Problem Solving with AI Techniques
Ensemble Methods

Paul Weng

UM-SJTU Joint Institute

VE593, Fall 2018

1 Motivation

2 Bagging

3 Boosting

4 Stacking

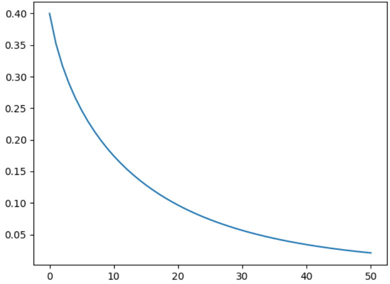
Ensemble Methods

- Idea: Several models (=ensemble) are better than one!
  - Learn several models
  - Combine their predictions (e.g., majority vote, average)
- Motivations:
  - **Variance reduction**: combined models less dependent on training set
  - **Bias reduction**: errors made by one model may be cancelled by another
  - Combined models may be more expressive
  - ML competitions (e.g., Netflix, Yahoo, HiggsML) won with such methods
- Ensemble methods = meta-algorithms to be used on top of other models

Classification: Why This Idea May Work?

- Using an ensemble may help reduce error rate
- Assume you have trained $2n+1$ classifiers, each with **independent** error rate $\varepsilon = 0.4$
- Error rate of ensemble of $2n+1$ classifiers with majority rule:

$$\sum_{k=n+1}^{2n+1} \binom{2n+1}{k} \varepsilon^k (1-\varepsilon)^{2n+1-k}$$



- Note: in practice, error rates are not independent

Regression: Why This Idea May Work?

- Averaging an ensemble's predictions may help reduce variance
- Assume you have trained n regressors $h_1, \dots, h_n$, each with MSE:

$$\int_{\mathcal{X}} (h_i(x) - f(x))^2 \mu(x) dx = \int_{\mathcal{X}} \varepsilon_i(x)^2 \mu(x) dx$$

- MSE of the average prediction:

$$\begin{aligned} \int_{\mathcal{X}} \Big(\frac{1}{n}\sum_i h_i(x) - f(x)\Big)^2 \mu(x) dx &= \int_{\mathcal{X}} \Big(\frac{1}{n}\sum_i \varepsilon_i(x)\Big)^2 \mu(x) dx \\ &\leq \frac{1}{n}\sum_i \int_{\mathcal{X}} \varepsilon_i(x)^2 \mu(x) dx \end{aligned}$$

- Note: if ε_i's are independent with zero mean, MSE is reduced by $1/n$

Bagging: Principle

- Issue: Independent models can be obtained by training them on different i.i.d datasets, but we have only one dataset
- Bootstrap (statistics): Use training dataset $\mathcal{D}$ as an approx. of true distribution μ
- Bagging = Bootstrap aggregating
- Basic procedure:
  - Generate $\mathcal{D}_1, \dots, \mathcal{D}_K$ by random sampling with replacement from $\mathcal{D}$
  - For each $\mathcal{D}_k$, train a model h_k
  - Combine $h_1, \dots, h_K$ as final model
- Random forest = bagging with decision trees

Bootstrap Sampling

- Example of sampling with replacement:

- Probability of data point not being selected in $\mathcal{D}_k$:
  $(1 - \frac{1}{N})^N \to e^{-1} \approx 0.3678$
- Therefore, less than 2/3 (i.e., 63.2%) of data points in $\mathcal{D}$ is any $\mathcal{D}_k$

Boosting: Principle

- Idea:
  - Train models sequentially
  - Later models focus on instances mispredicted by earlier models
  - Weighted combination of models with weights given by their errors

- Can reduce bias and variance
- Base model here called weak learner
- How to focus on mispredicted instances?

Adaboost

- Adaboost = Adaptive Boosting
- Assume a classification problem where $h_t : \mathcal{X} \to \{-1, 1\}$
- Idea: Increase probabilities of mispredicted instances (and therefore decrease those of well-classified instances) and train next model on weighted instances

- Procedure: Start with $\hat{\mu}_0$ uniform distribution over $\mathcal{D}$ and repeat:
  - Train model h_t with distribution $\hat{\mu}_t$
    $\Rightarrow \varepsilon_t = \mathbb{P}(h_t(\boldsymbol{X}) \neq Y \mid (\boldsymbol{X}, Y) \sim \hat{\mu}_t)$
  - Compute $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\epsilon_t}$
  - $\hat{\mu}_{t+1}(\boldsymbol{x}, y) \propto \hat{\mu}_t(\boldsymbol{x}, y) \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(\boldsymbol{x}) = y \\ e^{\alpha_t} & \text{if } h_t(\boldsymbol{x}) \neq y \end{cases}$

Bagging vs Boosting

	Bagging	Boosting
Dataset $\mathcal{D}_k$ for h_k	Boostrap sample	Focuses on difficult instances
Independence of h_k's	Yes	No
Final model	Linear combination	Linear combination
Weights	Uniform weights	Depend on performance
Variance reduction	Yes	Yes
Bias reduction	No	Yes

Gradient Boosting

- Gradient boosting: adding a new model is seen as a gradient step
- Idea: Train next model on gradient of loss function
  (e.g. $\frac{\partial \ell(y,y')}{\partial y'} = y - y'$ for squared loss)
- Procedure: Repeat with $\mathcal{D}_1 = \mathcal{D}$:
  - Train model h_t on $\mathcal{D}_t$
  - Compute $\mathcal{D}_{t+1} = \{(\boldsymbol{x}, \frac{\partial \ell(y,h_t(y))}{\partial y'}) \,|\, (\boldsymbol{x}, y) \in \mathcal{D}_t\}$

- Adaboost is a special case of Gradient Boosting with exp. loss
- XGBoost = gradient boosting with decision trees

Stacking: Principle

- Can we do better than averaging or voting?
- Idea: Learn to combine the prediction of ensemble
- Basic procedure for two-level stacking:
  - Split $\mathcal{D}$ in K-folds: $\mathcal{D}_1, \ldots, \mathcal{D}_K$
  - For each fold $\mathcal{D}_k$
    - Train n models on $\mathcal{D}_{-k}$
    - Predict on $\mathcal{D}_k$ to obtain $\mathcal{D}'_k = \{\big((h_1(\boldsymbol{x}), \ldots, h_K(\boldsymbol{x})), y\big) \mid (\boldsymbol{x}, y) \in \mathcal{D}_k\}$
  - Train a model on $\cup_k \mathcal{D}'_k$