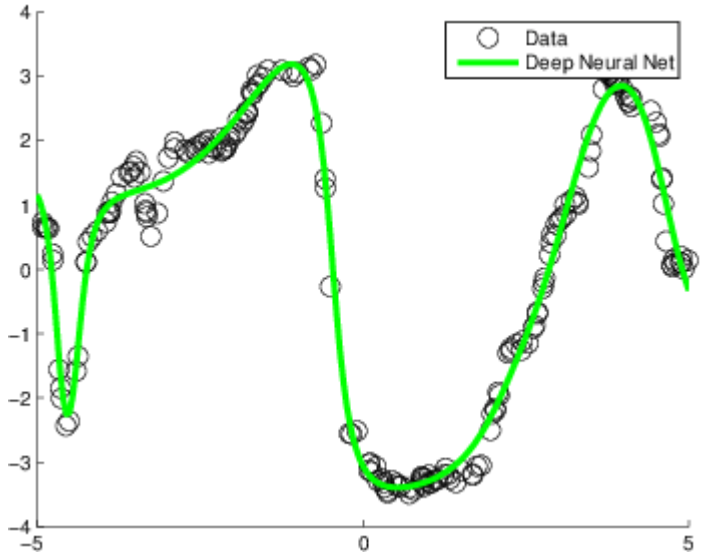
Reduced regularization of output layer weights [way more wiggly/sensitive]



Reduced regularization of output layer bias [curves shifting vertically]

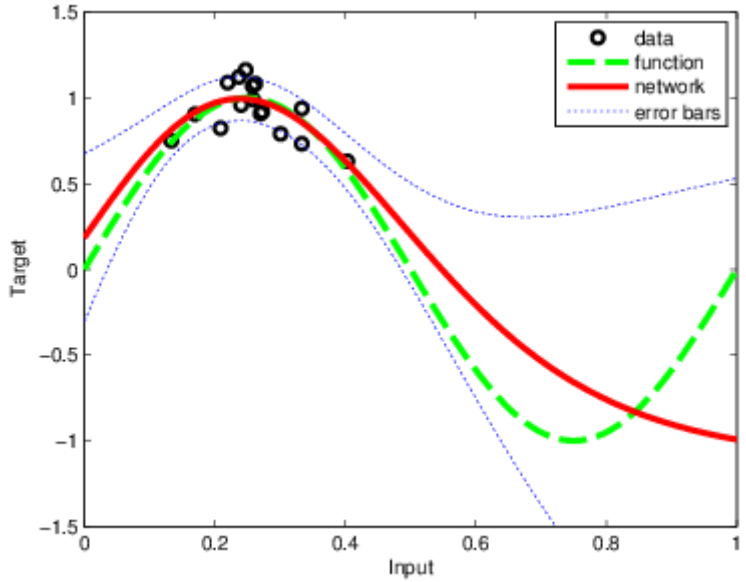
Deep but Sparse Neural Network

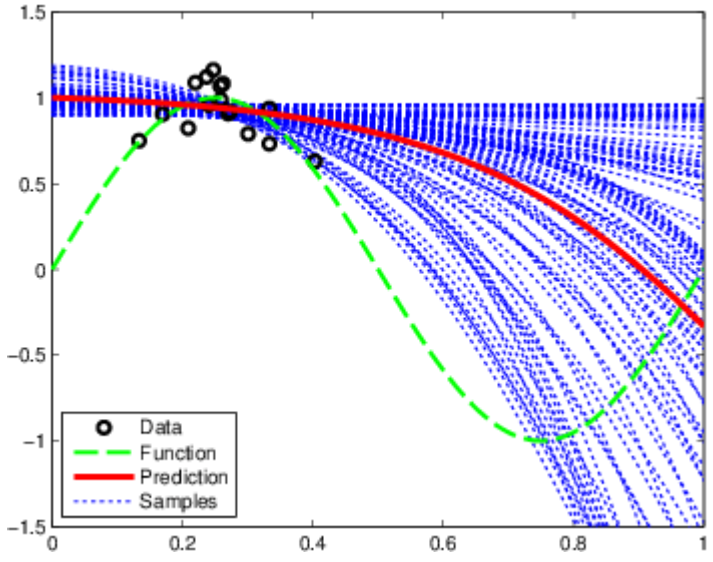




Not every node in layer “i” is connected to every node in layer “i” + 1

Posterior Predictive Density for an MLP





$$p(y|\mathbf{x}, \mathcal{D}, \alpha, \beta) \approx \mathcal{N}(y|f(\mathbf{x}, \mathbf{w}_{MP}), \sigma^2(\mathbf{x}))$$

Ensemble Learning

- Stacking: using the output of one model as an input to another model
- Blending: using the outputs of several models as inputs to another model
- Do *NOT* use the same training data for both “layers”

$$f(y|\mathbf{x}, \boldsymbol{\pi}) = \sum_{m \in \mathcal{M}} w_m f_m(y|\mathbf{x})$$

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} w_m \hat{f}_m^{-i}(\mathbf{x}))$$

Experimental Comparison: Take 1

MODEL	1ST	2ND	3RD	4TH	5TH	6TH	7TH	8TH	9TH	10TH
BST-DT	0.580	0.228	0.160	0.023	0.009	0.000	0.000	0.000	0.000	0.000
RF	0.390	0.525	0.084	0.001	0.000	0.000	0.000	0.000	0.000	0.000
BAG-DT	0.030	0.232	0.571	0.150	0.017	0.000	0.000	0.000	0.000	0.000
SVM	0.000	0.008	0.148	0.574	0.240	0.029	0.001	0.000	0.000	0.000
ANN	0.000	0.007	0.035	0.230	0.606	0.122	0.000	0.000	0.000	0.000
KNN	0.000	0.000	0.000	0.009	0.114	0.592	0.245	0.038	0.002	0.000
BST-STMP	0.000	0.000	0.002	0.013	0.014	0.257	0.710	0.004	0.000	0.000
DT	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.616	0.291	0.089
LOGREG	0.000	0.000	0.000	0.000	0.000	0.000	0.040	0.312	0.423	0.225
NB	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.030	0.284	0.686

BST-DT: boosted decision trees
RF: random forest
BAG-DT: bagged decision trees
SVM: support vector machine
ANN: artificial neural network
KNN: K-nearest neighbors

They used 11 different data sets, with 5000 training cases and around 10000 testing cases for each data set. The number of features ranged from 9 to 200. Five-fold cross validation was used to assess average test error. Separate ("internal") cross validation was used for model selection.

Experimental Comparison: Take 2

Dataset	Domain	Type	D	% probes	N_{train}	N_{val}	N_{test}
Aracene	Mass spectrometry	Dense	10,000	30	100	100	700
Dexter	Text classification	Sparse	20,000	50	300	300	2000
Dorothea	Drug discovery	Sparse	100,000	50	800	350	800
Gisette	Digit recognition	Dense	5000	30	6000	1000	6500
Madelon	Artificial	Dense	500	96	2000	600	1800

	Screened features		ARD	
Method	Avg rank	Avg time	Avg rank	Avg time
HMC MLP	1.5	384 (138)	1.6	600 (186)
Boosted MLP	3.8	9.4 (8.6)	2.2	35.6 (33.5)
Bagged MLP	3.6	3.5 (1.1)	4.0	6.4 (4.4)
Boosted trees	3.4	3.03 (2.5)	4.0	34.1 (32.4)
Random forests	2.7	1.9 (1.7)	3.2	11.2 (9.3)

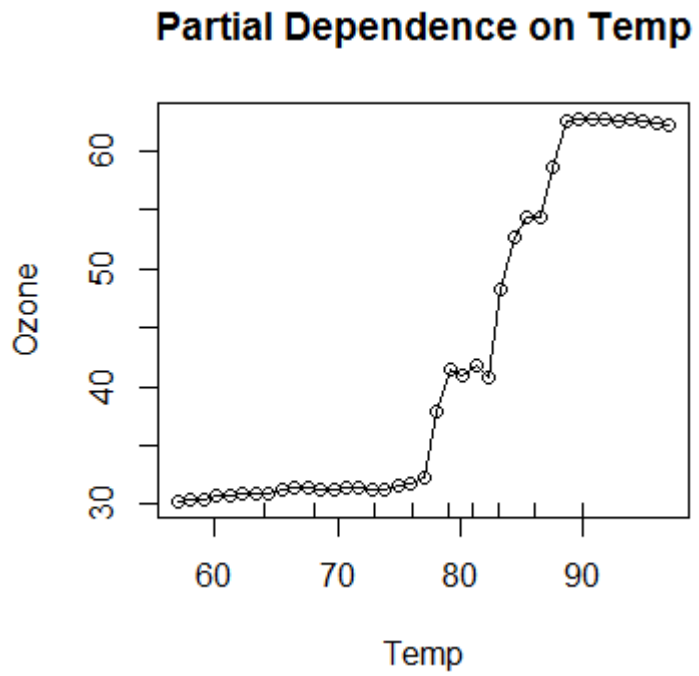
Measuring Error Reduction for Variable Importance

```
> library(randomForest)
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.
> data(airquality)
> airquality <- na.omit(airquality)
> airquality <- airquality[sample(1:nrow(airquality)),]
> library(gbm)
Loading required package: survival
Loading required package: lattice
Loading required package: splines
Loading required package: parallel
Loaded gbm 2.1.1
> ozone.gbm <- gbm(Ozone ~ ., data = airquality, n.trees = 2,
+                  interaction.depth = 1, shrinkage = 1.0,
+                  bag.fraction = 1)
Distribution not specified, assuming gaussian ...
> # Parameters for demo only!  Use validation for optimization!
>
> predictions <- predict(ozone.gbm, airquality, n.trees = 0)
> mse0 <- mean((airquality$Ozone - predictions) ^ 2)
> predictions <- predict(ozone.gbm, airquality, n.trees = 1)
> mse1 <- mean((airquality$Ozone - predictions) ^ 2)
> predictions <- predict(ozone.gbm, airquality, n.trees = 2)
> mse2 <- mean((airquality$Ozone - predictions) ^ 2)
```

```
> ozone.gbm$train.error
[1] 565.7910 434.1239
> mse1
[1] 565.791
> mse2
[1] 434.1239
>
> summary(ozone.gbm)
             var  rel.inf
Temp        Temp 80.14641
Wind        Wind 19.85359
Solar.R Solar.R  0.00000
Month     Month  0.00000
Day         Day  0.00000
> 100 * (mse1 - mse0) / (mse2 - mse0)
[1] 80.14641
> 100 * (mse2 - mse1) / (mse2 - mse0)
[1] 19.85359
```

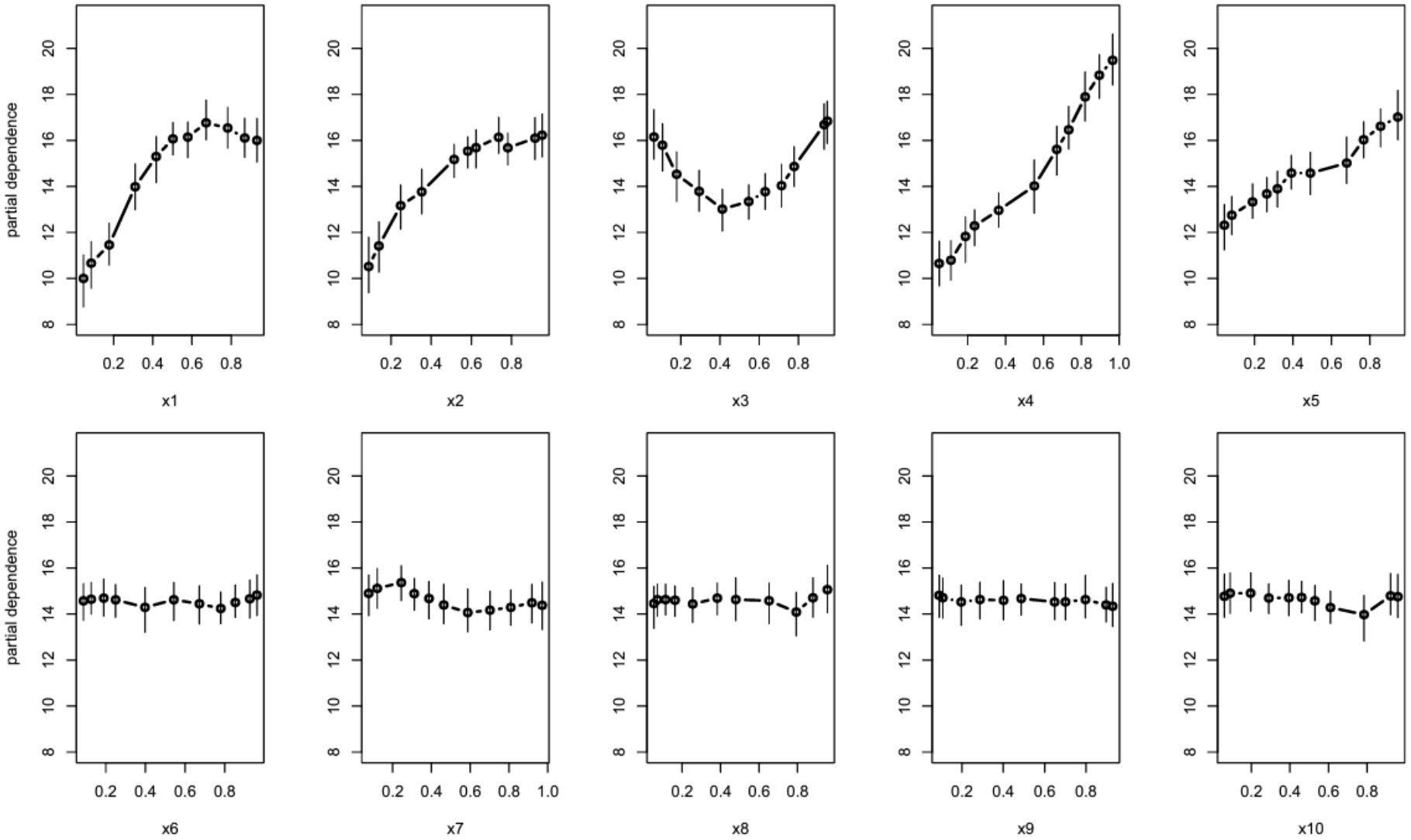
Partial Dependence Plot Example

```
> library(randomForest)
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.
> data(airquality)
> airquality <- na.omit(airquality)
> airquality <- na.omit(airquality)
> set.seed(131)
> ozone.rf <- randomForest(Ozone ~ ., airquality, importance=TRUE)
> imp <- importance(ozone.rf)
> impvar <- rownames(imp)[order(imp[, 1], decreasing=TRUE)]
> coordinates <- partialPlot(ozone.rf, airquality, impvar[1],
+                            xlab = impvar[1],
+                            main = paste("Partial Dependence on", impvar[1]))
> averagePredictions <- array(0, length(coordinates$x))
> for (i in 1:length(coordinates$x)) {
+     modified <- airquality
+     modified$Temp <- coordinates$x[i]    # set all Temp values to current value
+     averagePredictions[i] <- mean(predict(ozone.rf, modified))
+ }
> points(coordinates$x, averagePredictions)
```



$$\bar{f}_S(X_S) = \frac{1}{N}\sum_{i=1}^{N} f(X_S, x_{iC})$$

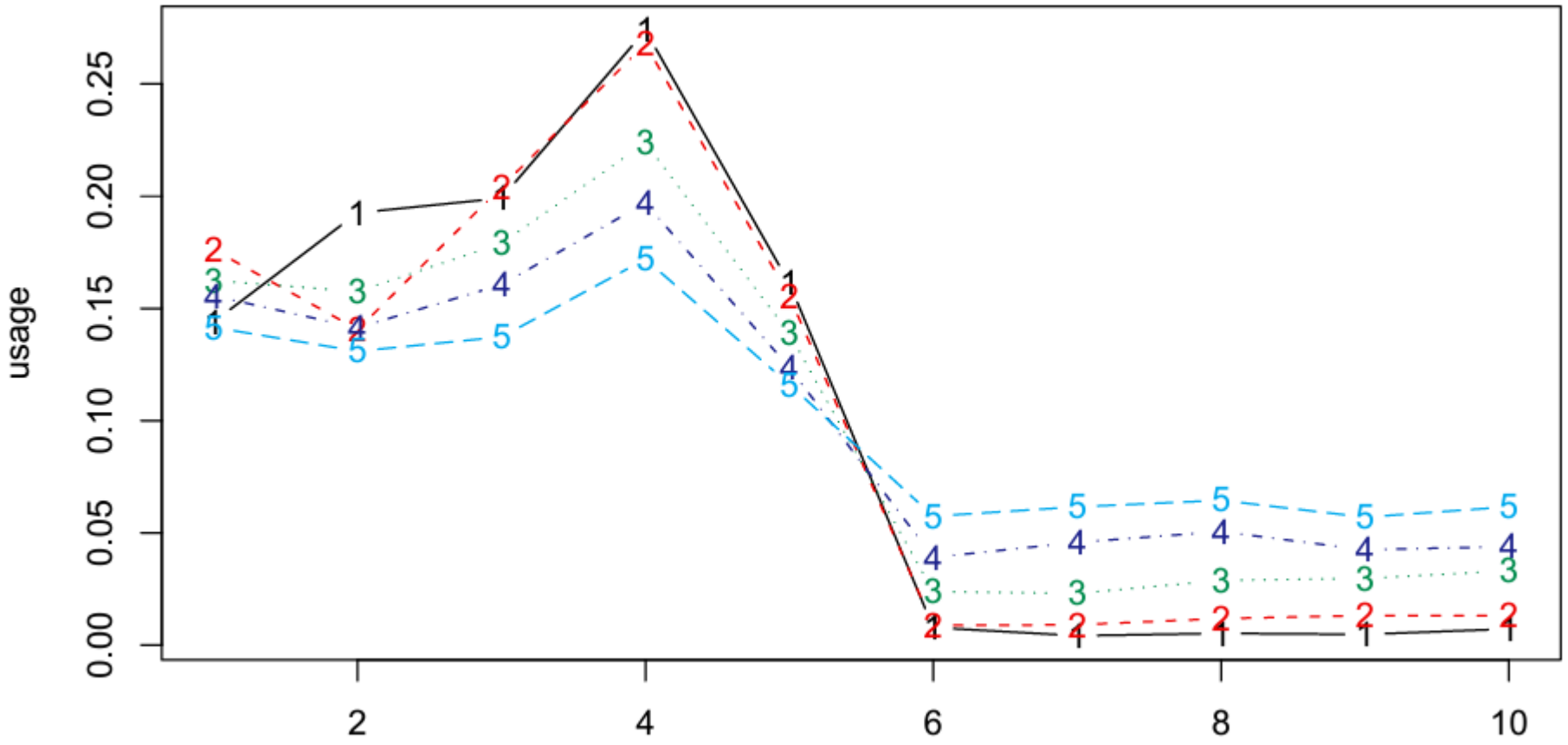
Partial Dependence Plot Example



Average Usage of Each Variable

$$v_j = \frac{1}{M}\sum_{m=1}^{M}\mathbb{I}(j \in T_m)$$

The lines show what happens as we increase the number of trees



Online Learning

Online Learning Topics

- Definition
- Stochastic Gradient Descent Example
- Vowpal Wabbit Example
- Online Bandits Example

Definition

- Offline (batch) learning
  - The entire training set is used at once to construct a model
- Online learning
  - Our model is updated after each observation [as soon as we get a label]
  - Example: after an advertisement is presented, the user may click the ad or not click the ad [yup: online learning for online advertising ☺]

Stochastic Gradient Descent Example

learningRate <− 0.1

bias <− 0

weight <− c(0.181175749183, 0.075938900706)

input <− c(-2.29193354108, -2.83416998336)

output <− 0

... learning happens here ... How do we get from model above to model below?

bias = -0.0347404766401

weight = [0.260798612828, 0.174399316807]

Stochastic Gradient Descent Example

- Recall from the “First Derivative (Gradient) of Log Loss” slide that the negative first derivative with respect to f(x) is (output - prediction)
- If we differentiate f(x)=w*x with respect to w, we get (output - prediction) * x
- So …

```
> learningRate <- 0.1
>
> bias <- 0
> weight <- c(0.181175749183, 0.075938900706)
>
> input <- c(-2.29193354108, -2.83416998336)
> output <- 0
>
> prediction <- 1 / (1 + exp(- sum(bias * 1 + weight * input)))
>
> bias + learningRate * (output - prediction) * 1
[1] -0.03474048
> weight + learningRate * (output - prediction) * input
[1] 0.2607986 0.1743993
```

Stochastic Gradient Descent Example

Refer to the stochastic_gradient_descent.ipynb notebook [from the class on logistic regression]

Vowpal Wabbit Example

Refer to the vowpal-wabbit_console.txt output file

Online Bandits Example

Refer to the online-bandits.ipynb notebook