Cross Validation & Ensembling

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

Machine Learning

Outline

1. **Cross Validation**
   - How Many Folds?
2. **Ensemble Methods**
   - Voting
   - Bagging
   - Boosting
   - Why AdaBoost Works?

Outline

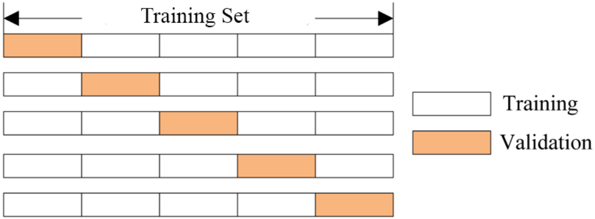
1. **Cross Validation**
   - How Many Folds?
2. **Ensemble Methods**
   - Voting
   - Bagging
   - Boosting
   - Why AdaBoost Works?

Cross Validation

- So far, we use the ***hold out*** method for:
  - Hyperparameter tuning: validation set
  - Performance reporting: testing set
- What if we get an "unfortunate" split?

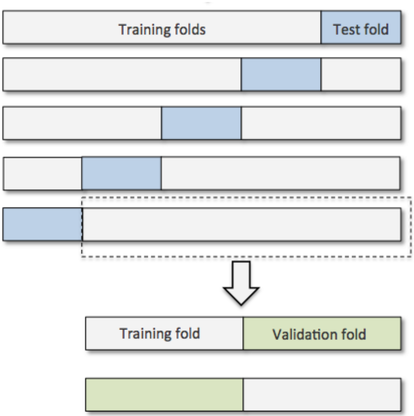
Cross Validation

- So far, we use the ***hold out*** method for:
  - Hyperparameter tuning: validation set
  - Performance reporting: testing set
- What if we get an "unfortunate" split?
- ***K-fold cross validation***:
  1. Split the data set $\mathbb{X}$ evenly into K subsets $\mathbb{X}^{(i)}$ (called ***folds***)
  2. For $i = 1, \cdots, K$, train $f_{-N^{(i)}}$ using all data but the i-th fold ($\mathbb{X} \backslash \mathbb{X}^{(i)}$)
  3. Report the ***cross-validation error*** C_{CV} by averaging all testing errors $C[f_{-N^{(i)}}]$'s on $\mathbb{X}^{(i)}$



Nested Cross Validation

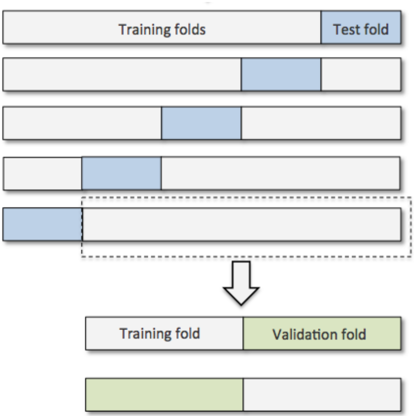
- Cross validation (CV) can be applied to ***both*** hyperparameter tuning and performance reporting



- E.g, 5×2 ***nested CV***

Nested Cross Validation

- Cross validation (CV) can be applied to ***both*** hyperparameter tuning and performance reporting

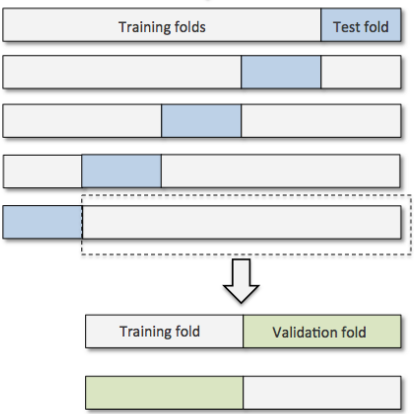


- E.g, 5×2 ***nested CV***

1. Inner (2 folds): select hyperparameters giving lowest C_{CV}
   - Can be wrapped by grid search

Nested Cross Validation

- Cross validation (CV) can be applied to ***both*** hyperparameter tuning and performance reporting

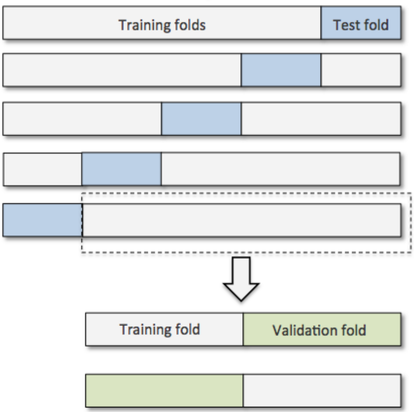


- E.g, 5×2 ***nested CV***

1. Inner (2 folds): select hyperparameters giving lowest C_{CV}
   - Can be wrapped by grid search
2. Train final model using ***both*** training and validation sets with the selected hyperparameters

Nested Cross Validation

- Cross validation (CV) can be applied to ***both*** hyperparameter tuning and performance reporting



- E.g, 5×2 ***nested CV***

1. Inner (2 folds): select hyperparameters giving lowest C_{CV}
    - Can be wrapped by grid search
2. Train final model using ***both*** training and validation sets with the selected hyperparameters
3. Outer (5 folds): report C_{CV} as test error

Outline

1. **Cross Validation**
   - How Many Folds?
2. **Ensemble Methods**
   - Voting
   - Bagging
   - Boosting
   - Why AdaBoost Works?

How Many Folds K? I

How Many Folds K? I

- The cross-validation error C_{CV} is an average of $C[f_{-N^{(i)}}]$'s

How Many Folds K? I

- The cross-validation error C_{CV} is an average of $C[f_{-N^{(i)}}]$'s
- Regard each $C[f_{-N^{(i)}}]$ as an estimator of the expected generalization error $\mathrm{E}_{\mathbb{X}}(C[f_N])$

How Many Folds K? I

- The cross-validation error C_{CV} is an average of $C[f_{-N^{(i)}}]$'s
- Regard each $C[f_{-N^{(i)}}]$ as an estimator of the expected generalization error $\mathrm{E}_{\mathbb{X}}(C[f_N])$
- C_{CV} is an estimator too, and we have

$$\mathrm{MSE}(C_{\text{CV}}) = \mathrm{E}_{\mathbb{X}}[(C_{\text{CV}} - \mathrm{E}_{\mathbb{X}}(C[f_N]))^2] = \mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) + \mathrm{bias}(C_{\text{CV}})^2$$

Point Estimation Revisited: Mean Square Error

- Let $\hat{\theta}_n$ be an estimator of quantity θ related to random variable $\mathbf{x}$ mapped from n i.i.d samples of $\mathbf{x}$
- ***Mean square error*** of $\hat{\theta}_n$:

$$\mathrm{MSE}(\hat{\theta}_n) = \mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right]$$

Point Estimation Revisited: Mean Square Error

- Let $\hat{\theta}_n$ be an estimator of quantity θ related to random variable $\mathbf{x}$ mapped from n i.i.d samples of $\mathbf{x}$
- ***Mean square error*** of $\hat{\theta}_n$:

$$\mathrm{MSE}(\hat{\theta}_n) = \mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right]$$

- Can be decomposed into the bias and variance:

$$\mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right] = \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n] + \mathrm{E}[\hat{\theta}_n] - \theta)^2\right]$$

Point Estimation Revisited: Mean Square Error

- Let $\hat{\theta}_n$ be an estimator of quantity θ related to random variable $\mathbf{x}$ mapped from n i.i.d samples of $\mathbf{x}$
- ***Mean square error*** of $\hat{\theta}_n$:

$$\mathrm{MSE}(\hat{\theta}_n) = \mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right]$$

- Can be decomposed into the bias and variance:

$$\begin{aligned}\mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right] &= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n] + \mathrm{E}[\hat{\theta}_n] - \theta)^2\right] \\ &= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])^2 + (\mathrm{E}[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])(\mathrm{E}[\hat{\theta}_n] - \theta)\right]\end{aligned}$$

Point Estimation Revisited: Mean Square Error

- Let $\hat{\theta}_n$ be an estimator of quantity θ related to random variable $\mathbf{x}$ mapped from n i.i.d samples of $\mathbf{x}$
- *Mean square error* of $\hat{\theta}_n$:

$$\mathrm{MSE}(\hat{\theta}_n) = \mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right]$$

- Can be decomposed into the bias and variance:

$$\begin{aligned}\mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right] &= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n] + \mathrm{E}[\hat{\theta}_n] - \theta)^2\right] \\ &= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])^2 + (\mathrm{E}[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])(\mathrm{E}[\hat{\theta}_n] - \theta)\right] \\ &= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])^2\right] + \mathrm{E}\left[(\mathrm{E}[\hat{\theta}_n] - \theta)^2\right] + 2\mathrm{E}\left(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n]\right)(\mathrm{E}[\hat{\theta}_n] - \theta)\end{aligned}$$

Point Estimation Revisited: Mean Square Error

- Let $\hat{\theta}_n$ be an estimator of quantity θ related to random variable $\mathbf{x}$ mapped from n i.i.d samples of $\mathbf{x}$
- ***Mean square error*** of $\hat{\theta}_n$:

$$\mathrm{MSE}(\hat{\theta}_n) = \mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right]$$

- Can be decomposed into the bias and variance:

$$\begin{aligned}
\mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right] &= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n] + \mathrm{E}[\hat{\theta}_n] - \theta)^2\right] \\
&= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])^2 + (\mathrm{E}[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])(\mathrm{E}[\hat{\theta}_n] - \theta)\right] \\
&= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])^2\right] + \mathrm{E}\left[(\mathrm{E}[\hat{\theta}_n] - \theta)^2\right] + 2\mathrm{E}\left(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n]\right)(\mathrm{E}[\hat{\theta}_n] - \theta) \\
&= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])^2\right] + \left(\mathrm{E}[\hat{\theta}_n] - \theta\right)^2 + 2 \cdot 0 \cdot (\mathrm{E}[\hat{\theta}_n] - \theta)
\end{aligned}$$

Point Estimation Revisited: Mean Square Error

- Let $\hat{\theta}_n$ be an estimator of quantity θ related to random variable $\mathbf{x}$ mapped from n i.i.d samples of $\mathbf{x}$
- *Mean square error* of $\hat{\theta}_n$:

$$\mathrm{MSE}(\hat{\theta}_n) = \mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right]$$

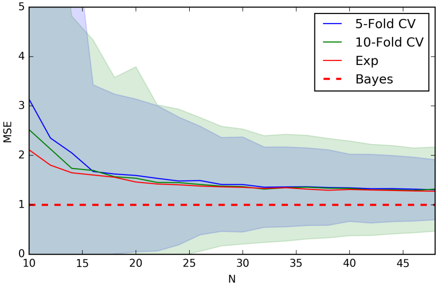
- Can be decomposed into the bias and variance:

$$\begin{aligned}
\mathrm{E}_{\mathbb{X}}\left[(\hat{\theta}_n - \theta)^2\right] &= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n] + \mathrm{E}[\hat{\theta}_n] - \theta)^2\right] \\
&= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])^2 + (\mathrm{E}[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])(\mathrm{E}[\hat{\theta}_n] - \theta)\right] \\
&= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])^2\right] + \mathrm{E}\left[(\mathrm{E}[\hat{\theta}_n] - \theta)^2\right] + 2\mathrm{E}\left(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n]\right)(\mathrm{E}[\hat{\theta}_n] - \theta) \\
&= \mathrm{E}\left[(\hat{\theta}_n - \mathrm{E}[\hat{\theta}_n])^2\right] + \left(\mathrm{E}[\hat{\theta}_n] - \theta\right)^2 + 2 \cdot 0 \cdot (\mathrm{E}[\hat{\theta}_n] - \theta) \\
&= \mathrm{Var}_{\mathbb{X}}(\hat{\theta}_n) + \mathrm{bias}(\hat{\theta}_n)^2
\end{aligned}$$

- MSE of an unbiased estimator is its variance

Example: 5-Fold vs. 10-Fold CV

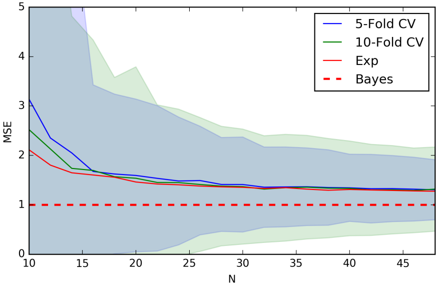
$$\mathrm{MSE}(C_{\mathrm{CV}}) = \mathrm{E}_{\mathbb{X}}[(C_{\mathrm{CV}} - \mathrm{E}_{\mathbb{X}}(C[f_N]))^2] = \mathrm{Var}_{\mathbb{X}}(C_{\mathrm{CV}}) + \mathrm{bias}(C_{\mathrm{CV}})^2$$



Example: 5-Fold vs. 10-Fold CV

$$\mathrm{MSE}(C_{\mathrm{CV}}) = \mathrm{E}_{\mathbb{X}}[(C_{\mathrm{CV}} - \mathrm{E}_{\mathbb{X}}(C[f_N]))^2] = \mathrm{Var}_{\mathbb{X}}(C_{\mathrm{CV}}) + \mathrm{bias}(C_{\mathrm{CV}})^2$$

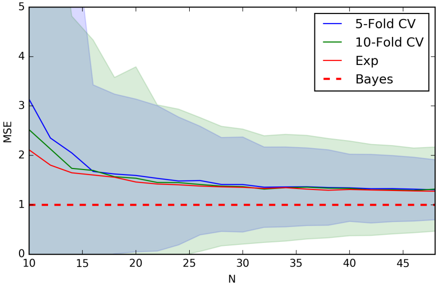
- Consider polynomial regression where $\mathrm{P}(\mathrm{y}|\mathrm{x}) = \sin(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$



Example: 5-Fold vs. 10-Fold CV

$$\mathrm{MSE}(C_{\mathsf{CV}}) = \mathrm{E}_{\mathbb{X}}[(C_{\mathsf{CV}} - \mathrm{E}_{\mathbb{X}}(C[f_N]))^2] = \mathrm{Var}_{\mathbb{X}}(C_{\mathsf{CV}}) + \mathrm{bias}(C_{\mathsf{CV}})^2$$

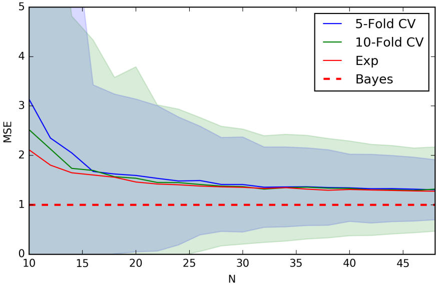
- Consider polynomial regression where $\mathrm{P}(\mathrm{y}|\mathrm{x}) = \sin(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$
- Let $C[\cdot]$ be the MSE of predictions (made by a function) to true labels



Example: 5-Fold vs. 10-Fold CV

$$\mathrm{MSE}(C_{\mathrm{CV}}) = \mathrm{E}_{\mathbb{X}}[(C_{\mathrm{CV}} - \mathrm{E}_{\mathbb{X}}(C[f_N]))^2] = \mathrm{Var}_{\mathbb{X}}(C_{\mathrm{CV}}) + \mathrm{bias}(C_{\mathrm{CV}})^2$$

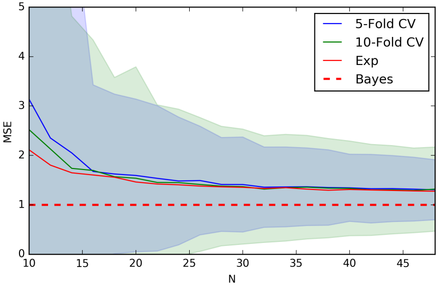
- Consider polynomial regression where $\mathrm{P}(\mathrm{y}|\mathrm{x}) = \sin(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$
- Let $C[\cdot]$ be the MSE of predictions (made by a function) to true labels
- $\mathrm{E}_{\mathbb{X}}(C[f_N])$: read line



Example: 5-Fold vs. 10-Fold CV

$$\text{MSE}(C_{\text{CV}}) = \text{E}_{\mathbb{X}}[(C_{\text{CV}} - \text{E}_{\mathbb{X}}(C[f_N]))^2] = \text{Var}_{\mathbb{X}}(C_{\text{CV}}) + \text{bias}(C_{\text{CV}})^2$$

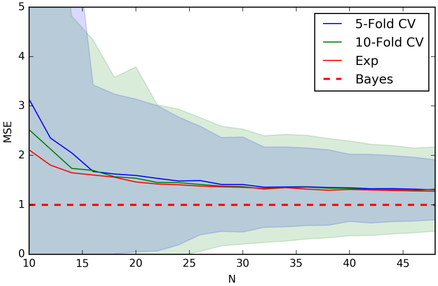
- Consider polynomial regression where $\text{P}(\text{y}|\text{x}) = \sin(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$
- Let $C[\cdot]$ be the MSE of predictions (made by a function) to true labels
- $\text{E}_{\mathbb{X}}(C[f_N])$: read line
- $\text{bias}(C_{\text{CV}})$: gaps between the red and other solid lines ($\text{E}_{\mathbb{X}}[C_{\text{CV}}]$)



Example: 5-Fold vs. 10-Fold CV

$$\mathrm{MSE}(C_{\mathrm{CV}}) = \mathrm{E}_{\mathbb{X}}[(C_{\mathrm{CV}} - \mathrm{E}_{\mathbb{X}}(C[f_N]))^2] = \mathrm{Var}_{\mathbb{X}}(C_{\mathrm{CV}}) + \mathrm{bias}(C_{\mathrm{CV}})^2$$

- Consider polynomial regression where $\mathrm{P}(\mathrm{y}|\mathrm{x}) = \sin(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$
- Let $C[\cdot]$ be the MSE of predictions (made by a function) to true labels
- $\mathrm{E}_{\mathbb{X}}(C[f_N])$: read line
- $\mathrm{bias}(C_{\mathrm{CV}})$: gaps between the red and other solid lines ($\mathrm{E}_{\mathbb{X}}[C_{\mathrm{CV}}]$)
- $\mathrm{Var}_{\mathbb{X}}(C_{\mathrm{CV}})$: shaded areas



Decomposing Bias and Variance

- C_{CV} is an estimator of the expected generalization error $\mathrm{E}_{\mathbb{X}}(C[f_N])$:

$$\mathrm{MSE}(C_{\text{CV}}) = \mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) + \mathrm{bias}(C_{\text{CV}})^2, \text{ where}$$

Decomposing Bias and Variance

- C_{CV} is an estimator of the expected generalization error $\mathrm{E}_{\mathbb{X}}(C[f_N])$:

$$\mathrm{MSE}(C_{\text{CV}}) = \mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) + \mathrm{bias}(C_{\text{CV}})^2, \text{ where}$$

$$\mathrm{bias}\left(C_{\text{CV}}\right) = \mathrm{E}_{\mathbb{X}}\left(C_{\text{CV}}\right) - \mathrm{E}_{\mathbb{X}}(C[f_N]) = \mathrm{E}\left(\sum_i \frac{1}{K} C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N])$$

Decomposing Bias and Variance

- C_{CV} is an estimator of the expected generalization error $\mathrm{E}_{\mathbb{X}}(C[f_N])$:

$$\mathrm{MSE}(C_{\text{CV}}) = \mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) + \mathrm{bias}(C_{\text{CV}})^2, \text{ where}$$

$$\begin{aligned}\mathrm{bias}(C_{\text{CV}}) &= \mathrm{E}_{\mathbb{X}}(C_{\text{CV}}) - \mathrm{E}_{\mathbb{X}}(C[f_N]) = \mathrm{E}\left(\sum_i \frac{1}{K} C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\ &= \frac{1}{K}\sum_i \mathrm{E}\left(C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N])\end{aligned}$$

Decomposing Bias and Variance

- C_{CV} is an estimator of the expected generalization error $\mathrm{E}_{\mathbb{X}}(C[f_N])$:

$$\mathrm{MSE}(C_{\text{CV}}) = \mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) + \mathrm{bias}(C_{\text{CV}})^2, \text{ where}$$

$$\begin{aligned}
&\mathrm{bias}(C_{\text{CV}}) = \mathrm{E}_{\mathbb{X}}(C_{\text{CV}}) - \mathrm{E}_{\mathbb{X}}(C[f_N]) = \mathrm{E}\left(\textstyle\sum_i \frac{1}{K} C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\
&\quad = \tfrac{1}{K}\textstyle\sum_i \mathrm{E}\left(C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\
&\quad = \mathrm{E}\left(C[f_{-N^{(s)}}]\right) - \mathrm{E}(C[f_N]), \forall s
\end{aligned}$$

Decomposing Bias and Variance

- C_{CV} is an estimator of the expected generalization error $\mathrm{E}_{\mathbb{X}}(C[f_N])$:

$$\mathrm{MSE}(C_{\text{CV}}) = \mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) + \mathrm{bias}(C_{\text{CV}})^2, \text{ where}$$

$$\begin{aligned}
\mathrm{bias}\,(C_{\text{CV}}) = \mathrm{E}_{\mathbb{X}}\,(C_{\text{CV}}) - \mathrm{E}_{\mathbb{X}}(C[f_N]) = \mathrm{E}\left(\textstyle\sum_i \frac{1}{K} C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\
= \tfrac{1}{K}\textstyle\sum_i \mathrm{E}\left(C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\
= \mathrm{E}\left(C[f_{-N^{(s)}}]\right) - \mathrm{E}(C[f_N]), \forall s \\
= \mathrm{bias}\left(C[f_{-N^{(s)}}]\right), \forall s
\end{aligned}$$

Decomposing Bias and Variance

- C_{CV} is an estimator of the expected generalization error $\mathrm{E}_{\mathbb{X}}(C[f_N])$:

$$\mathrm{MSE}(C_{\text{CV}}) = \mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) + \mathrm{bias}(C_{\text{CV}})^2, \text{ where}$$

$$\begin{aligned}\mathrm{bias}(C_{\text{CV}}) &= \mathrm{E}_{\mathbb{X}}(C_{\text{CV}}) - \mathrm{E}_{\mathbb{X}}(C[f_N]) = \mathrm{E}\left(\sum_i \frac{1}{K} C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\ &= \frac{1}{K}\sum_i \mathrm{E}\left(C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\ &= \mathrm{E}\left(C[f_{-N^{(s)}}]\right) - \mathrm{E}(C[f_N]), \forall s \\ &= \mathrm{bias}\left(C[f_{-N^{(s)}}]\right), \forall s\end{aligned}$$

$$\mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) = \mathrm{Var}\left(\sum_i \frac{1}{K} C[f_{-N^{(i)}}]\right) = \frac{1}{K^2}\mathrm{Var}\left(\sum_i C[f_{-N^{(i)}}]\right)$$

Decomposing Bias and Variance

- C_{CV} is an estimator of the expected generalization error $\mathrm{E}_{\mathbb{X}}(C[f_N])$:

$$\mathrm{MSE}(C_{\text{CV}}) = \mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) + \mathrm{bias}(C_{\text{CV}})^2, \text{ where}$$

$$\begin{aligned}\mathrm{bias}(C_{\text{CV}}) &= \mathrm{E}_{\mathbb{X}}(C_{\text{CV}}) - \mathrm{E}_{\mathbb{X}}(C[f_N]) = \mathrm{E}\left(\sum_i \tfrac{1}{K} C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\ &= \tfrac{1}{K}\sum_i \mathrm{E}\left(C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\ &= \mathrm{E}\left(C[f_{-N^{(s)}}]\right) - \mathrm{E}(C[f_N]), \forall s \\ &= \mathrm{bias}\left(C[f_{-N^{(s)}}]\right), \forall s\end{aligned}$$

$$\begin{aligned}\mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) &= \mathrm{Var}\left(\sum_i \tfrac{1}{K} C[f_{-N^{(i)}}]\right) = \tfrac{1}{K^2}\mathrm{Var}\left(\sum_i C[f_{-N^{(i)}}]\right) \\ &= \tfrac{1}{K^2}\left(\sum_i \mathrm{Var}\left(C[f_{-N^{(i)}}]\right) + 2\sum_{i,j,j>i}\mathrm{Cov}_{\mathbb{X}}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right)\right)\end{aligned}$$

Decomposing Bias and Variance

- C_{CV} is an estimator of the expected generalization error $\mathrm{E}_{\mathbb{X}}(C[f_N])$:

$$\mathrm{MSE}(C_{\text{CV}}) = \mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) + \mathrm{bias}(C_{\text{CV}})^2, \text{ where}$$

$$\begin{aligned}
\mathrm{bias}(C_{\text{CV}}) &= \mathrm{E}_{\mathbb{X}}(C_{\text{CV}}) - \mathrm{E}_{\mathbb{X}}(C[f_N]) = \mathrm{E}\left(\sum_i \frac{1}{K} C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\
&= \frac{1}{K}\sum_i \mathrm{E}\left(C[f_{-N^{(i)}}]\right) - \mathrm{E}(C[f_N]) \\
&= \mathrm{E}\left(C[f_{-N^{(s)}}]\right) - \mathrm{E}(C[f_N]), \forall s \\
&= \mathrm{bias}\left(C[f_{-N^{(s)}}]\right), \forall s
\end{aligned}$$

$$\begin{aligned}
\mathrm{Var}_{\mathbb{X}}(C_{\text{CV}}) &= \mathrm{Var}\left(\sum_i \frac{1}{K} C[f_{-N^{(i)}}]\right) = \frac{1}{K^2}\mathrm{Var}\left(\sum_i C[f_{-N^{(i)}}]\right) \\
&= \frac{1}{K^2}\left(\sum_i \mathrm{Var}\left(C[f_{-N^{(i)}}]\right) + 2\sum_{i,j,j>i} \mathrm{Cov}_{\mathbb{X}}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right)\right) \\
&= \frac{1}{K}\mathrm{Var}\left(C[f_{-N^{(s)}}]\right) + \frac{2}{K^2}\sum_{i,j,j>i} \mathrm{Cov}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right), \forall s
\end{aligned}$$

How Many Folds K? II

$$\text{MSE}(C_{\text{CV}}) = \text{Var}_{\mathbb{X}}(C_{\text{CV}}) + \text{bias}(C_{\text{CV}})^2, \text{ where}$$
$$\text{bias}(C_{\text{CV}}) = \text{bias}\left(C[f_{-\mathbb{N}^{(s)}}]\right), \forall s$$
$$\text{Var}(C_{\text{CV}}) = \frac{1}{K}\text{Var}\left(C[f_{-\mathbb{N}^{(s)}}]\right) + \frac{2}{K^2}\sum_{i,j,j>i}\text{Cov}\left(C[f_{-\mathbb{N}^{(i)}}], C[f_{-\mathbb{N}^{(j)}}]\right), \forall s$$

- We can reduce $\text{bias}(C_{\text{CV}})$ and $\text{Var}(C_{\text{CV}})$ by ***learning theory***
  - Choosing the right model complexity avoiding both underfitting and overfitting
  - Collecting more training examples (N)

How Many Folds K? II

$$\text{MSE}(C_{\text{CV}}) = \text{Var}_{\mathbb{X}}(C_{\text{CV}}) + \text{bias}(C_{\text{CV}})^2, \text{ where}$$
$$\text{bias}\left(C_{\text{CV}}\right) = \text{bias}\left(C[f_{-N^{(s)}}]\right), \forall s$$
$$\text{Var}\left(C_{\text{CV}}\right) = \frac{1}{K}\text{Var}\left(C[f_{-N^{(s)}}]\right) + \frac{2}{K^2}\sum_{i,j,j>i}\text{Cov}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right), \forall s$$

- We can reduce $\text{bias}\left(C_{\text{CV}}\right)$ and $\text{Var}\left(C_{\text{CV}}\right)$ by ***learning theory***
  - Choosing the right model complexity avoiding both underfitting and overfitting
  - Collecting more training examples (N)
- Furthermore, we can reduce $\text{Var}\left(C_{\text{CV}}\right)$ by ***making*** $f_{-N^{(i)}}$ ***and*** $f_{-N^{(j)}}$ ***uncorrelated***

How Many Folds K? III

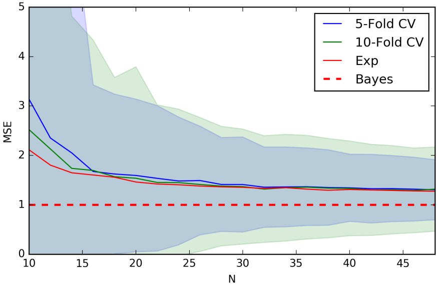
$$\text{bias}\left(C_{\text{CV}}\right) = \text{bias}\left(C[f_{-N^{(s)}}]\right), \forall s$$
$$\text{Var}_{\mathbb{X}}\left(C_{\text{CV}}\right) = \frac{1}{K}\text{Var}\left(C[f_{-N^{(s)}}]\right) + \frac{2}{K^2}\sum_{i,j,j>i}\text{Cov}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right), \forall s$$

- With a large K, the C_{CV} tends to have:

How Many Folds K? III

$$\text{bias}\left(C_{\text{CV}}\right) = \text{bias}\left(C[f_{-N^{(s)}}]\right), \forall s$$
$$\text{Var}_{\mathbb{X}}\left(C_{\text{CV}}\right) = \frac{1}{K}\text{Var}\left(C[f_{-N^{(s)}}]\right) + \frac{2}{K^2}\sum_{i,j,j>i}\text{Cov}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right), \forall s$$

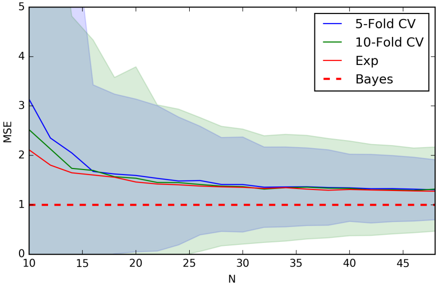
- With a large K, the C_{CV} tends to have:
  - Low $\text{bias}\left(C[f_{-N^{(s)}}]\right)$ and $\text{Var}\left(C[f_{-N^{(s)}}]\right)$, as $f_{-N^{(s)}}$ is trained on more examples



How Many Folds K? III

$$\text{bias}\left(C_{\text{CV}}\right) = \text{bias}\left(C[f_{-N^{(s)}}]\right), \forall s$$
$$\text{Var}_{\mathbb{X}}\left(C_{\text{CV}}\right) = \frac{1}{K}\text{Var}\left(C[f_{-N^{(s)}}]\right) + \frac{2}{K^2}\sum_{i,j,j>i}\text{Cov}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right), \forall s$$

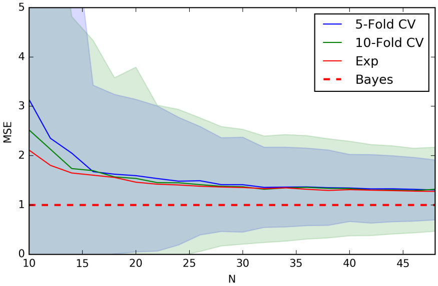
- With a large K, the C_{CV} tends to have:
    - Low $\text{bias}\left(C[f_{-N^{(s)}}]\right)$ and $\text{Var}\left(C[f_{-N^{(s)}}]\right)$, as $f_{-N^{(s)}}$ is trained on more examples
    - High $\text{Cov}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right)$, as training sets $\mathbb{X}\backslash\mathbb{X}^{(i)}$ and $\mathbb{X}\backslash\mathbb{X}^{(j)}$ are more similar thus $C[f_{-N^{(i)}}]$ and $C[f_{-N^{(j)}}]$ are more positively correlated



How Many Folds K? IV

$$\text{bias}(C_{\text{CV}}) = \text{bias}\left(C[f_{-N^{(s)}}]\right), \forall s$$
$$\text{Var}_{\mathbb{X}}(C_{\text{CV}}) = \frac{1}{K}\text{Var}\left(C[f_{-N^{(s)}}]\right) + \frac{2}{K^2}\sum_{i,j,j>i}\text{Cov}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right), \forall s$$

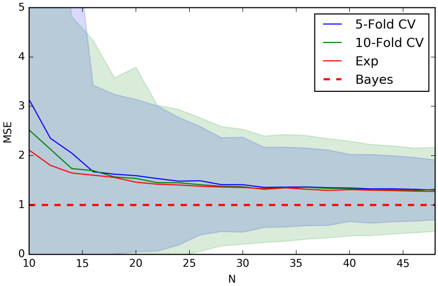
- Conversely, with a small K, the cross-validation error tends to have a high $\text{bias}\left(C[f_{-N^{(s)}}]\right)$ and $\text{Var}\left(C[f_{-N^{(s)}}]\right)$ but low $\text{Cov}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right)$



How Many Folds K? IV

$$\text{bias}(C_{\text{CV}}) = \text{bias}\left(C[f_{-N^{(s)}}]\right), \forall s$$
$$\text{Var}_{\mathbb{X}}(C_{\text{CV}}) = \frac{1}{K}\text{Var}\left(C[f_{-N^{(s)}}]\right) + \frac{2}{K^2}\sum_{i,j,j>i}\text{Cov}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right), \forall s$$

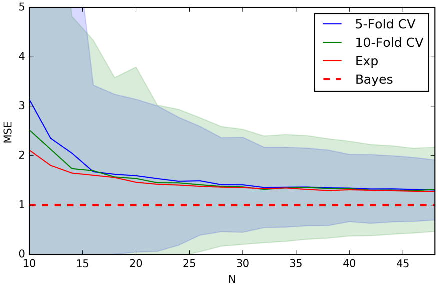
- Conversely, with a small K, the cross-validation error tends to have a high $\text{bias}\left(C[f_{-N^{(s)}}]\right)$ and $\text{Var}\left(C[f_{-N^{(s)}}]\right)$ but low $\text{Cov}\left(C[f_{-N^{(i)}}], C[f_{-N^{(j)}}]\right)$
- In practice, we usually set $K = 5$ or 10



Leave-One-Out CV

$$\text{bias}\left(C_{\text{CV}}\right) = \text{bias}\left(C[f_{-\mathbb{N}^{(s)}}]\right), \forall s$$
$$\text{Var}_{\mathbb{X}}\left(C_{\text{CV}}\right) = \frac{1}{K}\text{Var}\left(C[f_{-\mathbb{N}^{(s)}}]\right) + \frac{2}{K^2}\sum_{i,j,j>i}\text{Cov}\left(C[f_{-\mathbb{N}^{(i)}}], C[f_{-\mathbb{N}^{(j)}}]\right), \forall s$$

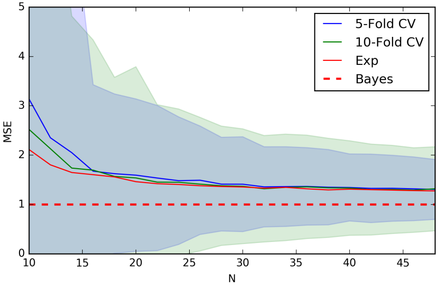
- For very small dataset:
  - $\text{MSE}(C_{\text{CV}})$ is dominated by $\text{bias}\left(C[f_{-\mathbb{N}^{(s)}}]\right)$ and $\text{Var}\left(C[f_{-\mathbb{N}^{(s)}}]\right)$
  - Not $\text{Cov}\left(C[f_{-\mathbb{N}^{(i)}}], C[f_{-\mathbb{N}^{(j)}}]\right)$



Leave-One-Out CV

$$\text{bias}\left(C_{\text{CV}}\right) = \text{bias}\left(C[f_{-\mathbb{N}^{(s)}}]\right), \forall s$$
$$\text{Var}_{\mathbb{X}}\left(C_{\text{CV}}\right) = \frac{1}{K}\text{Var}\left(C[f_{-\mathbb{N}^{(s)}}]\right) + \frac{2}{K^2}\sum_{i,j,j>i}\text{Cov}\left(C[f_{-\mathbb{N}^{(i)}}], C[f_{-\mathbb{N}^{(j)}}]\right), \forall s$$

- For very small dataset:
  - $\text{MSE}(C_{\text{CV}})$ is dominated by $\text{bias}\left(C[f_{-\mathbb{N}^{(s)}}]\right)$ and $\text{Var}\left(C[f_{-\mathbb{N}^{(s)}}]\right)$
  - Not $\text{Cov}\left(C[f_{-\mathbb{N}^{(i)}}], C[f_{-\mathbb{N}^{(j)}}]\right)$
- We can choose $K = N$, which we call the ***leave-one-out CV***



Outline

1. Cross Validation
   - How Many Folds?
2. **Ensemble Methods**
   - Voting
   - Bagging
   - Boosting
   - Why AdaBoost Works?

Ensemble Methods

- ***No free lunch theorem***: there is no single ML algorithm that always outperforms the others in all domains/tasks

Ensemble Methods

- ***No free lunch theorem***: there is no single ML algorithm that always outperforms the others in all domains/tasks
- Can we combine multiple base-learners to improve
  - Applicability across different domains, and/or
  - Generalization performance in a specific task?

Ensemble Methods

- ***No free lunch theorem***: there is no single ML algorithm that always outperforms the others in all domains/tasks
- Can we combine multiple base-learners to improve
  - Applicability across different domains, and/or
  - Generalization performance in a specific task?
- These are the goals of ***ensemble learning***

Ensemble Methods

- ***No free lunch theorem***: there is no single ML algorithm that always outperforms the others in all domains/tasks
- Can we combine multiple base-learners to improve
  - Applicability across different domains, and/or
  - Generalization performance in a specific task?
- These are the goals of ***ensemble learning***
- How to "combine" multiple base-learners?

Outline

1. Cross Validation
   - How Many Folds?
2. Ensemble Methods
   - Voting
   - Bagging
   - Boosting
   - Why AdaBoost Works?

Voting

- ***Voting***: linear combining the predictions of base-learners for each $\boldsymbol{x}$:

$$\tilde{y}_k = \sum_j w_j \hat{y}_k^{(j)} \text{ where } w_j \geq 0, \sum_j w_j = 1.$$

- If all learners are given equal weight $w_j = 1/L$, we have the ***plurality vote*** (multi-class version of majority vote)

Voting Rule	Formular
Sum	$\tilde{y}_k = \frac{1}{L}\sum_{j=1}^{L} \hat{y}_k^{(j)}$
Weighted sum	$\tilde{y}_k = \sum_j w_j \hat{y}_k^{(j)}, w_j \geq 0, \sum_j w_j = 1$
Median	$\tilde{y}_k = \text{median}_j \hat{y}_k^{(j)}$
Minimum	$\tilde{y}_k = \min_j \hat{y}_k^{(j)}$
Maximum	$\tilde{y}_k = \max_j \hat{y}_k^{(j)}$
Product	$\tilde{y}_k = \prod_j \hat{y}_k^{(j)}$

Why Voting Works? I

Why Voting Works? I

- Assume that each $\hat{y}^{(j)}$ has the expected value $\mathrm{E}_{\mathbb{X}}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)$ and variance $\mathrm{Var}_{\mathbb{X}}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)$
- When $w_j = 1/L$, we have:

$$\mathrm{E}_{\mathbb{X}}\left(\tilde{y}\,|\,\boldsymbol{x}\right) = \mathrm{E}\left(\sum_j \frac{1}{L}\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) = \frac{1}{L}\sum_j \mathrm{E}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) = \mathrm{E}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)$$

$$\begin{aligned}\mathrm{Var}_{\mathbb{X}}\left(\tilde{y}\,|\,\boldsymbol{x}\right) &= \mathrm{Var}\left(\sum_j \frac{1}{L}\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) = \frac{1}{L^2}\mathrm{Var}\left(\sum_j \hat{y}^{(j)}\,|\,\boldsymbol{x}\right) \\ &= \frac{1}{L}\mathrm{Var}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) + \frac{2}{L^2}\sum_{i,j,i<j}\mathrm{Cov}\left(\hat{y}^{(i)},\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)\end{aligned}$$

Why Voting Works? I

- Assume that each $\hat{y}^{(j)}$ has the expected value $\mathrm{E}_{\mathbb{X}}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)$ and variance $\mathrm{Var}_{\mathbb{X}}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)$
- When $w_j = 1/L$, we have:

$$\mathrm{E}_{\mathbb{X}}(\tilde{y}\,|\,\boldsymbol{x}) = \mathrm{E}\left(\sum_j \frac{1}{L}\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) = \frac{1}{L}\sum_j \mathrm{E}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) = \mathrm{E}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)$$

$$\begin{aligned}\mathrm{Var}_{\mathbb{X}}(\tilde{y}\,|\,\boldsymbol{x}) &= \mathrm{Var}\left(\sum_j \frac{1}{L}\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) = \frac{1}{L^2}\mathrm{Var}\left(\sum_j \hat{y}^{(j)}\,|\,\boldsymbol{x}\right) \\ &= \frac{1}{L}\mathrm{Var}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) + \frac{2}{L^2}\sum_{i,j,i<j}\mathrm{Cov}\left(\hat{y}^{(i)},\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)\end{aligned}$$

- The expected value doesn't change, so the bias doesn't change

Why Voting Works? II

$$\mathrm{Var}_{\mathbb{X}}(\tilde{y}\,|\,\boldsymbol{x}) = \frac{1}{L}\mathrm{Var}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) + \frac{2}{L^2}\sum_{i,j,i<j}\mathrm{Cov}\left(\hat{y}^{(i)},\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)$$

Why Voting Works? II

$$\mathrm{Var}_{\mathbb{X}}(\tilde{y}\,|\,\boldsymbol{x}) = \frac{1}{L}\mathrm{Var}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) + \frac{2}{L^2}\sum_{i,j,i<j}\mathrm{Cov}\left(\hat{y}^{(i)},\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)$$

- If $\hat{y}^{(i)}$ and $\hat{y}^{(j)}$ are uncorrelated, the variance can be reduced

Why Voting Works? II

$$\mathrm{Var}_{\mathbb{X}}(\tilde{y}\,|\,\boldsymbol{x}) = \frac{1}{L}\mathrm{Var}\left(\hat{y}^{(j)}\,|\,\boldsymbol{x}\right) + \frac{2}{L^2}\sum_{i,j,i<j}\mathrm{Cov}\left(\hat{y}^{(i)},\hat{y}^{(j)}\,|\,\boldsymbol{x}\right)$$

- If $\hat{y}^{(i)}$ and $\hat{y}^{(j)}$ are uncorrelated, the variance can be reduced
- Unfortunately, $\hat{y}^{(j)}$'s may ***not*** be i.i.d. in practice
- If voters are positively correlated, variance increases

Outline

Bagging

- ***Bagging*** (short for ***bootstrap aggregating***) is a voting method, but base-learners are made different deliberately
- How?

Bagging

- ***Bagging*** (short for ***bootstrap aggregating***) is a voting method, but base-learners are made different deliberately
- How? Why not train them using slightly different training sets?

Bagging

- ***Bagging*** (short for ***bootstrap aggregating***) is a voting method, but base-learners are made different deliberately
- How? Why not train them using slightly different training sets?

1. Generate L slightly different samples from a given sample is done by ***bootstrap***: given $\mathbb{X}$ of size N, we draw N points randomly from $\mathbb{X}$ ***with replacement*** to get $\mathbb{X}^{(j)}$
   - It is possible that some instances are drawn more than once and some are not at all

Bagging

- ***Bagging*** (short for ***bootstrap aggregating***) is a voting method, but base-learners are made different deliberately
- How? Why not train them using slightly different training sets?

1. Generate L slightly different samples from a given sample is done by ***bootstrap***: given $\mathbb{X}$ of size N, we draw N points randomly from $\mathbb{X}$ ***with replacement*** to get $\mathbb{X}^{(j)}$
    - It is possible that some instances are drawn more than once and some are not at all
2. Train a base-learner for each $\mathbb{X}^{(j)}$

Outline

Boosting

- In bagging, generating "uncorrelated" base-learners is left to chance and unstability of the learning method

Boosting

- In bagging, generating "uncorrelated" base-learners is left to chance and unstability of the learning method
- In ***boosting***, we generate ***complementary*** base-learners
- How?

Boosting

- In bagging, generating "uncorrelated" base-learners is left to chance and unstability of the learning method
- In ***boosting***, we generate ***complementary*** base-learners
- How? Why not train the next learner on the mistakes of the previous learners

Boosting

- In bagging, generating "uncorrelated" base-learners is left to chance and unstability of the learning method
- In ***boosting***, we generate ***complementary*** base-learners
- How? Why not train the next learner on the mistakes of the previous learners
- For simplicity, let's consider the binary classification here: $d^{(j)}(\boldsymbol{x}) \in \{1, -1\}$
- The original boosting algorithm combines three ***weak learners*** to generate a ***strong learner***
  - A week learner has error probability less than $1/2$ (better than random guessing)
  - A strong learner has arbitrarily small error probability

Original Boosting Algorithm

- Training

1. Given a large training set, randomly divide it into three

Original Boosting Algorithm

- Training

1. Given a large training set, randomly divide it into three
2. Use $\mathbb{X}^{(1)}$ to train the first learner $d^{(1)}$ and feed $\mathbb{X}^{(2)}$ to $d^{(1)}$

Original Boosting Algorithm

- Training

1. Given a large training set, randomly divide it into three
2. Use $\mathbb{X}^{(1)}$ to train the first learner $d^{(1)}$ and feed $\mathbb{X}^{(2)}$ to $d^{(1)}$
3. Use all points misclassified by $d^{(1)}$ and $\mathbb{X}^{(2)}$ to train $d^{(2)}$. Then feed $\mathbb{X}^{(3)}$ to $d^{(1)}$ and $d^{(2)}$

Original Boosting Algorithm

- Training

1. Given a large training set, randomly divide it into three
2. Use $\mathbb{X}^{(1)}$ to train the first learner $d^{(1)}$ and feed $\mathbb{X}^{(2)}$ to $d^{(1)}$
3. Use all points misclassified by $d^{(1)}$ and $\mathbb{X}^{(2)}$ to train $d^{(2)}$. Then feed $\mathbb{X}^{(3)}$ to $d^{(1)}$ and $d^{(2)}$
4. Use the points on which $d^{(1)}$ and $d^{(2)}$ disagree to train $d^{(3)}$

Original Boosting Algorithm

- Training

1. Given a large training set, randomly divide it into three
2. Use $\mathbb{X}^{(1)}$ to train the first learner $d^{(1)}$ and feed $\mathbb{X}^{(2)}$ to $d^{(1)}$
3. Use all points misclassified by $d^{(1)}$ and $\mathbb{X}^{(2)}$ to train $d^{(2)}$. Then feed $\mathbb{X}^{(3)}$ to $d^{(1)}$ and $d^{(2)}$
4. Use the points on which $d^{(1)}$ and $d^{(2)}$ disagree to train $d^{(3)}$

- Testing

1. Feed a point it to $d^{(1)}$ and $d^{(2)}$ first. If their outputs agree, use them as the final prediction

Original Boosting Algorithm

- Training

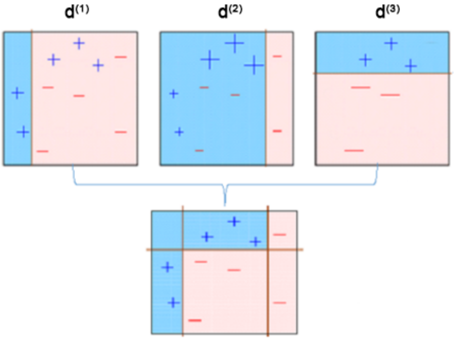
1. Given a large training set, randomly divide it into three
2. Use $\mathbb{X}^{(1)}$ to train the first learner $d^{(1)}$ and feed $\mathbb{X}^{(2)}$ to $d^{(1)}$
3. Use all points misclassified by $d^{(1)}$ and $\mathbb{X}^{(2)}$ to train $d^{(2)}$. Then feed $\mathbb{X}^{(3)}$ to $d^{(1)}$ and $d^{(2)}$
4. Use the points on which $d^{(1)}$ and $d^{(2)}$ disagree to train $d^{(3)}$

- Testing

1. Feed a point it to $d^{(1)}$ and $d^{(2)}$ first. If their outputs agree, use them as the final prediction
2. Otherwise the output of $d^{(3)}$ is taken

Example

- Assuming $\mathbb{X}^{(1)}$, $\mathbb{X}^{(2)}$, and $\mathbb{X}^{(3)}$ are the same:



- Disadvantage: requires a large training set to afford the three-way split

AdaBoost

- ***AdaBoost***: uses the same training set over and over again
- How to make some points "larger?"

AdaBoost

- ***AdaBoost***: uses the same training set over and over again
- How to make some points "larger?"
- Modify the probabilities of drawing the instances as a function of error

AdaBoost

- ***AdaBoost***: uses the same training set over and over again
- How to make some points "larger?"
- Modify the probabilities of drawing the instances as a function of error
- Notation:
- $\mathrm{Pr}^{(i,j)}$: probability that an example $(\boldsymbol{x}^{(i)}, y^{(i)})$ is drawn to train the jth base-learner $d^{(j)}$
- $\varepsilon^{(j)} = \sum_i \mathrm{Pr}^{(i,j)} 1(y^{(i)} \neq d^{(j)}(\boldsymbol{x}^{(i)}))$: error rate of $d^{(j)}$ on its training set

Algorithm

- Training

1. Initialize $\Pr^{(i,1)} = \frac{1}{N}$ for all i
2. Start from $j = 1$:
    1. Randomly draw N examples from $\mathbb{X}$ with probabilities $\Pr^{(i,j)}$ and use them to train $d^{(j)}$

Algorithm

- Training

1. Initialize $\Pr^{(i,1)} = \frac{1}{N}$ for all i
2. Start from $j = 1$:
    1. Randomly draw N examples from $\mathbb{X}$ with probabilities $\Pr^{(i,j)}$ and use them to train $d^{(j)}$
    2. Stop adding new base-learners if $\varepsilon^{(j)} \geq \frac{1}{2}$

Algorithm

- Training

1. Initialize $\Pr^{(i,1)} = \frac{1}{N}$ for all i
2. Start from $j = 1$:
    1. Randomly draw N examples from $\mathbb{X}$ with probabilities $\Pr^{(i,j)}$ and use them to train $d^{(j)}$
    2. Stop adding new base-learners if $\varepsilon^{(j)} \geq \frac{1}{2}$
    3. Define $\alpha_j = \frac{1}{2}\log\left(\frac{1-\varepsilon^{(j)}}{\varepsilon^{(j)}}\right) > 0$ and set $\Pr^{(i,j+1)} = \Pr^{(i,j)} \cdot \exp(-\alpha_j y^{(i)} d^{(j)}(\boldsymbol{x}^{(i)}))$ for all i

Algorithm

- Training

1. Initialize $\Pr^{(i,1)} = \frac{1}{N}$ for all i
2. Start from $j = 1$:
   1. Randomly draw N examples from $\mathbb{X}$ with probabilities $\Pr^{(i,j)}$ and use them to train $d^{(j)}$
   2. Stop adding new base-learners if $\varepsilon^{(j)} \geq \frac{1}{2}$
   3. Define $\alpha_j = \frac{1}{2} \log \left(\frac{1-\varepsilon^{(j)}}{\varepsilon^{(j)}} \right) > 0$ and set $\Pr^{(i,j+1)} = \Pr^{(i,j)} \cdot \exp(-\alpha_j y^{(i)} d^{(j)}(\boldsymbol{x}^{(i)}))$ for all i
   4. Normalize $\Pr^{(i,j+1)}$, $\forall i$, by multiplying $\left(\sum_i \Pr^{(i,j+1)} \right)^{-1}$

Algorithm

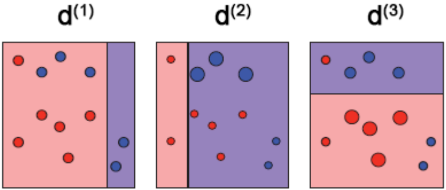
- Training

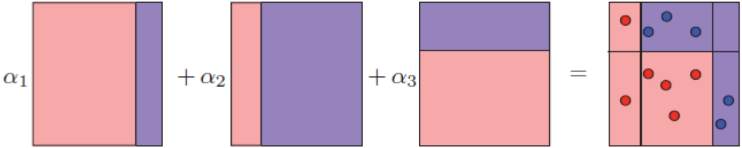
1. Initialize $\Pr^{(i,1)} = \frac{1}{N}$ for all i
2. Start from $j = 1$:
   1. Randomly draw N examples from $\mathbb{X}$ with probabilities $\Pr^{(i,j)}$ and use them to train $d^{(j)}$
   2. Stop adding new base-learners if $\varepsilon^{(j)} \geq \frac{1}{2}$
   3. Define $\alpha_j = \frac{1}{2}\log\left(\frac{1-\varepsilon^{(j)}}{\varepsilon^{(j)}}\right) > 0$ and set $\Pr^{(i,j+1)} = \Pr^{(i,j)} \cdot \exp(-\alpha_j y^{(i)} d^{(j)}(\boldsymbol{x}^{(i)}))$ for all i
   4. Normalize $\Pr^{(i,j+1)}$, $\forall i$, by multiplying $\left(\sum_i \Pr^{(i,j+1)}\right)^{-1}$

- Testing

1. Given $\boldsymbol{x}$, calculate $\hat{y}^{(j)}$ for all j
2. Make final prediction $\tilde{y}$ by voting: $\tilde{y} = \sum_j \alpha_j d^{(j)}(\boldsymbol{x})$

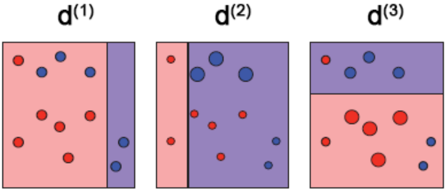
Example

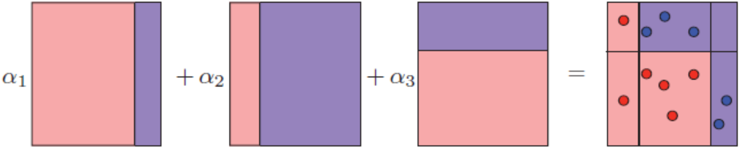




- $d^{(j+1)}$ complements $d^{(j)}$ and $d^{(j-1)}$ by focusing on predictions they disagree

Example





- $d^{(j+1)}$ complements $d^{(j)}$ and $d^{(j-1)}$ by focusing on predictions they disagree
- Voting weights ($\alpha_j = \frac{1}{2}\log\left(\frac{1-\varepsilon^{(j)}}{\varepsilon^{(j)}}\right)$) in predictions are proportional to the base-learner's accuracy

Outline

Why AdaBoost Works

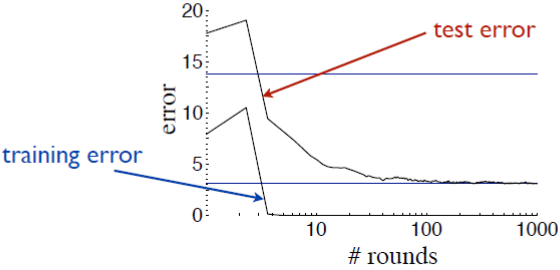
- Why AdaBoost improves performance?

Why AdaBoost Works

- Why AdaBoost improves performance?
- By increasing model complexity?

Why AdaBoost Works

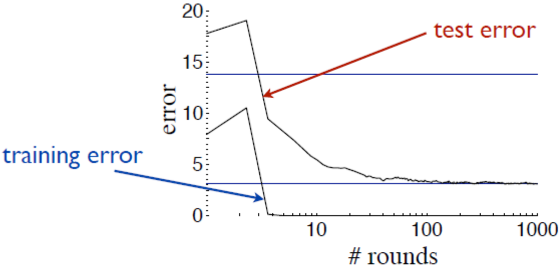
- Why AdaBoost improves performance?
- By increasing model complexity? Not exactly
  - Empirical study: AdaBoost ***reduces overfitting*** as L grows, even when there is no training error



C4.5 decision trees (Schapire et al., 1998).

Why AdaBoost Works

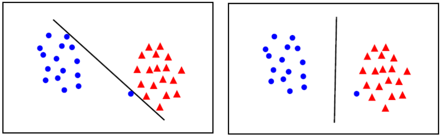
- Why AdaBoost improves performance?
- By increasing model complexity? Not exactly
  - Empirical study: AdaBoost ***reduces overfitting*** as L grows, even when there is no training error
- AdaBoost ***increases margin*** [1, 2]



C4.5 decision trees (Schapire et al., 1998).

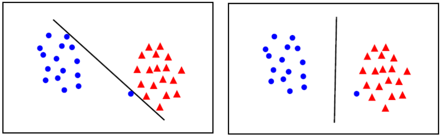
Margin as Confidence of Predictions

- Recall in SVC, a larger margin improves generalizability



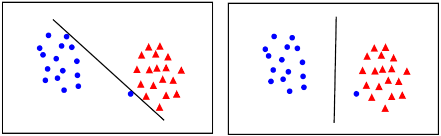
Margin as Confidence of Predictions

- Recall in SVC, a larger margin improves generalizability
- Due to ***higher confidence predictions*** over training examples



Margin as Confidence of Predictions

- Recall in SVC, a larger margin improves generalizability
- Due to ***higher confidence predictions*** over training examples



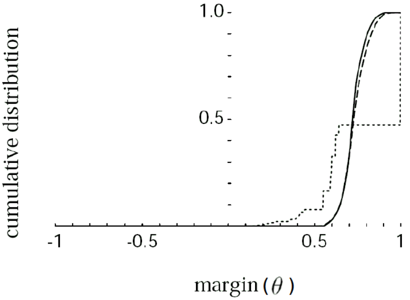
- We can define the margin for AdaBoost similarly
- In binary classification, define ***margin*** of a prediction of an example $(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathbb{X}$ as:

$$margin(\boldsymbol{x}^{(i)}, y^{(i)}) = y^{(i)} f(\boldsymbol{x}^{(i)}) = \sum_{j: y^{(i)} = d^{(j)}(\boldsymbol{x}^{(i)})} \alpha_j - \sum_{j: y^{(i)} \neq d^{(j)}(\boldsymbol{x}^{(i)})} \alpha_j$$

Margin Distribution

- Margin distribution over θ:

$$\Pr_{\mathbb{X}}(y^{(i)}f(\boldsymbol{x}^{(i)}) \leq \theta) \approx \frac{|(\boldsymbol{x}^{(i)}, y^{(i)}) : y^{(i)}f(\boldsymbol{x}^{(i)}) \leq \theta|}{|\mathbb{X}|}$$

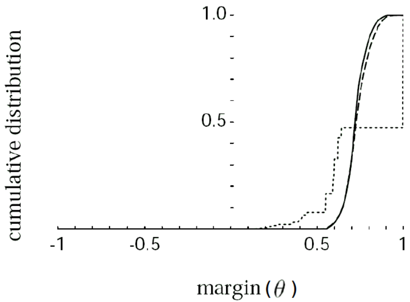


LEGEND: (small dash, large dash, solid) lines equal (5, 100, 1000) rounds of boosting

Margin Distribution

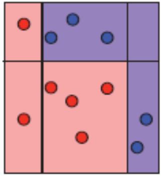
- Margin distribution over θ:

$$\Pr_{\mathbb{X}}(y^{(i)}f(\boldsymbol{x}^{(i)}) \leq \theta) \approx \frac{|(\boldsymbol{x}^{(i)}, y^{(i)}) : y^{(i)}f(\boldsymbol{x}^{(i)}) \leq \theta|}{|\mathbb{X}|}$$



LEGEND: (small dash, large dash, solid) lines equal (5, 100, 1000) rounds of boosting

- A complementary learner:
- Clarifies low confidence areas
- Increases margin of points in these areas



Reference I

[1] Yoav Freund, Robert Schapire, and N Abe.
A short introduction to boosting.
Journal-Japanese Society For Artificial Intelligence, 14(771-780):1612, 1999.

[2] Liwei Wang, Masashi Sugiyama, Cheng Yang, Zhi-Hua Zhou, and Jufu Feng.
On the margin explanation of boosting algorithms.
In *COLT*, pages 479–490. Citeseer, 2008.