Ensemble learning

© Victor Kitov

v.v.kitov@yandex.ru

Summer school on Machine Learning in High Energy Physics

in partnership with

Yandex Data Factory

August 2015

Ensemble learning

Definition 1

Ensemble learning - using multiple machine learning methods for a given problem and integrating their output to obtain final result.

Synonyms: committee-based learning, multiple classifier systems. Closely relates to **data fusion** - integration of multiple data representing the same real-world object into useful representation.

Applications:

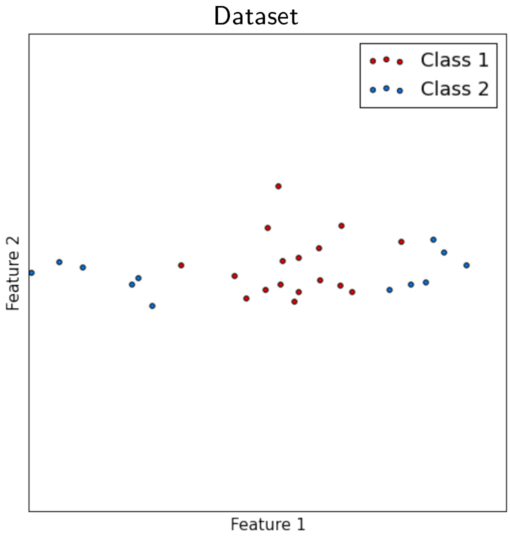
- supervised methods: regression, classification
- unsupervised methods: clustering

Motivation of ensembles

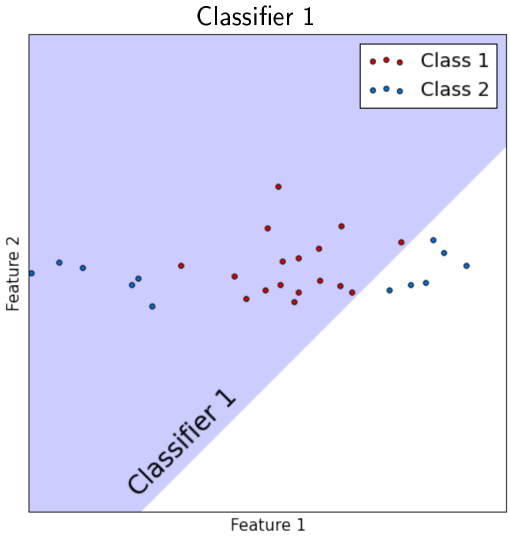
- Benefits for prediction:
    - increased accuracy
    - increased robustness.
- Justification: some predictors are compensating the errors of other predictors
- When to use:
    - existing model hypothesis space is too narrow to explain the true one
    - too small dataset to figure out concretely the exact model hypothesis
    - avoid local optima of optimization methods
- Frequently the task itself promotes usage of ensembles (such as computer security):
    - multiple sources of diverse information
    - different abstraction levels need to be united

Table of Contents

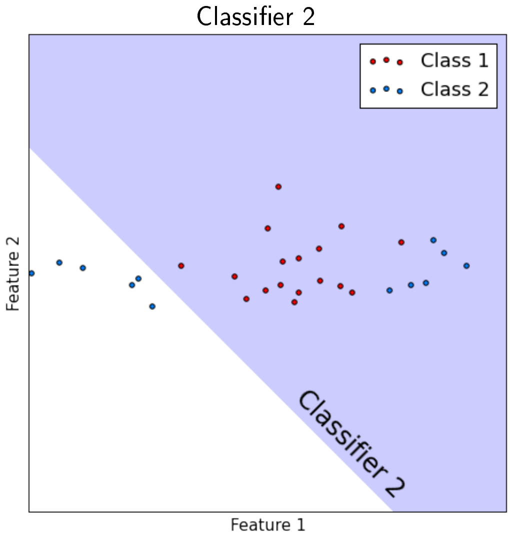
Motivation for classification



Motivation for classification

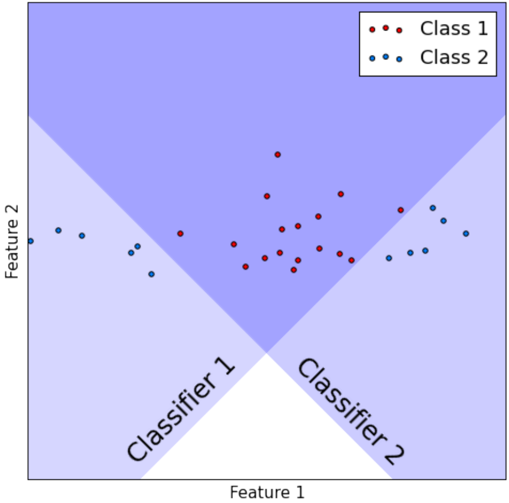


Motivation for classification

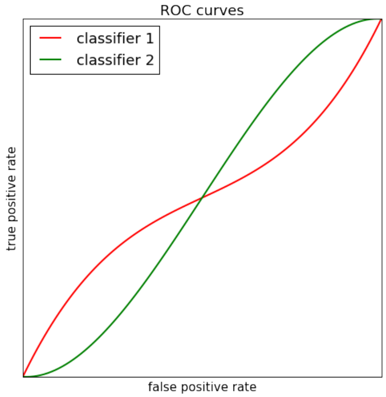


Motivation for classification

Classifier 1 and classifier 2 combined using AND rule

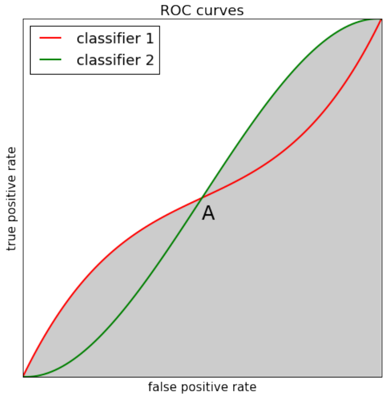


ROC curve of classifier combination



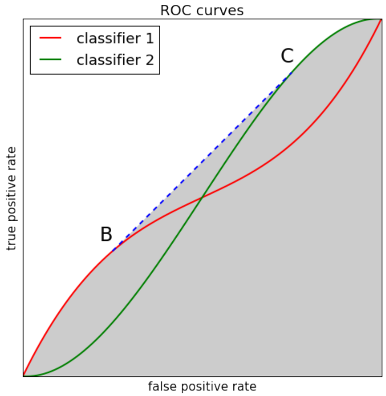
Suppose we have two classifiers and want to optimize AUC of their combination.

ROC curve of classifier combination



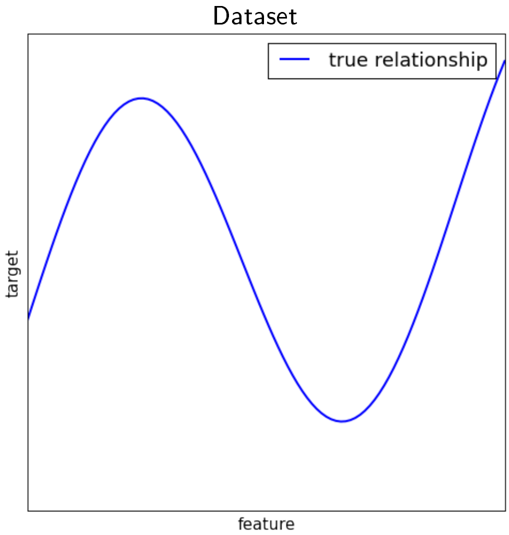
By taking best performing classifier at each point we can achieve $\int \max ROC_1(t), ROC_2(t) dt$.

ROC curve of classifier combination

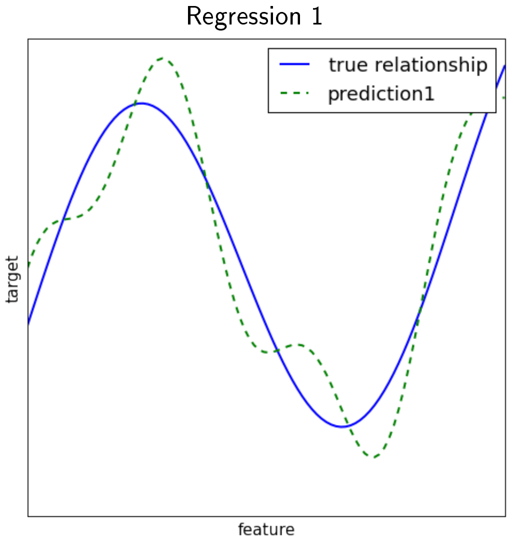


Interval [B,C]: by selecting classifier1(A) with probability p and classifier2(C) with probability $1 - p$ we can obtain AUC as the convex hull of the ROC curves.

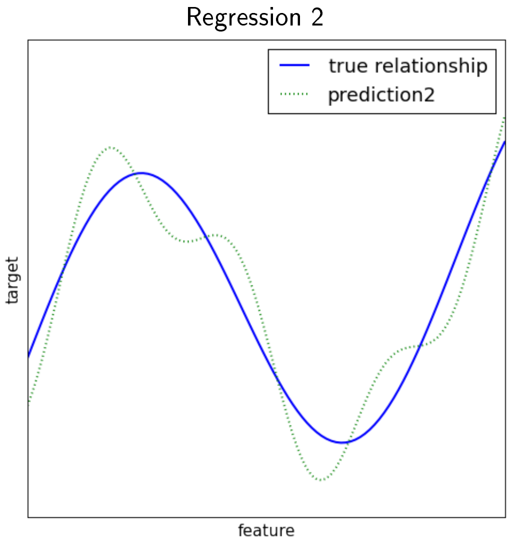
Motivation for regression



Motivation for regression

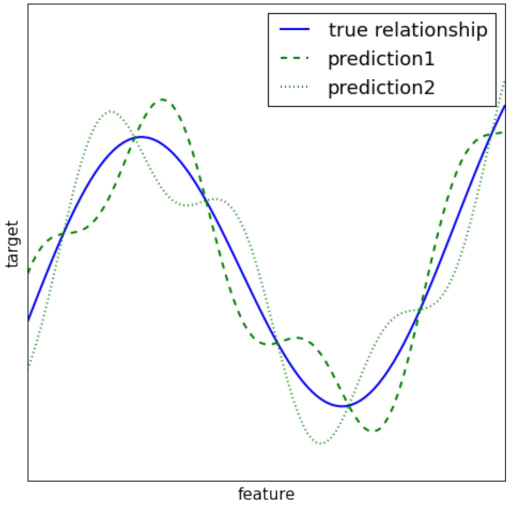


Motivation for regression



Motivation for regression

Regression 1 and regression 2 combined using averaging



Regression - convex loss function

- Training set (x_1, y_1), (x_2, y_2), ...(x_N, y_N)
- Forecasting function $\widehat{y} = F(x, \theta)$, where θ is a vector of adjustable parameters.
- Objective: $\sum_{n=1}^{N} \mathcal{L}(F(x_n, \theta), y_n) \to \min_\theta$
- Most commonly $\mathcal{L}(\widehat{y}, y) = G(\widehat{y} - y)$, such as $(\widehat{y} - y)^2$ or $|\widehat{y} - y|$

Regression - convex loss function

Theorem 1

Suppose the loss function be $\mathcal{L}(\widehat{y}_i, y_i) = G(\widehat{y}_i - y_i)$ for some convex function $G(\cdot)$ and K regressions are used for prediction, which yield K losses on the training set:

$$L_1 = \sum_{n=1}^{N} \mathcal{L}(\widehat{y}_n^1, y_n),\ L_2 = \sum_{n=1}^{N} \mathcal{L}(\widehat{y}_n^2, y_n),\ \dots, L_K = \sum_{n=1}^{N} \mathcal{L}(\widehat{y}_n^K, y_n).$$

Then the loss L of the convex combination of these regressions with coefficients $\lambda_1, \lambda_2, ...\lambda_K$ is no greater than weighted loss of individual models:

$$L_{comb} \leq \sum_{k=1}^{K} \lambda_k L_k$$

If $L_1 = L_2 = \ldots = L_K = L$, then loss of their arbitrary convex combination is no greater than L.

Bias-variance decomposition

Theorem 2

Unknown relationship $y = f(x) + \varepsilon$ is reconstructed using a set of points (x_n, y_n), $n = 1, 2...N$ as $\widehat{f}(x)$. Noise ε is independent of any x, $\mathbb{E}\varepsilon = 0$ and $Var[\varepsilon] = \sigma^2$. Then

$$\mathbb{E}[\widehat{f} - f]^2 = [\mathbb{E}\widehat{f} - f]^2 + \mathbb{E}[\widehat{f} - \mathbb{E}\widehat{f}]^2$$

$$\mathbb{E}[\widehat{f} - y]^2 = [\mathbb{E}\widehat{f} - f]^2 + \mathbb{E}[\widehat{f} - \mathbb{E}\widehat{f}]^2 + \sigma^2$$

Essentially this means:

$$MSE = \text{bias}^2 + \text{variance} + \text{irreducible error}$$

Different models of ensemble have different bias and variance deviations, which we hope to average away.

More motivation

- By combining B classifiers with independent misclassification and majority vote combiner $P(\mathit{misclassification}) \overset{B\to\infty}{\to} 0$.
- By combining B regressions with i.i.d. errors (zero mean, $\mathbb{E}|\varepsilon| < \infty$) and output averaging, $P(\lim_{B\to\infty} |\widehat{\varepsilon}_B| = 0) = 1$.
- Ensembling is the winning strategy for most machine learning competitions (Netflix, Kaggle, KDD Cup, etc.).
- Computational cost is often not greater: when building individual model, we need to estimate and evaluate multiple candidates anyway.

Table of Contents

2. Categorization of ensemble methods
   - General scheme
   - Error decompositions for regression

Process of ensemble generation

$$\begin{tikzcd}\boxed{\text{generation}} \arrow[r] & \boxed{\text{pruning}} \arrow[r] & \boxed{\text{integration}}\end{tikzcd}$$

Generation: generation a set of models using training data.
Pruning: removing redundant models
Integration: combination of outputs of each model to produce final output

Synonyms list:

model	learner, base learner, expert, predictor,regression, classifier
ensemble	committee, multiple models
integration	combination, fusion, selection (if one is selected).

Remarks

Ensemble fitting:

- Generation of models and their fitting is done using the training set.
- Pruning and integration algorithm fitting is done using the validation set
- Ensemble quality is evaluated using the test set.

Diversity

What is a good ensemble?

A good ensemble is the one that consists of accurate predictors that make errors in different regions of input space.

Definition 2

Diversity is a property that characterizes complementarity of base learner errors. It is greater when, all other factors being equal, the classifiers that make incorrect decisions for a given example spread their decisions more evenly over the possible incorrect decisions.

- A lot of diversity definitions exist (see [Zhou, 2012])
- No universal diversity definition yet found.

Ambiguity decomposition

Theorem 3

Let $\widehat{F}(x) = \sum_{k=1}^{K} \alpha_k \widehat{f}_k(x)$, *where* $\sum_{k=1}^{K} \alpha_k = 1$ *and* $\alpha_k \geq 0,\ k = 1, 2, ...K$. *Then for every instance:*

$$(\widehat{F} - f)^2 = \sum_{k=1}^{K} [\alpha_k (\widehat{f}_k - f)^2] - \sum_{k=1}^{K} [\alpha_k (\widehat{f}_k - \widehat{F})^2]$$

Implications:

- ensemble loss is less or equal than weighted loss of individual learners.
- the factor that increases accuracy is ambiguity of individual forecasts.

Shortcoming: when accuracy increases, ambiguity decreases and vice versa.

Bias/variance/covariance decomposition

Regression framework, combiner $\widehat{F}(x) = \frac{1}{M}\sum_{m=1}^{M} \widehat{f}_m(x)$

$$\mathbb{E}\left[(\widehat{F}-y)\right] = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1-\frac{1}{M}\right)\overline{covar}$$

where

$$\overline{bias} = \frac{1}{M}\sum_{m=1}^{M}[\mathbb{E}f_m - f]$$

$$\overline{var} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_i\{[\widehat{f}_i - \mathbb{E}\widehat{f}_i]^2\}$$

$$\overline{covar} = \frac{1}{M(M-1)}\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\mathbb{E}_{i,j}\{[\widehat{f}_i - \mathbb{E}\widehat{f}_i][\widehat{f}_j - \mathbb{E}\widehat{f}_j]\}$$

- Naonori Ueda and Ryohei Nakano, “Generalization error of ensemble estimators”, in IEEE International Conference on Neural Networks, 1996, vol. 1, pp. 90-95.
- Gavin Brown, Diversity in neural network ensembles, Phd thesis, University of Birmingham - United Kingdom, 2004.

Table of Contents

3 Model generation
- Categorization of approaches
- Modifying training dataset
- Features subset selection
- Features transformation
- Output manipulation
- Generation algorithm manipulation

Model generation approaches

- Model generation algorithm may be
  - the same (homogeneous generation)
  - different (heterogeneous generation)
- Heterogeneous generation
  - gives more diverse models
  - it is harder to control diversity
  - it should be controlled by predictions on the pruning stage
- Homogeneous generation
  - by data manipulation
    - subsampling from the training dataset (*bagging*)
    - features manipulation
  - by manipulation of the generation algorithm
- *Model should be unstable with respect to variations in generation.*

Subsampling from the training dataset

different algorithms are trained on different subsets of training data

- works for unstable algorithms with respect to training data: DT, NN, rule learners, MARS, etc.
- Bagging: subsampling of training data with replacement
- K learners trained on K-fold cross-validation subsets
- Adaboost may also be an example (reweighting of samples)

Training data manipulation

Representation of training samples is changed

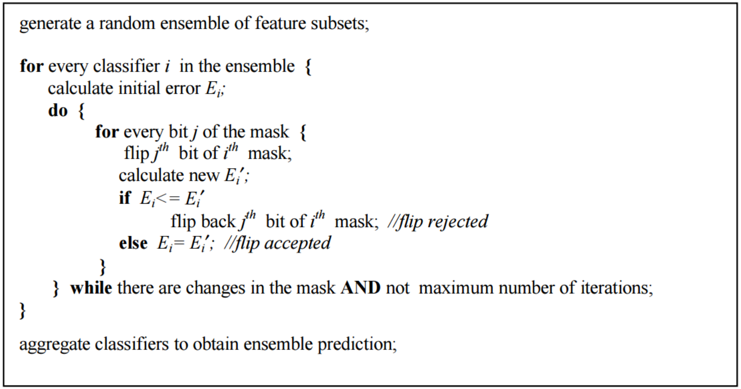
- Features manipulation:
    - Feature subset selection
    - Feature transformation
        - input smearing $X \leftarrow X + \alpha N(0, \widehat{\sigma}_X)$
- Output manipulation:
    - output smearing $Y \leftarrow Y + \alpha N(0, \widehat{\sigma}_Y)$
    - learning new models on errors of previous models

Features subset selection

- features selected from uniform distribution (known as *random subspace method*)
  - RandomForest - uses combination of random subsampling and training set sampling
- the probability to preserve a feature is proportional to feature informativeness
  - less diverse, but more accurate
  - less probability that irrelevant base learners are added
  - successfully applied to K-NN method[1]
- *input smearing:* $x \leftarrow x + \alpha N(0, \widehat{\sigma}_X)$
  - may be combined with bagging
- directed search of feature subsets
  - hill-climbing
  - genetic

[1]Carlotta Domeniconi and Bojun Yan, “Nearest neighbor ensemble”, in International Conference on Pattern Recognition, 2004, vol. 1, pp. 228–231.

Hill climbing

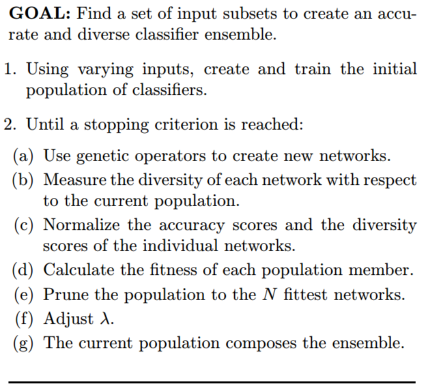


```
generate a random ensemble of feature subsets;

for every classifier $i$ in the ensemble {
    calculate initial error $E_i$;
    do {
        for every bit $j$ of the mask {
            flip $j^{th}$ bit of $i^{th}$ mask;
            calculate new $E_i'$;
            if $E_i <= E_i'$
                flip back $j^{th}$ bit of $i^{th}$ mask; //flip rejected
            else $E_i = E_i'$; //flip accepted
        }
    } while there are changes in the mask AND not maximum number of iterations;
}

aggregate classifiers to obtain ensemble prediction;
```

E_i may denote any loss of model i - it may include combinations of accuracy, ambiguity, diversity and other performance metrics.

Gabriele Zenobi and P´adraig Cunningham, "Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error", in European Conference on Machine Learning. 2001, vol. LNCS 2167, pp. 576–587, Springer.

Search with genetic algorithms



GOAL: Find a set of input subsets to create an accurate and diverse classifier ensemble.

1. Using varying inputs, create and train the initial population of classifiers.
2. Until a stopping criterion is reached:
   (a) Use genetic operators to create new networks.
   (b) Measure the diversity of each network with respect to the current population.
   (c) Normalize the accuracy scores and the diversity scores of the individual networks.
   (d) Calculate the fitness of each population member.
   (e) Prune the population to the N fittest networks.
   (f) Adjust λ.
   (g) The current population composes the ensemble.

Fitness of model *i* is measured as

$$Fitness_i = Accuracy_i + \lambda Diversity_i$$

David W. Opitz, "Feature selection for ensembles", in 16th National Conference on Artificial Intelligence, Orlando - U.S.A., 1999, pp. 379–384, AAAI Press.

Rotation forest

```
INPUT: design matrix $X$, outputs $Y$, number of models $M$.

ALGORITHM:
for each model $m = 1, 2, ...M$:
    divide feature space into $K$ disjoint regions $S_1, S_2, ...S_K$.
    for each $k = 1, 2, ...K$:
        $X_m \leftarrow X[:, S_k]$
        $X_m^{ss}, Y^{ss} \leftarrow$ random subsampling of $(X_m, Y)$ // optional step
        $P_m \leftarrow PCA(X_m^{ss})$
        train $f_i$ on $(P_m, Y)$

PREDICTION for new observation $x$:
    for each model $m = 1, 2, ...M$:
        $x_m \leftarrow x[:, S_k]$
        $p_m \leftarrow PCA_m(x_m)$
        $\hat{y}_m = f_m(p_m)$
    output $\frac{1}{M} \sum_{m=1}^{M} \hat{y}_m$
```

PCA is equivalent to rotation, hence the name.

Forward stagewise additive modeling

Input: training dataset (x_i, y_i), $i = 1, 2, ...n$; loss function $L(f, y)$, general form of additive classifier $h(x, \gamma)$ (dependent from parameter γ) and the number M of successive additive approximations.

1. Fit initial approximation $f^0(x)$ (might be taken $f^0(x) \equiv 0$)
2. For $m = 1, 2, ...M$:
   1. find next best classifier
   $$(c_m, \gamma_m) = \arg\min \sum_{i=1}^{n} L(f_{m-1}(x_i) + c_m h(x, \gamma_m),\ y_i)$$
   2. set
   $$f_m(x) = f_{m-1}(x) + c_m h(x, \gamma_m)$$

Output: approximation function $f^M(x) = f^0(x) + \sum_{j=1}^{M} c_j h(x, \gamma_m)$

Adaboost algorithm is obtained for $L(y, f(x)) = e^{-yf(x)}$

Adaboost (discrete version)

Assumptions: loss function $L(y, f(x)) = e^{-yf(x)}$, classification task: $y \in \}$
Input: training dataset (x_i, y_i), $i = 1, 2, ...n$; number of additive weak classifiers M, a family of weak classifiers $h(x)$, outputting only +1 or -1 (binary classification) and trainable on weighted datasets.

1. Initialize observation weights $w_i = 1/n$, $i = 1, 2, ...n$.
2. for $m = 1, 2, ...M$:
    1. fit $h^m(x)$ to training data using weights w_i
    2. compute weighted misclassification rate:
    $$E_m = \frac{\sum_{i=1}^n w_i \mathbb{I}[h^m(x) \neq y_i]}{\sum_{i=1}^n w_i}$$
    3. compute $\alpha_m = \ln((1 - E_m)/E_m)$
    4. increase all weights, where misclassification with $h^m(x)$ was made:
    $$w_i \leftarrow w_i e^{\alpha_m},\ i \in \{i :\ h^m(x_i) \neq y_i\}$$

Output: composite classifier $f(x) = \operatorname{sign}\left(\sum_{m=1}^M \alpha_m h^m(x)\right)$

Gradient boosting of trees - regression

Input: training dataset (x_i, y_i), $i = 1, 2, ...n$; loss function $L(f, y)$ and the number M of successive additive approximations.

1. Fit constant initial approximation $f^0(x)$:
   $f^0(x) = \arg\min_\gamma \sum_{i=1}^n L(\gamma,\ y_i)$
2. For each step $m = 1, 2, ...M$:
   1. calculate derivatives $z_i = -\frac{\partial L(r,y)}{\partial r}|_{r=f^{m-1}(x)}$
   2. train regression tree h^m on (x_i, z_i), $i = 1, 2, ...n$ with squared loss function $\sum_{i=1}^n (h^m(x_i) - z_i)^2$ and extract terminal regions R_{jm}, $j = 1, 2, ...J_m$.
   3. for each terminal region R_{jm}, $j = 1, 2, ...J_m$ solve univariate optimization problem:
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(f^{m-1}(x_i) + \gamma,\ y_i)$$
   4. update $f^m(x) = f^{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{I}[x \in R_{jm}]$

Output: approximation function $f^M(x)$

Generation algorithm manipulation

- Manipulation of parameter sets - apply learning algorithms with different parameters
  - K-NN: K
  - NN: initial weights, number of nodes, number of layers
  - etc.
- Manipulation of induction algorithm - change the way model induction is done
  - Negative correlation algorithm
    - add correlation with ensemble prediction to loss function
  - score correct outputs, when ensemble was wrong, more:

$$score_m = \sum_{n=1}^{N} \mathbb{I}[f_m(x_n) = y_n] / \max(1, \sum_{i=1}^{M} \mathbb{I}[f_i(x_n) = y_n])$$

$$loss_m = |F(x_n) - y_n|^{\lambda} (f_i(x_n) - y_n)^2$$

Negative correlation algorithm

MSE loss $[F(x)-y]^2$ is augmented with $\lambda(f_i(x)-F(x))\sum_{j\neq i}(f_j(x)-F(x))$.

```
INPUT:
   training set $(x_1, y_1), ...(x_N, y_N)$
   $M$: number of predictors required

ALGORITHM:
   while not converged:
      for each training pattern $x_n, y_n$ in training set:
         calculate $F(x) = \frac{1}{M}\sum_i f_i(x_n)$
            for each network $i = 1, 2, ...M$:
               for each weight in network:
                  perform single update, using:
                  $e_i = \frac{1}{2}(f_i(x_n)-y_n)^2 + \lambda(f_i(x_n)-F(x_n))\sum_{j\neq i}(f_j(x_n)-F(x_n))$

PREDICTION:
   output $F(x) = \frac{1}{M}\sum_i f_i(x_n)$
```

Gavin Brown and Jeremy L. Wyatt. Managing Diversity In Regression Ensembles. 2005. Journal of Machine Learning Research 6 1621–1650.

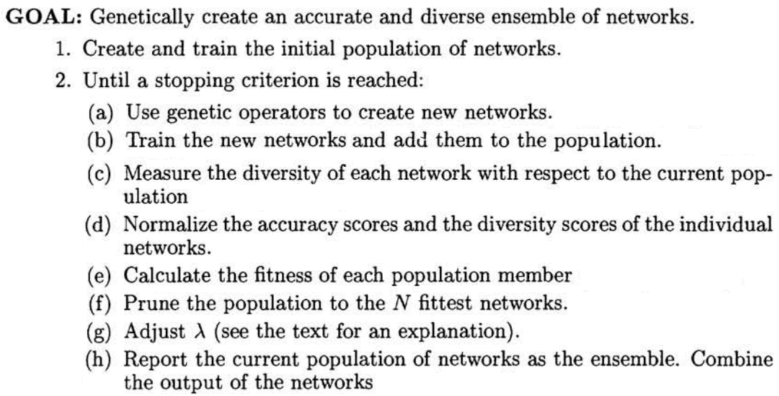
Cooperative Neural Network Ensembles

Cooperative Neural Network Ensembles - constructive heuristic ensemble building:

```
start with randomly initialized ensemble of 2 NN.
while ensemble validation error is not acceptamble:
   for each NN:
      train NN
   if all NN sufficiently trained
      add hidden nodes to NN
   if all NN sufficiently expanded
      add new NN
   if maximum number of iteration reached
      stop
```

- Md. Monirul Islam, Xin Yao, and Kazuyuki Murase, “A constructive algorithm for training cooperative neural network ensembles”, IEEE Transactions on Neural Networks, vol. 14, no. 4, pp. 820–834, 2003.

ADDEMUP algorithm



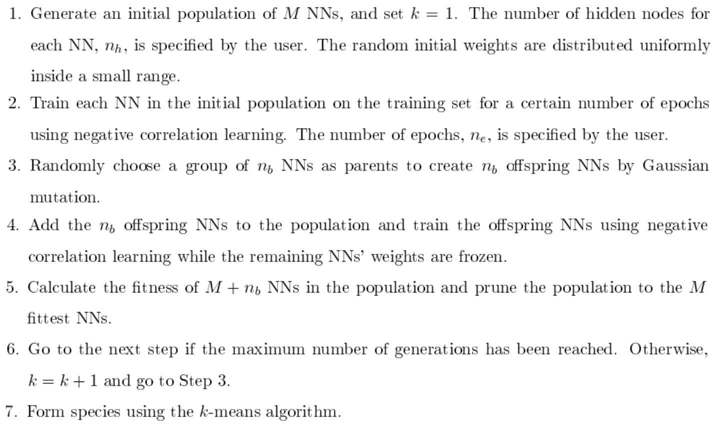
GOAL: Genetically create an accurate and diverse ensemble of networks.

1. Create and train the initial population of networks.
2. Until a stopping criterion is reached:
    - (a) Use genetic operators to create new networks.
    - (b) Train the new networks and add them to the population.
    - (c) Measure the diversity of each network with respect to the current population
    - (d) Normalize the accuracy scores and the diversity scores of the individual networks.
    - (e) Calculate the fitness of each population member
    - (f) Prune the population to the N fittest networks.
    - (g) Adjust λ (see the text for an explanation).
    - (h) Report the current population of networks as the ensemble. Combine the output of the networks

Optimized loss of individual model is weighted by ensemble performance.
Networks are combined using accuracy dependent weights.

David W. Opitz and Jude W. Shavlik, "Generating accurate and diverse members of a neural-network ensemble", Advances in Neural Information Processing Systems, vol. 8, pp. 535–541, 1996.

Evolutionary ensembles with negative correlation



1. Generate an initial population of M NNs, and set $k = 1$. The number of hidden nodes for each NN, n_h, is specified by the user. The random initial weights are distributed uniformly inside a small range.
2. Train each NN in the initial population on the training set for a certain number of epochs using negative correlation learning. The number of epochs, n_e, is specified by the user.
3. Randomly choose a group of n_b NNs as parents to create n_b offspring NNs by Gaussian mutation.
4. Add the n_b offspring NNs to the population and train the offspring NNs using negative correlation learning while the remaining NNs' weights are frozen.
5. Calculate the fitness of $M + n_b$ NNs in the population and prune the population to the M fittest NNs.
6. Go to the next step if the maximum number of generations has been reached. Otherwise, $k = k + 1$ and go to Step 3.
7. Form species using the k-means algorithm.

Genetic operation contain only mutation - adding gaussian noise to weights of NN.
Exchange of information - through *negative correlation* approach and weighted accuracy (success score for x_i=1 / #[models that were correct]).

Yong Liu, Xin Yao, and Tetsuya Higuchi, "Evolutionary ensembles with negative correlation learning", IEEE Transactions on Evolutionary Computation, vol. 4, no. 4, pp. 380–387, 2000.

Not mentioned before

- Random forest
- Extra random trees

- Not covereded topics:
  - Bayesian model averaging
  - Making ensembles from infinite model space

Table of Contents

General information

- Ensemble pruning is used to:
    - decrease computational costs
    - increase accuracy
    - avoid multi-collinearity problem
- Very similar to feature selection
    - forecasts have the same scale: no standardization needed
    - very correlated: estimation of combiner may be unstable
- Pruning a set of M models requires considering $2^M - 1$ variants.
    - unfeasible for $M > 30$

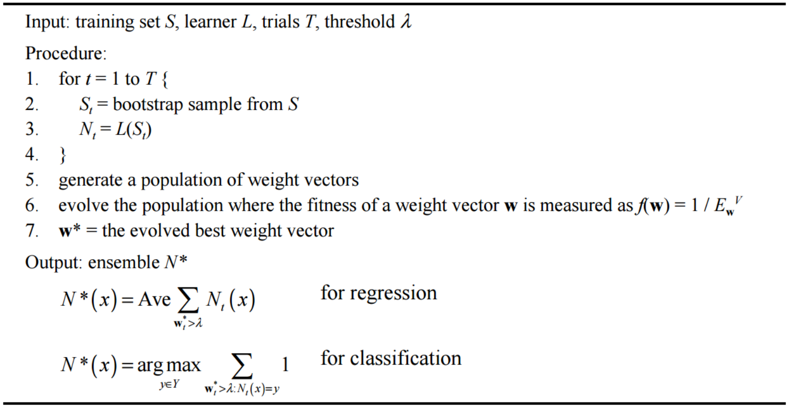
Algorithms

- Consider metric
  - global: ensemble accuracy, diversity
  - average pairwise metric: correlation, diversity
- Optimize metric
  - through full search (for small M)
  - suboptimal greedy search
    - top K from list (ordered by accuracy, diversity or combination)
    - with score above pre-specified threshold
    - forward, backward, forward-backward
    - genetic
- Number of models M
  - predefined as parameter
  - auto-defined (process stops when validation accuracy stops to improve)
- Many algorithms not included here, see [Zhou, 2012] for more.

Forward and backward suboptimal search

- Score measure:
  - individual accuracy of model
  - accuracy of ensemble when given model is added to it
  - diversity with respect to the current ensemble
- Forward-backward:
  - iteratively one model is added (with maximum score) and one discarded (with minimum score)
  - process stops when the same model was added and discarded
- Forward-replacement:
  - iteratively one model is added (with maximum score) and one worst model substituted with the best not included model.

GASEN - genetic optimization algorithm



```
Input: training set $S$, learner $L$, trials $T$, threshold $\lambda$
Procedure:
1. for $t = 1$ to $T$ {
2.     $S_t$ = bootstrap sample from $S$
3.     $N_t = L(S_t)$
4. }
5. generate a population of weight vectors
6. evolve the population where the fitness of a weight vector $\mathbf{w}$ is measured as $f(\mathbf{w}) = 1 / E_{\mathbf{w}}^{V}$
7. $\mathbf{w}^*$ = the evolved best weight vector
Output: ensemble $N^*$
    $N^*(x) = \text{Ave} \sum_{\mathbf{w}_i^* > \lambda} N_t(x)$          for regression
    $N^*(x) = \arg\max_{y \in Y} \sum_{\mathbf{w}_i^* > \lambda : N_t(x) = y} 1$          for classification
```

Pruning is performed by leaving models with weights above threshold.

Zhi-Hua Zhou, Jianxin Wu, and Wei Tang, "Ensembling neural networks: many could be better than all", Artificial Intelligence, vol. 137, pp. 239–263, 2002.

Clustering pruning approach

- **Clustering pruning** groups models into clusters with respect to:
  - their predictions
  - their prediction similarity (e.g. correlations)
- Approaches:
  - from each cluster select most single model
    - most accurate
    - most representative (centroid model)
    - most distant model from other clusters
  - iteratively select most accurate models from each cluster until ensemble accuracy stops to improve

Table of Contents

Fixed combiner at class level

Output of base learner k

Exact class: ω_1 or ω_2.

Combiner predicts ω_1 if:

- all classifiers predict ω_1 (AND rule)
- at least one classifier predicts ω_1 (OR rule)
- at least k classifiers predict ω_1 (k-out-of-N)
- majority of classifiers predict ω_1 (majority vote)

Each classifier may be assigned a weight, based on its performance:

- weighted majority vote
- weighted k-out-of-N (based on score sum)

Fixed combiner - ranking level

Output of base learner k

Ranking of classes:

$$\omega_{k_1} \succeq \omega_{k_2} \succeq \ldots \succeq \omega_{k_C}$$

Ranking is equivalent to scoring of each class (with incomparable scoring between classifiers).

Definition 3

Let $B_k(i)$ be the count of classes scored below ω_i by classifier k. **Borda count $B(i)$ of class ω_i** is the total number of classes scored below ω_i by all classifiers:

$$B(i) = \sum_{k=1}^{K} B_k(i)$$

Combiner predicts ω_i where $i = \arg\max_i B(i)$

Fixed combiner at class probability level

Output of base learner k

Vectors of class probabilities:

$$[p^k(\omega_1),\ p^k(\omega_2),\ \dots\ p^k(\omega_C)]$$

Combiner predicts ω_i if $i = \arg\max_i F(p^1(\omega_i), p^2(\omega_i), \dots p^K(\omega_i))$

- mean (equivalent to sum), median.
- product, min, max - possible but less robust

Error correcting codes

- Used in classification
- Each class ω_i is coded as a binary codeword W_i consisting of B bits:

$$\omega_i \to W_i$$

- Minimum sufficient amount of bits to code C classes is $\lceil \log_2 C \rceil$
- Given x, B binary classifiers predict each bit of the class codeword.
- Class is predicted as

$$\hat{c}(x) = \arg\min_{c} \sum_{b=1} |W_{cb} - \widehat{p}_b(x)|$$

- where W_{cb} is the b-th bit of codeword, corresponding to class c.
- More bits are used to make classification more robust to errors of individual binary classifiers.
- Codewords are selected to have maximum mutual Hamming distance or randomly.

Linear combination

Ensemble of regressions / scores for classification:

$$F(x) = \sum_{m=1}^{M} w_m(x) f_m(x)$$

- Type of weights:
  - constant $w_m(x) \equiv w_m$
  - dependent on x
- Problem - multi-collinearity of $f_m(x)$
  - may be solved at generation or pruning step or with regularization

Constant weights

$$F(x) = \sum_{m=1}^{M} w_m f_m(x)$$

- Methods:
  - Basic ensemble method (BEM): $w_1 = w_2 = ... = w_M = \frac{1}{M}$
  - find w_m by regressing y_n on $f_1(x), f_2(x), ...f_m(x)$ on the validation set
    - may add constraints $\sum w_m = 1$ (get *generalized ensemble method*)
    - may add constraints $w_m \geq 0,\ m = 1, 2, ...M.$
    - may add constant term
  - Problem of regression: multi-collinearity of $f_1(x), ...f_M(x)$

Solving multi-collinearity problem

- Add constraints $w_m \geq 0,\ m = 1, 2, \dots M$.
- Add regularization (e.g. $\sum_m (w_m - \frac{1}{M})^2$)
- Use suboptimal search: evolutionary search or gradient descent search, average multiple optimization results.
- Integrate weights estimation into greedy ensemble forward selection procedure:
  - how many times each model was selected indicates the respective weight
- Use principal component regression:
  1. convert X to PCA representation Z
  2. find regression coefficients for Z
  3. using found coefficients and principal directions recover w_m

  - needs tuning of the number of principal components

Finding constant weights

Weighted averaging combiner

$$f(x) = \sum_{k=1}^{K} w_k f_k(x)$$

Naive fitting

$$\widehat{w} = \arg\min_{w} \sum_{i=1}^{N} \mathcal{L}(y_i, \sum_{k=1}^{K} w_k f_k(x_i))$$

will overfit. The mostly overfitted method will get the most weight.

Linear stacking

- Let training set $\{(x_i, y_i), i = 1, 2, ...N\}$ be split into M folds.
- Denote $fold(i)$ to be the fold, containing observation i
- Denote $f_k^{-fold(i)}$ be predictor k trained on all folds, except $fold(i)$.

Definition

Linear stacking (or stacked generalization) is weighted averaging combiner, where weights are found using

$$\widehat{w} = \arg\min_{w} \sum_{i=1}^{N} \mathcal{L}(y_i, \sum_{k=1}^{K} w_k f_k^{-fold(i)}(x_i))$$

5 Integration of forecasts
- Fixed integration schemes for classification
- Weighted linear integration with constant weights
- Weighted linear integration with variable weights
- Additional categorization of ensemble methods

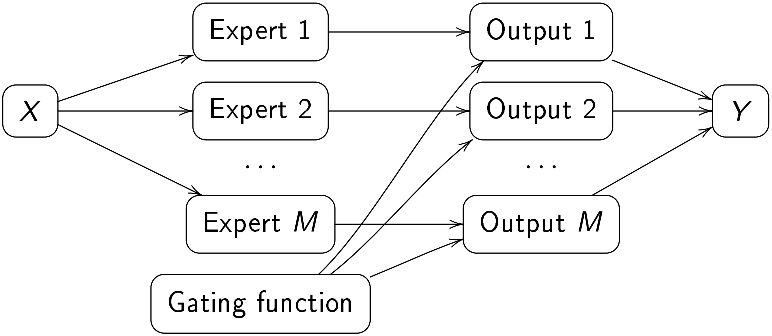
Non-constant weighting types

- *areas of expertise*: input space is divided into regions where weights are constant before prediction
    - example: meta decision trees
- weights estimated using given parametric function
    - example: mixture of experts, generalized stacking
- dynamic approach:
    1. given predicted object x, find K nearest neighbors to it.
    2. weight base predictors by their performance on nearest neighbours

Meta decision trees

- Classification task
- Decision tree is built
- Instead of classes the tree predicts which classifier to use
- Instead of initial features, meta features are used
  - $p(\omega_i|f_m, x)$, $i = 1, 2, \dots C$, $m = 1, 2, \dots M$
  - $\max_i p(\omega_i|f_m, x)$, $m = 1, 2, \dots M$.
  - $-\sum p(\omega_i|f_m, x) \ln p(\omega_i|f_m, x)$, $m = 1, 2, \dots M$.
- Initial features can be added to meta-features
- Minimized impurity function: $Imp(S) = \min_m [error\ rate(S|f_m)]$, where S is the data subset falling inside the node.
- Meta-decision tree is domain independent (depends only on base learners if only meta-features used)

Mixture of experts



- Gating function provides averaging weights $w_m(x, \theta_{GF})$:

$$F(x) = \sum_{m=1}^{M} w_m(x, \theta_{GF}) f_m(x, \theta_m)$$

Mixture of experts

- Parameters of gating function θ_{GF} and of individual experts θ_m are estimated using EM procedure.
- Weights are calculated using soft-max (with smoothing parameter γ):

$$w_m(x, \theta_{GF}) = \frac{\exp(\gamma g(x, \theta_m))}{\sum_{i=1}^{M} \exp(\gamma g(x, \theta_i))}$$

- $g(x, \theta_m)$ specify area of competence of model m, examples: $g(x, \theta_m) = x^T \theta_m$, $g(x, \alpha_m, \theta_m) = exp(-\alpha_m ||x - \theta_m||)$, etc.
- Weight $w_m(x, \theta_{GF})$ means the probability that output for x would be generated by m-th expert
    - $w_m(x_n)$ specifies the weight of instance (x_n, y_n) while training expert m.
    - $w_m(x)$ specifies the contribution of expert m into final prediction
- As the same training sample is weighted differently for each expert, the problem of multi-collinearity is lesser (each expert specializes on its own subsample of data).

Generalized stacking

Definition

Generalized stacking is prediction

$$f(x) = A_\theta \left(f_1(x), f_2(x), \dots f_K(x)\right),$$

where A is some general form predictor and θ is a vector of parameters, estimated by

$$\widehat{\theta} = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}\left(y_i,\, A_\theta\left(f_1^{-fold(i)}(x), f_2^{-fold(i)}(x), \dots f_K^{-fold(i)}(x)\right)\right)$$

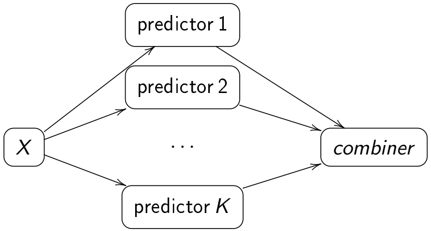
- Stacking is the most general approach
- It is a winning strategy in most ML competitions.
- $f_i(x)$ may be:
  - class number (coded using one-hot encoding).
  - vector of class probabilities
  - any initial or generated feature

5. Integration of forecasts
   - Fixed integration schemes for classification
   - Weighted linear integration with constant weights
   - Weighted linear integration with variable weights
   - Additional categorization of ensemble methods

Feature space

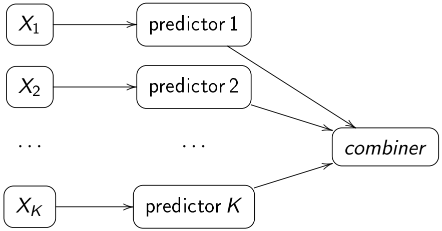
Common feature space X
Further categorization:

- different methods (stacking, blending)
- the same method applied to different data (bagging, boosting).
- the same method with different initialization (neural networks, EM)



Different feature spaces X_1, X_2, ...X_K:
Example: person identification using

- signature
- face detection
- voice identification



Level of combination

Classification pipeline:

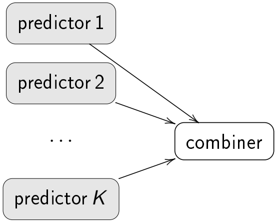
$$\begin{tikzcd}\boxed{\text{raw data}} \arrow[r] & \boxed{\text{preprocessed data}} \arrow[r] & \boxed{\text{classifier}} \arrow[r] & \boxed{\text{decision}}\end{tikzcd}$$

Any of this data can be used in combiner.

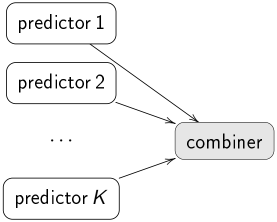
- **Raw data** - combiner is a usual classifier
    - may need to operate in high dimensional space
    - needs implicit feature selection/extraction (such as DT, NN, methods with regularization)
- **Preprocessed data**
    - PCA, other dimensionality reduction techniques
    - coding of object (vector quantisation, clustering)
    - probability of each class
- **Classifier output** - class number.

Degree of training

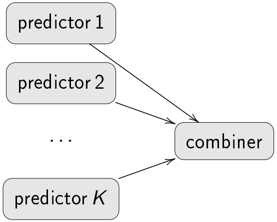
Predictors trainable, combiner fixed:



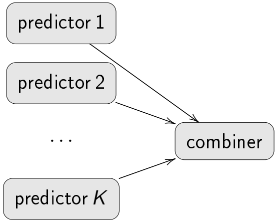
Predictors fixed, combiner trainable:



Predictors trained, then combiner trained:



Predictors and combiner trained jointly:



References

Keith D. Copsey Andrew R. Webb.
Statistical Pattern Recognition.
Wiley, 3rd edition, 2011.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman.
The elements of statistical learning: data mining, inference and prediction.
Springer, 2 edition, 2008.

João Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa.
Ensemble approaches for regression: A survey.
ACM Comput. Surv., 45(1):10:1–10:40, December 2012.

Zhi-Hua Zhou.
Ensemble Methods: Foundations and Algorithms.
Chapman & Hall/CRC, 1st edition, 2012.

Proof of theorem (1)

$\mathcal{L}$ is convex in the first argument, since

$$\mathcal{L}(\lambda\widehat{y}^1 + (1-\lambda)\widehat{y}^2, y) = G(\lambda\widehat{y}^1 + (1-\lambda)\widehat{y}^2 - y) =$$
$$G(\lambda(\widehat{y}^1 - y) + (1-\lambda)(\widehat{y}^2 - y)) \leq \lambda G(\widehat{y}^1 - y) + (1-\lambda)G(\widehat{y}^2 - y)$$
$$= \lambda\mathcal{L}(\widehat{y}^1, y) + (1-\lambda)\mathcal{L}(\widehat{y}^2, y)$$

By induction $\forall K > 2$, $\forall \lambda_1, \lambda_2, ...\lambda_K : \lambda_i \geq 0 \, \forall i$, $\sum_{i=1}^K \lambda_i = 1$

$$\mathcal{L}(\sum_{k=1}^K \lambda_k \widehat{y}^k, y) \leq \sum_{k=1}^K \lambda_k \mathcal{L}(\widehat{y}^k, y)$$

Since last equation holds for arbitrary $y, \widehat{y}^1, \widehat{y}^2, ...\widehat{y}^K$, it holds for $y_n, \widehat{y}_n^1, \widehat{y}_n^2, ...\widehat{y}_n^K$ for $n = 1, 2, ...N$.

By summing over n, we obtain $L_{comb} \leq \sum_{k=1}^K \lambda_k L_k$.

Proof of bias-variance decomposition (2)

$$\begin{aligned}\mathbb{E}\left(\widehat{f}-f\right)^2 = E\left(\widehat{f}-\mathbb{E}\widehat{f}+\mathbb{E}\widehat{f}-f\right) = \mathbb{E}\left(\widehat{f}-\mathbb{E}\widehat{f}\right)^2+\left(\mathbb{E}\widehat{f}-f\right)^2 \\ +2\mathbb{E}\left[(\widehat{f}-\mathbb{E}\widehat{f})(\mathbb{E}\widehat{f}-f)\right] \\ = \mathbb{E}\left(\widehat{f}-\mathbb{E}\widehat{f}\right)^2+\left(\mathbb{E}\widehat{f}-f\right)^2\end{aligned}$$

We used that $(\mathbb{E}\widehat{f}-f)$ is a constant number and hence $\mathbb{E}\left[(\widehat{f}-\mathbb{E}\widehat{f})(\mathbb{E}\widehat{f}-f)\right] = (\mathbb{E}\widehat{f}-f)\mathbb{E}(\widehat{f}-\mathbb{E}\widehat{f}) = 0.$

$$\begin{aligned}\mathbb{E}\left(\widehat{f}-y\right)^2 &= \mathbb{E}\left(\widehat{f}-f-\varepsilon\right)^2 = \mathbb{E}\left(\widehat{f}-f\right)^2+\mathbb{E}\varepsilon^2-2\mathbb{E}\left[(\widehat{f}-f)\varepsilon\right] \\ &= \mathbb{E}\left(\widehat{f}-\mathbb{E}\widehat{f}\right)^2+\left(\mathbb{E}\widehat{f}-f\right)^2+\sigma^2\end{aligned}$$

We used that $\mathbb{E}\left[(\widehat{f}-f)\varepsilon\right] = \mathbb{E}\left[(\widehat{f}-f)\right]\mathbb{E}\varepsilon = 0$ since ε is independent of x.

Proof of ambiguity decomposition (3)

$$\begin{aligned}
\sum_{k=1}^{K}\alpha_k(f_k-y)^2 &= \sum_{k=1}^{K}\alpha_k(f_k-F+F-y) \\
&= \sum_{k=1}^{K}\alpha_k\left[(f_k-F)^2+(F-y)^2+2(f_k-f)(f-y)\right] \\
&= \sum_{k=1}^{K}\alpha_k\,(f_k-F)^2+(F-y)^2
\end{aligned}$$

since, by definition $F=\sum_{k=1}^{K}\alpha_k f_k$. So it follows that

$$(F-y)^2=\sum_{k=1}^{K}\alpha_k(f_k-y)^2-\sum_{k=1}^{K}\alpha_k\,(f_k-F)^2$$