Workflow of Supervised Techniques

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm,

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm, possibly matched in some way to nature of problem.

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm, possibly matched in some way to nature of problem.
3. Adjust algorithm for optimal performance,

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm, possibly matched in some way to nature of problem.
3. Adjust algorithm for optimal performance, perhaps using validation set and/or some kind of cross-validation.

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).
2. Decide on the algorithm, possibly matched in some way to nature of problem.
3. Adjust algorithm for optimal performance, perhaps using validation set and/or some kind of cross-validation.
4. Report accuracy in test set,

Workflow of Supervised Techniques

1. Obtain/code the training set and decide on relevant features (preprocess).

2. Decide on the algorithm, possibly matched in some way to nature of problem.

3. Adjust algorithm for optimal performance, perhaps using validation set and/or some kind of cross-validation.

4. Report accuracy in test set, possibly combine with other learners in ensemble.

Notes and Issues

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data.

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data. Often used interchangeably with *machine learning*: idea that computers could 'learn' relationships without specific programming.

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data. Often used interchangeably with *machine learning*: idea that computers could 'learn' relationships without specific programming.

Typically used when $p >> n$:

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data. Often used interchangeably with *machine learning*: idea that computers could 'learn' relationships without specific programming.

Typically used when $p >> n$: number of parameters (e.g. coefficients on terms) is far larger than number of observations (e.g. speakers or documents).

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data. Often used interchangeably with *machine learning*: idea that computers could 'learn' relationships without specific programming.

Typically used when $p >> n$: number of parameters (e.g. coefficients on terms) is far larger than number of observations (e.g. speakers or documents).

$\rightarrow$ results in general curse of dimensionality wherein feature matrix is large (e.g. $100k$ columns) and sparse and thus obtaining meaningful estimates is difficult.

Notes and Issues

Supervised techniques are about learning relationship between X and labeled data. Often used interchangeably with *machine learning*: idea that computers could 'learn' relationships without specific programming.

Typically used when $p >> n$: number of parameters (e.g. coefficients on terms) is far larger than number of observations (e.g. speakers or documents).

$\rightarrow$ results in general curse of dimensionality wherein feature matrix is large (e.g. $100k$ columns) and sparse and thus obtaining meaningful estimates is difficult.

So techniques may require careful tuning of *regularization parameters* to obtain good performance.

Notes and Issues II

Notes and Issues II

Once we have the training set we face a dilemma...

Notes and Issues II

Once we have the training set we face a dilemma. . .

1 we can use our technique to fit a very complicated model to this data (perfectly),

Notes and Issues II

Once we have the training set we face a dilemma. . .

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements.

Notes and Issues II

Once we have the training set we face a dilemma. . .

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

Notes and Issues II

Once we have the training set we face a dilemma...

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set,

Notes and Issues II

Once we have the training set we face a dilemma...

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

Notes and Issues II

Once we have the training set we face a dilemma. . .

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

2. we can use be more relaxed about the fit of our algorithm in the training set,

Notes and Issues II

Once we have the training set we face a dilemma...

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

→ we have overfit to our training set, and may do poorly on our test set.

2. we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance.

Notes and Issues II

Once we have the training set we face a dilemma...

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

2 we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance,

Notes and Issues II

Once we have the training set we face a dilemma...

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

2. we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance, but may induce bias in the sense that we miss important relationships in the data.

Notes and Issues II

Once we have the training set we face a dilemma...

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

→ we have overfit to our training set, and may do poorly on our test set.

2. we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance, but may induce bias in the sense that we miss important relationships in the data.

→ we have underfit to our training set,

Notes and Issues II

Once we have the training set we face a dilemma. . .

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

2. we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance, but may induce bias in the sense that we miss important relationships in the data.

$\rightarrow$ we have underfit to our training set, and may do poorly on our test set.

Notes and Issues II

Once we have the training set we face a dilemma. . .

1. we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

2. we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance, but may induce bias in the sense that we miss important relationships in the data.

$\rightarrow$ we have underfit to our training set, and may do poorly on our test set.

So managing the *bias-variance tradeoff* is a key element of supervised learning,

Notes and Issues II

Once we have the training set we face a dilemma...

1 we can use our technique to fit a very complicated model to this data (perfectly), including noisy elements. This avoids bias, but it incurs variance (when we move to the test set).

$\rightarrow$ we have overfit to our training set, and may do poorly on our test set.

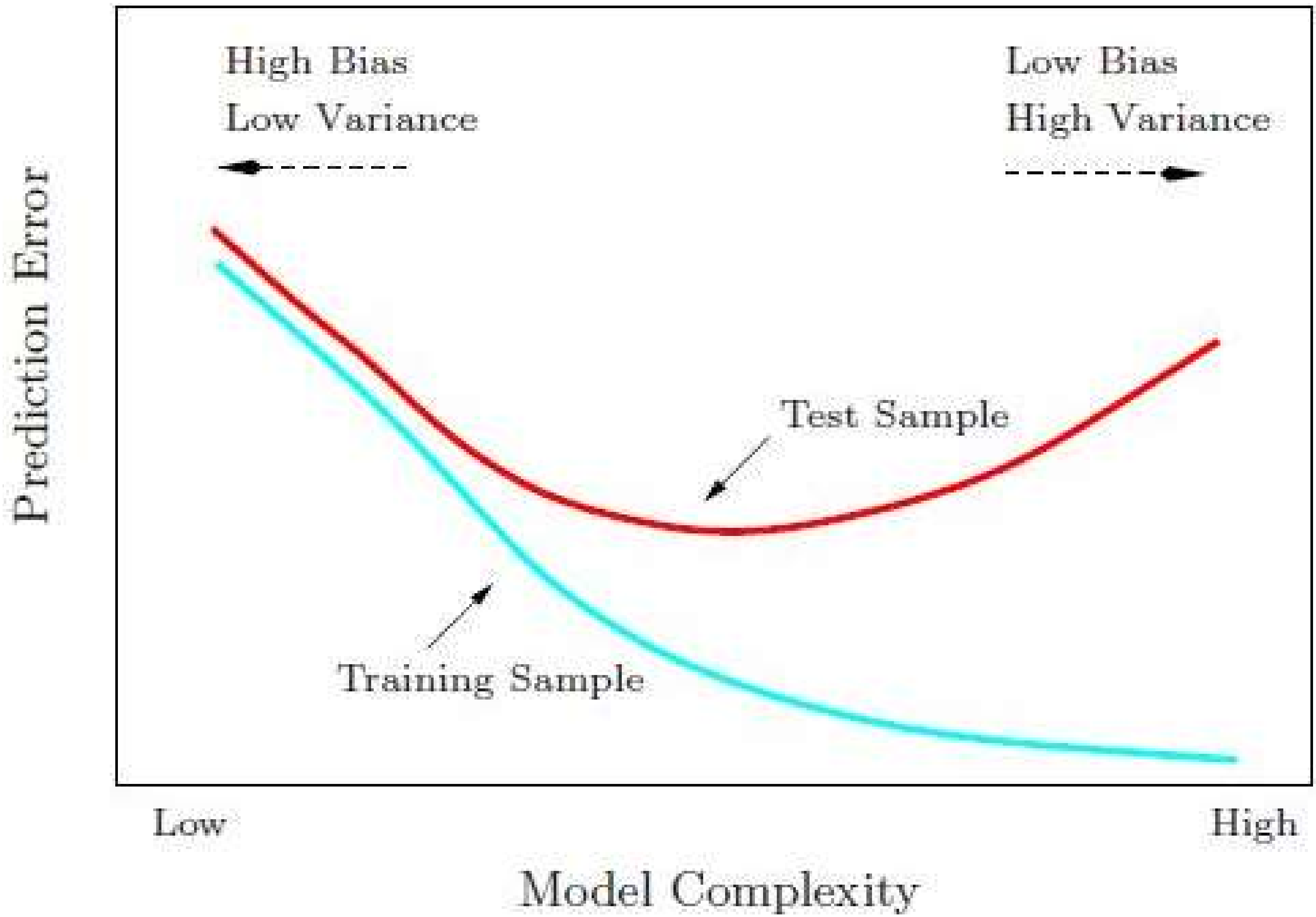
2 we can use be more relaxed about the fit of our algorithm in the training set, and accept some imperfections in performance. This avoids high variance, but may induce bias in the sense that we miss important relationships in the data.

$\rightarrow$ we have underfit to our training set, and may do poorly on our test set.

So managing the *bias-variance tradeoff* is a key element of supervised learning, and we may need to tune our algorithms with that in mind.

Bias-Variance Tradeoff (Hastie et al, p38)

Bias-Variance Tradeoff (Hastie et al, p38)



Cross-Validation

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'),

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
    1. grab one of the k chunks as a validation set (each only used once)

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
   1. grab one of the k chunks as a validation set (each only used once)
   2. grab the other $k - 1$ chunks as a training set

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
   1. grab one of the k chunks as a validation set (each only used once)
   2. grab the other $k - 1$ chunks as a training set
   3. test on the validation set,

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
    1. grab one of the k chunks as a validation set (each only used once)
    2. grab the other $k - 1$ chunks as a training set
    3. test on the validation set, record prediction error

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
    1. grab one of the k chunks as a validation set (each only used once)
    2. grab the other $k - 1$ chunks as a training set
    3. test on the validation set, record prediction error
3. Average over runs to get prediction error estimate.

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
    1. grab one of the k chunks as a validation set (each only used once)
    2. grab the other $k - 1$ chunks as a training set
    3. test on the validation set, record prediction error
3. Average over runs to get prediction error estimate.

$\rightarrow$ Can follow same steps for models of different specifications; variant on this approach can be used for model selection, directly.

Cross-Validation

Want to properly estimate model performance (bias and especially variance).

Popular and efficient approach is k-fold cross validation, esp when we don't have enough data to form a completely separate validation set.

1. divide all data into k equal size chunks ($k = 10$ is common; $k = n$ is 'leave one out'), and set the parameter(s) of model at particular value,
2. repeat the following k times (folds):
   1. grab one of the k chunks as a validation set (each only used once)
   2. grab the other $k - 1$ chunks as a training set
   3. test on the validation set, record prediction error
3. Average over runs to get prediction error estimate.

$\rightarrow$ Can follow same steps for models of different specifications; variant on this approach can be used for model selection, directly.

Graphically

Graphically