Supervised learning: training sets, testing sets, overfitting, correlation versus cause and effect, cross-validation, learning curve, inductive bias

Relevant Readings: None

CS495 - Machine Learning, Fall 2009

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs
  - Our first few projects will be on *concept learning* - the special case where the output is boolean

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs
  - Our first few projects will be on *concept learning* - the special case where the output is boolean
- Here's a general process for it:

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs
  - Our first few projects will be on *concept learning* - the special case where the output is boolean
- Here's a general process for it:
  - 1) First, reframe the problem so that you have a function defined in terms of a fixed number of features (attributes)

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs
  - Our first few projects will be on *concept learning* - the special case where the output is boolean
- Here's a general process for it:
  - 1) First, reframe the problem so that you have a function defined in terms of a fixed number of features (attributes)
    - May need to do some "feature engineering" to make it work

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs
  - Our first few projects will be on *concept learning* - the special case where the output is boolean
- Here's a general process for it:
  - 1) First, reframe the problem so that you have a function defined in terms of a fixed number of features (attributes)
    - May need to do some "feature engineering" to make it work
  - 2) Collect a big set of example data

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs
  - Our first few projects will be on *concept learning* - the special case where the output is boolean
- Here's a general process for it:
  - 1) First, reframe the problem so that you have a function defined in terms of a fixed number of features (attributes)
    - May need to do some "feature engineering" to make it work
  - 2) Collect a big set of example data
  - 3) Partition your data samples into two sets: the *training set* and the *testing set*

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs
  - Our first few projects will be on *concept learning* - the special case where the output is boolean
- Here's a general process for it:
  - 1) First, reframe the problem so that you have a function defined in terms of a fixed number of features (attributes)
    - May need to do some "feature engineering" to make it work
  - 2) Collect a big set of example data
  - 3) Partition your data samples into two sets: the *training set* and the *testing set*
  - 4) Select a learning algorithm

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs
  - Our first few projects will be on *concept learning* - the special case where the output is boolean
- Here's a general process for it:
  - 1) First, reframe the problem so that you have a function defined in terms of a fixed number of features (attributes)
    - May need to do some "feature engineering" to make it work
  - 2) Collect a big set of example data
  - 3) Partition your data samples into two sets: the *training set* and the *testing set*
  - 4) Select a learning algorithm
  - 5) Train the learning algorithm on the training set

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs
  - Our first few projects will be on *concept learning* - the special case where the output is boolean
- Here's a general process for it:
  - 1) First, reframe the problem so that you have a function defined in terms of a fixed number of features (attributes)
    - May need to do some "feature engineering" to make it work
  - 2) Collect a big set of example data
  - 3) Partition your data samples into two sets: the *training set* and the *testing set*
  - 4) Select a learning algorithm
  - 5) Train the learning algorithm on the training set
  - 6) Use the testing set to estimate the algorithm's performance

Supervised Learning

- *Supervised Learning* is learning a function from example inputs and outputs
  - Our first few projects will be on *concept learning* - the special case where the output is boolean
- Here's a general process for it:
  - 1) First, reframe the problem so that you have a function defined in terms of a fixed number of features (attributes)
    - May need to do some "feature engineering" to make it work
  - 2) Collect a big set of example data
  - 3) Partition your data samples into two sets: the *training set* and the *testing set*
  - 4) Select a learning algorithm
  - 5) Train the learning algorithm on the training set
  - 6) Use the testing set to estimate the algorithm's performance

Overfitting

- Many learning algorithms can be trained to the point where they fit the training set at or near 100% accuracy

Overfitting

- Many learning algorithms can be trained to the point where they fit the training set at or near 100% accuracy
- How could that a bad thing?

Overfitting

- Many learning algorithms can be trained to the point where they fit the training set at or near 100% accuracy
- How could that a bad thing?
  - Because it may do poorly on the testing set (i.e. the “real thing”)

Overfitting

- Many learning algorithms can be trained to the point where they fit the training set at or near 100% accuracy
- How could that a bad thing?
  - Because it may do poorly on the testing set (i.e. the "real thing")
  - The reason why is the algorithm is picking up too much on random noise and spurious correlations

Overfitting

- Many learning algorithms can be trained to the point where they fit the training set at or near 100% accuracy
- How could that a bad thing?
  - Because it may do poorly on the testing set (i.e. the "real thing")
  - The reason why is the algorithm is picking up too much on random noise and spurious correlations
  - This is called *overfitting*

Overfitting

- Many learning algorithms can be trained to the point where they fit the training set at or near 100% accuracy
- How could that a bad thing?
  - Because it may do poorly on the testing set (i.e. the "real thing")
  - The reason why is the algorithm is picking up too much on random noise and spurious correlations
  - This is called *overfitting*
- The same thing happens in human learning:

Overfitting

- Many learning algorithms can be trained to the point where they fit the training set at or near 100% accuracy
- How could that a bad thing?
  - Because it may do poorly on the testing set (i.e. the "real thing")
  - The reason why is the algorithm is picking up too much on random noise and spurious correlations
  - This is called *overfitting*
- The same thing happens in human learning:
  - It can manifest as superstition

Overfitting

- Many learning algorithms can be trained to the point where they fit the training set at or near 100% accuracy
- How could that a bad thing?
  - Because it may do poorly on the testing set (i.e. the "real thing")
  - The reason why is the algorithm is picking up too much on random noise and spurious correlations
  - This is called *overfitting*
- The same thing happens in human learning:
  - It can manifest as superstition
  - Computer crash example

Overfitting

- Many learning algorithms can be trained to the point where they fit the training set at or near 100% accuracy
- How could that a bad thing?
  - Because it may do poorly on the testing set (i.e. the "real thing")
  - The reason why is the algorithm is picking up too much on random noise and spurious correlations
  - This is called *overfitting*
- The same thing happens in human learning:
  - It can manifest as superstition
  - Computer crash example

Correlation $\neq$ cause and effect

- Consider the following example [William C. Burns]:

Correlation $\neq$ cause and effect

- Consider the following example [William C. Burns]:
  - 1) Get data on all the fires in San Francisco for the last ten years

Correlation $\neq$ cause and effect

- Consider the following example [William C. Burns]:
  - 1) Get data on all the fires in San Francisco for the last ten years
  - 2) Correlate the number of fire engines at each fire and the damages in dollars at each fire

Correlation $\neq$ cause and effect

- Consider the following example [William C. Burns]:
  - 1) Get data on all the fires in San Francisco for the last ten years
  - 2) Correlate the number of fire engines at each fire and the damages in dollars at each fire
  - 3) Note the significant relationship between number of fire engines and the amount of damage

Correlation $\neq$ cause and effect

- Consider the following example [William C. Burns]:
  - 1) Get data on all the fires in San Francisco for the last ten years
  - 2) Correlate the number of fire engines at each fire and the damages in dollars at each fire
  - 3) Note the significant relationship between number of fire engines and the amount of damage
  - 4) Conclude that fire engines cause the damage!

Correlation $\neq$ cause and effect

- Consider the following example [William C. Burns]:
  - 1) Get data on all the fires in San Francisco for the last ten years
  - 2) Correlate the number of fire engines at each fire and the damages in dollars at each fire
  - 3) Note the significant relationship between number of fire engines and the amount of damage
  - 4) Conclude that fire engines cause the damage!

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate
- The idea is simple:

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate
- The idea is simple:
  - Repeat the training and testing process using many different partitions of your data

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate
- The idea is simple:
  - Repeat the training and testing process using many different partitions of your data
- This helps reduce statistical error

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate
- The idea is simple:
  - Repeat the training and testing process using many different partitions of your data
- This helps reduce statistical error
- Several variants are available, for example:

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate
- The idea is simple:
  - Repeat the training and testing process using many different partitions of your data
- This helps reduce statistical error
- Several variants are available, for example:
  - k-fold cross-validation

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate
- The idea is simple:
  - Repeat the training and testing process using many different partitions of your data
- This helps reduce statistical error
- Several variants are available, for example:
  - k-fold cross-validation
    - Partition data into k sets

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate
- The idea is simple:
  - Repeat the training and testing process using many different partitions of your data
- This helps reduce statistical error
- Several variants are available, for example:
  - k-fold cross-validation
    - Partition data into k sets
    - Do this k times: train on $k-1$ sets, test on the remaining 1

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate
- The idea is simple:
  - Repeat the training and testing process using many different partitions of your data
- This helps reduce statistical error
- Several variants are available, for example:
  - k-fold cross-validation
    - Partition data into k sets
    - Do this k times: train on $k - 1$ sets, test on the remaining 1
  - Repeated random sub-sampling validation

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate
- The idea is simple:
  - Repeat the training and testing process using many different partitions of your data
- This helps reduce statistical error
- Several variants are available, for example:
  - k-fold cross-validation
    - Partition data into k sets
    - Do this k times: train on $k-1$ sets, test on the remaining 1
  - Repeated random sub-sampling validation
    - Just keep choosing the training and testing set completely at random at some fixed ratio; say 80% training set and 20% testing set

Cross-validation

- Splitting your data into a single training set and testing set may not give the best estimate of future performance
- Some form of *cross-validation* is usually employed to get a more rigorous estimate
- The idea is simple:
  - Repeat the training and testing process using many different partitions of your data
- This helps reduce statistical error
- Several variants are available, for example:
  - k-fold cross-validation
    - Partition data into k sets
    - Do this k times: train on $k - 1$ sets, test on the remaining 1
  - Repeated random sub-sampling validation
    - Just keep choosing the training and testing set completely at random at some fixed ratio; say 80% training set and 20% testing set

Learning curves

- Trial 1: Do training on one sample and test on the rest

Learning curves

- Trial 1: Do training on one sample and test on the rest
- Trial 2: Do training on two samples and test on the rest

Learning curves

- Trial 1: Do training on one sample and test on the rest
- Trial 2: Do training on two samples and test on the rest
- ...

Learning curves

- Trial 1: Do training on one sample and test on the rest
- Trial 2: Do training on two samples and test on the rest
- ...
- Trial n: Do training on all but one sample and test on the last one

Learning curves

- Trial 1: Do training on one sample and test on the rest
- Trial 2: Do training on two samples and test on the rest
- ...
- Trial n: Do training on all but one sample and test on the last one
- Graph the performance over these trials - this is called a *learning curve*

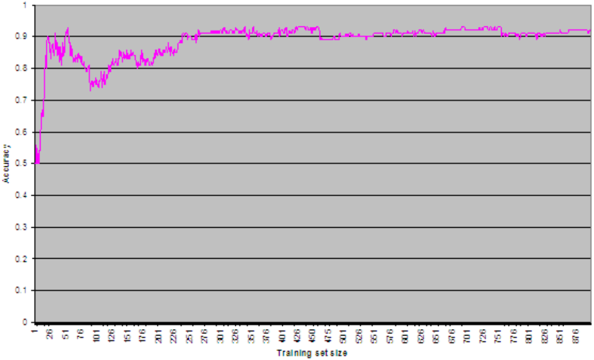
Learning curves

- Trial 1: Do training on one sample and test on the rest
- Trial 2: Do training on two samples and test on the rest
- ...
- Trial *n*: Do training on all but one sample and test on the last one
- Graph the performance over these trials - this is called a *learning curve*
- Better: do cross-validation in each trial to "smooth out" the curve

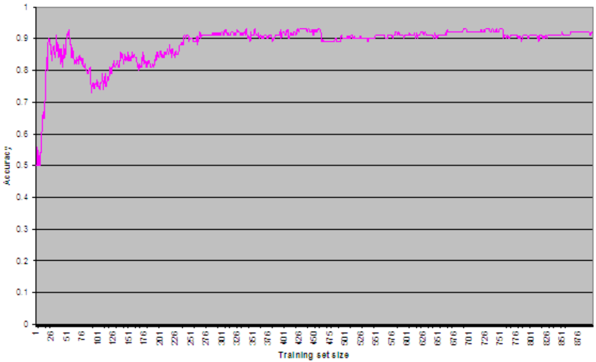
Learning curves

- Trial 1: Do training on one sample and test on the rest
- Trial 2: Do training on two samples and test on the rest
- ...
- Trial *n*: Do training on all but one sample and test on the last one
- Graph the performance over these trials - this is called a *learning curve*
- Better: do cross-validation in each trial to "smooth out" the curve

Learning curves



Learning curves



Inductive bias

- If you use (say) a neural network to model some target function, you are making an implicit assumption that the function can actually be modeled by that neural network

Inductive bias

- If you use (say) a neural network to model some target function, you are making an implicit assumption that the function can actually be modeled by that neural network
  - This may exclude a great number of possible functions from consideration

Inductive bias

- If you use (say) a neural network to model some target function, you are making an implicit assumption that the function can actually be modeled by that neural network
  - This may exclude a great number of possible functions from consideration
  - The hidden assumption that we are only considering certain possibilities for the target function is called *inductive bias*

Inductive bias

- If you use (say) a neural network to model some target function, you are making an implicit assumption that the function can actually be modeled by that neural network
  - This may exclude a great number of possible functions from consideration
  - The hidden assumption that we are only considering certain possibilities for the target function is called *inductive bias*
- Inductive bias is not necessarily a bad thing. Without it, learning is too difficult.

Inductive bias

- If you use (say) a neural network to model some target function, you are making an implicit assumption that the function can actually be modeled by that neural network
  - This may exclude a great number of possible functions from consideration
  - The hidden assumption that we are only considering certain possibilities for the target function is called *inductive bias*
- Inductive bias is not necessarily a bad thing. Without it, learning is too difficult.
- Another example of inductive bias

Inductive bias

- If you use (say) a neural network to model some target function, you are making an implicit assumption that the function can actually be modeled by that neural network
  - This may exclude a great number of possible functions from consideration
  - The hidden assumption that we are only considering certain possibilities for the target function is called *inductive bias*
- Inductive bias is not necessarily a bad thing. Without it, learning is too difficult.
- Another example of inductive bias
  - *Ockham's Razor*: "The simplest hypothesis that fits the data is the most likely one"

Inductive bias

- If you use (say) a neural network to model some target function, you are making an implicit assumption that the function can actually be modeled by that neural network
  - This may exclude a great number of possible functions from consideration
  - The hidden assumption that we are only considering certain possibilities for the target function is called *inductive bias*
- Inductive bias is not necessarily a bad thing. Without it, learning is too difficult.
- Another example of inductive bias
  - *Ockham's Razor*: "The simplest hypothesis that fits the data is the most likely one"
  - Works well in practice

Inductive bias

- If you use (say) a neural network to model some target function, you are making an implicit assumption that the function can actually be modeled by that neural network
  - This may exclude a great number of possible functions from consideration
  - The hidden assumption that we are only considering certain possibilities for the target function is called *inductive bias*
- Inductive bias is not necessarily a bad thing. Without it, learning is too difficult.
- Another example of inductive bias
  - *Ockham's Razor*: "The simplest hypothesis that fits the data is the most likely one"
  - Works well in practice
- Different learning algorithms include different kinds of inductive bias.

Inductive bias

- If you use (say) a neural network to model some target function, you are making an implicit assumption that the function can actually be modeled by that neural network
  - This may exclude a great number of possible functions from consideration
  - The hidden assumption that we are only considering certain possibilities for the target function is called *inductive bias*
- Inductive bias is not necessarily a bad thing. Without it, learning is too difficult.
- Another example of inductive bias
  - *Ockham's Razor*: "The simplest hypothesis that fits the data is the most likely one"
  - Works well in practice
- Different learning algorithms include different kinds of inductive bias.