Regression and the Bias-Variance Decomposition

William Cohen
10-601 April 2008

Readings: Bishop 3.1, 3.2
Regression

• Technically: learning a function \( f(x) = y \) where \( y \) is real-valued, rather than discrete.
  – Replace \( \text{livesInSquirrelHill}(x_1, x_2, \ldots, x_n) \) with \( \text{averageCommuteDistanceInMiles}(x_1, x_2, \ldots, x_n) \)
  – Replace \( \text{userLikesMovie}(u, m) \) with \( \text{usersRatingForMovie}(u, m) \)
  – …
Example: univariate linear regression

- Example: predict age from number of publications
Linear regression

• **Model:** \( y_i = ax_i + b + \varepsilon_i \) where \( \varepsilon_i \sim N(0,\sigma) \)

• **Training Data:** \((x_1,y_1),\ldots,(x_n,y_n)\)

• **Goal:** estimate \( a, b \) with \( w=(\hat{a},\hat{b}) \)

\[
w = \arg \max Pr(w \mid D) \\
= \arg \max Pr(D \mid w) Pr(w) \\
\approx \arg \max \sum_i \log Pr(y_i \mid x_i, w) \\
\approx \arg \min \sum_i [\hat{\varepsilon}_i(w)]^2 \quad \hat{\varepsilon}_i(w) \equiv y_i - (\hat{a}x_i + \hat{b})
\]
Linear regression

- **Model:** \( y_i = ax_i + b + \varepsilon_i \) where \( \varepsilon_i \sim N(0,\sigma) \)
- **Training Data:** \((x_1,y_1), \ldots, (x_n,y_n)\)
- **Goal:** estimate \( a, b \) with \( w=(\hat{a},\hat{b}) \) \( w = \arg \min \sum_{i} \hat{\varepsilon}_i^2 \)
- **Ways to estimate parameters**
  - Find derivative wrt parameters \( a,b \)
  - Set to zero and solve
    - Or use gradient ascent to solve
    - Or ....
Linear regression

How to estimate the slope?

slope = \frac{\Delta y}{\Delta x}

\approx \frac{(y_1 - \bar{y})}{(x_1 - \bar{x})} \approx \frac{(y_2 - \bar{y})}{(x_2 - \bar{x})} \approx \ldots

\approx \frac{1}{n} \sum_{i} (y_i - \bar{y})

\approx \frac{1}{n} \sum_{i} (x_i - \bar{x})

\approx \frac{1}{n} \sum_{i} \frac{(x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2}

n*\text{cov}(X, Y)

n*\text{var}(X)
Linear regression

How to estimate the intercept?

\[ y = \hat{a}x + \hat{b} \]

\[ \hat{b} = \bar{y} - \hat{a}\bar{x} \]
Bias/Variance Decomposition of Error
Bias – Variance decomposition of error

Return to the simple regression problem \( f : X \rightarrow Y \)

\[ y = f(x) + \varepsilon \]

What is the expected error for a learned \( h \)?
Bias – Variance decomposition of error

\[ E_D \left[ \int \int (f(x) + \varepsilon - h_D(x))^2 \Pr(\varepsilon) \Pr(x) d\varepsilon dx \right] \]

Experiment (the error of which I’d like to predict):
1. Draw size \( n \) sample \( D=(x_1, y_1), \ldots, (x_n, y_n) \)
2. Train linear function \( h_D \) using \( D \)
3. Draw a test example \( (x, f(x) + \varepsilon) \)
4. Measure squared error of \( h_D \) on that example
Bias – Variance decomposition of error (2)

\[ E_{D,\epsilon} \left\{ \left( f(x) + \epsilon - h_D(x) \right)^2 \right\} \]

Fix \( x \), then do this experiment:

1. Draw size \( n \) sample \( D = (x_1, y_1), \ldots, (x_n, y_n) \)
2. Train linear function \( h_D \) using \( D \)
3. Draw the test example \( (x, f(x) + \epsilon) \)
4. Measure squared error of \( h_D \) on that example
Bias – Variance decomposition of error

\[ E_{D,\epsilon} \left\{ (f(x) + \epsilon - h_D(x))^2 \right\} \]

\[
\begin{align*}
E \left\{ (t - \hat{y})^2 \right\} \\
= E \left\{ ([t - f] + [f - \hat{y}])^2 \right\} \quad \text{why not?} \\
= E \left\{ [t - f]^2 + [f - \hat{y}]^2 + 2[t - f][f - \hat{y}] \right\} \\
= E \left\{ [t - f]^2 + [f - \hat{y}]^2 + 2[tf - t\hat{y} - f^2 + f\hat{y}] \right\}
\end{align*}
\]
Bias – Variance decomposition of error

\[ E_{D,\varepsilon}\{ (t - \hat{y})^2 \} \]

\[ = E\left\{ (\left[ t - f \right] + [f - \hat{y}])^2 \right\} \]

\[ = E\left\{ [t - f]^2 + [f - \hat{y}]^2 + 2[t - f][f - \hat{y}] \right\} \]

\[ = E\left\{ [t - f]^2 + [f - \hat{y}]^2 + 2[tf - t\hat{y} - f\hat{y}] \right\} \]

\[ = E[\varepsilon^2] + E[(f - \hat{y})^2] + 2\left( E[tf] - E[t\hat{y}] - E[f^2] + E[f\hat{y}] \right) \]

Intrinsic noise

Depends on how well learner approximates \( f \)
Bias – Variance decomposition of error

\[ E\{ (f - \hat{y})^2 \} \]

\[ = E\{ (f - h) + (h - \hat{y}) \}^2 \]

\[ = E\{ (f - h)^2 + (h - \hat{y})^2 + 2(f - h)(h - \hat{y}) \} \]

\[ = E\{ (f - h)^2 + (h - \hat{y})^2 + 2(fh - f\hat{y} - h^2 + h\hat{y}) \} \]

\[ = E[(f - h)^2] + E[(h - \hat{y})^2] + 2(E[fh] - E[f\hat{y}] - E[h^2] + E[h\hat{y}]) \]

Squared difference between best possible prediction for \( x \), \( f(x) \), and our “long-term” expectation for what the learner will do if we averaged over many datasets \( D \), \( E_D[h_D(x)] \).

Squared difference between what we expect in a representative run on a dataset \( D \) (\( \hat{y} \)) and what we expect in a representative run on a dataset \( D \) (\( h \)).
Bias-variance decomposition

\[ E_D \left[ \int_y \int_x (h(x) - f(x))^2 p(y|x)p(x)dydx \right] \]

Make the long-term average better approximate the true function \( f(x) \)

\[
\text{bias}^2 = \int (E_D[h(x)] - f(x))^2 p(x)dx
\]

\[
\text{variance} = \int E_D[(h(x) - E_D[h(x)])^2]p(x)dx
\]

Make the learner less sensitive to variations in the data

How can you reduce \textbf{bias} of a learner?

How can you reduce \textbf{variance} of a learner?
A generalization of bias-variance decomposition to other loss functions

- “Arbitrary” real-valued loss $L(t,y)$
  
  But $L(y,y')=L(y',y)$, $L(y,y)=0$,
  and $L(y,y')!=0$ if $y!=y'$

- Define “optimal prediction”:
  \[ y^* = \arg\min_{y'} L(t,y') \]

- Define “main prediction of learner”
  \[ y_m = y_m^D = \arg\min_{y'} E_D\{L(t,y')\} \]

- Define “bias of learner”:
  \[ B(x) = L(y^*,y_m) \]

- Define “variance of learner”:
  \[ V(x) = E_D[L(y_m,y)] \]

- Define “noise for x”:
  \[ N(x) = E_t[L(t,y^*)] \]

Claim:
\[
E_{D,t}[L(t,y)] = c_1 N(x) + B(x) + c_2 V(x)
\]

where
\[
c_1 = \Pr_D[y=y^*] - 1
\]
\[
c_2 = 1 \text{ if } y_m = y^*, -1 \text{ else }
\]
Other regression methods
Example: univariate linear regression

- Example: predict age from number of publications

\[ \hat{y} \approx \frac{1}{7} x + 26 \]

Paul Erdős

Hungarian mathematician, 1913-1996

\( x \sim 1500 \)

age about 240
Linear regression

Summary:

\[ \hat{a} = \sum_{i} \frac{(x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2} \]

\[ \hat{b} = \bar{y} - \hat{a}\bar{x} \]

To simplify:

- assume zero-centered data, as we did for PCA
- let \( \mathbf{x} = (x_1, \ldots, x_n) \) and \( \mathbf{y} = (y_1, \ldots, y_n) \)
- then...

\[ \hat{a} = \mathbf{x}^T \mathbf{y} (\mathbf{x}^T \mathbf{x})^{-1} \]

\[ \hat{b} = 0 \]
Onward: multivariate linear regression

**Univariate**

\[
\begin{align*}
x & = \langle x_1, \ldots, x_n \rangle \\
y & = \langle y_1, \ldots, y_n \rangle \\
\hat{w} & = x^T y (x^T x)^{-1}
\end{align*}
\]

**Multivariate**

\[
\begin{align*}
X & = \begin{bmatrix} x_1^1, \ldots, x_1^k \\
& \ldots \\
& \ldots \\
& x_n^1, \ldots, x_n^k \end{bmatrix} \\
y & = \begin{bmatrix} y_1 \\
& \ldots \\
& \ldots \\
& y_n \end{bmatrix}
\end{align*}
\]

\[
\hat{y} = \hat{w}^1 x_1^1 + \ldots + \hat{w}^k x_1^k
\]

\[
\hat{w} = X^T y (X^T X)^{-1}
\]

**Example**

\[
\begin{align*}
w & = \arg \min \sum [\hat{e}_i(w)]^2 \\
\hat{e}_i(w) & \equiv y_i - w^T x_i
\end{align*}
\]
Onward: multivariate linear regression

\[ w = \arg \min \sum [\hat{e}_i(w)]^2 \]
\[ \hat{e}_i(w) \equiv y_i - w^T x_i \]
\[ w = \arg \min \sum [\hat{e}_i(w)]^2 + \frac{\lambda}{2} w^T w \]

\[ \hat{w} = X^T y (\lambda I + X^T X)^{-1} \]

\[ X = \begin{bmatrix} x_1^1, \ldots, x_1^k \\ \vdots \\ x_n^1, \ldots, x_n^k \end{bmatrix} \]
\[ y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \]

\[ \hat{y} = \hat{w}^1 x_1^1 + \ldots + \hat{w}^k x_1^k \]
\[ \hat{w} = X^T y (X^T X)^{-1} \]
Onward: multivariate linear regression

Multivariate, multiple outputs

\[ X = \begin{bmatrix}
  x_1^1, & \ldots, & x_1^m \\
  \vdots & & \vdots \\
  x_n^1, & \ldots, & x_n^m 
\end{bmatrix} \quad y = \begin{bmatrix}
  y_1 \\
  \vdots \\
  y_n 
\end{bmatrix} \quad Y = \begin{bmatrix}
  y_1^1, & \ldots, & y_1^k \\
  \vdots \\
  y_n^1, & \ldots, & y_n^k 
\end{bmatrix} \]

\[ \hat{y} = \hat{w}^1 x^1 + \ldots + \hat{w}^k x^k \]

\[ \hat{w} = X^T y (X^T X)^{-1} \]

\[ \hat{y} = WX \]

\[ W = X^T Y (X^T X)^{-1} \]
Onward: multivariate linear regression

\[ w = \arg \min \sum [\hat{e}_i(w)]^2 \]

\[ \hat{e}_i(w) \equiv y_i - w^T x_i \]

\[ w = \arg \min \sum [\hat{e}_i(w)]^2 + \frac{\lambda}{2} w^T w \]

\[ X = \begin{bmatrix} x_1^1, \ldots, x_1^k \\ \vdots \\ x_n^1, \ldots, x_n^k \end{bmatrix} \]

\[ y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \]

\[ \hat{y} = \hat{W}^1 x_1^1 + \ldots + \hat{W}^k x_1^k \]

\[ \hat{w} = X^T y (\lambda I + X^T X)^{-1} \]

bias \[ = \int (E_D[h(x)] - f(x))^2 p(x) dx \]

variance \[ = \int E_D[(h(x) - E_D[h(x)])^2] p(x) dx \]
Onward: multivariate linear regression

\[ w = \arg \min \sum [\hat{e}_i(w)]^2 \]
\[ \hat{e}_i(w) \equiv y_i - w^T x_i \]
\[ w = \arg \min \sum [\hat{e}_i(w)]^2 + \frac{\lambda}{2} w^T w \]

\[ w = (w^1, w^2) \] What does fixing \( w^2 = 0 \) do (if \( \lambda = 0 \))? 

\[ \hat{w} = X^T y (\lambda I + X^T X)^{-1} \]

\[ \hat{y} = \hat{w}^1 x^1 + \ldots + \hat{w}^k x^k \]

\[ bias^2 = \int (E_D[h(x)] - f(x))^2 p(x) dx \]

\[ variance = \int E_D[(h(x) - E_D[h(x)])^2] p(x) dx \]
Regression trees - summary

• Growing tree:
  – Split to optimize

\[ \Delta \text{error} = \text{sd}(T) - \sum_i \frac{|T_i|}{|T|} \times \text{sd}(T_i). \]

• At each leaf node
  – Predict the majority class

• Pruning tree:
  – Prune to reduce estimated error on training data
    
  estimates are adjusted by \((n+k)/(n-k)\):

  \( n = \#\text{cases}, \ k = \#\text{features} \)

• Prediction:
  – Trace path to a leaf and predict
    
    using to a linear interpolation of every prediction made by every node on the path

If the case follows branch \( S_i \) of subtree \( S \), let \( n_i \) be the number of training cases at \( S_i \), \( PV(S_i) \) the predicted value at \( S_i \), and \( M(S) \) the value given by the model at \( S \). The predicted value backed up to \( S \) is

\[
PV(S) = \frac{n_i \times PV(S_i) + k \times M(S)}{n_i + k}
\]
Regression trees: example - 1

CHMIN ≤ 7: RM0
CHMIN > 7:
  MMAX > 24000: RM1
  MMAX ≤ 24000:
    CACH ≤ 48: RM2
    CACH > 48: average 217.5

<table>
<thead>
<tr>
<th>Model</th>
<th>RM0</th>
<th>RM1</th>
<th>RM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle time</td>
<td>11.5</td>
<td>-101.4</td>
<td>11.9</td>
</tr>
<tr>
<td>Min mem</td>
<td></td>
<td>0.030</td>
<td></td>
</tr>
<tr>
<td>Max mem</td>
<td>0.003</td>
<td>0.008</td>
<td></td>
</tr>
<tr>
<td>Cache size</td>
<td>0.902</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min chans</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max chans</td>
<td>0.518</td>
<td>4.686</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 1: Model tree for CPU performance*

<table>
<thead>
<tr>
<th></th>
<th>Original Attributes</th>
<th>Transformed Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correlation</td>
<td>Percentage Deviation</td>
</tr>
<tr>
<td>Ein-Dor (retrial)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>IBP</td>
<td>–</td>
<td>35.0%</td>
</tr>
<tr>
<td>M5</td>
<td>.921</td>
<td>34.9%</td>
</tr>
<tr>
<td>M5 (no smoothing)</td>
<td>.908</td>
<td>37.2%</td>
</tr>
<tr>
<td>M5 (no models)</td>
<td>.803</td>
<td>49.9%</td>
</tr>
</tbody>
</table>

*Table 1: CPU performance data*
Regression trees – example 2

What does pruning do to bias and variance?

- *servo*: (167 cases)

  This interesting collection of data, provided by Karl Ulrich, refers to an extremely non-linear phenomenon – predicting the rise time of a servomechanism in terms of two (continuous) gain settings and two (discrete) choices of mechanical linkages.

---

<table>
<thead>
<tr>
<th>Model</th>
<th>LM1</th>
<th>LM2</th>
<th>LM3</th>
<th>LM4</th>
<th>LM5</th>
<th>LM6</th>
<th>LM7</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>constant term</em></td>
<td>-0.44</td>
<td>2.6</td>
<td>3.5</td>
<td>0.18</td>
<td>0.52</td>
<td>0.36</td>
<td>0.23</td>
</tr>
<tr>
<td>pgain</td>
<td>0.82</td>
<td>0.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.06</td>
</tr>
<tr>
<td><em>motor = D</em> vs E, C, B, A*</td>
<td>3.3</td>
<td>0.24</td>
<td>0.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>motor = D, E</em> vs C, B, A*</td>
<td>1.8</td>
<td>-0.16</td>
<td>0.15</td>
<td>0.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>motor = D, E, C</em> vs B, A*</td>
<td></td>
<td>0.1</td>
<td>0.09</td>
<td>0.07</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>motor = D, E, C, B</em> vs A*</td>
<td></td>
<td></td>
<td>0.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>screw = D</em> vs E, C, B, A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>screw = D, E</em> vs C, B, A*</td>
<td>0.47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>screw = D, E, C</em> vs B, A*</td>
<td>0.63</td>
<td>0.28</td>
<td>0.34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>screw = D, E, C, B</em> vs A*</td>
<td></td>
<td>0.9</td>
<td>0.16</td>
<td>0.14</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 2.** Model tree and linear models for data set *servo*
Kernel regression

- *aka* locally weighted regression, locally linear regression, LOESS, ...

**Figure 2:** In locally weighted regression, points are weighted by proximity to the current $x$ in question using a kernel. A regression is then computed using the weighted points.
Kernel regression

• *aka* locally weighted regression, locally linear regression, …

• Close approximation to kernel regression:
  – Pick a few values $z^1, \ldots, z^k$ up front
  – Preprocess: for each example $(x, y)$, replace $x$ with
    \[
    x = \langle K(x, z^1), \ldots, K(x, z^k) \rangle
    \]
    where
    \[
    K(x, z) = \exp\left( -\frac{(x-z)^2}{2\sigma^2} \right)
    \]
  – Use multivariate regression on $x, y$ pairs
Kernel regression

• *aka* locally weighted regression, locally linear regression, LOESS, …

What does making the kernel wider do to bias and variance?

**Figure 3:** The estimator variance is minimized when the kernel includes as many training points as can be accommodated by the model. Here the linear LOESS model is shown. Too large a kernel includes points that degrade the fit; too small a kernel neglects points that increase confidence in the fit.
Additional readings


