Admin

• **Auditing/registration forms:**
  – Submit them at end of class, pick them up next class.
    • I may have written notes on your form (especially when I didn’t sign).

• **Website/Piazza:**
  – [https://piazza.com/ubc.ca/winterterm22018/cpsc540](https://piazza.com/ubc.ca/winterterm22018/cpsc540)

• **Tutorials:** start Monday after class (no need to formally register).

• **Office hours:** start Wednesday after class.

• **Assignment 1** due Friday.
  – Sign up for CS account so you can handin.
Supervised Learning Notation

- We are given **training data** where we know labels:

  \[
  \begin{array}{cccccc}
  \text{Egg} & \text{Milk} & \text{Fish} & \text{Wheat} & \text{Shellfish} & \text{Peanuts} \\
  0 & 0.7 & 0 & 0.3 & 0 & 0 \\
  0.3 & 0.7 & 0 & 0.6 & 0 & 0.01 \\
  0 & 0 & 0 & 0.8 & 0 & 0 \\
  0.3 & 0.7 & 1.2 & 0 & 0.10 & 0.01 \\
  0.3 & 0 & 1.2 & 0.3 & 0.10 & 0.01 \\
  \end{array}
  \]

  \[X = \begin{pmatrix}
  \text{Egg} \\
  \text{Milk} \\
  \text{Fish} \\
  \text{Wheat} \\
  \text{Shellfish} \\
  \text{Peanuts} \\
  \end{pmatrix}, \quad y = \begin{pmatrix}
  \text{Sick?} \\
  1 \\
  1 \\
  0 \\
  1 \\
  1 \\
  \end{pmatrix}\]

- But the goal is to do well on **any possible testing data**:

  \[
  \begin{array}{cccccc}
  \text{Egg} & \text{Milk} & \text{Fish} & \text{Wheat} & \text{Shellfish} & \text{Peanuts} \\
  0.5 & 0 & 1 & 0.6 & 2 & 1 \\
  0 & 0.7 & 0 & 1 & 0 & 0 \\
  3 & 1 & 0 & 0.5 & 0 & 0 \\
  \end{array}
  \]

  \[\tilde{X} = \begin{pmatrix}
  \text{Egg} \\
  \text{Milk} \\
  \text{Fish} \\
  \text{Wheat} \\
  \text{Shellfish} \\
  \text{Peanuts} \\
  \end{pmatrix}, \quad \tilde{y} = \begin{pmatrix}
  \text{Sick?} \\
  ? \\
  ? \\
  \end{pmatrix}\]
“Test Set” vs. “Test Error”

• Formally, the “test error” is the expected error of our model:
  \[ \mathbb{E} \left[ |\hat{y}_i - \tilde{y}_i| \right] \]
  – Here I’m using absolute error between predictions and true labels.
    • But you could use squared error or other losses.
  – The expectation is taken over distribution of test examples.
    • Think of this as the “error with infinite data”.
  – We assume that our training examples are drawn IID from this distribution.
    • Otherwise, “training” might not help to reduce “test error”.

• Unfortunately, we cannot compute the test error.
  – We don’t have access to the distribution over all test examples.
“Test Set” vs. “Test Error”

• We often approximate “test error” with the error on a “test set”:
  \[
  \frac{1}{t} \sum_{i=1}^{t} |\hat{y}_i - \tilde{y}_i |
  \]
  – Here, we are using ‘t’ examples drawn IID from the test distribution.

• Note that “test set error” is not the “test error”.
  – The goal is have a low “test error”, not “test set error”.

• The “golden rule” of machine learning:
  – A “test set” cannot influence the “training” in any way.
  – Otherwise, “test set error” is not an unbiased “test error” approximation.
  – We run the risk of “overfitting” to the “test set”.
Typical Supervised Learning Steps (Are Bad?)

• Given data \{X,y\}, a typical set of supervised learning steps:
  – Data splitting:
    • Split \{X,y\} into a train set \{X_{train},y_{train}\} and a validation set \{X_{valid},y_{valid}\}.
    • We’re going to use the validation set error as approximation of test error.
  – Tune hyper-parameters (number of hidden units, \(\lambda\), polynomial degree, etc.):
    • For each candidate value “\(\lambda\)” of the hyper-parameters:
      – Fit a model to the train set \{X_{train},y_{train}\} using the given hyper-parameters “\(\lambda\)”.
      – Evaluate the model on the validation set \{X_{valid},y_{valid}\}.
    – Choose the model with the best performance on the validation set.
      • And maybe re-train using hyper-parameter “\(\lambda\)” on the full dataset.

• Can this overfit, even though we used a validation set?
  – Yes, we’ve violated the golden rule. But maybe it’s not too bad...
Validation Error, Test Error, and Approximation Error

• 340 discusses the “Fundamental Trade-Off of Machine Learning”.
  – Simple identity relating training set error to test error.

• We have a similar identity for the validation error.
  – If $E_{\text{test}}$ is the test error and $E_{\text{valid}}$ is the error on the validation set, then:

\[
E_{\text{test}} = (E_{\text{test}} - E_{\text{valid}}) + E_{\text{valid}}
\]

• If $E_{\text{approx}}$ is small, then $E_{\text{valid}}$ is a good approximation of $E_{\text{test}}$.
  – We can’t measure $E_{\text{test}}$, so how do we know if $E_{\text{approx}}$ is small?
Bounding $E_{\text{approx}}$

• Let’s consider a simple case:
  – Labels $y_i$ are binary, and we try 1 hyper-parameter setting.
  – IID assumption on validation set implies $E_{\text{valid}}$ is unbiased: $E[E_{\text{valid}}] = E_{\text{test}}$.

• We can bound probability $E_{\text{approx}}$ is greater than $\varepsilon$.
  – Assumptions: data is IID (so $E_{\text{valid}}$ is unbiased) and loss is in $[0,1]$.
  – By using Hoeffding’s inequality:
    $$\Pr\left(\left|\frac{1}{E_{\text{approx}}} E_{\text{test}} - E_{\text{valid}}\right| > \varepsilon\right) \leq 2 \exp\left(-\frac{2 \varepsilon^2 \cdot t}{E_{\text{approx}}}\right)$$
    – Probability that $E_{\text{valid}}$ is far from $E_{\text{test}}$ goes down exponentially with ‘$t$’.

• This is great: the bigger your validation set, the better approximation you get.
Bounding $E_{\text{approx}}$

• Let’s consider a slightly less-simple case:
  – Labels are binary, and we tried ‘k’ hyper-parameter values.
  – In this case it’s unbiased for each ‘k’: $E[E_{\text{valid}(\lambda)}] = E_{\text{test}}$.
  – So for each validation error $E_{\text{valid}(\lambda)}$ we have:

\[
p( | E_{\text{test}} - E_{\text{valid}(\lambda)} | > \varepsilon ) \leq 2 \exp (-2 \varepsilon^2 t)
\]

  – But our final $E_{\text{valid}} = \min\{E_{\text{valid}(\lambda)}\}$, which is biased.
    • We can’t apply Hoeffding because we chose best among ‘k’ values.

• Fix: bound on probability that all $|E_{\text{test}} - E_{\text{valid}(\lambda)}|$ values are $\leq \varepsilon$.
  – Since we showed it holds for all values, it holds for the best value.
Bounding $E_{\text{approx}}$

• The “union bound” for any events $\{A_1, A_2, ..., A_k\}$ is that:

$$P(A_1 \cup A_2 \cup ... \cup A_k) \leq \sum_{i=1}^{k} P(A_i)$$

• Combining with Hoeffding we can get:

$$P\left( |E_{\text{test}} - \min_{a} [E_{\text{valid}}(a)]^3| > \varepsilon \right) \leq \sum_a P\left( |E_{\text{test}} - E_{\text{valid}(a)}| > \varepsilon \right)$$

$$\leq \sum_a P\left( |E_{\text{test}} - E_{\text{valid}(a)}| > \varepsilon \right)$$

$$\leq \sum_a 2 \exp(-2\varepsilon^2 t)$$

$$= k \, 2 \exp(-2\varepsilon^2 t)$$
Bounding $E_{\text{approx}}$

- So if we choose best $E_{\text{valid}}(\lambda)$ among ‘k’ $\lambda$ values, we have:

$$p\left(\left|E_{\text{test}},t - E_{\text{valid}}(\lambda)\right| > \varepsilon \text{ for any } \lambda\right) \leq k \cdot 2 \cdot \exp\left(-2 \varepsilon^2 t\right)$$

- So optimizing over ‘k’ models is ok if we have large ‘t’.
  - But if ‘k’ is too large or ‘t’ is too small the validation error isn’t useful.

- Examples:
  - If $k=10$ and $t=1000$, probability that $|E_{\text{approx}}| > .05$ is less than 0.14.
  - If $k=10$ and $t=10000$, probability that $|E_{\text{approx}}| > .05$ is less than $10^{-20}$.
  - If $k=10$ and $t=1000$, probability that $|E_{\text{approx}}| > .01$ is less than 2.7 (useless).
  - If $k=100$ and $t=100000$, probability that $|E_{\text{approx}}| > .01$ is less than $10^{-6}$.
Bounding $E_{\text{approx}}$

- Validation error vs. test error for fixed ‘t’.
  - $E_{\text{valid}}$ goes down as we increase ‘k’, but $E_{\text{approx}}$ can go up.
  - Overfitting of validation set.
Discussion

• Bound is usually very loose, but data is probably not fully IID.
  – Similar bounds are possible for cross-validation.

• Similar arguments apply for the $E_{\text{approx}}$ of the training error.
  – Value ‘$k$’ is the number of hyper-parameters you are optimizing over (even if don’t try them all).
  – So ‘$k$’ is usually huge: you try out $k=O(nd)$ decision stumps.

• What if we train by gradient descent?
  – We’re optimizing on continuous space, so $k=\infty$ and **the bound is useless**.
  – In this case, **VC-dimension** is one way to replace ‘$k$’ (doesn’t need union bound).
    • “Simpler” models like decision stumps and linear models will have lower VC-dimension.

• **Learning theory** keywords if you want to go deeper into this topic:
  – Bias-variance (see bonus slides for details and why this is weird), sample complexity, PAC learning, VC dimension, Rademacher complexity.
  – A gentle place to start is the [Learning from Data book](https://www.adaptive.com/content/materials/widmayer/).
(pause)
Generalization Error

• An alternative measure of performance is the generalization error:
  – Average error over the set of $x^i$ values that are not seen in the training set.
  – “How well we expect to do for a completely unseen feature vector”.

• Test error vs. generalization error when labels are deterministic:

$$E_{test} = \mathbb{E} \left[ |\hat{y}^i - \tilde{y}^i| \right]$$

$$E_{general} = \frac{1}{\text{number of } x^i \text{ values not in training set}} \sum_{x^i \notin \text{training set}} |\hat{y}^i - \tilde{y}^i|$$

↑ Labels are deterministic, but we still take expectation over data distribution.

↓ Average error over unseen $x^i$ values.
“Best” and the “Good” Machine Learning Models

• Question 1: what is the “best” machine learning model?
  – The model that gets lower generalization error than all other models.

• Question 2: which models always do better than random guessing?
  – Models with lower generalization error than random for all problems.

• No free lunch theorem:
  – There is no “best” model achieving the best generalization error for every problem.
  – If model A generalizes better to new data than model B on one dataset, there is another dataset where model B works better.
No Free Lunch Theorem

• Let’s show the “no free lunch” theorem in a simple setting:
  – The \( x^i \) and \( y^i \) are binary, and \( y^i \) being a deterministic function of \( x^i \).

• With ‘d’ features, each “learning problem” is a map from \( \{0,1\}^d \rightarrow \{0,1\} \).
  – Assigning a binary label to each of the \( 2^d \) feature combinations.

<table>
<thead>
<tr>
<th>Feature 1</th>
<th>Feature 2</th>
<th>Feature 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Map 1</th>
<th>Map 2</th>
<th>Map 3</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>…</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>…</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

• Let’s pick one of these maps (“learning problems”) and:
  – Generate a set training set of ‘n’ IID samples.
  – Fit model A (convolutional neural network) and model B (naïve Bayes).
No Free Lunch Theorem

• Define the “unseen” examples as the \((2^d - n)\) not seen in training.
  – Assuming no repetitions of \(x^i\) values, and \(n < 2^d\).
  – Generalization error is the average error on these “unseen” examples.

• Suppose that model A got 1% error and model B got 60% error.
  – We want to show model B beats model A on another “learning problem”.

• Among our set of “learning problems” find the one where:
  – The labels \(y^i\) agree on all training examples.
  – The labels \(y^i\) disagree on all “unseen” examples.

• On this other “learning problem”:
  – Model A gets 99% error and model B gets 40% error.
No Free Lunch Theorem

• Further, across all “learning problems” with these ‘n’ examples:
  – Average generalization error of every model is 50% on unseen examples.
    • It’s right on each unseen example in exactly half the learning problems.
  – With ‘k’ classes, the average error is \((k-1)/k\) (random guessing).

• This is kind of depressing:
  – For general problems, no “machine learning” is better than “predict 0”.


(pause)
Limit of No Free Lunch Theorem

• Fortunately, the world is structured:
  – Some “learning problems” are more likely than others.

• For example, it’s usually the case that “similar” $x^i$ have similar $y^i$.
  – Datasets with properties like this are more likely.
  – Otherwise, you probably have no hope of learning.

• Models with the right “similarity” assumptions can beat “predict 0”.

• With assumptions like this, you can consider consistency:
  – As ‘n’ grows, model A converges to the optimal test error.
Refined Fundamental Trade-Off

- Let $E_{\text{best}}$ be the **irreducible error** (lowest possible error for *any* model).
  - For example, irreducible error for predicting coin flips is 0.5.
- Some learning theory results use $E_{\text{best}}$ to further decompose $E_{\text{test}}$:
  $$E_{\text{test}} = (E_{\text{test}} - E_{\text{train}}) + (E_{\text{train}} - E_{\text{best}}) + E_{\text{best}}$$
  - $E_{\text{approx}}$ measures how sensitive we are to training data (like “variance”).
  - $E_{\text{model}}$ measures if our model is complicated enough to fit data (like “bias”).
  - $E_{\text{best}}$ measures how low can *any* model make test error (“irreducible” error).
Refined Fundamental Trade-Off

• Let $E_{\text{best}}$ be the irreducible error (lowest possible error for any model).
  – For example, irreducible error for predicting coin flips is 0.5.
• Some learning theory results use $E_{\text{best}}$ to further decompose $E_{\text{test}}$:

\[ E_{\text{test}} = (E_{\text{test}} - E_{\text{train}}) + (E_{\text{train}} - E_{\text{best}}) + E_{\text{best}} \]

  - $E_{\text{approx}}$ and $E_{\text{model}}$ (bias-variability trade-off)

• This is similar to the bias-variance trade-off (bonus slide):
  – You need to trade between having low $E_{\text{approx}}$ and having low $E_{\text{model}}$.
  – Powerful models have low $E_{\text{model}}$ but can have high $E_{\text{approx}}$.
  – $E_{\text{best}}$ does not depend on what model you choose.
Consistency and Universal Consistency

• A model is **consistent** for a **particular learning problem** if:
  – $E_{\text{test}}$ converges to $E_{\text{best}}$ as ‘n’ goes to infinity, for that particular problem.

• A model is **universally consistent** for a **class of learning problems** if:
  – $E_{\text{test}}$ converges to $E_{\text{best}}$ as ‘n’ goes to infinity, for all problems in the class.

• Typically, the class would consist of:
  – A **continuity assumption** on the labels $y^i$ as a function of $x^i$.
    • E.g., if $x^i$ is close to $x^j$ then they are likely to receive the same label.
  – A boundedness assumption of the set of $x^i$.
K-Nearest Neighbours (KNN)

• Classical consistency results focus on \( k \)-nearest neighbours (KNN).

• To classify an object \( \tilde{x}_i \):
  1. Find the ‘k’ training examples \( x_i \) that are “nearest” to \( \tilde{x}_i \).
  2. Classify using the most common label of “nearest” examples.
Consistency of KNN

- **KNN consistency** properties (under reasonable assumptions):
  - As ‘n’ goes to ∞, \( E_{\text{test}} \leq 2E_{\text{best}} \).
    - For fixed ‘k’ and binary labels.

- **Stone’s Theorem**: KNN is “universally consistent”.
  - If ‘k’ converges to ∞ as ‘n’ converges to ∞, but \( k/n \) converges to 0, \( E_{\text{test}} \) converges to \( E_{\text{best}} \).
    - For example, \( k = O(\log n) \).
    - First algorithm shown to have this property.

- Consistency says nothing about finite ‘n’.
  - See "Don't Trust Asymptotics".
Consistency of Non-Parametric Models

• **Universal consistency** has been shown for several models:
  – Linear models with polynomial basis.
  – Linear models with Gaussian RBFs.
  – Neural networks with one hidden layer and standard activations.
    • Sigmoid, tanh, ReLU, etc.

• It’s **non-parametric** versions that are consistent:
  – Size of model is a function of ‘n’.
  – Examples:
    • KNN needs to store all ‘n’ training examples.
    • Degree of polynomial must grow with ‘n’ (not true for fixed polynomial).
    • Number of hidden units must grow with ‘n’ (not true for fixed neural network).
Parametric vs. Non-Parametric Models

Test error decreases with increasing number of examples. The gap may never go to zero.

- Parametric model
  - \( E_{\text{best}} \)
Parametric vs. Non-Parametric Models

Test error vs. number of examples (n).

Parametric model vs. Non-parametric model.

Gap can go to zero.
Summary

• Test error vs. test set error
  – What we care about is the test error.

• Overfitting hyper-parameters on a validation set:
  – Depends on how many hyper-parameters you try and number of validation examples.

• No free lunch theorem:
  – There is no “best” or even “good” machine learning models across all problems.

• Universal consistency:
  – Some non-parametric models can solve any continuous learning problem.

• Post-lecture bonus slides: bias-variance decomposition.

• Next time:
  – Besides least squares, what other problems can be solved in 1 line of code?
Bias-Variance Decomposition

• Analysis of **expected test error** of any learning algorithm:

Assume \( y_i = f(x_i) + \epsilon_i \) for some function 'f' and random error \( \epsilon \) with a mean of 0 and a variance of \( \sigma^2 \).

Assume we have a "learner" that can take a training set \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \) and use these to make predictions \( \hat{f}(x) \).

Then for a new example \((x_i, y_i)\) the error averaged over training sets is:

\[
E [ (y_i - \hat{f}(x_i))^2 ] = \text{Bias} [ \hat{f}(x_i) ]^2 + \text{Var} [ \hat{f}(x_i) ] + \sigma^2
\]

"Irreducible error": best we can hope for given the noise level.

**Expected error due to having wrong model.**

How sensitive is the model to the particular training set?

\[
\text{Var} [ \hat{f}(x_i) ] = E [ (\hat{f}(x_i) - E[\hat{f}(x_i)])^2 ]
\]
Learning Theory

• Bias-variance decomposition is a bit weird:
  – Considers expectation over *possible training sets*.

• Bias-variance says *nothing about your training set*.
  – This is different than Hoeffding bounds:
    • Bound the test error based on your actual training set and training/validation error.