# 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 ~ $1/\sigma^2$)
  - Code each $w_i$ to its accuracy (bits ~ $n^{1/2}$)
- **We know that y and $w_i$ are both normally distributed**
  - $y \sim N(\mathbf{x}\cdot\mathbf{w}, \sigma^2)$
  - $w_j^{est} \sim N(w_j, \sigma^2/n)$
- **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

- $\log(\text{likelihood}) = -\log(\Pi_i p(y_i|\mathbf{x}_i)) =$

- $\log[\Pi(1/\text{sqrt}(2\pi)\boldsymbol{\sigma}) \exp(-|\mathbf{y}-\hat{\mathbf{y}}|_2^2/2\boldsymbol{\sigma}^2)] =$

$n \ln(\text{sqrt}(2\pi)\, \boldsymbol{\sigma}) + \text{Err}_q/ 2\boldsymbol{\sigma}^2$

## 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**

- $\log(\text{likelihood}) = -\log(\Pi_i p(y_i|\mathbf{x}_i)) =$

  $n \ln(\text{sqrt}(2\pi)\ \sigma) + Err_q / 2\sigma^2$

**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**

- $\log(\text{likelihood}) = - \log(\Pi_i p(y_i|\mathbf{x}_i)) =$

  $n \ln(\text{sqrt}(2\pi)\, \sigma) + \text{Err}_q / 2\sigma^2$

**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**

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

**Option 2 – use estimate from current iteration**

$\sigma^2 = \text{Err}_q /n$ pay $n \ln(\text{sqrt}(2\pi)\ \text{Err}_q/n)$

# 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**

$$p\,[-(q/p)\log(q/p) - ((p-q)/p)\log((p-q)/p)]$$

**If q/p = ½ total cost is *p***

cost/selected feature = 2 bits

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

cost/selected feature = log(p) bits

# Code each coefficient

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

(1/2) $log(n)$ bits/feature

# MDL regression penalty

**Minimize**

$Err_q / 2\sigma^2$ **+** $\lambda\ |w|_0$

$L_0$ 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 $q \ll p$, 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*

# MDL regression penalty - aside

**Entropy of features being present or absent:**

$\Sigma(-\pi\log(\pi) - (1-\pi)\log(1-\pi)) =$

$(-\pi\log(\pi) - (1-\pi)\log(1-\pi))p =$

$-q \log(\pi) - (p-q) \log (1-\pi)$

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**

$Err_q / 2\sigma^2 + \lambda |w|_0$

$L_0$ penalty on coefficients

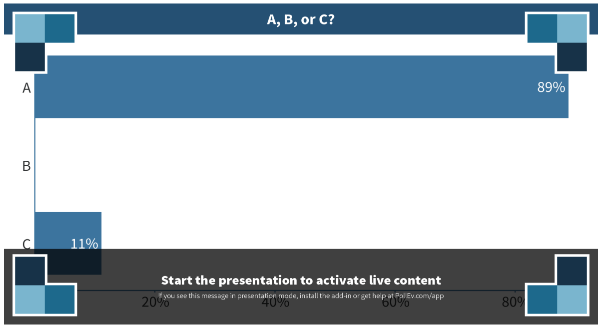
SSE with the $|w|_0 = q$ features

| **Method** | **penalty ($\lambda$)** | |
|---|---|---|
| **AIC** | *1* | code coefficient using 1 bit |
| **BIC** | (1/2) *log(n)* | code coefficient using $n^{1/2}$ bits |
| **RIC** | *log(p)* | code feature presence/absence<br>*prior: one feature will come in* |

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**

$Err_q / 2\sigma^2 + \lambda \, |w|_0$

| **Method** | **penalty ($\lambda$)** | |
|---|---|---|
| **A)** AIC | *1* | code coefficient using 1 bit |
| **B)** BIC | (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' $C_p$**
  - $C_p = Err_q/\sigma^2 + 2\ q - n$
- **AIC**
  - $AIC = -2 \log(\text{likelihood}) + 2\ q$

$$= -2 \log[\Pi(1/\text{sqrt}(2\pi)\sigma) \exp(- Err_q/2\sigma^2)] + 2q$$

**But the best estimate we have is $\sigma^2 = Err_q/n$**

$$AIC \sim 2n \log (Err_q/n)^{1/2} - 2 \log \exp(- Err_q/(2Err_q/n)) + 2q$$

$$\sim n \log(Err_q/n) + 2\ q$$

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

*n* doesn't effect maximization

# **BIC is MDL** (as n goes to infinity)

$Err/ 2\sigma^2 + (1/2)\, \mathit{log(n)}\, q$

- **BIC**

  $- 2 \log(\text{likelihood}) + 2 \log(\text{sqrt}(n))\, q$

  $= n \log(Err_q/n) + \log(n)\, q$

**using the exact same derivation as before.**

$\mathbf{q = |w|_0}$ 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**

  **Test error = training error + penalty**

  **= bias + variance**

  - **Training error = bias = bits to code residual**
    - $\Sigma_i (y_i-\hat{y}_i)^2 / 2\sigma^2 = -\, p(\mathbf{y}|\mathbf{X}) \log p(\mathbf{y}|\mathbf{X})$
  - **Penalty = variance = bits to code the model**

Ignoring irreducible uncertainty

# What you should know

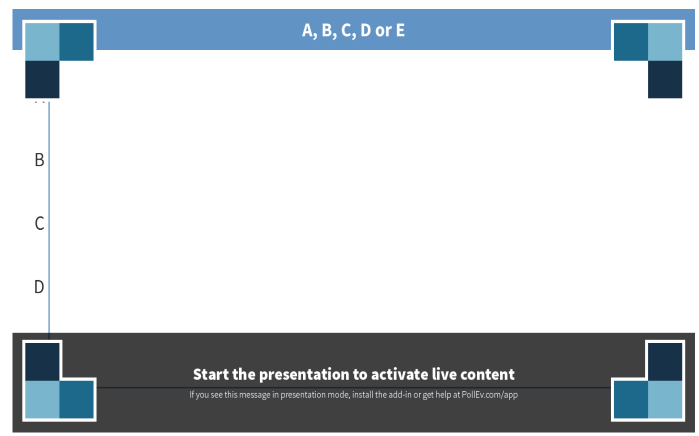
- **How to code (in the info-theory sense)**
  - decision trees, regression models,
  - classification and prediction errors
- **AIC, BIC and RIC**
  - Assumptions behind them
  - Why they are useful

**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)** $\Sigma_i\ (y_i-\hat{y}_i)^2$
- **B)** $(1/n)\ \Sigma_i\ (y_i-\hat{y}_i)^2$
- **C)** $sqrt((1/n)\ \Sigma_i\ (y_i-\hat{y}_i)^2\ )$
- **D)** something else

**Where does the $2\sigma^2$ come from?**

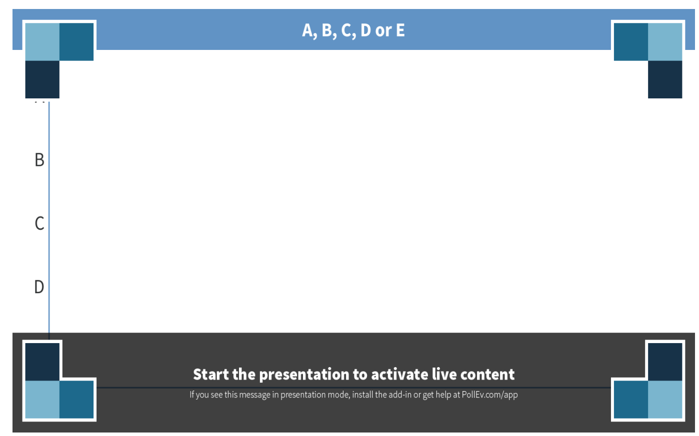
# Bonus: p-values

- **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)

**p = number of features**
1/p = prior probability

- **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)