Algorithm Independent Topics

Jason Corso

SUNY at Buffalo

Introduction

- Now that we've built an intuition for some pattern classifiers, let's take a step back and look at some of the more foundational underpinnings.
- Algorithm-Independent means
  - those mathematical foundations that do not depend upon the particular classifier or learning algorithm used;
  - techniques that can be used in conjunction with different learning algorithms, or provide guidance in their use.
- Specifically, we will cover
  1. Lack of inherent superiority of any one particular classifier;
  2. Some systematic ways for selecting a particular method over another for a given scenario;
  3. Methods for integrating component classifiers, including bagging, and (in depth) boosting.

No Free Lunch Theorem

- The key question we inspect: **If we are interested solely in the generalization performance, are there any reasons to prefer one classifier or learning algorithm over another?**

No Free Lunch Theorem

- The key question we inspect: **If we are interested solely in the generalization performance, are there any reasons to prefer one classifier or learning algorithm over another?**
- Simply put, **No!**

No Free Lunch Theorem

- The key question we inspect: **If we are interested solely in the generalization performance, are there any reasons to prefer one classifier or learning algorithm over another?**
- Simply put, **No!**
- If we find one particular algorithm outperforming another in a particular situation, it is a consequence of its fit to the particular pattern recognition problem, not the general superiority of the algorithm.

No Free Lunch Theorem

- The key question we inspect: **If we are interested solely in the generalization performance, are there any reasons to prefer one classifier or learning algorithm over another?**
- Simply put, **No!**
- If we find one particular algorithm outperforming another in a particular situation, it is a consequence of its fit to the particular pattern recognition problem, not the general superiority of the algorithm.
- NFLT should remind you that we need to focus on the particular problem at hand, the assumptions, the priors, the data and the cost. Recall the fish problem from earlier in the semester.

No Free Lunch Theorem

- The key question we inspect: **If we are interested solely in the generalization performance, are there any reasons to prefer one classifier or learning algorithm over another?**
- Simply put, **No!**
- If we find one particular algorithm outperforming another in a particular situation, it is a consequence of its fit to the particular pattern recognition problem, not the general superiority of the algorithm.
- NFLT should remind you that we need to focus on the particular problem at hand, the assumptions, the priors, the data and the cost. Recall the fish problem from earlier in the semester.
- For more information, refer to `http://www.no-free-lunch.org/`.

Judging Classifier Performance

- We need to be more clear about how we judge classifier performance.
- We've thus far considered the performance on an i.i.d. test data set.
- But, this has drawbacks in some cases:

Judging Classifier Performance

- We need to be more clear about how we judge classifier performance.
- We've thus far considered the performance on an i.i.d. test data set.
- But, this has drawbacks in some cases:
  1. In discrete situations with large training and testing sets, they necessarily overlap. We are hence testing on training patterns.

Judging Classifier Performance

- We need to be more clear about how we judge classifier performance.
- We've thus far considered the performance on an i.i.d. test data set.
- But, this has drawbacks in some cases:
    1. In discrete situations with large training and testing sets, they necessarily overlap. We are hence testing on training patterns.
    2. In these cases, most sufficiently powerful techniques (e.g., nearest neighbor methods) can perfectly learn the training set.

Judging Classifier Performance

- We need to be more clear about how we judge classifier performance.
- We've thus far considered the performance on an i.i.d. test data set.
- But, this has drawbacks in some cases:
  1. In discrete situations with large training and testing sets, they necessarily overlap. We are hence testing on training patterns.
  2. In these cases, most sufficiently powerful techniques (e.g., nearest neighbor methods) can perfectly learn the training set.
  3. For low-noise or low-Bayes error cases, if we use an algorithm powerful enough to learn the training set, then the upper limit of the i.i.d. error decreases as the training set size increases.

Judging Classifier Performance

- We need to be more clear about how we judge classifier performance.
- We've thus far considered the performance on an i.i.d. test data set.
- But, this has drawbacks in some cases:
  1. In discrete situations with large training and testing sets, they necessarily overlap. We are hence testing on training patterns.
  2. In these cases, most sufficiently powerful techniques (e.g., nearest neighbor methods) can perfectly learn the training set.
  3. For low-noise or low-Bayes error cases, if we use an algorithm powerful enough to learn the training set, then the upper limit of the i.i.d. error decreases as the training set size increases.
- Thus, we will use the **off-training set error**, which is the error on points not in the training set.
  - If the training set is very large, then the maximum size of the off-training set is necessarily small.

We need to tie things down with concrete notation.

- Consider a two-class problem.

We need to tie things down with concrete notation.

- Consider a two-class problem.
- We have a training set $\mathcal{D}$ consisting of pairs $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i$ being the patterns and $y_i = \pm 1$ being the classes for $i = 1, \ldots, n$.
    - Assume the data is discrete.

We need to tie things down with concrete notation.

- Consider a two-class problem.
- We have a training set $\mathcal{D}$ consisting of pairs $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i$ being the patterns and $y_i = \pm 1$ being the classes for $i = 1, \ldots, n$.
    - Assume the data is discrete.
- We assume our data has been generated by some unknown target function $F(\mathbf{x})$ which we want to learn (i.e., $y_i = F(\mathbf{x}_i)$).
    - Typically, we have some random component in the data giving a non-zero Bayes error rate.

We need to tie things down with concrete notation.

- Consider a two-class problem.
- We have a training set $\mathcal{D}$ consisting of pairs $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i$ being the patterns and $y_i = \pm 1$ being the classes for $i = 1, \dots, n$.
    - Assume the data is discrete.
- We assume our data has been generated by some unknown target function $F(\mathbf{x})$ which we want to learn (i.e., $y_i = F(\mathbf{x}_i)$).
    - Typically, we have some random component in the data giving a non-zero Bayes error rate.
- Let $\mathcal{H}$ denote the finite set of hypotheses or possible sets of parameters to be learned. Each element in the set, $h \in \mathcal{H}$, comprises the necessary set of arguments to fully specify a classifier (e.g., parameters of a Gaussian).

We need to tie things down with concrete notation.

- Consider a two-class problem.
- We have a training set $\mathcal{D}$ consisting of pairs $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i$ being the patterns and $y_i = \pm 1$ being the classes for $i = 1, \ldots, n$.
    - Assume the data is discrete.
- We assume our data has been generated by some unknown target function $F(\mathbf{x})$ which we want to learn (i.e., $y_i = F(\mathbf{x}_i)$).
    - Typically, we have some random component in the data giving a non-zero Bayes error rate.
- Let $\mathcal{H}$ denote the finite set of hypotheses or possible sets of parameters to be learned. Each element in the set, $h \in \mathcal{H}$, comprises the necessary set of arguments to fully specify a classifier (e.g., parameters of a Gaussian).
- $P(h)$ is the prior that the algorithm will learn hypothesis h.

We need to tie things down with concrete notation.

- Consider a two-class problem.
- We have a training set $\mathcal{D}$ consisting of pairs $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i$ being the patterns and $y_i = \pm 1$ being the classes for $i = 1, \ldots, n$.
  - Assume the data is discrete.
- We assume our data has been generated by some unknown target function $F(\mathbf{x})$ which we want to learn (i.e., $y_i = F(\mathbf{x}_i)$).
  - Typically, we have some random component in the data giving a non-zero Bayes error rate.
- Let $\mathcal{H}$ denote the finite set of hypotheses or possible sets of parameters to be learned. Each element in the set, $h \in \mathcal{H}$, comprises the necessary set of arguments to fully specify a classifier (e.g., parameters of a Gaussian).
- $P(h)$ is the prior that the algorithm will learn hypothesis h.
- $P(h|\mathcal{D})$ is the probability that the algorithm will yield hypothesis h when trained on the data $\mathcal{D}$.

- Let E be the error for our cost function (zero-one or for some general loss function).

- Let E be the error for our cost function (zero-one or for some general loss function).
- We cannot compute the error directly (i.e., based on distance of h to the unknown target function F).

- Let E be the error for our cost function (zero-one or for some general loss function).
- We cannot compute the error directly (i.e., based on distance of h to the unknown target function F).
- So, what we can do is compute the expected value of the error given our dataset, which will require us to marginalize over all possible targets.

$$\mathcal{E}[E|\mathcal{D}] = \sum_{h,F} \sum_{\mathbf{x} \notin \mathcal{D}} P(\mathbf{x}) \left[1 - \delta(F(\mathbf{x}), h(\mathbf{x}))\right] P(h|\mathcal{D}) P(F|\mathcal{D}) \tag{1}$$

- We can view this as a weighted inner product between the distributions $P(h|\mathcal{D})$ and $P(F|\mathcal{D})$.
- It says that the expected error is related to
  1. all possible inputs and their respective weights $P(\mathbf{x})$;
  2. the "match" between the hypothesis h and the target F.

Cannot Prove Much About Generalization

- The key point here, however, is that **without prior knowledge of the target distribution $P(F|\mathcal{D})$, we can prove little about any paricular learning algorithm $P(h|\mathcal{D})$, including its generalization performance.**

Cannot Prove Much About Generalization

- The key point here, however, is that **without prior knowledge of the target distribution $P(F|\mathcal{D})$, we can prove little about any paricular learning algorithm $P(h|\mathcal{D})$, including its generalization performance.**
- The expected off-training set error when the true function is $F(\mathbf{x})$ and the probability for the kth candidate learning algorithm is $P_k(h(\mathbf{x})|\mathcal{D})$ follows:

$$\mathcal{E}_k[E|F,n] = \sum_{\mathbf{x}\notin\mathcal{D}} P(\mathbf{x})\left[1-\delta(F(\mathbf{x}),h(\mathbf{x}))\right] P_k(h(\mathbf{x})|\mathcal{D}) \qquad (2)$$

No Free Lunch Theorem

For any two learning algorithms $P_1(h|\mathcal{D})$ and $P_2(h|\mathcal{D})$, the following are true, independent of the sampling distribution $P(\mathbf{x})$ and the number n of training points:

1. Uniformly averaged over all target functions F,

$$\mathcal{E}_1[E|F,n] - \mathcal{E}_2[E|F,n] = 0. \tag{3}$$

No Free Lunch Theorem

For any two learning algorithms $P_1(h|\mathcal{D})$ and $P_2(h|\mathcal{D})$, the following are true, independent of the sampling distribution $P(\mathbf{x})$ and the number n of training points:

1. Uniformly averaged over all target functions F,

$$\mathcal{E}_1[E|F,n] - \mathcal{E}_2[E|F,n] = 0. \tag{3}$$

   - No matter how clever we are at choosing a "good" learning algorithm $P_1(h|\mathcal{D})$ and a "bad" algorithm $P_2(h|\mathcal{D})$, if all target functions are equally likely, then the "good" algorithm will not outperform the "bad" one.

No Free Lunch Theorem

For any two learning algorithms $P_1(h|\mathcal{D})$ and $P_2(h|\mathcal{D})$, the following are true, independent of the sampling distribution $P(\mathbf{x})$ and the number n of training points:

1. Uniformly averaged over all target functions F,

$$\mathcal{E}_1[E|F,n] - \mathcal{E}_2[E|F,n] = 0. \tag{3}$$

- No matter how clever we are at choosing a "good" learning algorithm $P_1(h|\mathcal{D})$ and a "bad" algorithm $P_2(h|\mathcal{D})$, if all target functions are equally likely, then the "good" algorithm will not outperform the "bad" one.
- There are no i and j such that for all $F(\mathbf{x})$, $\mathcal{E}_i[E|F,n] > \mathcal{E}_j[E|F,n]$.

No Free Lunch Theorem

For any two learning algorithms $P_1(h|\mathcal{D})$ and $P_2(h|\mathcal{D})$, the following are true, independent of the sampling distribution $P(\mathbf{x})$ and the number n of training points:

1. Uniformly averaged over all target functions F,

$$\mathcal{E}_1[E|F,n] - \mathcal{E}_2[E|F,n] = 0. \tag{3}$$

- No matter how clever we are at choosing a "good" learning algorithm $P_1(h|\mathcal{D})$ and a "bad" algorithm $P_2(h|\mathcal{D})$, if all target functions are equally likely, then the "good" algorithm will not outperform the "bad" one.
- There are no i and j such that for all $F(\mathbf{x})$, $\mathcal{E}_i[E|F,n] > \mathcal{E}_j[E|F,n]$.
- Furthermore, there is at least one target function for which random guessing is a better algorithm!

1. For any fixed training set $\mathcal{D}$, uniformly averaged over F,

$$\mathcal{E}_1[E|F,\mathcal{D}] - \mathcal{E}_2[E|F,\mathcal{D}] = 0. \tag{4}$$

1. For any fixed training set $\mathcal{D}$, uniformly averaged over F,

$$\mathcal{E}_1[E|F,\mathcal{D}] - \mathcal{E}_2[E|F,\mathcal{D}] = 0. \tag{4}$$

2. Uniformly averaged over all priors $P(F)$,

$$\mathcal{E}_1[E|n] - \mathcal{E}_2[E|n] = 0. \tag{5}$$

1. For any fixed training set $\mathcal{D}$, uniformly averaged over F,

$$\mathcal{E}_1[E|F,\mathcal{D}] - \mathcal{E}_2[E|F,\mathcal{D}] = 0. \tag{4}$$

2. Uniformly averaged over all priors $P(F)$,

$$\mathcal{E}_1[E|n] - \mathcal{E}_2[E|n] = 0. \tag{5}$$

3. For any fixed training set $\mathcal{D}$, uniformly averaged over $P(F)$,

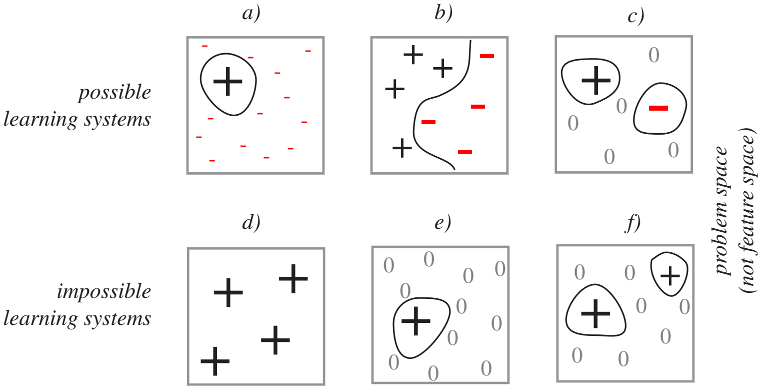
$$\mathcal{E}_1[E|\mathcal{D}] - \mathcal{E}_2[E|\mathcal{D}] = 0. \tag{6}$$

An NFLT Example

	$\mathbf{x}$	F	h_1	h_2
$\mathcal{D}$	000	1	1	1
	001	-1	-1	-1
	010	1	1	1
	011	-1	1	-1
	100	1	1	-1
	101	-1	1	-1
	110	1	1	-1
	111	1	1	-1

- Given 3 binary features.
- Expected off-training set errors are $\mathcal{E}_1 = 0.4$ and $\mathcal{E}_2 = 0.6$.
- The fact that we do not know $F(\mathbf{x})$ beforehand means that all targets are equally likely and we therefore must average over all possible ones.
- For each of the consistent 2^5 distinct target functions, there is exactly one other target function whose output is inverted for each of the patterns outside of the training set. Thus, they're behaviors are inverted and cancel.

Illustration of Part 1: A Conservation Generalization



- There is a conservation idea one can take from this: For every possible learning algorithm for binary classification the sum of performance over all possible target functions is exactly zero.

NFLT Take Home Message

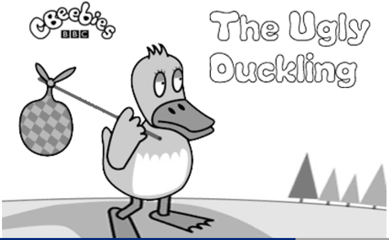
- It is the assumptions about the learning algorithm and the particular pattern recognition scenario that are relevant.

NFLT Take Home Message

- It is the assumptions about the learning algorithm and the particular pattern recognition scenario that are relevant.
- This is a particularly important issue in practical pattern recognition scenarios when even strongly theoretically grounded methods may behave poorly.

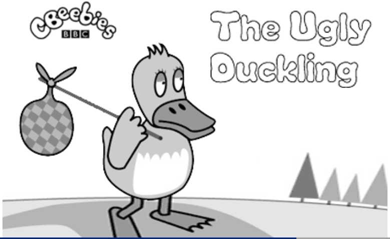
Ugly Duckling Theorem

- Now, we turn our attention to a related question: is there any one feature or pattern representation that will yield better classification performance in the absence of assumptions?
- No! – you guessed it!



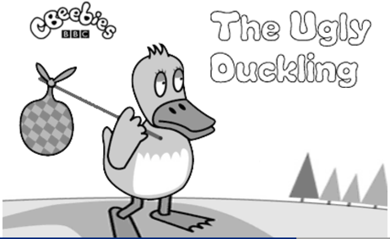
Ugly Duckling Theorem

- Now, we turn our attention to a related question: is there any one feature or pattern representation that will yield better classification performance in the absence of assumptions?
- No! – you guessed it!
- The ugly duckling theorem states that in the absence of assumptions, there is no privileged or "best" feature. Indeed, even the way we compute similarity between patterns depends implicitly on the assumptions.



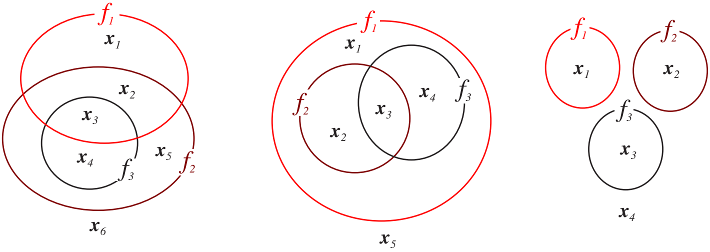
Ugly Duckling Theorem

- Now, we turn our attention to a related question: is there any one feature or pattern representation that will yield better classification performance in the absence of assumptions?
- No! – you guessed it!
- The ugly duckling theorem states that in the absence of assumptions, there is no privileged or "best" feature. Indeed, even the way we compute similarity between patterns depends implicitly on the assumptions.
- And, of course, these assumptions may or may not be correct...



Feature Representation

- We can use logical expressions or “predicates” to describe a pattern (we will get to this later in non-metric methods).
- Denote a binary feature attribute by f_i, then a particular pattern might be described by the predicate $f_1 \wedge f_2$.
- We could also define such predicates on the data itself: $\mathbf{x}_1 \vee \mathbf{x}_2$.
- Define the **rank** of a predicate to be the number of elements it contains.



Impact of Prior Assumptions on Features

- In the absence of prior information, is there a principled reason to judge any two distinct patterns as more or less similar than two other distinct patterns?
- A natural measure of similarity is the number of features shared by the two patterns.

Impact of Prior Assumptions on Features

- In the absence of prior information, is there a principled reason to judge any two distinct patterns as more or less similar than two other distinct patterns?
- A natural measure of similarity is the number of features shared by the two patterns.
- Can you think of any issues with such a similarity measure?

Impact of Prior Assumptions on Features

- In the absence of prior information, is there a principled reason to judge any two distinct patterns as more or less similar than two other distinct patterns?
- A natural measure of similarity is the number of features shared by the two patterns.
- Can you think of any issues with such a similarity measure?
- Suppose we have two features: f_1 represents *blind_in_right_eye*, and f_2 represents *blind_in_left_eye*. Say person $\mathbf{x}_1 = (1, 0)$ is blind in the right eye only and person $\mathbf{x}_2 = (0, 1)$ is blind only in the left eye. $\mathbf{x}_1$ and $\mathbf{x}_2$ are maximally different.

Impact of Prior Assumptions on Features

- In the absence of prior information, is there a principled reason to judge any two distinct patterns as more or less similar than two other distinct patterns?
- A natural measure of similarity is the number of features shared by the two patterns.
- Can you think of any issues with such a similarity measure?
- Suppose we have two features: f_1 represents *blind_in_right_eye*, and f_2 represents *blind_in_left_eye*. Say person $\mathbf{x}_1 = (1, 0)$ is blind in the right eye only and person $\mathbf{x}_2 = (0, 1)$ is blind only in the left eye. $\mathbf{x}_1$ and $\mathbf{x}_2$ are maximally different.
- This is conceptually a problem: person $\mathbf{x}_1$ is more similar to a totally blind person and a normally sighted person than to person $\mathbf{x}_2$.

Multiple Feature Representations

- One can always find multiple ways of representing the same features.
- For example, we might use f'_1 and f'_2 to represent *blind_in_right_eye* and *same_in_both_eyes*, resp. Then we would have the following four types of people (in both cases):

	f_1	f_2		f'_1	f'_2
$\mathbf{x}_1$	0	0		0	1
$\mathbf{x}_2$	0	1		0	0
$\mathbf{x}_3$	1	0		1	0
$\mathbf{x}_4$	1	1		1	1

Back To Comparing Features/Patterns

- The plausible similarity measure is thus the number of predicates the two patterns share, rather than the number of shared features.
- Consider two distinct patterns in some representation $\mathbf{x}_i$ and $\mathbf{x}_j$, $i \neq j$.

Back To Comparing Features/Patterns

- The plausible similarity measure is thus the number of predicates the two patterns share, rather than the number of shared features.
- Consider two distinct patterns in some representation $\mathbf{x}_i$ and $\mathbf{x}_j$, $i \neq j$.
  - There are no predicates of rank 1 that are shared by the two patterns.

Back To Comparing Features/Patterns

- The plausible similarity measure is thus the number of predicates the two patterns share, rather than the number of shared features.
- Consider two distinct patterns in some representation $\mathbf{x}_i$ and $\mathbf{x}_j$, $i \neq j$.
    - There are no predicates of rank 1 that are shared by the two patterns.
    - There is but one predicate of rank 2, $\mathbf{x}_i \vee \mathbf{x}_j$.

Back To Comparing Features/Patterns

- The plausible similarity measure is thus the number of predicates the two patterns share, rather than the number of shared features.
- Consider two distinct patterns in some representation $\mathbf{x}_i$ and $\mathbf{x}_j$, $i \neq j$.
    - There are no predicates of rank 1 that are shared by the two patterns.
    - There is but one predicate of rank 2, $\mathbf{x}_i \vee \mathbf{x}_j$.
    - For predicates of rank 3, two of the patterns must be $\mathbf{x}_i$ and $\mathbf{x}_j$. So, with d patterns in total, there are $\binom{d-2}{1} = d - 2.$ predicates of rank 3.

Back To Comparing Features/Patterns

- The plausible similarity measure is thus the number of predicates the two patterns share, rather than the number of shared features.
- Consider two distinct patterns in some representation $\mathbf{x}_i$ and $\mathbf{x}_j$, $i \neq j$.
  - There are no predicates of rank 1 that are shared by the two patterns.
  - There is but one predicate of rank 2, $\mathbf{x}_i \vee \mathbf{x}_j$.
  - For predicates of rank 3, two of the patterns must be $\mathbf{x}_i$ and $\mathbf{x}_j$. So, with d patterns in total, there are $\binom{d-2}{1} = d - 2.$ predicates of rank 3.
  - For arbitrary rank, the total number of predicates shared by the two patterns is

$$\sum_{r-2}^{d} \binom{d-2}{r-2} = (1+1)^{d-2} = 2^{d-2} \tag{7}$$

Back To Comparing Features/Patterns

- The plausible similarity measure is thus the number of predicates the two patterns share, rather than the number of shared features.
- Consider two distinct patterns in some representation $\mathbf{x}_i$ and $\mathbf{x}_j$, $i \neq j$.
  - There are no predicates of rank 1 that are shared by the two patterns.
  - There is but one predicate of rank 2, $\mathbf{x}_i \vee \mathbf{x}_j$.
  - For predicates of rank 3, two of the patterns must be $\mathbf{x}_i$ and $\mathbf{x}_j$. So, with d patterns in total, there are $\binom{d-2}{1} = d - 2.$ predicates of rank 3.
  - For arbitrary rank, the total number of predicates shared by the two patterns is

$$\sum_{r-2}^{d} \binom{d-2}{r-2} = (1+1)^{d-2} = 2^{d-2} \tag{7}$$

- **Key:** This result is independent of the choice of $\mathbf{x}_i$ and $\mathbf{x}_j$ (as long as they are distinct). Thus, the total number of predicates shared by two distinct patterns is constant and independent of the patterns themselves.

Ugly Duckling Theorem

- Given that we use a finite set of predicates that enables us to distinguish any two patterns under consideration, the number of predicates shared by any two such patterns is constant and independent of the choice of those patterns. Furthermore, if pattern similarity is based on the total number of predicates shared by the two patterns, then any two patterns are “equally similar.”

Ugly Duckling Theorem

- Given that we use a finite set of predicates that enables us to distinguish any two patterns under consideration, the number of predicates shared by any two such patterns is constant and independent of the choice of those patterns. Furthermore, if pattern similarity is based on the total number of predicates shared by the two patterns, then any two patterns are "equally similar."
- I.e., there is no problem-independent or privileged "best" set of features or feature attributes.

Ugly Duckling Theorem

- Given that we use a finite set of predicates that enables us to distinguish any two patterns under consideration, the number of predicates shared by any two such patterns is constant and independent of the choice of those patterns. Furthermore, if pattern similarity is based on the total number of predicates shared by the two patterns, then any two patterns are "equally similar."
- I.e., there is no problem-independent or privileged "best" set of features or feature attributes.
- The theorem forces us to acknowledge that even the apparently simply notion of similarity between patterns is fundamentally based on implicit assumptions about the problem domain.

Bias and Variance Introduction

- Bias and Variance are two measures of how well a learning algorithm matches a classification problem.
- **Bias** measures the accuracy or quality of the match: high bias is a poor match.
- **Variance** measures the precision or specificity of the match: a high variance implies a weak match.
- One can generally adjust the bias and the variance, but **they are not independent**.

Bias and Variance in Terms of Regression

- Suppose there is a true but unknown function $F(\mathbf{x})$ (having continuous valued output with noise).
- We seek to estimate $F(\cdot)$ based on n samples in set $\mathcal{D}$ (assumed to have been generated by the true $F(\mathbf{x})$).
- The regression function estimated is denoted $g(\mathbf{x}; \mathcal{D})$.
- How does this approximation depend on $\mathcal{D}$?

Bias and Variance in Terms of Regression

- Suppose there is a true but unknown function $F(\mathbf{x})$ (having continuous valued output with noise).
- We seek to estimate $F(\cdot)$ based on n samples in set $\mathcal{D}$ (assumed to have been generated by the true $F(\mathbf{x})$).
- The regression function estimated is denoted $g(\mathbf{x};\mathcal{D})$.
- How does this approximation depend on $\mathcal{D}$?
- The natural measure of effectiveness is the mean square error. And, note we need to average over all training sets $\mathcal{D}$ of fixed size n:

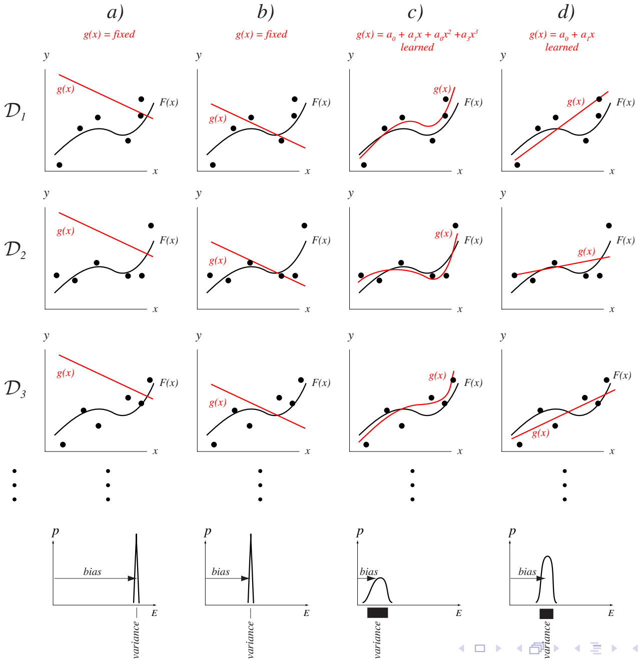
$$\mathcal{E}_{\mathcal{D}}\left[(g(\mathbf{x};\mathcal{D}) - F(\mathbf{x}))^2\right] = \underbrace{\left(\mathcal{E}_{\mathcal{D}}\left[g(\mathbf{x};\mathcal{D}) - F(\mathbf{x})\right]\right)^2}_{\text{bias}^2} + \underbrace{\mathcal{E}_{\mathcal{D}}\left[(g(\mathbf{x};\mathcal{D}) - \mathcal{E}_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})])^2\right]}_{\text{variance}} \qquad (8)$$

- **Bias** is the difference between the expected value and the true (generally unknown) value).
    - A low bias means that on average we will accurately estimate F from $\mathcal{D}$.

- **Bias** is the difference between the expected value and the true (generally unknown) value).
  - A low bias means that on average we will accurately estimate F from $\mathcal{D}$.
- A low variance means that the estimate of F does not change much as the training set varies.

- **Bias** is the difference between the expected value and the true (generally unknown) value).
    - A low bias means that on average we will accurately estimate F from $\mathcal{D}$.
- A low variance means that the estimate of F does not change much as the training set varies.
- Note, that even if an estimator is unbiased, there can nevertheless be a large mean-square error arising from a large variance term.

- **Bias** is the difference between the expected value and the true (generally unknown) value).
    - A low bias means that on average we will accurately estimate F from $\mathcal{D}$.
- A low variance means that the estimate of F does not change much as the training set varies.
- Note, that even if an estimator is unbiased, there can nevertheless be a large mean-square error arising from a large variance term.
- The **bias-variance dilemma** describes the trade-off between the two terms above: procedures with increased flexibility to adapt to the training data tend to have lower bias but higher variance.


a)
b)
c)
d)
g(x) = fixed
g(x) = fixed
g(x) = a_0 + a_1x + a_0x^2 + a_3x^3
learned
g(x) = a_0 + a_1x
learned
$\mathcal{D}_1$
$\mathcal{D}_2$
$\mathcal{D}_3$
y
x
g(x)
F(x)
p
E
bias
variance

Bias-Variance for Classification

How can we map the regression results to classification?

- For a two-category classification problem, we can define the target function as follows

$$F(\mathbf{x}) = P(y = 1|\mathbf{x}) = 1 - P(y = 0|\mathbf{x}) \tag{9}$$

Bias-Variance for Classification

How can we map the regression results to classification?

- For a two-category classification problem, we can define the target function as follows
$$F(\mathbf{x}) = P(y = 1|\mathbf{x}) = 1 - P(y = 0|\mathbf{x}) \tag{9}$$
- To recast classification in the framework of regression, define a discriminant function
$$y = F(\mathbf{x}) + \epsilon \tag{10}$$
where ϵ is a zero-mean r.v. assumed to be binomial with variance $F(\mathbf{x})(1 - F(\mathbf{x}))$.

Bias-Variance for Classification

How can we map the regression results to classification?

- For a two-category classification problem, we can define the target function as follows

$$F(\mathbf{x}) = P(y = 1|\mathbf{x}) = 1 - P(y = 0|\mathbf{x}) \tag{9}$$

- To recast classification in the framework of regression, define a discriminant function

$$y = F(\mathbf{x}) + \epsilon \tag{10}$$

  where ϵ is a zero-mean r.v. assumed to be binomial with variance $F(\mathbf{x})(1 - F(\mathbf{x}))$.

- The target function can thus be expressed as

$$F(\mathbf{x}) = \mathcal{E}[y|\mathbf{x}] \tag{11}$$

- We want to find an estimate $g(\mathbf{x}; \mathcal{D})$ that minimizes the mean-square error:

$$\mathcal{E}_D \left[(g(\mathbf{x}; \mathcal{D}) - y)^2 \right] \tag{12}$$

And thus the regression method ideas can translate here to classification.

- We want to find an estimate $g(\mathbf{x}; \mathcal{D})$ that minimizes the mean-square error:

$$\mathcal{E}_D \left[\left(g(\mathbf{x}; \mathcal{D}) - y \right)^2 \right] \tag{12}$$

  And thus the regression method ideas can translate here to classification.
- Assume equal priors $P(\omega_1) = P(\omega_1) = 1/2$ giving a Bayesian discriminant y_B with a threshold 1/2. The Bayesian decision boundary is the set of points for which $F(\mathbf{x}) = 1/2$.

- We want to find an estimate $g(\mathbf{x}; \mathcal{D})$ that minimizes the mean-square error:

$$\mathcal{E}_D \left[(g(\mathbf{x}; \mathcal{D}) - y)^2 \right] \tag{12}$$

  And thus the regression method ideas can translate here to classification.

- Assume equal priors $P(\omega_1) = P(\omega_1) = 1/2$ giving a Bayesian discriminant y_B with a threshold $1/2$. The Bayesian decision boundary is the set of points for which $F(\mathbf{x}) = 1/2$.
- For a given training set $\mathcal{D}$, we have the lowest error if the classifier error rate agrees with the Bayes error rate:

$$P(g(\mathbf{x}; \mathcal{D}) \neq y) = P(y_B(\mathbf{x}) \neq y) = \min[F(\mathbf{x}), 1 - F(\mathbf{x})] \tag{13}$$

- If not, then we have some increase on the Bayes error

$$P(g(\mathbf{x};\mathcal{D})) = \max[F(\mathbf{x}), 1 - F(\mathbf{x})] \tag{14}$$

$$= |2F(\mathbf{x}) - 1| + P(y_B(\mathbf{x}) = y) \ . \tag{15}$$

And averaging over all datasets of size n yields

$$P(g(\mathbf{x};\mathcal{D}) \neq y) = |2F(\mathbf{x}) - 1| P(g(\mathbf{x};\mathcal{D}) \neq y_B) + P(y_B \neq y) \tag{16}$$

- If not, then we have some increase on the Bayes error

$$P(g(\mathbf{x};\mathcal{D})) = \max[F(\mathbf{x}), 1 - F(\mathbf{x})] \tag{14}$$

$$= |2F(\mathbf{x}) - 1| + P(y_B(\mathbf{x}) = y) \ . \tag{15}$$

And averaging over all datasets of size n yields

$$P(g(\mathbf{x};\mathcal{D}) \neq y) = |2F(\mathbf{x}) - 1| P(g(\mathbf{x};\mathcal{D}) \neq y_B) + P(y_B \neq y) \tag{16}$$

- Hence, the classification error rate is linearly proportional to the **boundary error** $P(g(\mathbf{x};\mathcal{D}) \neq y_b)$, the incorrect estimation of the Bayes boundary, which is the "opposite" tail as we saw earlier in lecture 2.

$$P(g(\mathbf{x};\mathcal{D}) \neq y_B) = \begin{cases} \int_{1/2}^{\infty} p(g(\mathbf{x};\mathcal{D}))dg & \text{if } F(\mathbf{x}) < 1/2 \\ \int_{-\infty}^{1/2} p(g(\mathbf{x};\mathcal{D}))dg & \text{if } F(\mathbf{x}) \geq 1/2 \end{cases} \tag{17}$$

Bias-Variance Classification Boundary Error for Gaussian Case

- If we assume $p(g(\mathbf{x}; \mathcal{D}))$ is a Gaussian, we find

$$P(g(\mathbf{x}; \mathcal{D}) \neq y_B) =$$

$$\Phi \left[\underbrace{\mathrm{Sgn}[F(\mathbf{x}) - 1/2][\mathcal{E}_{\mathcal{D}}[g(\mathbf{x}; \mathcal{D})] - 1/2]}_{\text{boundary bias}} \underbrace{\mathrm{Var}[g(\mathbf{x}; \mathcal{D})]^{-1/2}}_{\text{variance}} \right] \tag{18}$$

where $\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_t^\infty \exp\left[-1/2u^2\right] du$.

Bias-Variance Classification Boundary Error for Gaussian Case

- If we assume $p(g(\mathbf{x};\mathcal{D}))$ is a Gaussian, we find

$$P(g(\mathbf{x};\mathcal{D}) \neq y_B) = \Phi\left[\underbrace{\text{Sgn}[F(\mathbf{x})-1/2][\mathcal{E}_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})]-1/2]}_{\text{boundary bias}}\underbrace{\text{Var}[g(\mathbf{x};\mathcal{D})]^{-1/2}}_{\text{variance}}\right] \quad (18)$$

where $\Phi(t) = \frac{1}{\sqrt{2\pi}}\int_t^\infty \exp\left[-1/2u^2\right]du$.

- You can visualize the boundary bias by imagining taking the spatial average of decision boundaries obtained by running the learning algorithm on all possible data sets.

Bias-Variance Classification Boundary Error for Gaussian Case

- If we assume $p(g(\mathbf{x};\mathcal{D}))$ is a Gaussian, we find

$$P(g(\mathbf{x};\mathcal{D}) \neq y_B) = \Phi\left[\underbrace{\mathrm{Sgn}[F(\mathbf{x})-1/2][\mathcal{E}_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})]-1/2]}_{\text{boundary bias}}\underbrace{\mathrm{Var}[g(\mathbf{x};\mathcal{D})]^{-1/2}}_{\text{variance}}\right] \quad (18)$$

  where $\Phi(t) = \frac{1}{\sqrt{2\pi}}\int_t^\infty \exp\left[-1/2u^2\right] du$.

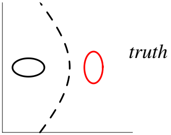
- You can visualize the boundary bias by imagining taking the spatial average of decision boundaries obtained by running the learning algorithm on all possible data sets.
- Hence, the effect of the variance term on the boundary error is highly nonlinear and depends on the value of the boundary bias.

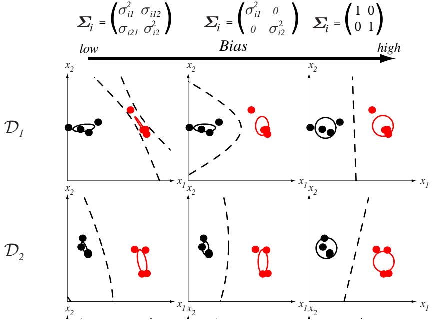
- With small variance, the sign of the boundary bias is increasingly a player.

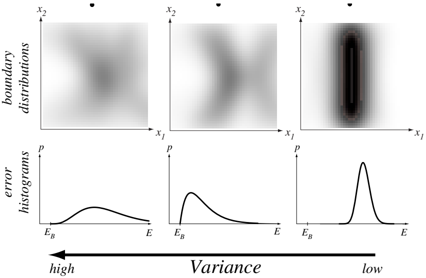
- With small variance, the sign of the boundary bias is increasingly a player.
- Whereas in regression the estimation error is additive in bias and variance, in classification there is a **non-linear** and **multiplicative** interaction.

- With small variance, the sign of the boundary bias is increasingly a player.
- Whereas in regression the estimation error is additive in bias and variance, in classification there is a **non-linear** and **multiplicative** interaction.
- In classification, the sign of the bounary bias affects the role of the variance in the error. So, low variance is generally important for accurate classification, while low boundary bias need not be.

- With small variance, the sign of the boundary bias is increasingly a player.
- Whereas in regression the estimation error is additive in bias and variance, in classification there is a **non-linear** and **multiplicative** interaction.
- In classification, the sign of the bounary bias affects the role of the variance in the error. So, low variance is generally important for accurate classification, while low boundary bias need not be.
- Similarly, in classification, **variance dominates bias**.


truth


$\Sigma_i = \begin{pmatrix} \sigma_{i1}^2 & \sigma_{i12} \\ \sigma_{i21} & \sigma_{i2}^2 \end{pmatrix}$
$\Sigma_i = \begin{pmatrix} \sigma_{i1}^2 & 0 \\ 0 & \sigma_{i2}^2 \end{pmatrix}$
$\Sigma_i = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
low
Bias
high
$\mathcal{D}_1$
$\mathcal{D}_2$
x_1
x_2


boundary distributions
error histograms
x_1
x_2
p
E_B
E
high
Variance
low

How Can We Determine the Bias and Variance?

- The results from the discussion in bias and variance suggest a way to estimate these two values (and hence a way to quantify the match between a method and a problem).
- What is it?

How Can We Determine the Bias and Variance?

- The results from the discussion in bias and variance suggest a way to estimate these two values (and hence a way to quantify the match between a method and a problem).
- What is it?
- **Resampling**. I.e., taking multiple datasets, perform the estimation and evaluate the boundary distributions and error histograms.

How Can We Determine the Bias and Variance?

- The results from the discussion in bias and variance suggest a way to estimate these two values (and hence a way to quantify the match between a method and a problem).
- What is it?
- **Resampling**. I.e., taking multiple datasets, perform the estimation and evaluate the boundary distributions and error histograms.
- Let's make these ideas clear in presenting two methods for resampling:
  1. Jackknife
  2. Bootstrap

Jackknife

- Let’s first attempt to demonstrate how resampling can be used to yield a more informative estimate of a general statistic.

Jackknife

- Let's first attempt to demonstrate how resampling can be used to yield a more informative estimate of a general statistic.
- Suppose we have a set $\mathcal{D}$ of n points x_i sampled from a one-dimensional distribution.

Jackknife

- Let's first attempt to demonstrate how resampling can be used to yield a more informative estimate of a general statistic.
- Suppose we have a set $\mathcal{D}$ of n points x_i sampled from a one-dimensional distribution.
- The familiar estimate of the mean is

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{19}$$

Jackknife

- Let's first attempt to demonstrate how resampling can be used to yield a more informative estimate of a general statistic.
- Suppose we have a set $\mathcal{D}$ of n points x_i sampled from a one-dimensional distribution.
- The familiar estimate of the mean is

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{19}$$

- And, the estimate of the accuracy of the mean is the sample variance, given by

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \hat{\mu})^2 \tag{20}$$

- Now, suppose we were instead interested in the median (the point for which half the distribution is higher, half lower).

- Now, suppose we were instead interested in the median (the point for which half the distribution is higher, half lower).
- Of course, we can determine the median explicitly.

- Now, suppose we were instead interested in the median (the point for which half the distribution is higher, half lower).
- Of course, we can determine the median explicitly.
- But, how can we compute the accuracy of the median, or a measure of the error or spread of it? This problem would translate to many other statistics that we could compute, such as the mode.

- Now, suppose we were instead interested in the median (the point for which half the distribution is higher, half lower).
- Of course, we can determine the median explicitly.
- But, how can we compute the accuracy of the median, or a measure of the error or spread of it? This problem would translate to many other statistics that we could compute, such as the mode.
- Resampling will help us here.

- Now, suppose we were instead interested in the median (the point for which half the distribution is higher, half lower).
- Of course, we can determine the median explicitly.
- But, how can we compute the accuracy of the median, or a measure of the error or spread of it? This problem would translate to many other statistics that we could compute, such as the mode.
- Resampling will help us here.
- First, define some new notation for **leave-one-out** statistic in which the statistic, such as the mean, is computed by leaving a particular datum out of the estimate. For the mean, we have

$$\mu_{(i)} = \frac{1}{n-1} \sum_{j \neq i} x_j = \frac{n\hat{\mu} - x_i}{n-1} \quad . \tag{21}$$

- Now, suppose we were instead interested in the median (the point for which half the distribution is higher, half lower).
- Of course, we can determine the median explicitly.
- But, how can we compute the accuracy of the median, or a measure of the error or spread of it? This problem would translate to many other statistics that we could compute, such as the mode.
- Resampling will help us here.
- First, define some new notation for **leave-one-out** statistic in which the statistic, such as the mean, is computed by leaving a particular datum out of the estimate. For the mean, we have

$$\mu_{(i)} = \frac{1}{n-1}\sum_{j\neq i} x_j = \frac{n\hat{\mu} - x_i}{n-1} \quad . \tag{21}$$

- Next, we can define the **jackknife** estimate of the mean, that is, the mean of the leave-one-out means:

$$\mu_{(\cdot)} = \frac{1}{n}\sum_{i=1}^{n} \mu_{(i)} \tag{22}$$

- And, we can show that the jackknife mean is the same of the traditional mean.

- And, we can show that the jackknife mean is the same of the traditional mean.
- The variance of the jacknife estimate is given by

$$\text{Var}[\hat{\mu}] = \frac{n-1}{n} \sum_{i=1}^{n} (\mu_{(i)} - \mu_{(\cdot)})^2 \tag{23}$$

which is equivalent to the traditional variance (for the mean case).

- And, we can show that the jackknife mean is the same of the traditional mean.
- The variance of the jacknife estimate is given by

$$\text{Var}[\hat{\mu}] = \frac{n-1}{n}\sum_{i=1}^{n}(\mu_{(i)} - \mu_{(\cdot)})^2 \tag{23}$$

  which is equivalent to the traditional variance (for the mean case).
- The benefit of expressing the variance of a jackknife estimate in this way is that it generalizes to any estimator $\hat{\theta}$, such as the median. To do so, we would need to similarly compute the set of n leave-one-out statistics, $\theta_{(i)}$.

$$\text{Var}[\hat{\theta}] = \mathcal{E}\left[[\hat{\theta}(x_1, x_2, \ldots, x_n) - \mathcal{E}[\hat{\theta}]]^2\right] \tag{24}$$

$$\text{Var}_{\text{jack}}[\hat{\theta}] = \frac{n-1}{n}\sum_{i=1}^{n}(\theta_{(i)} - \theta_{(\cdot)})^2 \tag{25}$$

Jackknife Bias Estimate

- The general bias of an estimator θ is the difference between its true value and its expected value:

$$\text{bias}[\hat{\theta}] = \theta - \mathcal{E}[\hat{\theta}] \tag{26}$$

Jackknife Bias Estimate

- The general bias of an estimator θ is the difference between its true value and its expected value:

$$\text{bias}[\hat{\theta}] = \theta - \mathcal{E}[\hat{\theta}] \tag{26}$$

- And, we can use the jackknife method to estimate such a bias.

$$\text{bias}_{\text{jack}}[\hat{\theta}] = (n-1)\left(\hat{\theta}_{(\cdot)} - \hat{\theta}\right) \tag{27}$$

Jackknife Bias Estimate

- The general bias of an estimator θ is the difference between its true value and its expected value:

$$\mathsf{bias}[\hat{\theta}] = \theta - \mathcal{E}[\hat{\theta}] \tag{26}$$

- And, we can use the jackknife method to estimate such a bias.

$$\mathsf{bias}_{\mathsf{jack}}[\hat{\theta}] = (n-1)\left(\hat{\theta}_{(\cdot)} - \hat{\theta}\right) \tag{27}$$

- We can rearrange the terms to see that the jackknife estimate of $\hat{\theta}$ is

$$\tilde{\theta} = \hat{\theta} - \mathsf{bias}_{\mathsf{jack}} = n\hat{\theta} - (n-1)\hat{\theta}_{(\cdot)} \tag{28}$$

Jackknife Bias Estimate

- The general bias of an estimator θ is the difference between its true value and its expected value:

$$\text{bias}[\hat{\theta}] = \theta - \mathcal{E}[\hat{\theta}] \tag{26}$$

- And, we can use the jackknife method to estimate such a bias.

$$\text{bias}_{\text{jack}}[\hat{\theta}] = (n-1)\left(\hat{\theta}_{(\cdot)} - \hat{\theta}\right) \tag{27}$$

- We can rearrange the terms to see that the jackknife estimate of $\hat{\theta}$ is

$$\tilde{\theta} = \hat{\theta} - \text{bias}_{\text{jack}} = n\hat{\theta} - (n-1)\hat{\theta}_{(\cdot)} \tag{28}$$

- The jacknife resampling technique often gives a more satisfactory estimate of a statistic (because we can measure accuracy) than do traditional methods, but it is more computationally expensive.

Jackknife Estimation of Classification Accuracy

- We are ultimately interested in computing the accuracy of a classifier.

Jackknife Estimation of Classification Accuracy

- We are ultimately interested in computing the accuracy of a classifier.
- We can again apply the Jackknife method, at, albeit, a very high computational cost.

Jackknife Estimation of Classification Accuracy

- We are ultimately interested in computing the accuracy of a classifier.
- We can again apply the Jackknife method, at, albeit, a very high computational cost.
- Train the classifier n times, each time using a training set $\mathcal{D}$ from which a single training point has been deleted.

Jackknife Estimation of Classification Accuracy

- We are ultimately interested in computing the accuracy of a classifier.
- We can again apply the Jackknife method, at, albeit, a very high computational cost.
- Train the classifier n times, each time using a training set $\mathcal{D}$ from which a single training point has been deleted.
- Then, each resulting classifier is tested on the single deleted point, and the jackknife accuracy is the mean of the leave-one-out accuracies.

Jackknife Estimation of Classification Accuracy

- We are ultimately interested in computing the accuracy of a classifier.
- We can again apply the Jackknife method, at, albeit, a very high computational cost.
- Train the classifier n times, each time using a training set $\mathcal{D}$ from which a single training point has been deleted.
- Then, each resulting classifier is tested on the single deleted point, and the jackknife accuracy is the mean of the leave-one-out accuracies.
- Note the high computational complexity.

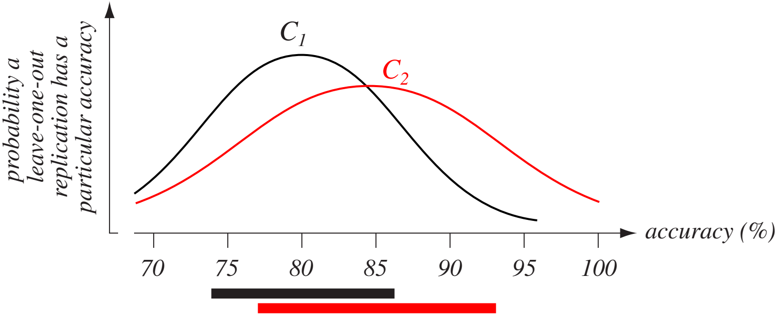
Jackknife Estimation of Classification Accuracy

- We are ultimately interested in computing the accuracy of a classifier.
- We can again apply the Jackknife method, at, albeit, a very high computational cost.
- Train the classifier n times, each time using a training set $\mathcal{D}$ from which a single training point has been deleted.
- Then, each resulting classifier is tested on the single deleted point, and the jackknife accuracy is the mean of the leave-one-out accuracies.
- Note the high computational complexity.
- One can apply this method to estimate the statistical significance in the comparison of two classifier designs.

Jackknife Estimation of Classification Accuracy

- We are ultimately interested in computing the accuracy of a classifier.
- We can again apply the Jackknife method, at, albeit, a very high computational cost.
- Train the classifier n times, each time using a training set $\mathcal{D}$ from which a single training point has been deleted.
- Then, each resulting classifier is tested on the single deleted point, and the jackknife accuracy is the mean of the leave-one-out accuracies.
- Note the high computational complexity.
- One can apply this method to estimate the statistical significance in the comparison of two classifier designs.
- Suppose we have two trained classifiers C_1 with an accuracy of 80% and C_2 with an accuracy of 85% (both as estimated with the jackknife procedure). Is C_2 really better than C_1?

- To answer this, calculate the jackknife estimate of the variance and use traditional hypothesis testing to test statistical significance.



Bootstrap

- A **bootstrap** dataset is one created by randomly selecting n points from the training set $\mathcal{D}$ with replacement.
    - Because $\mathcal{D}$ contains itself n points, there is nearly always duplication of individual points in a bootstrap dataset.

Bootstrap

- A **bootstrap** dataset is one created by randomly selecting n points from the training set $\mathcal{D}$ with replacement.
    - Because $\mathcal{D}$ contains itself n points, there is nearly always duplication of individual points in a bootstrap dataset.
- We repeat this process B times to yield B bootstrap datasets, which are treated as independent sets (although they are clearly not).

Bootstrap

- A **bootstrap** dataset is one created by randomly selecting n points from the training set $\mathcal{D}$ with replacement.
    - Because $\mathcal{D}$ contains itself n points, there is nearly always duplication of individual points in a bootstrap dataset.
- We repeat this process B times to yield B bootstrap datasets, which are treated as independent sets (although they are clearly not).
- The bootstrap estimate of a statistic θ is denoted $\hat{\theta}^{*(\cdot)}$, and, again, it is the mean of the B estimate on the individual bootstrap datasets.

$$\hat{\theta}^{*(\cdot)} = \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}^{*(b)} \tag{29}$$

where $\hat{\theta}^{*(b)}$ is the estimate on bootstrap dataset b.

Bootstrap Bias Estimate

- The bootstrap estimate of the bias is

$$\text{bias}_{\text{boot}} = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{*(b)} - \hat{\theta} = \hat{\theta}^{*(\cdot)} - \hat{\theta} \tag{30}$$

- To increase robustness to outliers, the bootstrap method is useful for computing a **trimmed** statistic, such as the trimmed mean, in which the statistic is computed with some portion of the highest and lowest being deleted.

Bootstrap Variance Estimate

- The boostrap estimate of the variance is

$$\text{Var}_{\text{boot}}[\theta] = \frac{1}{B}\sum_{b=1}^{B}\left[\hat{\theta}^{*(b)} - \hat{\theta}^{*(\cdot)}\right]^2 \tag{31}$$

- If the statistic θ is the mean, then in the limit of $B \to \infty$, the bootstrap estimate of the variance is the traditional variance of the mean.

Bootstrap Variance Estimate

- The boostrap estimate of the variance is

$$\mathrm{Var}_{\mathrm{boot}}[\theta] = \frac{1}{B} \sum_{b=1}^{B} \left[\hat{\theta}^{*(b)} - \hat{\theta}^{*(\cdot)} \right]^2 \tag{31}$$

- If the statistic θ is the mean, then in the limit of $B \to \infty$, the bootstrap estimate of the variance is the traditional variance of the mean.
- In general, the higher B the better the estimates of the statistic and its variance.

Bootstrap Variance Estimate

- The boostrap estimate of the variance is

$$\mathrm{Var}_{\mathrm{boot}}[\theta] = \frac{1}{B}\sum_{b=1}^{B}\left[\hat{\theta}^{*(b)} - \hat{\theta}^{*(\cdot)}\right]^2 \tag{31}$$

- If the statistic θ is the mean, then in the limit of $B \to \infty$, the bootstrap estimate of the variance is the traditional variance of the mean.
- In general, the higher B the better the estimates of the statistic and its variance.
- A benefit of boostrap (say over jackknife) is that one can adjust B based on the available resources...

Bootstrap Estimate of Classification Accuracy

- The bootstrap method can also be applied to estimate the classification accuracy.

Bootstrap Estimate of Classification Accuracy

- The bootstrap method can also be applied to estimate the classification accuracy.
- Train B classifiers, each with a different bootstrap dataset and test on the other bootstrap datasets.

Bootstrap Estimate of Classification Accuracy

- The bootstrap method can also be applied to estimate the classification accuracy.
- Train B classifiers, each with a different bootstrap dataset and test on the other bootstrap datasets.
- The bootstrap estimate of the accuracy is then simply the mean of these bootstrap accuracies.

Bootstrap Estimate of Classification Accuracy

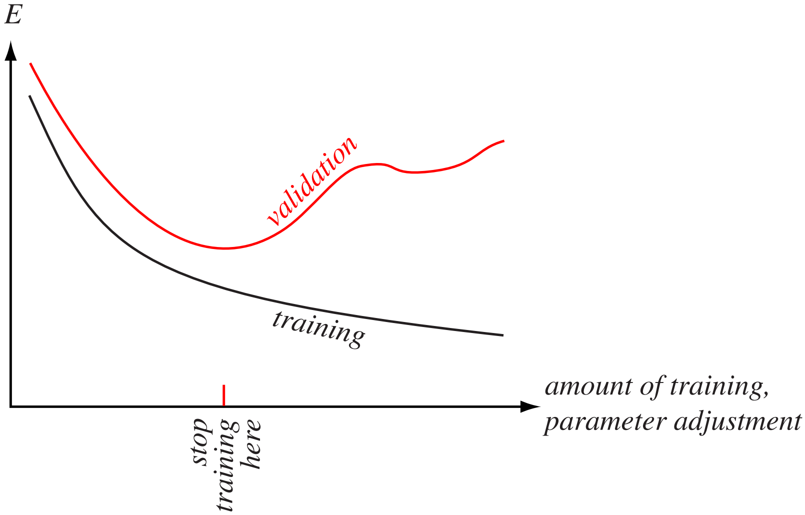
- The bootstrap method can also be applied to estimate the classification accuracy.
- Train B classifiers, each with a different bootstrap dataset and test on the other bootstrap datasets.
- The bootstrap estimate of the accuracy is then simply the mean of these bootstrap accuracies.
- But, the high computational complexity of bootstrapping rarely makes this a worthwhile practice.

Cross-Validation

- The jackknife and bootstrap estimates of classification accuracy are closely related (or special cases of in some instances) of the method of **cross-validation**.

Cross-Validation

- The jackknife and bootstrap estimates of classification accuracy are closely related (or special cases of in some instances) of the method of **cross-validation**.
- **Simple Validation**
  - Split the initial dataset into two parts a training part and a **validation** part.
  - Train the classifier on the training part of the dataset.
  - But, to estimate generalization accuracy, test it on the validation part during training and halt training when we reach a minimum of this validation error.
  - It is imperative that the validation set not have points also in the training set.


E
validation
training
amount of training,
parameter adjustment
stop
training
here

- A generalization of this method is **m-fold cross-validation**.

- A generalization of this method is **m-fold cross-validation**.
- Divide the training set into m disjoint sets of equal size n/m where n is the size of the initial set.

- A generalization of this method is **m-fold cross-validation**.
- Divide the training set into m disjoint sets of equal size n/m where n is the size of the initial set.
- Then, train the classifier m times, each time with a different set held out as a validation set.

- A generalization of this method is **m-fold cross-validation**.
- Divide the training set into m disjoint sets of equal size n/m where n is the size of the initial set.
- Then, train the classifier m times, each time with a different set held out as a validation set.
- The estimated performance is the mean of these m errors.

- A generalization of this method is **m-fold cross-validation**.
- Divide the training set into m disjoint sets of equal size n/m where n is the size of the initial set.
- Then, train the classifier m times, each time with a different set held out as a validation set.
- The estimated performance is the mean of these m errors.
- In what case is this a jackknife estimate of the accuracy?

- A generalization of this method is **m-fold cross-validation**.
- Divide the training set into m disjoint sets of equal size n/m where n is the size of the initial set.
- Then, train the classifier m times, each time with a different set held out as a validation set.
- The estimated performance is the mean of these m errors.
- In what case is this a jackknife estimate of the accuracy?
- When $m = n$.

Resampling for Classifier Design

- We just covered resampling for estimating statistics (even classification accuracies).
- Now, we want to see how resampling can help to directly improve upone classifier design.
- **Arcing**, adaptive reweighting and combing, refers to reusing or selecting data in order to improve classification.
- We will discuss two such methods
- Bagging
- Boosting (in some detail)

- Bagging gets its name from bootstrap aggregation.

- Bagging gets its name from bootstrap aggregation.
- We create B bootstrap datasets by drawing $n' < n$ samples from $\mathcal{D}$ with replacement to create each bootstrap dataset.

- Bagging gets its name from bootstrap aggregation.
- We create B bootstrap datasets by drawing $n' < n$ samples from $\mathcal{D}$ with replacement to create each bootstrap dataset.
- Each of these datasets is used to learn a different component classifier.

- Bagging gets its name from bootstrap aggregation.
- We create B bootstrap datasets by drawing $n' < n$ samples from $\mathcal{D}$ with replacement to create each bootstrap dataset.
- Each of these datasets is used to learn a different component classifier.
- The final classification is based on the vote of each component classifier.

- Bagging gets its name from bootstrap aggregation.
- We create B bootstrap datasets by drawing $n' < n$ samples from $\mathcal{D}$ with replacement to create each bootstrap dataset.
- Each of these datasets is used to learn a different component classifier.
- The final classification is based on the vote of each component classifier.
- The component classifier are typically of the same form, but they need not be.

- Bagging gets its name from bootstrap aggregation.
- We create B bootstrap datasets by drawing $n' < n$ samples from $\mathcal{D}$ with replacement to create each bootstrap dataset.
- Each of these datasets is used to learn a different component classifier.
- The final classification is based on the vote of each component classifier.
- The component classifier are typically of the same form, but they need not be.
- Bagging seems to improve recognition for unstable classifiers, but this is not theoretically grounded.
  - An unstable classifier is one for which small changes in the training data lead to significantly different classifiers and relatively large changes in accuracy.

- Bagging gets its name from bootstrap aggregation.
- We create B bootstrap datasets by drawing $n' < n$ samples from $\mathcal{D}$ with replacement to create each bootstrap dataset.
- Each of these datasets is used to learn a different component classifier.
- The final classification is based on the vote of each component classifier.
- The component classifier are typically of the same form, but they need not be.
- Bagging seems to improve recognition for unstable classifiers, but this is not theoretically grounded.
  - An unstable classifier is one for which small changes in the training data lead to significantly different classifiers and relatively large changes in accuracy.
- For regression, the bagged estimate is simply the average. For classification, one can count the most popular classification, or combine the classifiers in some other more sophisticated way (perhaps with yet another classifier).

-