Minimum Description Length (MDL)

Lyle Ungar

AIC – Akaike Information Criterion
BIC – Bayesian Information Criterion
RIC – Risk Inflation Criterion
MDL

- Sender and receiver both know X
- Want to send y using minimum number of bits
- Send y in two parts
  - Code (the model)
  - Residual (training error = “in-sample error”)
- Decision tree
  - Code = the tree
  - Residual = the misclassifications
- Linear regression
  - Code = the weights
  - Residual = prediction errors

The MDL model is of optimal complexity
Trades off bias and variance
Complexity of a decision tree

- Need to code the structure of the tree and the values on the leaves

Homework?
Complexity of Classification Error

◆ Have two sequences y, ŷ
  ● Code the difference between them
  ● y' = (0, 1, 1, 0, 1, 1, 1, 0)
  ● ŷ' = (0, 0, 1, 1, 1, 1, 1, 0)
  ● y' - ŷ' = (0, 1, 0, -1, 0, 0, 0, 0)

◆ How to code the differences?
◆ How many bits would the best possible code take?
MDL for linear regression

◆ Need to code the model
  ● For example
    ■ \( y = 3.1 x_1 + 97.2 x_{321} - 17.4 x_{5204} \)
  ● Two part code for model
    ■ Which features are in the model
    ■ Coefficients for those features

◆ Need to code the residual
  ● \( \Sigma_i (y_i - \hat{y}_i)^2 \)

◆ How to code a real number?

How to code which features are in the model?
How to code the feature values?
Complexity of a real number

- A real number could require infinite number of bits
- So we need to decide what accuracy to code it to
  - Code Sum of Square Error (SSE) to the accuracy given by the irreducible variance (bits $\sim 1/\sigma^2$)
  - Code each $w_i$ to its accuracy (bits $\sim n^{1/2}$)
- We know that $y$ and $w_i$ are both normally distributed
  - $y \sim N(x \cdot w, \sigma^2)$
  - $w_{j,est} \sim N(w_j, \sigma^2/n)$
- Code a real value by dividing it into regions of equal probability mass
  - Optimal coding length is given by entropy: $\int p(y) \log(p(y))dy$
MDL regression penalty

Code the residual / data log-likelihood

- $\log(\text{likelihood}) = - \log(\prod_i p(y_i|x_i)) =$
- $- \log[\prod (1/\sqrt{2\pi}\sigma) \exp(-|y-\hat{y}|^2/2\sigma^2)] =$
  \begin{align*}
    n \ln(\sqrt{2\pi} \sigma) + \frac{\text{Err}_q}{2\sigma^2}
  \end{align*}$

Code the model

For each feature: is it in the model?
If it is included, what is its coefficient?
Code the residual

Code the residual / data log-likelihood

\[- \log(\text{likelihood}) = - \log(\prod_i p(y_i|x_i)) =\]
\[n \ln(\sqrt{2\pi} \sigma) + \frac{\text{Err}_q}{2\sigma^2}\]

If we know \(\sigma^2\)

then the first term is a constant (and has no effect) and
the second term gives a scaled version of the Error
Code the residual

Code the residual / data log-likelihood

\[- \log(\text{likelihood}) = - \log(\prod_i p(y_i|x_i)) = \]
\[n \ln(\sqrt{2\pi} \sigma) + \frac{\text{Err}_q}{2\sigma^2}\]

But we don’t know \(\sigma^2\).

\[\sigma^2 = E[(y-\hat{y})^2)] = (1/n) \text{Err}\]

Option 1 – use estimate from previous iteration

\[\sigma^2 = \text{Err}_{q-1} / n \quad \text{pay} \quad \text{Err}_q / 2 \text{Err}_{q-1}\]

Option 2 – use estimate from current iteration

\[\sigma^2 = \text{Err}_q / n \quad \text{pay} \quad n \ln(\sqrt{2\pi} \text{Err}_q/n)\]
For each feature: is it in the model?

If you expect q features in the model, each will come in with probability $(q/p)$

The total cost is then

$$p \left[ -(q/p) \log(q/p) - ((p-q)/p) \log((p-q)/p) \right]$$

If $q/p = \frac{1}{2}$ total cost is $p$

$$\text{cost/selected feature} = 2 \text{ bits}$$

If $q = 1$, total cost is roughly $\log(p)$

$$\text{cost/selected feature} = \log(p) \text{ bits}$$
Code each coefficient

Code each **coefficient** with accuracy proportional to $n^{1/2}$

$(1/2) \log(n)$ bits/feature
MDL regression penalty

Minimize
\[ \text{Err}_q / 2\sigma^2 + \lambda \ |w|_0 \]

L₀ penalty on coefficients
SSE with the \( |w|_0 = q \) features

Penalty \( \lambda = - \log(\pi) + (1/2) \log(n) \)

Code each feature presence using \(-\log(\pi)\) bits/feature
\( \pi = q/p \) assume \( q << p \), so \( \log(1-\pi) \) is near 0
if \( q=1 \), then \(-\log(\pi) = \log(p)\)

Code each coefficient with accuracy proportional to \( n^{1/2} \)
\( (1/2) \log(n) \) bits/feature

n observations \( q = |w|_0 \) actual features
p potential features
Entrophy of features being present or absent:
\[
\sum (-\pi \log(\pi) - (1-\pi) \log(1-\pi)) =
\]
\[
(-\pi \log(\pi) - (1-\pi) \log(1-\pi))p =
\]
\[
-qp \log(\pi) - (p-q) \log(1-\pi)
\]
If \( \pi \ll 1 \) then \( \log (1-\pi) \) is roughly 0

\(-q \log(\pi)\) - is the cost of coding the q features

So each feature costs \( \log(\pi) \) bits

\( \pi = q/p \) probability of a feature being selected

n observations \( q = |w|_0 \) actual features

p potential features
### Regression penalty methods

Minimize
\[
\text{Err}_q/2\sigma^2 + \lambda |w|_0
\]

$L_0$ penalty on coefficients
SSE with the $|w|_0 = q$ features

<table>
<thead>
<tr>
<th>Method</th>
<th>penalty ($\lambda$)</th>
<th>code coefficient using 1 bit</th>
<th>code coefficient using $n^{1/2}$ bits</th>
<th>code feature presence/absence prior: one feature will come in</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIC</td>
<td>$(1/2) \log(n)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIC</td>
<td>$\log(p)$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How do you estimate $\sigma^2$?
Regression penalty methods

Which penalty should you use if

- You expect 10 out of 100,000 features, n = 100
- You expect 200 out of 1,000 features, n = 1,000,000
- You expect 500 out of 1,000 features, n = 1,000

Minimize

\[ \frac{\text{Err}_q}{2\sigma^2} + \lambda |w|_0 \]

Method penalty (\(\lambda\))

A) AIC \(1\) code coefficient using 1 bit
B) BIC \(\frac{1}{2}\log(n)\) code coefficient using \(n^{1/2}\) bits
C) RIC \(\log(p)\) code feature presence/absence
Mallows’ $C_p$, AIC as MDL

Err/ $2\sigma^2 + q$

- Mallows’ $C_p$
  - $C_p = \frac{\text{Err}_q}{\sigma^2} + 2q - n$

- AIC
  - $AIC = -2 \log(\text{likelihood}) + 2q$
    
    \begin{align*}
    &= -2 \log\left[\prod\left(1/\sqrt{2\pi}\sigma\right) \exp\left(- \frac{\text{Err}_q}{2\sigma^2}\right)\right] + 2q
    \end{align*}

But the best estimate we have is $\sigma^2 = \frac{\text{Err}_q}{n}$

$AIC \sim 2n \log \left(\frac{\text{Err}_q}{n}\right)^{1/2} - 2 \log \exp(- \frac{\text{Err}_q}{2\text{Err}_q/n}) + 2q$

$\sim n \log(\text{Err}_q/n) + 2q$

$q = |w|_0$ features in the model

$n$ doesn’t effect maximization
BIC is MDL (as n goes to infinity)

\[
\text{Err/ } 2\sigma^2 + (1/2) \log(n) q
\]

◆ BIC

\[
- 2 \log(\text{likelihood}) + 2 \log(\sqrt{n}) q
\]

\[
= n \log(\text{Err}_q/n) + \log(n) q
\]

using the exact same derivation as before.

\[q = |w|_0 \text{ features in the model}\]
Why does MDL work?

◆ **We want to tune model complexity**
  - How many bits should we use to code the model?

◆ **Minimize**
  
  \[
  \text{Test error} = \text{training error} + \text{penalty} = \text{bias} + \text{variance}
  \]
  
  - Training error = bias = bits to code residual
    - \( \sum_i (y_i - \hat{y}_i)^2 / 2\sigma^2 = -p(y|\mathbf{X}) \log p(y|\mathbf{X}) \)
  
  - Penalty = variance = bits to code the model

Ignoring irreducible uncertainty
What you should know

- **How to code (in the info-theory sense)**
  - decision trees, regression models,
  - classification and prediction errors
- **AIC, BIC and RIC**
  - Assumptions behind them
  - Why they are useful
You think maybe 10 out of 100,000 features will be significant. Use
A) $L_2$ with CV
B) $L_1$ with CV
C) $L_0$ with AIC
D) $L_0$ with BIC
E) $L_0$ with RIC
You think maybe 500 out of 1,000 features will be significant. Do not use
A) $L_2$ with CV
B) $L_1$ with CV
C) $L_0$ with AIC
D) $L_0$ with BIC
E) $L_0$ with RIC
Regression penalty methods

Minimize
Err/ 2\sigma^2 + \lambda |w|_0

Err is
A) \sum_i (y_i - \hat{y}_i)^2
B) (1/n) \sum_i (y_i - \hat{y}_i)^2
C) \sqrt{((1/n) \sum_i (y_i - \hat{y}_i)^2} 
D) something else

Where does the 2\sigma^2 come from?
**Bonus: p-values**

- **P-value**: the probability of getting a false positive

If I check 1,000 univariate correlations between $x_j$ and some $y$, and accept those with $p < 0.01$

I should expect roughly ___ false positives

A) 0
B) 1
C) 10
D) 100
E) 1,000

How would you ‘fix’ this?
Bonus: p-values

- **Bonferroni**
  - require a p-value to be \( p \) times smaller
  - p-value < 0.01(1/p)

- **Simes: sequential feature selection** (a.k.a. Benjamini-Hochberg)
  - Sort features by their p-values
  - For the first feature to be accepted use Bonferroni
    - p-value < 0.01(1/p)  -- if nothing passes, then stop
  - If it is accepted the p-value for the second feature is:
    - p-value < 0.01(2/p) -- if nothing passes, then stop
  - If it is accepted the p-value for the third feature is
    - p-value < 0.01(3/p)

\( p = \text{number of features} \)
\( 1/p = \text{prior probability} \)