732A95/TDDE01 Introduction to Machine Learning

Lecture 1a Block 2: Ensemble Methods

Jose M. Peña
IDA, Linköping University, Sweden

Contents

Literature

- Main source
  - Hastie, T., Tibshirani, R. and Friedman, J. *The Elements of Statistical Learning*. Springer, 2009. Sections 10.1-10.10 and 15.1-15.4.
- Additional source
  - Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006. Sections 14.1-14.4.

Bagging

- Boosting aggregation (bagging) is a technique to combine (weak) regressions and so produce a more accurate (committee) regression. It can also be applied to classification.
- Main steps:
  - Obtain B bootstrap samples of the original training data, i.e. draw B samples with replacement from the original data and of the same size as the original data.
  - Run the regression algorithm on each bootstrap sample b to obtain the regression $f^b(x)$.
  - Return the average of the regressions obtained, that is

$$f_{bag}(x) = \frac{1}{B} \sum_b f^b(x)$$

- Bagging (generalized) linear regressions is not particularly useful: The bagged regression will converge to the one learned from the original data as B increases.
- Then, it makes more sense to use bagging with non-linear regressions.
- For classification, we can average the individual models by averaging their posterior class probabilities, or use majority voting.

Bagging

- Let $h(x)$ denote the true regression. Then, $f^b(x) = h(x) + \epsilon^b(x)$.
- The error of $f^b(x)$ can be expressed as

$$E_X[(f^b(x) - h(x))^2] = E_X[\epsilon^b(x)^2]$$

- The error of $f_{bag}(x)$ can be expressed as

$$E_X[(\frac{1}{B}\sum_b f^b(x) - h(x))^2] = E_X[(\frac{1}{B}\sum_b \epsilon^b(x))^2]$$

- **Assume** that the error terms $\epsilon^b(x)$ have zero mean and are uncorrelated, i.e. $E_X[\epsilon^b(x)] = 0$ and $E_X[\epsilon^b(x)\epsilon^{b'}(x)] = 0$. Then,

$$E_X[(\frac{1}{B}\sum_b f^b(x) - h(x))^2] = \frac{1}{B}\Big[\frac{1}{B}\sum_b E_X[\epsilon^b(x)^2]\Big]$$

  which implies that bagging reduces the average error of the individual regressions by a factor of B.
- In practice, the reduction in error is less dramatic as the individual errors are highly correlated.
- At least, the bagged error is never larger than the average individual errors:

$$\frac{1}{B}\sum_b E_X[\epsilon^b(x)^2] = E_X[\sum_b \frac{1}{B}\epsilon^b(x)^2] \geq E_X[(\frac{1}{B}\sum_b \epsilon^b(x))^2]$$

  by Jensen's inequality, i.e. $\sum_i \lambda_i g(a_i) \geq g(\sum_i \lambda_i a_i)$.

Bagging

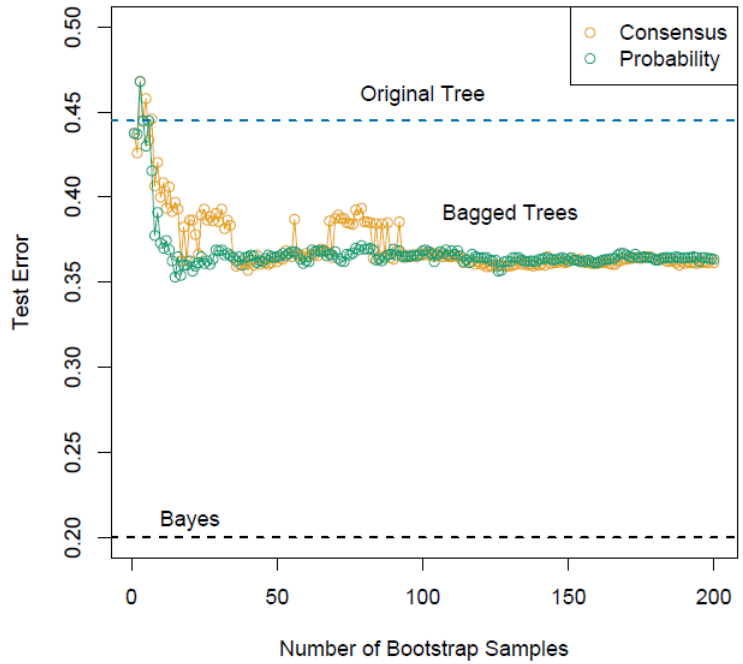


FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

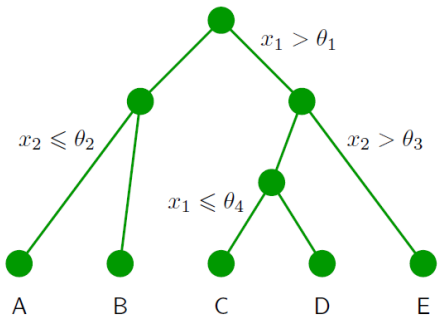
- Bagging reduces the variance (a.k.a. unstability) of the original algorithm (the more uncorrelated the individual errors, the larger the reduction).
- The reason is that the variance of the average of B identically distributed variables with positive pairwise correlation ρ is

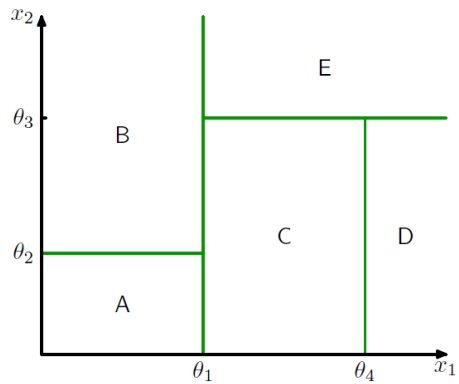
$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

- This is why bagging is used with decision trees, which have high variance.

Random Forests

- Recall from previous lectures that a decision tree partitions the input space into regions. Each region has associated a predictor.
- Then, decision trees can be seen as a form of committee predictor.





- The model in each region is typically a constant, specifically
  - the result of majority voting for classification, since the best classifier under the 0-1 loss function is $\arg\max_y p(y|x)$, and
  - the average for regression, since the best regression function under the squared error loss function is $E_{Y|x}[y]$.
- Decision trees have many advantages (popular, fast, interpretable, etc.) and two major disadvantages: Low accuracy and high variance.
- These disadvantages make them good candidates for bagging, at the expense of compromising some of their advantages.

Random Forests

- Random forest = bagging + decision trees + decorrelation.

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

- Note that step 1ai aims to decorrelate the individual decision trees.

Random Forests

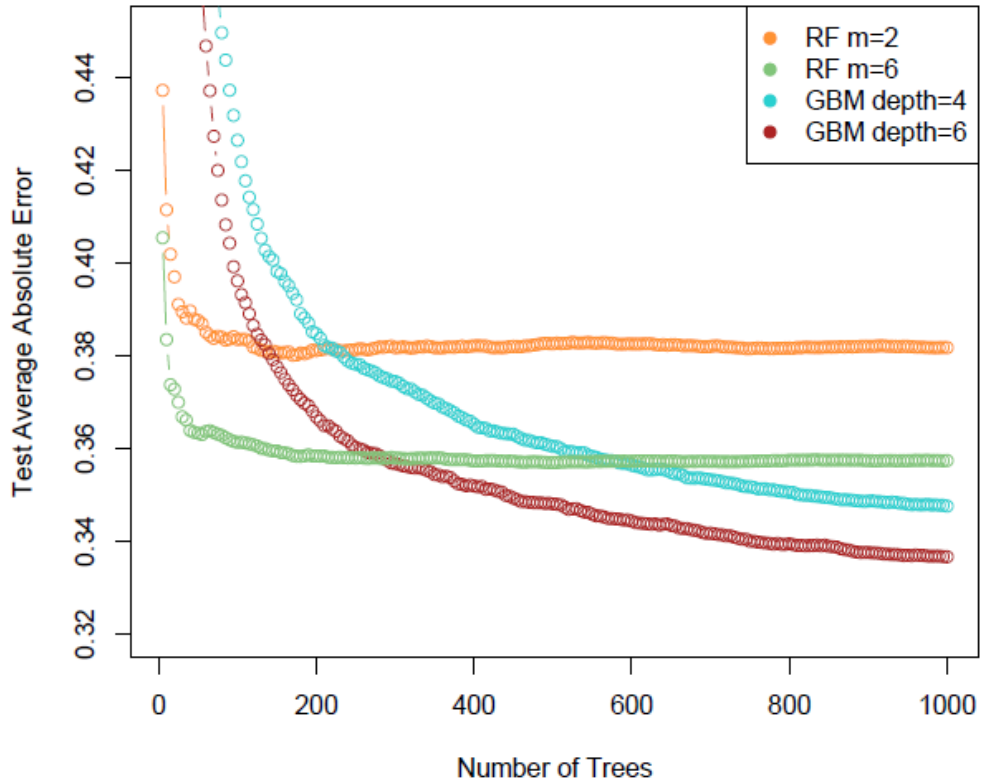
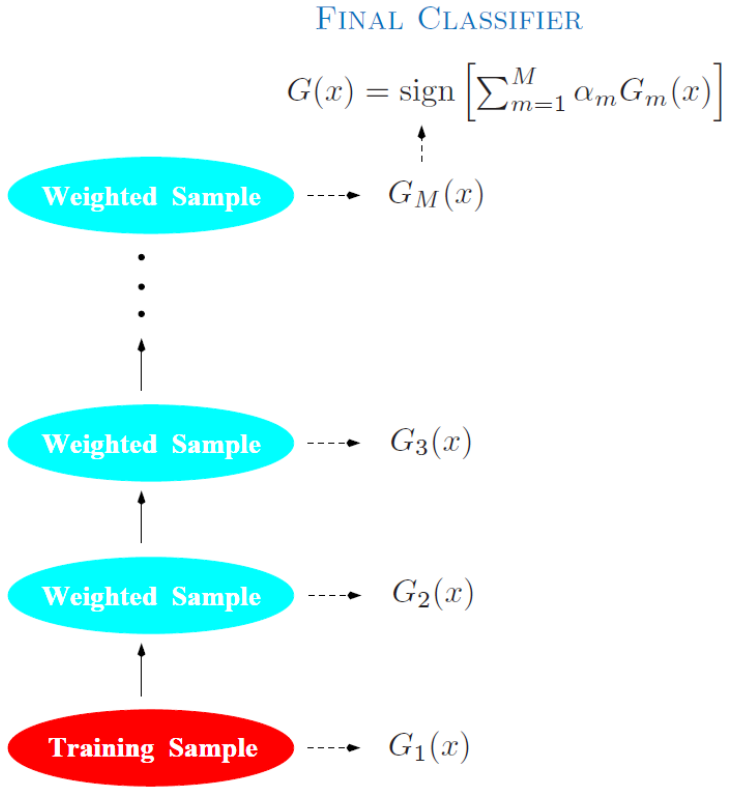


FIGURE 15.3. *Random forests compared to gradient boosting on the California housing data. The curves represent mean absolute error on the test data as a function of the number of trees in the models. Two random forests are shown, with* $m = 2$ *and* $m = 6$*. The two gradient boosted models use a shrinkage parameter* $\nu = 0.05$ *in (10.41), and have interaction depths of* 4 *and* 6*. The boosted models outperform random forests.*

Boosting

- Boosting is a technique to combine (weak) classifiers and so produce a more accurate (committee) classifier. It can also be applied to regression.
- Main steps:
  - Run the original classification algorithm on the original/modified training data.
  - Modify the training data by giving
    - more weight to the erroneously classified points, and
    - less weight to the correctly classified points.
  - Iterate through the previous steps a number of times.
  - Return a weighted average of the classifiers obtained.



AdaBoost

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

AdaBoost

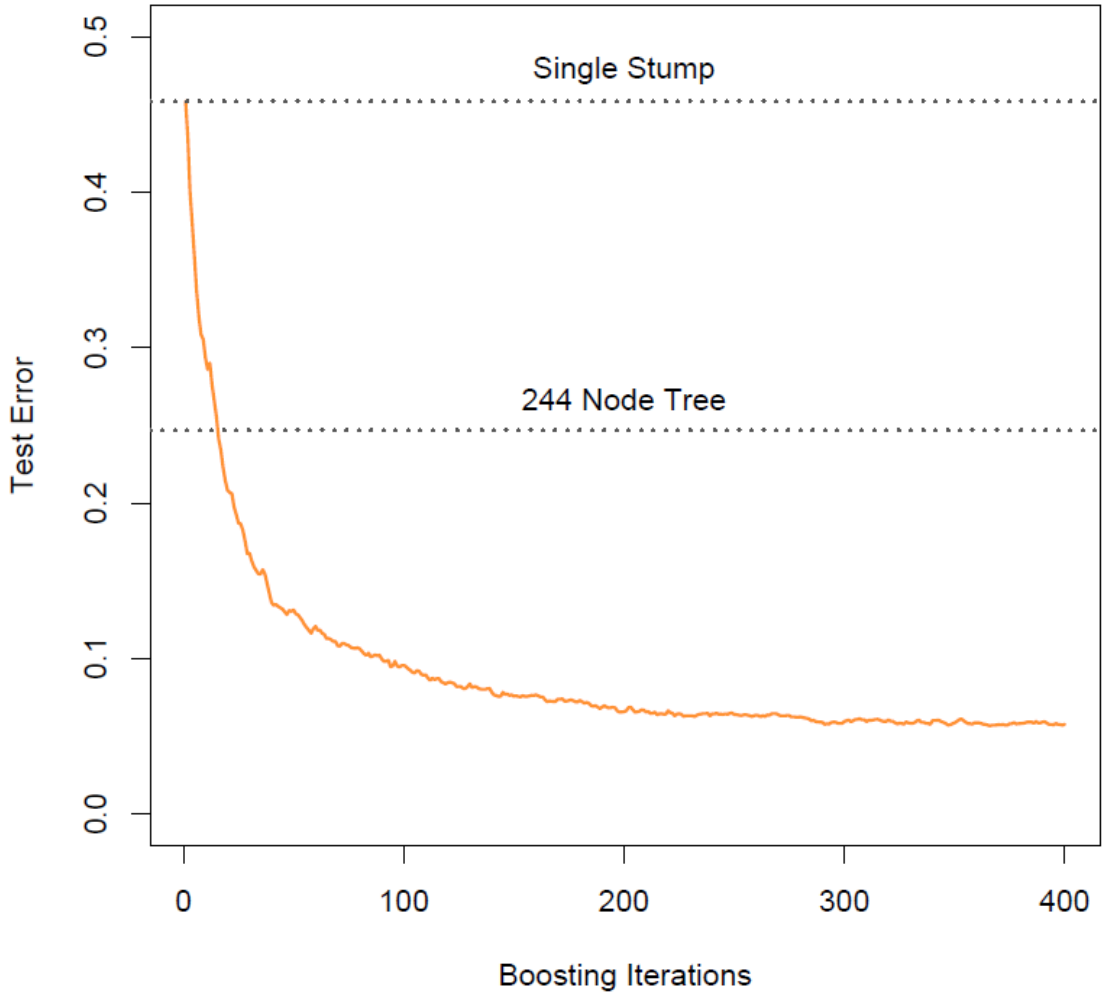


FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.*

Forward Stagewise Additive Modeling

- Boosting can be seen as fitting an additive expansion of a set of basis functions. That is, the boosted classifier

$$G(x) = \sum_m \alpha_m G_m(x)$$

can be rewritten as

$$f(x) = \sum_m \beta_m b(x; \gamma_m)$$

and the aim of boosting can be rephrased as minimizing a loss function over the training data $\{x_i, y_i\}$, that is

$$\min_{\{\beta_m, \gamma_m\}} \sum_i L(y_i, \sum_m \beta_m b(x_i; \gamma_m))$$

- The problem above is challenging for most loss functions and basis functions. Heuristic solution: Add the basis functions one at a time.

Forward Stagewise Additive Modeling

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (a) Compute

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$

   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

AdaBoost

- It can be shown that Adaboost is equivalent to forward stagewise additive modeling with exponential loss function, that is

$$L(y, f(x)) = \exp(-yf(x))$$

- In each step of forward stagewise additive modeling, we have to solve

$$\begin{aligned}(\beta_m, \gamma_m) &= \arg\min_{\beta,\gamma} \sum_i \exp[-y_i(f_{m-1}(x_i) + \beta b(x_i;\gamma))] \\ &= \arg\min_{\beta,\gamma} \sum_i w_i^{(m)} \exp(-y_i \beta b(x_i;\gamma))\end{aligned}$$

  where $w_i^{(m)} = exp(-y_i f_{m-1}(x_i))$ are the instance weights in AdaBoost.
- Note that

$$\sum_i w_i^{(m)} \exp(-y_i \beta b(x_i;\gamma)) = \exp(-\beta) \sum_{y_i = b(x_i;\gamma)} w_i^{(m)} + \exp(\beta) \sum_{y_i \neq b(x_i;\gamma)} w_i^{(m)}$$

  and, thus, for any given $\beta > 0$

$$\gamma_m = \arg\min_{\gamma} \sum_i w_i^{(m)} I(y_i \neq b(x_i;\gamma))$$

  which is what step 2a of AdaBoost does.
- Now, solving for β results in $\beta = (1/2)\log((1 - err_m)/err_m)$.
- Finally, take $\alpha_m = 2\beta$ and note that

$$w_i^{(m+1)} = exp(-y_i[f_{m-1}(x_i) + \beta b(x_i;\gamma)]) = w_i^m exp(\alpha_m I(y_i \neq b(x_i;\gamma)))exp(-\beta)$$

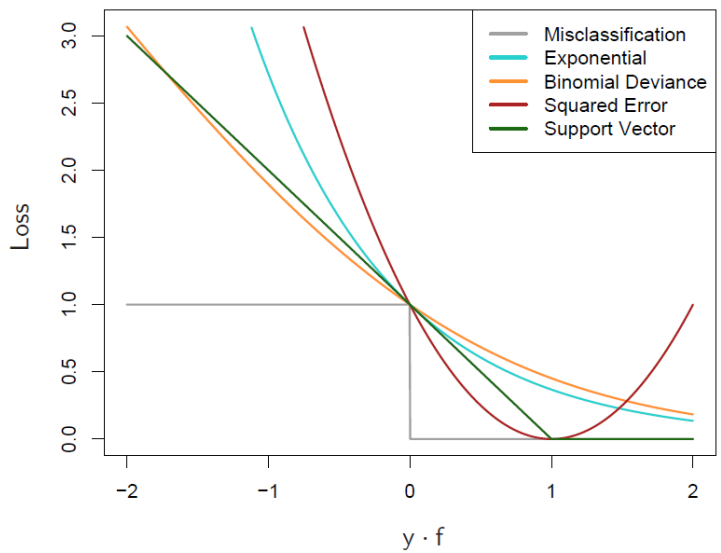
  where $exp(-\beta)$ has no effect as it multiplies all weights. This is steps 2b-d.

AdaBoost

- It can be shown that the population minimizer of the exponential loss function is

$$\arg\min_{f(x)} E_{Y|x}[exp(-yf(x))] = \frac{1}{2}\log\frac{p(Y=1|x)}{p(Y=-1|x)}$$

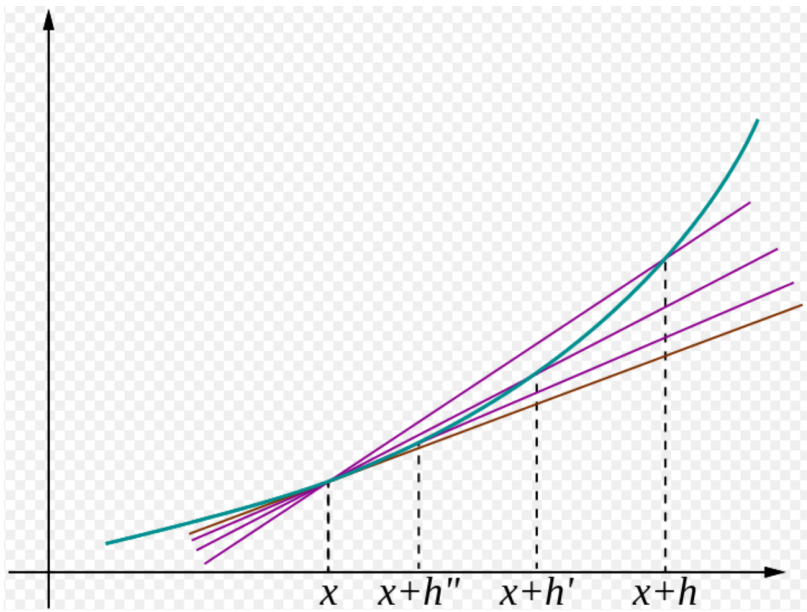
- Since Adaboost's output is an approximation to the population minimizer, it makes sense to use the sign as the classification rule in step 3.
- Note that the exponential loss is an upper bound on the 0-1 loss.



- Moreover, the exponential loss for a misclassified point increases exponentially with its margin $yf(x)$. This may make the performance of Adaboost to degrade in settings with wrongly labeled points: Adaboost may try to classify them correctly at the expense of other misclassified points that are correctly labeled.

Gradient Boosting

- For some loss functions and/or basis functions, forward stagewise additive modeling can be cumbersome. In those cases, gradient boosting may be the solution.
- Recall that $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$



- Recall that the gradient is a vector whose components are the partial derivatives.

Gradient Boosting

1. Initialize $f_0(x) = \arg\min_\gamma \sum_i L(y_i, b(x_i; \gamma))$
2. For $m = 1$ to M:
   (a) Compute the gradient residual $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$
   (b) $\gamma_m = \arg\min_\gamma \sum_i (r_{im} - b(x_i; \gamma))^2$
   (c) $\beta_m = \arg\min_\beta \sum_i L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma_m))$
   (d) $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$

- Note that step 2b aims for the basis function that is most parallel to the gradient.
- Step 2a is easy in some cases, e.g. if $L(y, f(x)) = \frac{1}{2}(y - f(x))^2$ then

$$\frac{\partial L(y, f(x))}{\partial f(x)} = y - f(x)$$

Summary

- Bagging: Combines weak predictors to reduce error and variance.
- Random forest = bagging + decision trees + decorrelation.
- AdaBoost: Combination of weak predictors under exponential loss.
- Gradient boosting: Extension to arbitrary loss functions.
- How many individual models in bagging, boosting and random forest ? (Nested) cross-validation.