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?

\[ K \]-fold cross validation:

1. Split the data set \( X \) evenly into \( K \) subsets \( X_i \) (called folds)
2. For \( i = 1, \ldots, K \), train \( f_{N(i)} \) using all data but the \( i \)-th fold \( (X \setminus X_i) \)
3. Report the cross-validation error \( C_{CV} \) by averaging all testing errors \( C[f_{N(i)}] \)'s on \( X_i \)
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 $X$ evenly into $K$ subsets $X^{(i)}$ (called *folds*).
2. For $i = 1, \cdots, K$, train $f_{-N(i)}$ using all data but the $i$-th fold ($X \setminus X^{(i)}$).
3. Report the **cross-validation error** $C_{CV}$ by averaging all testing errors $C[f_{-N(i)}]$'s on $X^{(i)}$. 

![Diagram of K-fold cross validation](image)
Nested Cross Validation

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

  E.g., $5 \times 2$ nested CV

- Inner (2 folds): select hyperparameters giving lowest $CV$

- Train final model using both training and validation sets with the selected hyperparameters

- Outer (5 folds): report $CV$ as test error
Nested Cross Validation

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

E.g, $5 \times 2$ 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, $5 \times 2$ 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, $5 \times 2$ 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 \( K \)?

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 \( E_X(C_f[N]) \). \( C_{CV} \) is an estimator too, and we have

\[
\text{MSE}(C_{CV}) = E_X(C_{CV})^2 = \text{Var}_X(C_{CV}) + \text{bias}(C_{CV})^2
\]
How Many Folds $K$? I

- The cross-validation error $C_{CV}$ is an average of $C[f_{-N(i)}]$’s
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 $E_X(C[f_N])$. 

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 $E_X(C[f_N])$
- $C_{CV}$ is an estimator too, and we have

$$
\text{MSE}(C_{CV}) = E_X[(C_{CV} - E_X(C[f_N]))^2] = Var_X(C_{CV}) + \text{bias}(C_{CV})^2
$$
Point Estimation Revisited: Mean Square Error

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

\[
\text{MSE}(\hat{\theta}_n) = E_X [(\hat{\theta}_n - \theta)^2]
\]
Point Estimation Revisited: Mean Square Error

- Let $\hat{\theta}_n$ be an estimator of quantity $\theta$ related to random variable $x$ mapped from $n$ i.i.d samples of $x$

- **Mean square error** of $\hat{\theta}_n$:

$$\text{MSE}(\hat{\theta}_n) = \mathbb{E}_X [(\hat{\theta}_n - \theta)^2]$$

- Can be decomposed into the bias and variance:

$$\mathbb{E}_X [(\hat{\theta}_n - \theta)^2] = \mathbb{E} [(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n] + \mathbb{E}[\hat{\theta}_n] - \theta)^2]$$
Point Estimation Revisited: Mean Square Error

- Let $\hat{\theta}_n$ be an estimator of quantity $\theta$ related to random variable $x$ mapped from $n$ i.i.d samples of $x$
- **Mean square error** of $\hat{\theta}_n$:
  \[ \text{MSE}(\hat{\theta}_n) = \mathbb{E}_x[(\hat{\theta}_n - \theta)^2] \]

- Can be decomposed into the bias and variance:
  \[
  \mathbb{E}_x[(\hat{\theta}_n - \theta)^2] = \mathbb{E}[(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n] + \mathbb{E}[\hat{\theta}_n] - \theta)^2] \\
  = \mathbb{E}[(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])^2 + (\mathbb{E}[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])(\mathbb{E}[\hat{\theta}_n] - \theta)]
  \]
Point Estimation Revisited: Mean Square Error

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

$$\text{MSE}(\hat{\theta}_n) = E_x[(\hat{\theta}_n - \theta)^2]$$

- Can be decomposed into the bias and variance:

$$E_x[(\hat{\theta}_n - \theta)^2] = E[(\hat{\theta}_n - E[\hat{\theta}_n] + E[\hat{\theta}_n] - \theta)^2]$$

$$= E[(\hat{\theta}_n - E[\hat{\theta}_n])^2 + (E[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - E[\hat{\theta}_n])(E[\hat{\theta}_n] - \theta)]$$

$$= E[(\hat{\theta}_n - E[\hat{\theta}_n])^2] + E[(E[\hat{\theta}_n] - \theta)^2] + 2E(\hat{\theta}_n - E[\hat{\theta}_n])(E[\hat{\theta}_n] - \theta)$$
Point Estimation Revisited: Mean Square Error

- Let $\hat{\theta}_n$ be an estimator of quantity $\theta$ related to random variable $x$ mapped from $n$ i.i.d samples of $x$
- **Mean square error** of $\hat{\theta}_n$:
  \[
  \text{MSE}(\hat{\theta}_n) = E_X[(\hat{\theta}_n - \theta)^2]
  \]

- Can be decomposed into the bias and variance:
  \[
  E_X[(\hat{\theta}_n - \theta)^2] = E\left[(\hat{\theta}_n - E[\hat{\theta}_n] + E[\hat{\theta}_n] - \theta)^2\right]
  = E\left[(\hat{\theta}_n - E[\hat{\theta}_n])^2 + (E[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - E[\hat{\theta}_n])(E[\hat{\theta}_n] - \theta)\right]
  = E\left[(\hat{\theta}_n - E[\hat{\theta}_n])^2\right] + E\left[(E[\hat{\theta}_n] - \theta)^2\right] + 2E\left(\hat{\theta}_n - E[\hat{\theta}_n]\right)(E[\hat{\theta}_n] - \theta)
  = E\left[(\hat{\theta}_n - E[\hat{\theta}_n])^2\right] + (E[\hat{\theta}_n] - \theta)^2 + 2 \cdot 0 \cdot (E[\hat{\theta}_n] - \theta)
  
  \text{MSE of an unbiased estimator is its variance}
Point Estimation Revisited: Mean Square Error

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

$$\text{MSE}(\hat{\theta}_n) = \mathbb{E}_{x}[(\hat{\theta}_n - \theta)^2]$$

- Can be decomposed into the bias and variance:

$$\mathbb{E}_{x}[(\hat{\theta}_n - \theta)^2] = \mathbb{E}[(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n] + \mathbb{E}[\hat{\theta}_n] - \theta)^2]$$

$$= \mathbb{E}[(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])^2 + (\mathbb{E}[\hat{\theta}_n] - \theta)^2 + 2(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])(\mathbb{E}[\hat{\theta}_n] - \theta)]$$

$$= \mathbb{E}[(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])^2] + \mathbb{E}[(\mathbb{E}[\hat{\theta}_n] - \theta)^2] + 2\mathbb{E}(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n]) (\mathbb{E}[\hat{\theta}_n] - \theta)$$

$$= \mathbb{E}[(\hat{\theta}_n - \mathbb{E}[\hat{\theta}_n])^2] + (\mathbb{E}[\hat{\theta}_n] - \theta)^2 + 2 \cdot 0 \cdot (\mathbb{E}[\hat{\theta}_n] - \theta)$$

$$= \text{Var}_x(\hat{\theta}_n) + \text{bias}(\hat{\theta}_n)^2$$

- MSE of an unbiased estimator is its variance
Example: 5-Fold vs. 10-Fold CV

\[
\text{MSE}(C_{\text{CV}}) = E_X[(C_{\text{CV}} - E_X(C[f_N]))^2] = \text{Var}_X(C_{\text{CV}}) + \text{bias}(C_{\text{CV}})^2
\]
Example: 5-Fold vs. 10-Fold CV

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

- Consider polynomial regression where
  \[ P(y|x) = \sin(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2) \]
Example: 5-Fold vs. 10-Fold CV

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

- Consider polynomial regression where
  \[P(y|x) = \sin(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)\]
- Let \(C[\cdot]\) be the MSE of predictions (made by a function) to true labels
Example: 5-Fold vs. 10-Fold CV

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

- Consider polynomial regression where
  \( P(y|x) = \sin(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2) \)
- Let \( C[\cdot] \) be the MSE of predictions (made by a function) to true labels
- \( \mathbb{E}_X(C[f_N]) \): read line
Example: 5-Fold vs. 10-Fold CV

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

- Consider polynomial regression where
  \[P(y|x) = \sin(x) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)\]
- Let \(C[\cdot]\) be the MSE of predictions (made by a function) to true labels
- \(\mathbb{E}_X(C[f_N])\): read line
- \(\text{bias}(C_{CV})\): gaps between the red and other solid lines (\(\mathbb{E}_X[C_{CV}]\))
Example: 5-Fold vs. 10-Fold CV

\[
\text{MSE}(C_{CV}) = E_X[(C_{CV} - E_X(C[f_N]))^2] = \text{Var}_X(C_{CV}) + \text{bias}(C_{CV})^2
\]

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

- $C_{CV}$ is an estimator of the expected generalization error $E_X(C[f_N])$:

$$\text{MSE}(C_{CV}) = \text{Var}_X(C_{CV}) + \text{bias}(C_{CV})^2, \text{ where}$$
Decomposing Bias and Variance

- $C_{CV}$ is an estimator of the expected generalization error $E_X(C[f_N])$:

$$\text{MSE}(C_{CV}) = \text{Var}_X(C_{CV}) + \text{bias}(C_{CV})^2,$$

where

$$\text{bias}(C_{CV}) = E_X(C_{CV}) - E_X(C[f_N]) = E \left( \sum_i \frac{1}{K} C[f_{-N(i)}] \right) - E(C[f_N])$$
Decomposing Bias and Variance

- $C_{CV}$ is an estimator of the expected generalization error $E_X(C[f_N])$:

$$\text{MSE}(C_{CV}) = \text{Var}_X(C_{CV}) + \text{bias}(C_{CV})^2,$$

where

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

$$= \frac{1}{K} \sum_i E\left(C[f_{-N(i)}]\right) - E(C[f_N])$$
Decomposing Bias and Variance

- $C_{CV}$ is an estimator of the expected generalization error $E_X(C[f_N])$:

\[
\text{MSE}(C_{CV}) = \text{Var}_X(C_{CV}) + \text{bias}(C_{CV})^2, \text{ where}
\]

\[
\text{bias}(C_{CV}) = E_X(C_{CV}) - E_X(C[f_N]) = E \left( \sum_i \frac{1}{K} C[f_{-N(i)}] \right) - E(C[f_N])
\]

\[
= \frac{1}{K} \sum_i E \left( C[f_{-N(i)}] \right) - E(C[f_N])
\]

\[
= E \left( C[f_{-N(s)}] \right) - E(C[f_N]), \forall s
\]
Decomposing Bias and Variance

- $C_{\text{CV}}$ is an estimator of the expected generalization error $E_X(C[f_N])$:

\[
\text{MSE}(C_{\text{CV}}) = \text{Var}_X(C_{\text{CV}}) + \text{bias}(C_{\text{CV}})^2, \text{ where}
\]

\[
\text{bias}(C_{\text{CV}}) = E_X(C_{\text{CV}}) - E_X(C[f_N]) = E \left( \sum_i \frac{1}{K} C[f_{-N(i)}] \right) - E(C[f_N])
\]
\[
= \frac{1}{K} \sum_i E(C[f_{-N(i)}]) - E(C[f_N])
\]
\[
= E(C[f_{-N(s)}]) - E(C[f_N]), \forall s
\]
\[
= \text{bias} \left( C[f_{-N(s)}] \right), \forall s
\]
Decomposing Bias and Variance

$C_{CV}$ is an estimator of the expected generalization error $E_X(C[f_N])$:

$$
MSE(C_{CV}) = \text{Var}_X(C_{CV}) + \text{bias}(C_{CV})^2,
$$

where

$$
bias(C_{CV}) = E_X(C_{CV}) - E_X(C[f_N]) = E \left( \sum_i \frac{1}{K} C[f_{-N(i)}] \right) - E(C[f_N])
= \frac{1}{K} \sum_i E \left( C[f_{-N(i)}] \right) - E(C[f_N])
= E \left( C[f_{-N(s)}] \right) - E(C[f_N]), \forall s
= \text{bias} \left( C[f_{-N(s)}] \right), \forall s
$$

$$
\text{Var}_X(C_{CV}) = \text{Var} \left( \sum_i \frac{1}{K} C[f_{-N(i)}] \right) = \frac{1}{K^2} \text{Var} \left( \sum_i C[f_{-N(i)}] \right)
$$
Decomposing Bias and Variance

- \( C_{CV} \) is an estimator of the expected generalization error \( E_X(C[f_N]) \):

\[
\text{MSE}(C_{CV}) = \text{Var}_X(C_{CV}) + \text{bias}(C_{CV})^2, \text{ where}
\]

\[
\text{bias}(C_{CV}) = E_X(C_{CV}) - E_X(C[f_N]) = E\left(\sum_i \frac{1}{K} C[f_{-N(i)}]\right) - E(C[f_N])
\]

\[
= \frac{1}{K} \sum_i E\left(C[f_{-N(i)}]\right) - E(C[f_N])
\]

\[
= E\left(C[f_{-N(s)}]\right) - E(C[f_N]), \forall s
\]

\[
= \text{bias}\left(C[f_{-N(s)}]\right), \forall s
\]

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

\[
= \frac{1}{K^2} \left(\sum_i \text{Var}\left(C[f_{-N(i)}]\right) + 2 \sum_{i,j,i > j} \text{Cov}_X\left(C[f_{-N(i)}], C[f_{-N(j)}]\right)\right)
\]
Decomposing Bias and Variance

- $C_{CV}$ is an estimator of the expected generalization error $E_X(C[f_N])$:

$$\text{MSE}(C_{CV}) = \text{Var}_X(C_{CV}) + \text{bias}(C_{CV})^2,$$

where

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

$$= \frac{1}{K} \sum_i E \left( C[f_{-N(i)}] \right) - E(C[f_N])$$

$$= E \left( C[f_{-N(s)}] \right) - E(C[f_N]), \forall s$$

$$= \text{bias} \left( C[f_{-N(s)}] \right), \forall s$$

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

$$= \frac{1}{K^2} \left( \sum_i \text{Var} \left( C[f_{-N(i)}] \right) + 2 \sum_{i,j,i>j} \text{Cov}_X \left( C[f_{-N(i)}], C[f_{-N(j)}] \right) \right)$$

$$= \frac{1}{K} \text{Var} \left( C[f_{-N(s)}] \right) + \frac{2}{K^2} \sum_{i,j,i>j} \text{Cov} \left( C[f_{-N(i)}], C[f_{-N(j)}] \right), \forall s$$
How Many Folds $K$? II

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

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

Shan-Hung Wu (CS, NTHU)

CV & Ensembling

Machine Learning 11 / 34
How Many Folds \( K \) ? II

\[
\text{MSE}(C_{\text{CV}}) = \text{Var}_X(C_{\text{CV}}) + \text{bias}(C_{\text{CV}})^2, \quad \text{where}
\]

\[
\text{bias}(C_{\text{CV}}) = \text{bias} \left( C[f_{-N(s)}] \right), \forall s
\]

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

- 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 \((N)\)
- Furthermore, we can reduce \( \text{Var}(C_{\text{CV}}) \) by **making** \( f_{-N(i)} \) **and** \( f_{-N(j)} \) **uncorrelated**
How Many Folds $K$? III

$$\text{bias} (C_{CV}) = \text{bias} (C[f_{-N(s)}]), \forall s$$

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

- With a large $K$, the $C_{CV}$ tends to have:
How Many Folds $K$? III

\[
\begin{align*}
\text{bias} \left( C_{CV} \right) &= \text{bias} \left( C[f_{-N(s)}] \right), \forall s \\
\text{Var}_X \left( C_{CV} \right) &= \frac{1}{K} \text{Var} \left( C[f_{-N(s)}] \right) + \frac{2}{K^2} \sum_{i,j,j>i} \text{Cov} \left( C[f_{-N(i)}], C[f_{-N(j)}] \right), \forall s
\end{align*}
\]

- 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

\[
\begin{align*}
\text{bias} (C_{\text{CV}}) &= \text{bias} \left( C[f_{-N(s)}] \right), \forall s \\
\text{Var}_X (C_{\text{CV}}) &= \frac{1}{K} \text{Var} (C[f_{-N(s)}]) + \frac{2}{K^2} \sum_{i,j,i>j} \text{Cov} \left( C[f_{-N(i)}], C[f_{-N(j)}] \right), \forall s
\end{align*}
\]

- With a large $K$, the $C_{\text{CV}}$ tends to have:
  - Low $\text{bias} (C[f_{-N(s)}])$ and $\text{Var} (C[f_{-N(s)}])$, 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 $X \setminus X^{(i)}$ and $X \setminus X^{(j)}$ are more similar thus $C[f_{-N(i)}]$ and $C[f_{-N(j)}]$ are more positively correlated
How Many Folds $K$? IV

\[
\text{bias}(C_{CV}) = \text{bias}(C[f_{-N(s)}]), \forall s
\]
\[
\text{Var}_X(C_{CV}) = \frac{1}{K} \text{Var}(C[f_{-N(s)}]) + \frac{2}{K^2} \sum_{i,j\neq i} \text{Cov}(C[f_{-N(i)}], C[f_{-N(j)}]), \forall s
\]

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

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

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

- In practice, we usually set $K = 5$ or 10
Leave-One-Out CV

\[
\text{bias} (C_{CV}) = \text{bias} (C[f_{-N(s)}]), \forall s
\]

\[
\text{Var}_X (C_{CV}) = \frac{1}{K} \text{Var} (C[f_{-N(s)}]) + \frac{2}{K^2} \sum_{i,j>i} \text{Cov} (C[f_{-N(i)}], C[f_{-N(j)}]), \forall s
\]

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

\[\text{MSE} \quad \text{5-Fold CV} \quad \text{10-Fold CV} \quad \text{Exp} \quad \text{Bayes}\]

\[\text{N} \quad 10 \quad 15 \quad 20 \quad 25 \quad 30 \quad 35 \quad 40 \quad 45\]

\[\text{MSE} \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5\]
Leave-One-Out CV

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

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

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

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

<table>
<thead>
<tr>
<th>Voting Rule</th>
<th>Formular</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sum</strong></td>
<td>$\tilde{y}<em>k = \frac{1}{L} \sum</em>{j=1}^{L} \hat{y}^{(j)}_k$</td>
</tr>
<tr>
<td><strong>Weighted sum</strong></td>
<td>$\tilde{y}_k = \sum_j w_j \hat{y}^{(j)}_k$, $w_j \geq 0$, $\sum_j w_j = 1$</td>
</tr>
<tr>
<td><strong>Median</strong></td>
<td>$\tilde{y}_k = \text{median}_j \hat{y}^{(j)}_k$</td>
</tr>
<tr>
<td><strong>Minimum</strong></td>
<td>$\tilde{y}_k = \text{min}_j \hat{y}^{(j)}_k$</td>
</tr>
<tr>
<td><strong>Maximum</strong></td>
<td>$\tilde{y}_k = \text{max}_j \hat{y}^{(j)}_k$</td>
</tr>
<tr>
<td><strong>Product</strong></td>
<td>$\tilde{y}_k = \prod_j \hat{y}^{(j)}_k$</td>
</tr>
</tbody>
</table>
Why Voting Works? I

Assume that each $\hat{y}(j)$ has the expected value $E_X \hat{y}(j) | x$ and variance $Var_X \hat{y}(j) | x$.

When $w_j = 1/L$, we have:

$$E_X (\tilde{y} | x) = E \frac{1}{L} \hat{y}(j) | x = E \hat{y}(j) | x = E \hat{y}(j) | x$$

$$Var_X (\tilde{y} | x) = Var \frac{1}{L} \hat{y}(j) | x = \frac{1}{L^2} Var \hat{y}(j) | x = Var \hat{y}(j) | x + 2 \frac{1}{L^2} \sum_{i, j, i < j} Cov \hat{y}(i), \hat{y}(j) | x$$

The expected value doesn't change, so the bias doesn't change.
Why Voting Works? 1

- Assume that each $\hat{y}^{(j)}$ has the expected value $E_X (\hat{y}^{(j)} | x)$ and variance $\text{Var}_X (\hat{y}^{(j)} | x)$
- When $w_j = 1/L$, we have:

\[
E_X (\tilde{y} | x) = E \left( \sum_j \frac{1}{L} \hat{y}^{(j)} | x \right) = \frac{1}{L} \sum_j E (\hat{y}^{(j)} | x) = E (\hat{y}^{(j)} | x)
\]

\[
\text{Var}_X (\tilde{y} | x) = \text{Var} \left( \sum_j \frac{1}{L} \hat{y}^{(j)} | x \right) = \frac{1}{L^2} \text{Var} \left( \sum_j \hat{y}^{(j)} | x \right) = \frac{1}{L} \text{Var} (\hat{y}^{(j)} | x) + \frac{2}{L^2} \sum_{i,j, i < j} \text{Cov} (\hat{y}^{(i)}, \hat{y}^{(j)} | x)
\]
Why Voting Works? I

- Assume that each $\hat{y}^{(j)}$ has the expected value $E_X (\hat{y}^{(j)} | x)$ and variance $\text{Var}_X (\hat{y}^{(j)} | x)$
- When $w_j = 1/L$, we have:

$$E_X (\tilde{y} | x) = E \left( \sum_j \frac{1}{L} \hat{y}^{(j)} | x \right) = \frac{1}{L} \sum_j E \left( \hat{y}^{(j)} | x \right) = E \left( \hat{y}^{(j)} | x \right)$$

$$\text{Var}_X (\tilde{y} | x) = \text{Var} \left( \sum_j \frac{1}{L} \hat{y}^{(j)} | x \right) = \frac{1}{L^2} \text{Var} \left( \sum_j \hat{y}^{(j)} | x \right)$$

$$= \frac{1}{L} \text{Var} \left( \hat{y}^{(j)} | x \right) + \frac{2}{L^2} \sum_{i,j, i < j} \text{Cov} \left( \hat{y}^{(i)}, \hat{y}^{(j)} | x \right)$$

- The expected value doesn’t change, so the bias doesn’t change
Why Voting Works? II

\[
\text{Var}_{\mathbf{x}} (\hat{y} \mid \mathbf{x}) = \frac{1}{L} \text{Var} (\hat{y}^{(j)} \mid \mathbf{x}) + \frac{2}{L^2} \sum_{i, j, i < j} \text{Cov} (\hat{y}^{(i)}, \hat{y}^{(j)} \mid \mathbf{x})
\]
Why Voting Works? II

$$\text{Var}_X(\tilde{y} | x) = \frac{1}{L} \text{Var}(\hat{y}^{(j)} | x) + \frac{2}{L^2} \sum_{i,j, i<j} \text{Cov}(\hat{y}^{(i)}, \hat{y}^{(j)} | x)$$

- If $\hat{y}^{(i)}$ and $\hat{y}^{(j)}$ are uncorrelated, the variance can be reduced.
Why Voting Works? II

\[
\text{Var}_X (\tilde{y} | x) = \frac{1}{L} \text{Var}(\hat{y}^{(j)} | x) + \frac{2}{L^2} \sum_{i,j,i \neq j} \text{Cov}(\hat{y}^{(i)}, \hat{y}^{(j)} | x)
\]

- 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

1. Cross Validation
   - How Many Folds?

2. Ensemble Methods
   - Voting
   - Bagging
   - Boosting
   - Why AdaBoost Works?
Bagging (short for bootstrap aggregating) is a voting method, but base-learners are made different deliberately.

How?

1. Generate $L$ slightly different samples from a given sample is done by bootstrap: given $X$ of size $N$, we draw $N$ points randomly from $X$ with replacement to get $X(j)$.

   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?
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 $X$ of size $N$, we draw $N$ points randomly from $X$ with replacement to get $X^{(j)}$.
   - 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 $\mathbf{X}$ of size $N$, we draw $N$ points randomly from $\mathbf{X}$ with replacement to get $\mathbf{X}^{(j)}$.
   - It is possible that some instances are drawn more than once and some are not at all.

2. Train a base-learner for each $\mathbf{X}^{(j)}$. 

Shan-Hung Wu (CS, NTHU)  CV & Ensembling  Machine Learning  22 / 34
Outline

1 Cross Validation
   - How Many Folds?

2 Ensemble Methods
   - Voting
   - Bagging
   - Boosting
   - Why AdaBoost Works?
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

\[
d(j)(x) \in \{1, -1\}
\]

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), while a strong learner has arbitrarily small error probability.
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: $d^{(j)}(x) \in \{1, -1\}$
- 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 $X^{(1)}$ to train the first learner $d^{(1)}$ and feed $X^{(2)}$ to $d^{(1)}$
Original Boosting Algorithm

Training

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

Training

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

- **Training**

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

Notation:

\[ P_{i,j} : \text{probability that a example } (x_i, y_i) \text{ is drawn to train the } j \text{th base-learner } d_j \]

\[ e_j = \frac{1}{y_i} P_{i,j} d_j (x_i) \]

Error of \( d_j \) on its training set
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:**
- $\Pr^{(i,j)}$: probability that an example $(x^{(i)}, y^{(i)})$ is drawn to train the $j$th base-learner $d^{(j)}$
- $\varepsilon^{(j)} = \sum_i \Pr^{(i,j)} 1(y^{(i)} \neq d^{(j)}(x^{(i)}))$: 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 $\text{Pr}^{(i,1)} = \frac{1}{N}$ for all $i$
2. Start from $j = 1$:
   1. Randomly draw $N$ examples from $\mathbb{X}$ with probabilities $\text{Pr}^{(i,j)}$ 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
      $\text{Pr}^{(i,j+1)} = \text{Pr}^{(i,j)} \cdot \exp(-\alpha_j y^{(i)} d^{(j)}(x^{(i)}))$ 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
      \[
      \Pr^{(i,j+1)} = \Pr^{(i,j)} \cdot \exp(-\alpha_j y^{(i)} d^{(j)}(x^{(i)})) \quad \text{for all } i
      \]
   4. Normalize \( \Pr^{(i,j+1)} \), \( \forall i \), by multiplying \( \left( \sum_i \Pr^{(i,j+1)} \right)^{-1} \)

Testing

Given \( x, c \) calculate \( \hat{y}^{(j)} \) for all \( j \)

Make final prediction \( \tilde{y} \) by voting:
\[
\tilde{y} = \sum_j \alpha_j d^{(j)}(x)
\]
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
      $\Pr^{(i,j+1)} = \Pr^{(i,j)} \cdot \exp(-\alpha_j y^{(i)} d^{(j)}(x^{(i)}))$ 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 $x$, calculate $\hat{y}^{(j)}$ for all $j$
2. Make final prediction $\tilde{y}$ by voting: $\tilde{y} = \sum_j \alpha_j d^{(j)}(x)$
Example

\[ d^{(j+1)} \text{ complements } d^{(j)} \text{ and } d^{(j-1)} \text{ by focusing on predictions they disagree} \]
Example

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

1 Cross Validation
   • How Many Folds?

2 Ensemble Methods
   • Voting
   • Bagging
   • Boosting
   • Why AdaBoost Works?
Why AdaBoost Works

- Why AdaBoost improves performance?
Why AdaBoost Works

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

Empirical study: AdaBoost reduces overfitting as $L$ grows, even when there is no training error. AdaBoost increases margin.

Shan-Hung Wu (CS, NTHU)
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]
Margin as Confidence of Predictions

- Recall in SVC, a larger margin improves generalizability.

\[
\text{margin}(x_i, y_i) = y_i f(x_i) = \sum_{j:\ y_i = d(j)(x_i)} a_j \cdot \mathbf{w}_j:
\]

Shan-Hung Wu (CS, NTHU) CV & Ensembling Machine Learning 32 / 34
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 \((x^{(i)}, y^{(i)}) \in \mathcal{X}\) as:

\[
\text{margin}(x^{(i)}, y^{(i)}) = y^{(i)}f(x^{(i)}) = \sum_{j:y^{(i)} = d^{(j)}(x^{(i)})} \alpha_j - \sum_{j:y^{(i)} \neq d^{(j)}(x^{(i)})} \alpha_j
\]
Margin Distribution

- Margin distribution over $\theta$:

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

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

- Margin distribution over $\theta$:

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

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

![Graph showing cumulative distribution with legend](image)