Approaches for the Tornados Analysis

Tornados Data

What is our goal with the Tornados data?

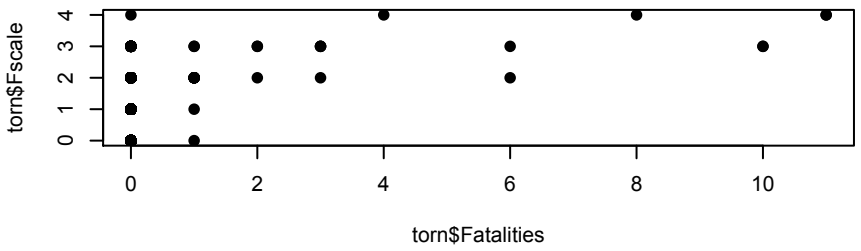
1. Develop an objective model that classifies tornados consistent with past data.

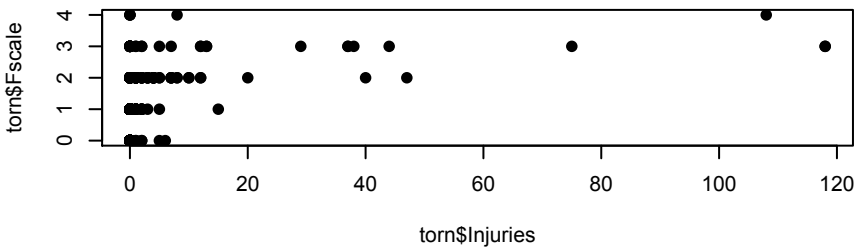
Why is this goal important?

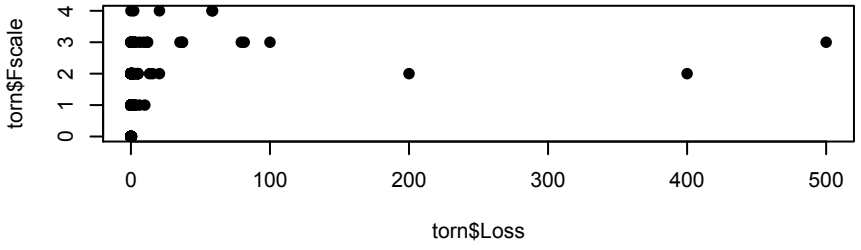
- More consistent classification which promotes scientific analyses.

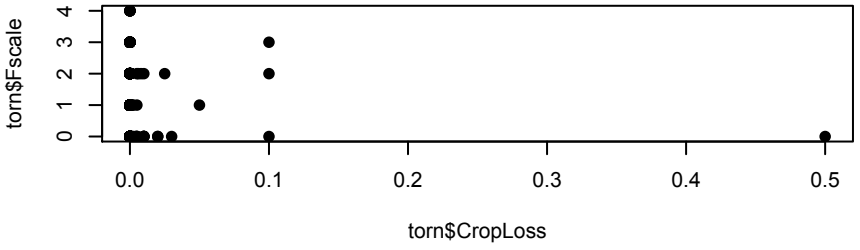
Joplin, MO May 22, 2011 – F5 Tornado

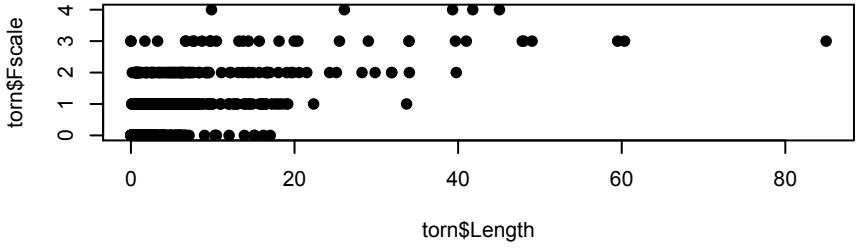
- 158 people died
- $2.8 Billion in damages

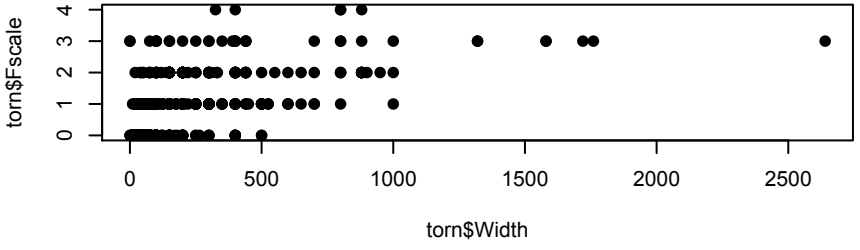












F-scale	0	1	2	3	4
Freq	578	242	100	32	5

Tornados Data

What are the challenges with the Tornados data?

1. Ordered Multiple-category response.
2. Non-linear.
3. Heavy Tails (Outliers)

Possible Approaches:

1. Linear Regression
2. Logistic Regression
3. Discriminant Analysis
4. K-Nearest Neighbors
5. Support Vector Machines
6. Trees

Linear Regression

Fit the linear model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}?$$

then predict using

$$\hat{y}_0 = \text{round}(\mathbf{x}_0'\boldsymbol{\beta})$$

Criticisms:

1. Predictions can be outside of 0-5.
2. Its not linear but we could uses basis expansions.
3. "Distance" between F0-F1 is same as F2-F3 (not true for tornados).

Logistic Regression

Logistic Regression Model:

$$Y_i \overset{ind}{\sim} \text{Bern}(p_i)$$

$$\log\left(\frac{p_i}{1-p_i}\right) = \mathbf{x}_i'\boldsymbol{\beta}$$

$$\Rightarrow p_i = \frac{\exp\{\mathbf{x}_i'\boldsymbol{\beta}\}}{1+\exp\{\mathbf{x}_i'\boldsymbol{\beta}\}}$$

Multinomial Logistic:

$$Y_i \overset{ind}{\sim} \text{Multinomial}(p_{1i}, \ldots, p_{Ki})$$

$$\log\left(\frac{p_{1i}}{p_{Ki}}\right) = \mathbf{x}_i'\boldsymbol{\beta}_1$$

$$\vdots$$

$$\log\left(\frac{p_{(K-1)i}}{p_{Ki}}\right) = \mathbf{x}_i'\boldsymbol{\beta}_{(K-1)}$$

$$p_{i1} = \frac{\exp\{\mathbf{x}_i'\boldsymbol{\beta}_1\}}{1+\sum_{k=1}^{K-1}\exp\{\mathbf{x}_i'\boldsymbol{\beta}_k\}}$$

$$\vdots$$

$$p_{i(K-1)} = \frac{\exp\{\mathbf{x}_i'\boldsymbol{\beta}_{(K-1)}\}}{1+\sum_{k=1}^{K-1}\exp\{\mathbf{x}_i'\boldsymbol{\beta}_k\}}$$

$$p_{iK} = \frac{1}{1+\sum_{k=1}^{K-1}\exp\{\mathbf{x}_i'\boldsymbol{\beta}_k\}}$$

MN-Logistic Regression

Multinomial Logistic Regression Model:

$$\log\left(\frac{p_{ki}}{p_{Ki}}\right) = \mathbf{x}_i'\boldsymbol{\beta}_k$$

How do we interpret β_{kj}?

1. For every unit increase in x_j, the log-odds ratio relative to category K increases by β_{kj}.
2. Just interpret the sign: If $\beta_{kj} > 0$, then p_{ki} increases as x_j increases.
3. Draw a picture.

MN-Logistic Regression

How do we estimate $\beta_1, \ldots, \beta_{K-1}$?

Maximum likelihood – remember that we assumed the multinomial distribution.

`library(nnet); multinom()`

How do we get uncertainty measurements (e.g. CIs)?

Asymptotics or bootstrap.

How do we do prediction?

Plug in and predict the probabilities.

Discriminant Analysis

Bayes Theorem for Classification (Revisited):

Recall in DA: Assume $\mathbf{X}$ is random and factor the joint distribution $[Y, \mathbf{X}] = [\mathbf{X} \mid Y][Y]$.

$$\text{Prior Probabilities: } \{\text{Prob}(Y = k)\}_{k=1}^{K} = \{\pi_k\}$$
$$\text{Conditional Dist: } [\mathbf{X} \mid Y = k] \sim \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Using Bayes Theorem:

$$[Y = k \mid \mathbf{X}] = \frac{\texttt{dmvnorm}(\mathbf{X} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k}{\sum_{j=1}^{K} \texttt{dmvnorm}(\mathbf{X} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\pi_j}$$

Discriminant Analysis

Discriminant Analysis Classification: Assign Y to the class that maximizes,

$$\max_k \ \texttt{dmvnorm}(\mathbf{X} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k$$

Note: Technically, this is "quadratic" discriminant analysis. "Linear" discriminant analysis assumes $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}\ \forall k$.

Discriminant Analysis

Discriminant Analysis Classification:

Issues:

1. What do we use for $\pi_1, \ldots, \pi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, and $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K$?

$$\hat{\pi}_k = \frac{1}{N_k} \sum_{i=1}^{N} I(y_i = k)$$

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{i:y_i=k} \mathbf{X}_i$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k} \sum_{i:y_i=k} (\mathbf{X}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{X}_i - \hat{\boldsymbol{\mu}}_k)'$$

Discriminant Analysis

Discriminant Analysis Classification:

Issues:

2. Still Makes NO SENSE when you have categorical predictors
3. Still no measure of uncertainty in your prediction.

K-Nearest Neighbors

K-Nearest Neighbors Algorithm (Non-parametric):

To classify a point which has covariates $\mathbf{x}_0$,

1. Find the K-nearest neighbors according to some distance $d_{0i} = d(\mathbf{x}_0, \mathbf{x}_i)$.
2. Approximate,

$$\text{Prob}(Y = k \mid \mathbf{x}_0) \approx \frac{1}{\text{Num Neigh}} \sum_{i \in \mathcal{N}_0} I(y_i = k)$$

where $\mathcal{N}_0$ is the set of nearest neighbor points.

3. Classify according to majority vote.

Support Vector Machines

Extending SVMs to more than 1 class is on-going work but here are a few ideas:

1. One vs. one
   - Construct "K choose 2" SVMs, tally number of times classified as class j, assign to largest majority.
2. One vs. All
   - Construct K SVMs, assign to class with largest

$$\beta_{0k} + \mathbf{x}_0' \boldsymbol{\beta}_k$$

Trees

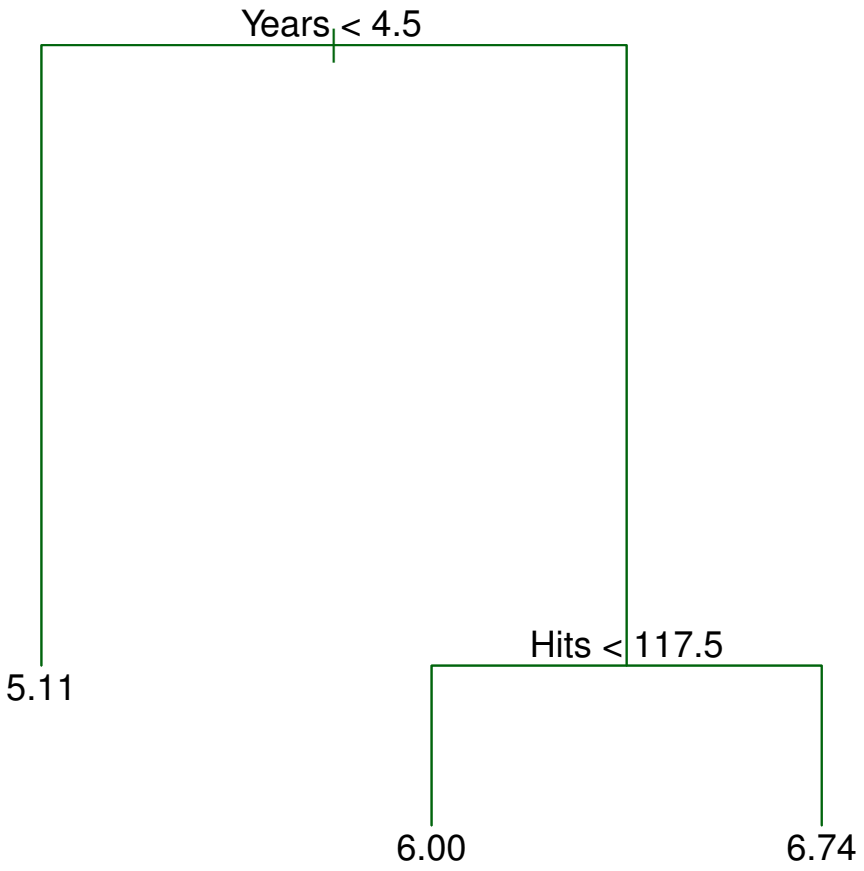
Big Idea:

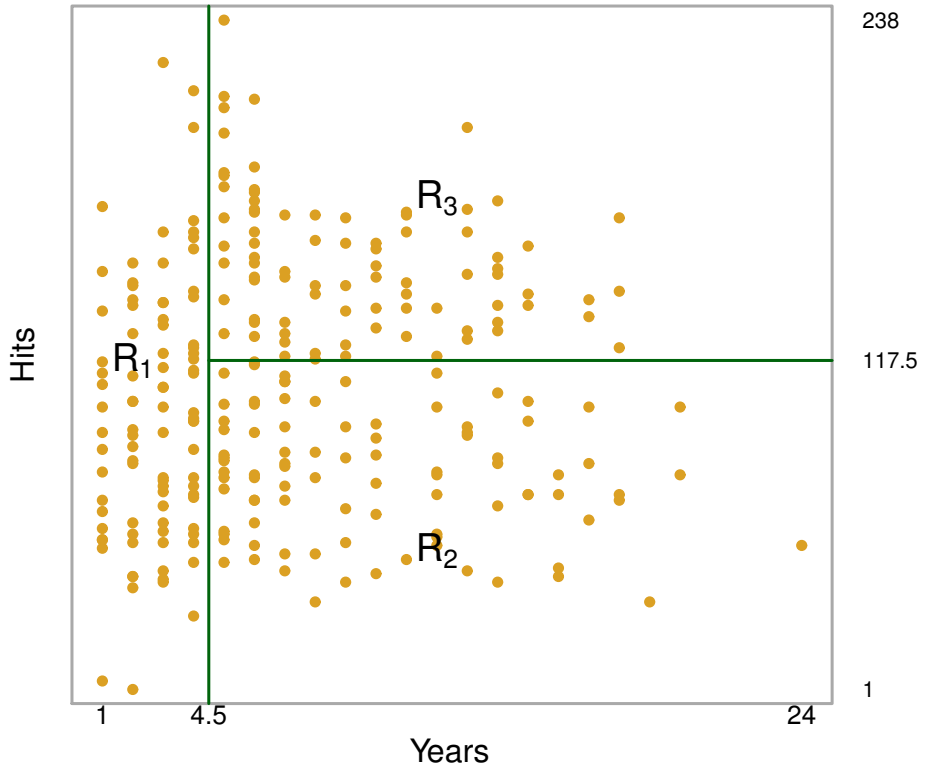
1. Divide predictor space into T non-overlapping regions $\mathcal{R}_1, \ldots, \mathcal{R}_T$.
2. For any $\mathbf{x}_0 \in \mathcal{R}_t$, make the same prediction.

Statistical Model:

$$\hat{y}(\mathbf{x}_0) = \sum_{t=1}^{T} \mu_t \mathbb{I}(\mathbf{x}_0 \in \mathcal{R}_t)$$

Trees


Years < 4.5
5.11
Hits < 117.5
6.00
6.74


Hits
238
117.5
1
R_1
R_2
R_3
1
4.5
24
Years

Trees

Statistical Model:

$$\hat{y}(\mathbf{x}_0) = \sum_{t=1}^{T} \mu_t \mathbb{I}(\mathbf{x}_0 \in \mathcal{R}_t)$$

Regression Trees:

$$\mu_t = \text{Mean}(\{y(\mathbf{x}_i) : \mathbf{x}_i \in \mathcal{R}_t\}_i)$$

Classification Trees:

$$\mu_t = (\pi_1, \ldots, \pi_K)'$$

$$\pi_k = \text{Proportion of } \{y(\mathbf{x}_i) : \mathbf{x}_i \in \mathcal{R}_t\} \text{ of class } k.$$

Classify prediction to highest probability

Trees

Growing a Tree: Find regions that minimize:

$$\text{Error}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda T$$

Controls Overfitting

$T = \text{Size of Tree}$

Regression Trees:

$$\text{Error}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$\hat{y}_i = \sum_{t=1}^{T} \hat{\mu}_t \mathbb{I}(\mathbf{x}_i \in \mathcal{R}_t)$$

Trees

Growing a Tree: Find regions that minimize:

$$\text{Error}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda T$$

$$T = \text{Size of Tree}$$

Classification Trees:

$$\text{Error}(\mathbf{y}, \hat{\mathbf{y}}) = \text{Classification Error (not optimal)}$$

$$\text{Error}(\mathbf{y}, \hat{\mathbf{y}}) = \text{Gini Index} = \sum_{k=1}^{K} \hat{\pi}_{tk}(1 - \hat{\pi}_{tk})$$

$$\text{Error}(\mathbf{y}, \hat{\mathbf{y}}) = \text{Cross Entropy} = -\sum_{k=1}^{K} \hat{\pi}_{tk} \log(\hat{\pi}_{tk})$$

$$\hat{\pi}_{tk} = \text{Proportion of Obs in Region } t \text{ of class } k$$

Trees

Growing a Tree: Recursive Binary Splitting

1. Find a predictor X_p and a cut point s such that splitting space into $\{X \mid X_p < s\}$ and $\{X \mid X_p \geq s\}$ leads to greatest reduction in error.
2. Repeat step 1 repeatedly but only split previously defined regions.

Trees

Growing a Tree: A few notes

1. Computationally simpler to build a big tree then "prune" it back to minimize:

$$\text{Error}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda T$$

$$T = \text{Size of Tree}$$

   Default in R is to build a big tree, you need to prune it yourself.

2. How do you choose λ?
   - Cross-validation

Trees

Advantages:

1. Intuitive
2. Easily handle qualitative predictors.
3. Fitting a "model" within each node can give a measure of uncertainty.
4. Outliers are less of an issue.

Disadvantages:

1. High variance
2. Have a tendency to overfit (need pruning)
3. Hard to reflect uncertainty of the tree (easy to reflect uncertainty within the node) – see Bayesian Trees

Ensemble Methods

Bagging

Disadvantage of Trees: High variance
Solution: Bootstrap Aggregating (Bagging)

Motiving Problem:

$$X_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$$

$$\mathbb{V}\text{ar}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{\sigma^2}{n} < \sigma^2$$

Big Idea: Average Trees

$$\hat{y}(\mathbf{x}_0) = \frac{1}{B}\sum_{b=1}^{B} \hat{y}^b(\mathbf{x}_0)$$

Bagging

Bagging Algorithm:

1. For $b = 1, \ldots, B$
   a. Take a bootstrapped (with replacement) sample of size n.
   b. Build a tree $\mathcal{T}_b$.
2. To predict for a point $\mathbf{x}_0$,
   a. For $b = 1, \ldots, B$, pass $\mathbf{x}_0$ down the b^{th} tree to get a bootstrapped prediction $\hat{y}^b(\mathbf{x}_0)$.
   b. Average predictions to get

$$\hat{y}(\mathbf{x}_0) = \frac{1}{B}\sum_{b=1}^{B} \hat{y}^b(\mathbf{x}_0)$$ Reg. Trees

$$\hat{y}(\mathbf{x}_0) = \text{mode}(\hat{y}^1(\mathbf{x}_0), \ldots, \hat{y}^B(\mathbf{x}_0))$$ Class. Tree – "majority vote"

Bagging

Why bagging is brilliant:

1. Bagging decreases variance so we can overfit our bootstrapped data.
    - Grow large trees (low bias) and the aggregate them.

Bagging

Why bagging is brilliant:

2. Eliminates the need to do cross validation.

Out-of-bag Error Estimation:

"In-bag" points: Obs. in the bootstrap sample.
"Out-of-bag" points: Obs. not in the bootstrap sample.

Use $\{\mathcal{T}_b\}$ that were built where i^{th} observation was out-of-bag to predict y_i. This is equivalent to a "hold-out" prediction that can be used to calculate misclassification error or MSE.

Bagging

Why bagging is brilliant:

3. Performs variable selection.

Downside: Hard to interpret because we are using many trees.

Solution: Look at how much MSE (or misclassification) is decreased when a variable is used. Rank variables according to how much they reduced error.

Note: In R it gives you how much MSE would increase if the variable would be left out (bigger = more important).

Bagging

Why bagging is brilliant:

4. Its not just for trees – any method can be bagged.

Random Forests

Motivating Problem

$$\mathbf{X} = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} \sim \mathcal{N}_n(0, \sigma^2 \mathbf{R})$$

$$\mathbb{V}\text{ar}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{\sigma^2}{n^2}\mathbf{1}_n'\mathbf{R}\mathbf{1}_n \overset{?}{<} \frac{\sigma^2}{n}$$

When observations are positively correlated,

$$\frac{\sigma^2}{n^2}\mathbf{1}_n'\mathbf{R}\mathbf{1}_n > \frac{\sigma^2}{n}$$

Independent obs. have more information than correlated obs.

Random Forests

Random Forest Algorithm:

1. For $b = 1, \ldots, B$
   a. Take a bootstrapped (with replacement) sample of size n.
   b. At each split, randomly consider $m < P$ vars
   c. Build a tree $\mathcal{T}_b$.
2. To predict for a point $\mathbf{x}_0$,
   a. For $b = 1, \ldots, B$, pass $\mathbf{x}_0$ down the b^{th} tree to get a bootstrapped prediction $\hat{y}^b(\mathbf{x}_0)$.
   b. Average predictions to get

$$\hat{y}(\mathbf{x}_0) = \frac{1}{B}\sum_{b=1}^{B} \hat{y}^b(\mathbf{x}_0)$$

Reg. Trees

$$\hat{y}(\mathbf{x}_0) = \text{mode}(\hat{y}^1(\mathbf{x}_0), \ldots, \hat{y}^B(\mathbf{x}_0))$$

Class. Tree – "majority vote"

Random Forests

Random Forest Intuition:

- Selecting only $m < P$ predictors to build a tree, "decorrelates" the trees.
- If $m = P$ then random forests = bagging.

What value do we use for m?

- It depends on how correlated your predictors are.
- Book suggests $m = \text{round}(\sqrt{P})$
- Cross validation

Bayesian Trees

Disadvantage: No uncertainty in the tree itself
Solution: Put a prior over the space of all trees

Tree: (Finite) set of splitting decisions (yes/no) and splitting locations.

Bayesian Tree Growing:

1. Split leaf j with probability $\pi_j(T) \in [0, 1)$.
2. If split in step 1, draw a split point
$$s_j \sim \text{Unif}(\text{All Possible Points})$$
3. Draw two new leaf "parameters" $\theta_{j_1,j_2} \sim \pi_\Theta(\theta)$.

Essentially, this is a prior over all trees. The hard part is sampling from the posterior (I have references if you're interested).

Boosting (Weak Learners)

Big Idea: Fit trees sequentially to residuals – each tree fits the part of the data that previous trees didn't capture.

Boosting Algorithm:

1. Fit the first tree to the data $\{(y_i, \mathbf{x}_i)\}$.
2. Obtain predictions $\{\hat{y}_i^{(1)}\}$ and calculate residuals $\{r_i^{(1)} = y_i - \lambda \hat{y}_i^{(1)}\}$.
3. For $b = 2, \ldots, B$
   a) Fit the b^{th} tree to the residuals $\{r_i^{(b-1)}\}$.
   b) Obtain predictions $\{\hat{r}_i^{(b)}\}$.
   c) Update the residuals $\{r_i^{(b)} = r_i^{(b-1)} - \lambda \hat{y}_i^{(b)}\}$.
4. Output the boosted model:

$$\hat{y} = \lambda \sum \hat{y}^{(b)}$$

Boosting (Weak Learners)

Intuition of Boosting:

1. Rather than using many trees which capture the same features of the data (bagging, random forests), use many trees which capture different aspects of the data.

A few notes:

1. If B is large, then you can overfit. Use cross-validation to choose B.
2. λ controls how "fast" you learn (i.e. how much of the data is learned from each piece). Book suggests to use $\lambda = 0.01$.
3. We can typically use small models at each step because learning happens over lots of steps.

Bayesian Additive Regression Trees (BART)

There is a Bayesian version of boosting called BART that I don't really want to get into. See me for a paper if you're interested.

My General Thoughts on Ensemble Methods

Ensemble Methods = Non-interpretable.

If you're just interested in prediction, they work great.

I'm a modeler, I like simple models/techniques that not only predict pretty well but are also easy to interpret.

Expectations for Tornados Analysis

1. Clean the Data
   - remove repeat observations
2. Determine an appropriate model for analyzing tornados that meets the goal of the analysis
   - State why the model you choose meets the goals
   - Justify linear vs. non-linear effects
3. Fit your chosen model
   - Give an overview of how to fit your model (e.g. ML or binary partitioning)
   - Justify any "tuning" parameters (e.g. cross-validation)
4. Results
   - "Show" your final model (e.g. a tree or parameter estimates)
   - Describe/show how a scientist would use your model to classify a tornado.
   - Do you see any out of the ordinary classifications?

My Tornados Analysis

1. Clean the Data
   - Most repeat observations occur because the tornado crossed state lines. I'd aggregate across state lines but only take 1 within each state.
2. Determine and appropriate model for analyzing tornados that meets the goal of the analysis
   - I'd use a tree – it deals with outliers well, intuitive, gives a measure of variable importance, nonlinear, etc.
3. Fit your chosen model
   - Fit a tree using recursive binary partitioning
   - Prune back to minimize cross validation error
4. Results
   - Plot my final tree
   - Intuitively explain how to classify according to the tree – report probabilities for a few observations
   - Check to see if anything looks "out of the ordinary"