Applied Statistics and Machine Learning

MDL, e-L2Boosting, Sparse Graphical Model Estimation, and Ridge Estimation

Bin Yu, IMA, June 24, 2013
MDL is a modern articulation of Occam’s Razor through Shannon’s Information Theory.

Jorma Rissanen (1978): One should choose the model that gives the shortest description of data.

Conceptually, the MDL framework removes the need for the postulation of a true model.


A (binary) code is a map from a set \( A \) of elements to a string of 0’s and 1’s. A prefix code corresponds to the leave nodes of a tree (no need to look ahead when decoding). To be a prefix code, it is necessary and sufficient for the Kraft’s inequality to hold:

\[
\sum_{x \in \mathcal{X}} 2^{-L(x)} \leq 1,
\]

where \( L \) is the number of 0’s and 1’s in the code of \( a \), integer valued and with unit (bits = binary digits, coined by Tukey to Shannon).

Example of a prefix code on the alphabet of \{a, b, c\}:
a maps into 0, b maps into 10 and c into 11.
Information Theoretic Foundation of MDL

Shannon’s source coding theorem says, $X$ is a random variable with probability distribution $P$. Given any prefix code $Q$ (code book), $EL_Q(X) \geq H(P)$. It can be proved by $KL \geq 0$ (Homework).

For discrete $X$,

$$H(P) = \sum_x P(x) [- \log P(X)] = \sum_x P(x) \log \frac{1}{P(X)}$$
Any distribution can be viewed as a code-length

For example: $P(a) = 1/3, P(b) = 1/3, P(c) = 1/3, L_Q(a) = 1, L_Q(b) = 1, L_Q(c) = 1$

$$EL_Q(X) = \frac{1}{3}(1 + 2 + 2) = \frac{5}{3}$$

$$H(P) = \log 3$$

The data generating probabilities for $X = a, b, c$ are equal, but $Q(a) = 1/2, Q(b) = 1/4, Q(c) = 1/4$ in the code book.

For any probability distribution on $Q$ on the alphabet,

- $-\log Q(a)$ could be viewed a code length.

$Q$ does not have to be the data generating probability distribution. When it is, the code is most “efficient” in the Shannon sense.
A valid code to be used in MDL

Rissenan (1978) generalized Shannon’s source coding theorem to a universal source coding theorem. That is, he showed that if a data generating distribution is known to belong to a nice parametric family of dim k, then the redundancy of any universal code is lower bounded by k/2 log n for n data points, for all members of the parametric family except for a zero measure set.

This means that one has to pay a price of k/2 log n for not knowing the particular member of a parametric family.

If a code achieves this lower bound, we call it “valid” to be used in MDL to compare model classes.
Different valid forms of code lengths to use with MDL in parametric models

(0) two-stage code

A class of models:

\[ M_k : k \in \{1, \ldots, M\} \]

“best” code \( Q_k \): choose

\[ \hat{k} = \arg \min_{1 \leq k \leq M} [-\log Q_k(\text{data})] \]

BIC(\( M_k \)) corresponds to a 2-stage “best” code book for data based on \( M_k \).

\[ \text{BIC}(M_k) = -\log P(\text{data}|\hat{\theta}_k) + \frac{k}{2} \log n, \]

\( \frac{k}{2} \log n \): # of bits needed to send \( \hat{\theta}_k \) over.

Each parameter takes roughly \(-\log \frac{1}{\sqrt{n}} = \frac{1}{2} \log n \) bits. \( \frac{1}{\sqrt{n}} \) is from local Gaussian property. Suppose that we have only one parameter, the estimation precision is \( 1/\sqrt{n} \).
Two-stage code length

- BIC balances fitting to data with model complexity and so does AIC.

  The penalty in AIC is motivated from the complexity penalty on prediction for a model while that in BIC is code length of the model – both go up with the complexity of the model.

- Using two stage coding MDL with sparse regression model will lead to something quite close to LS with a log form of penalty (Mairal’s lecture)
Predictive MDL

Two other forms of “valid” codes (or best to the first order after entropy term):

\[(1) \text{ Predictive coding: } x_1, \ldots, x_n. \text{ any probability on } X_1, \ldots, X_n. \]
\[p(x_1, \ldots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_n|x_1, \ldots, x_{n-1})\]

predictive best \( Q \)

\[- \log Q = - \sum_{t=1}^{n} \log P(x_t|\hat{\theta}_{t-1}),\]

where \( \hat{\theta}_{t-1} \) is the MLE based on \( x_1, \ldots, x_{n-1} \).

\[- \log Q - [ - \log P(x_1, \ldots, x_n|\theta^*)] = \sum_{t=1}^{n} \frac{k}{2t} = \frac{k}{2} \log n,\]

Predictive MDL is closely related to on-line learning.
Predictive MDL: example

Example: predictive MDL for regression model selection with known noise variance:

\[ M_k: \text{first } k \text{ predictor Normal regression model} \]

\[ p(y_t|\text{past}) = \frac{1}{\sqrt{2\pi}} \exp\left\{ -\frac{(y_t - x_t(k)^T \hat{\beta}_{t-1}(k))^2}{2\sigma^2} \right\}, \]

where \( \hat{\beta}_{t-1}(k) = \text{OLS based on first } k \text{ predictions and data units up to time } t-1. \)

That is, the response is \((y_1, \ldots, y_{t-1})^T\), and the design matrix is \((x_{ij}), i = 1, \ldots, t-1, j = 1, \ldots, k.\)

\[ -\log Q_k(y_1, \ldots, y_n) = \frac{1}{2\sigma^2} \sum_{t=1}^{n} \frac{(y_t - x_t(k)^T \hat{\beta}_{t-1}(k))^2}{2\sigma^2} \]
Predictive MDL for combining predictors in audio compression

Question:
how to compress, in a perceptually lossless manner, mixed audio signals (music and speech, songs) with low delay?
Application: internet radio, musicians playing together long-distance
Predictive MDL for combining predictors in audio compression

Key steps in Schuller et al (2002):
1. Perceptual pre-filter banks (related to wavelets)
2. Quantization on the filtered signals
3. Lossless predictive coding for quantized signals
Predictive MDL for combining predictors in audio compression

WCLMS predictor (weighted cascading LMS) in Step3:

Basic predictor P1: normalized LMS (Least Mean Square minimization with one data point at a time: a version of stochastic gradient) (LMS is an old algorithm from control theory) with lag L1. Fitting another N-LMS (with lag L2) to the residual time series to arrive at a new predictor P2; Repeating this with lag L3 gives P3. This cascading is really Tukey’s twicing or L2boosting in some sense.

For each of the 3 predictors, we get a weight at time $i$ ($c=2$, $\mu=0.9$) to linearly combine the three predictors:

$$\exp \left( -c(1-\mu) \sum_{i=1}^{n-1} |e_i(n-i)| \cdot \mu^{i-1} \right)$$
Compression results (bit rates) of LMS and WCLMS with different lags

<table>
<thead>
<tr>
<th>Order</th>
<th>LMS 10</th>
<th>40</th>
<th>80</th>
<th>200</th>
<th>400</th>
<th>WCLMS 40,80,200</th>
<th>WCLMS 200,80,40</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>44kHz</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chart44</td>
<td>2.05</td>
<td>1.98</td>
<td>1.98</td>
<td>1.93</td>
<td>1.92</td>
<td>1.87</td>
<td>1.81</td>
</tr>
<tr>
<td>jazz44</td>
<td>2.08</td>
<td>2.05</td>
<td>1.98</td>
<td>1.89</td>
<td>1.89</td>
<td>1.86</td>
<td>1.75</td>
</tr>
<tr>
<td>mspeech</td>
<td>2.11</td>
<td>2.06</td>
<td>2.07</td>
<td>2.09</td>
<td>2.10</td>
<td>1.99</td>
<td>1.98</td>
</tr>
<tr>
<td>spot44</td>
<td>2.01</td>
<td>1.99</td>
<td>2.00</td>
<td>2.01</td>
<td>2.01</td>
<td>1.90</td>
<td>1.88</td>
</tr>
<tr>
<td><strong>32kHz</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chart</td>
<td>2.18</td>
<td>2.13</td>
<td>2.11</td>
<td>2.04</td>
<td>2.02</td>
<td>2.01</td>
<td>1.94</td>
</tr>
<tr>
<td>jazz</td>
<td>2.39</td>
<td>2.34</td>
<td>2.22</td>
<td>2.11</td>
<td>2.10</td>
<td>2.10</td>
<td>1.99</td>
</tr>
<tr>
<td>mixed</td>
<td>2.29</td>
<td>2.26</td>
<td>2.26</td>
<td>2.25</td>
<td>2.25</td>
<td>2.18</td>
<td>2.16</td>
</tr>
<tr>
<td>spot</td>
<td>2.07</td>
<td>2.05</td>
<td>2.06</td>
<td>2.07</td>
<td>2.09</td>
<td>1.99</td>
<td>1.96</td>
</tr>
<tr>
<td><strong>16kHz</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chart</td>
<td>2.39</td>
<td>2.33</td>
<td>2.22</td>
<td>2.17</td>
<td>2.19</td>
<td>2.10</td>
<td>2.02</td>
</tr>
<tr>
<td>jazz</td>
<td>2.70</td>
<td>2.43</td>
<td>2.30</td>
<td>2.27</td>
<td>2.29</td>
<td>2.15</td>
<td>2.08</td>
</tr>
<tr>
<td>mixed</td>
<td>2.40</td>
<td>2.38</td>
<td>2.37</td>
<td>2.36</td>
<td>2.37</td>
<td>2.31</td>
<td>2.28</td>
</tr>
<tr>
<td>spot</td>
<td>2.32</td>
<td>2.33</td>
<td>2.33</td>
<td>2.32</td>
<td>2.35</td>
<td>2.25</td>
<td>2.21</td>
</tr>
<tr>
<td><strong>8kHz</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chart</td>
<td>2.51</td>
<td>2.28</td>
<td>2.22</td>
<td>2.21</td>
<td>2.27</td>
<td>2.09</td>
<td>2.03</td>
</tr>
<tr>
<td>jazz</td>
<td>2.83</td>
<td>2.31</td>
<td>2.25</td>
<td>2.27</td>
<td>2.35</td>
<td>2.10</td>
<td>2.06</td>
</tr>
<tr>
<td>mixed</td>
<td>2.42</td>
<td>2.39</td>
<td>2.37</td>
<td>2.39</td>
<td>2.41</td>
<td>2.33</td>
<td>2.31</td>
</tr>
<tr>
<td>spot</td>
<td>2.40</td>
<td>2.39</td>
<td>2.37</td>
<td>2.41</td>
<td>2.44</td>
<td>2.33</td>
<td>2.32</td>
</tr>
</tbody>
</table>
Comments on compression results

Single LMS does not work well compared to WCLMS.

For WCLMS, it is better to have long lags (200, 80, 40) to remove more harmonic signals first.
Mixture form of MDL

(2) Mixture form of MDL, or code length:

\[ M_k: (\beta_k, \sigma^2), y = x^T(k)\beta_k + \epsilon. \]

\[ Q(y_1, \ldots, y_n | M_k) = \int p(y_1, \ldots, y_n | \beta_k, \sigma^2)p(\beta_k, \sigma^2) \, d\beta_k \, d\sigma^2 \]

Bayes factor choose one of 2 models by looking at the ratio of the mixture probability. Choose \( k_1 \) if

\[ \frac{Q(y_1, \ldots, y_n | M_{k_1})}{Q(y_1, \ldots, y_n | M_{k_2})} > 1, \]

assuming \( M_{k_1} \) and \( M_{k_2} \) are equally likely.
Boosting: computation and estimation in one

Boosting was originated from Valiant’s Probability Approximate Correct (PAC) learning (Schapire, 1991), but made practical by Freund and Schapire (1996) (AdaBoost).

It has enjoyed impressive empirical successes.

Boosting idea:

choose a base procedure and apply it sequentially to modified data and then linearly combine the resulted estimators.

Cross-validation or a test set are commonly used to stop the iterations early.
Boosting: a gradient view

An important step forward for understanding boosting was made via the gradient view of Boosting by Breiman, Mason et al, Friedman et al, -- 2000

Friedman (01): L2 boosting or boosting with squared loss in the regression case: refitting of residuals (Twicing, Tukey, 1972).

Number of iteration steps is the regularization parameter.

Minimax optimality of L2 boosting in 1-dim case for Sobolev classes (Buhlmann & Yu, 03) (see Yu’s last lecture)
Stepwise regression: selected refs


Roughly, forward selection methods use tests (e.g. F-test) to find the most significant predictor to add and then the new model is refitted via OLS.

Backward-selection or backward elimination is done similarly.
Epsilon-L2Boosting (with fixed small steps)

Regression set-up:

standardized data \[ Z_i = (Y_i, X_i), \ i = 1, \ldots, n \]

\( X_i \): a p-dimensional predictor

\( Y_i \): response variable

We consider loss function \[ \sum_i L(Z_i; \beta) \] and let j index the jth predictor for \( j = 1, \ldots, p \).

- \( L^2 \) Boosting with a fixed step size (Forward Stagewise Fitting, Efron et al, 2004):
  \[ \epsilon > 0 \]
  \[
  (\hat{j}, \hat{s}) = \arg \min_{j, \, s = \pm \epsilon} \sum_{i=1}^{n} L(Z_i; \hat{\beta}^t + s1_j), \\
  \hat{\beta}^{t+1} = \hat{\beta}^t + \hat{s}1_{\hat{j}},
  \]

June 24, 2013
A Tidy Model of How the World Works

1. Investigation & Reporting
2. Policy & Decision
3. Agents & Actors
4. The World & Its State

News media text data
Comparative summarization (CCS) of text

Standards and Practices

... versus...

- To be persuasive we must be believable; to be believable we must be credible; to be credible we must be truthful.
  - Edward R. Murrow

- You supply the photographs, and I’ll supply the war.
  - William Randolph Hearst
The need for automatic text summarization

Improve Journalism Analysis, Improve Journalism, Improve How the World Works

- Holes in current approach
  - Comprehensive approach too time consuming
  - Case study approach too prone to bias

- Machine learning techniques
  - Fast, scale well
  - Reproducible results
  - Designed around predictive tasks

- Harness machine learning to power media studies
  - New predictive framework needed for media study
  - New design guidelines and metrics needed for machine learning
Concise and Comparative Summarization (CCS) on a given topic (Miratrix et al, 2012)

- Make the summarization problem predictive:
  - automatic labeling of documents into “positive” and “negative”
  - by thresholding the count of relevant words on the topic.
  - (e.g. if the topic is china, count words “china, chinese,…”)

- Predictors are counts of word phrases (uni-grams, bi-grams, tri-grams…)
  - what are very high dimension (1 million) and very sparse.

- Epsilon-L2 Boosting for picking a fixed number of word phrases

Our algorithm is a modification of BBR (Genkins et al, 2007) from L1 penalized logistic regression to Lasso for sparse, short and fat matrix text data.
Human comparison experiment

Human experiment findings:

Lasso with L2 column rescaling works better overall for different document sizes (articles vs paragraphs) than tf-idf normalization and stop-word removal and co-occurrence.

tf-idf works well for articles.
Some CCS results from NYT international section in 2009

<table>
<thead>
<tr>
<th>Iraq</th>
<th>Russia</th>
<th>Germany</th>
<th>Mexico</th>
</tr>
</thead>
<tbody>
<tr>
<td>american</td>
<td>a medvedev</td>
<td>angela merkel</td>
<td>and border protection</td>
</tr>
<tr>
<td>and afghanistan</td>
<td>caucasus</td>
<td>berlin</td>
<td>antonio betancourt</td>
</tr>
<tr>
<td>baghdad</td>
<td>europe</td>
<td>chancellor angela</td>
<td>cancun</td>
</tr>
<tr>
<td>brigade</td>
<td>gas</td>
<td>european</td>
<td>chihuahua</td>
</tr>
<tr>
<td>combat</td>
<td>georgia</td>
<td>france and</td>
<td>denise grady</td>
</tr>
<tr>
<td>gen</td>
<td>interfax news agency</td>
<td>frankfurt</td>
<td>drug cartels</td>
</tr>
<tr>
<td>in afghanistan</td>
<td>iran</td>
<td>group of mostly</td>
<td>guadalajara</td>
</tr>
<tr>
<td>invasion</td>
<td>moscow</td>
<td>hamburg</td>
<td>influenza</td>
</tr>
<tr>
<td>nuri</td>
<td>nuclear</td>
<td>marwa alsherbini</td>
<td>oaxaca</td>
</tr>
<tr>
<td>pentagon</td>
<td>president dmitri</td>
<td>matchfixing</td>
<td>outbreak</td>
</tr>
<tr>
<td>saddam</td>
<td>republics</td>
<td>minister karltheodor zu</td>
<td>president felipe</td>
</tr>
<tr>
<td>sergeant</td>
<td>sergi</td>
<td>munich</td>
<td>sinaloa</td>
</tr>
<tr>
<td>sunni</td>
<td>soviet</td>
<td>nazi</td>
<td>swine</td>
</tr>
<tr>
<td>troops</td>
<td>vladimir</td>
<td>world war</td>
<td>texas</td>
</tr>
<tr>
<td>war and who</td>
<td></td>
<td></td>
<td>tijuana</td>
</tr>
</tbody>
</table>

Table 1: Four Sample Summaries of Four Different Countries. The method used, a count rule with a threshold of 2, the Lasso for feature selection, and tf-idf reweighting of features, was one of the best identified for article-unit analysis by our validation experiment.
How does e-L2boosting compare with Lasso?

- Lasso (Least Absolute Sum of Squares Operator):

\[
\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} L(Z_i; \beta) + \lambda \|\beta\|_1 \right\}
\]

where \( \lambda > 0 \) is a regularization parameter.

e-L2Boosting also gives sparse models.

Are they the same?
Striking Similarity (Efron et al, 2004)

The paths are not always the same. (These plots are reverse of Mairal’s path plots that start with lambda=0.)
e-L2Boosting could be very different from Lasso
Lasso trade-off between empirical loss and penalty

If we run coordinate gradient descent with a fixed step size on the Lasso loss, at each iteration $t$:

$$
\beta^t \rightarrow \beta^{t+1} = \beta^t \pm \epsilon 1_j
$$

$$
\Delta \sum_{i=1}^{n} L = \sum_{i=1}^{n} L(Z_i; \beta^t \pm \epsilon 1_j) - \sum_{i=1}^{n} L(Z_i; \beta^t)
$$

Trade-off

$$
\Delta \| \beta \|_1 = \| \beta^{t+1} \|_1 - \| \beta^t \|_1 \equiv \pm \epsilon
$$
Forward and Backward

Two ways one can reduce the Lasso loss:

- Forward Step -- Reduce Empirical Loss (L2 boosting)

\[
\min_{j,s=\pm \epsilon} \sum_{i=1}^{n} L(Z_i; \beta^t + s 1_j)
\]

- Backward Step – Reduce Penalty

\[
\Delta \|\beta\|_1 = -\epsilon
\]

Let’s start with a forward step to get \( \beta^0 \), and use an initial

\[
\lambda^0 = \frac{1}{\epsilon} \left( \sum_{i=1}^{n} L(Z_i; 0) - \sum_{i=1}^{n} L(Z_i; \beta^0) \right)
\]
Blasso or Stagewise Lasso (Zhao & Yu, JMLR, 2004)

- Find the backward direction that leads to the minimal empirical loss
  \[ \hat{j} = \arg \min_{j \in I_A^{t}} \sum_{i = 1}^{n} L(Z_i; \hat{\beta}^t + s_j 1_j) \text{ where } s_j = -\text{sign}(\hat{\beta}^t_j) \varepsilon. \]

- Take the direction if it leads to a decrease in the Lasso loss
  \[ \Delta \sum L + \lambda \Delta \|\beta\|_1 < 0 \]

- Otherwise force a forward step
  \[ (\hat{j}, \hat{s}) = \arg \min_{j, s = \pm \varepsilon} \sum_{i = 1}^{n} L(Z_i; \hat{\beta}^t + s 1_j) \]

- Relax \( \lambda \) whenever necessary
  \[ \lambda^{t+1} = \min[\lambda^t, \frac{1}{\varepsilon} \left( \sum_{i = 1}^{n} L(Z_i; \hat{\beta}^t) - \sum_{i = 1}^{n} L(Z_i; \hat{\beta}^{t+1}) \right)] \]

T. Zhang’s work on backward-forward algorithms: see Mairal’s lecture.
BLasso Converges to Lasso Path as $\varepsilon$ goes to zero (Zhao and Y, 2004)
In what follows, $\mathbf{X}$ is a $\mathbb{R}^{n \times p}$ matrix containing in each of its $n$ rows observations of the zero-mean random vector $\mathbf{X}$ with covariance matrix $\Sigma$. Denote by $\mathbf{X}_j$ the $j$-th entry of $\mathbf{X}$ and by $\mathbf{X}_{J^*}$ the $(p - 1)$ dimensional vector resulting from deleting $\mathbf{X}_j$ from $\mathbf{X}$. For a given $j$, we can permute the order of the variables in $\mathbf{X}$ and partition $\Sigma$ to get:

$$\text{cov} \left( \begin{bmatrix} \mathbf{X}_j \\ \mathbf{X}_{J^*} \end{bmatrix} \right) = \begin{bmatrix} \sigma_{j,j} & \Sigma_{j,J^*} \\ \Sigma_{J^*,j} & \Sigma_{J^*,J^*} \end{bmatrix}$$

where $J^*$ corresponds to the indices in $\mathbf{X}_{J^*}$, so $\sigma_{j,j}$ is a scalar, $\Sigma_{j,J^*}$ is a $p - 1$ dimensional row vector and $\Sigma_{J^*,J^*}$ is a $(p - 1) \times (p - 1)$ square matrix. Inverting this partitioned matrix (see, for instance Hocking, 1996) yield:
Gaussian graphical model

\[
\begin{bmatrix}
\sigma_{j,j} & \Sigma_{j,j^*} \\
\Sigma_{j^*,j} & \Sigma_{j^*,j^*}
\end{bmatrix}^{-1} =
\begin{bmatrix}
\frac{1}{d_j^2} & -\frac{1}{d_j^2} \beta_j \\
M_1 & M_2^{-1}
\end{bmatrix},
\]

where:

\[
\beta_j = \left( \beta_{j,1}, \ldots, \beta_{j,j-1}, \beta_{j,j+1}, \ldots, \beta_{j,p} \right) = -\Sigma_{j,j^*} \Sigma_{j^*,j}^{-1} \in \mathbb{R}^{(p-1)},
\]

\[
d_j = \sqrt{\left( \sigma_{j,j} - \Sigma_{j,j^*} \Sigma_{j^*,j}^{-1} \Sigma_{j^*,j} \right)} \in \mathbb{R}_+,
\]

\[
M_2 = \Sigma_{j^*,j^*} - \Sigma_{j^*,j} \Sigma_{j,j}^{-1} \Sigma_{j^*,j},
\]

\[
M_1 = -M_2^{-1} \cdot \left( \Sigma_{j^*,j^*}^{-1} \Sigma_{j^*,j} \right) = -\frac{1}{d_j^2} \beta_j', \text{ (the second equality due to symmetry)}.
\]
Gaussian graphical model

The parameters $\beta_j$ and $d_j^2$ correspond respectively to the coefficients and the expected value of the squared residuals of the best linear model of $X_j$ based on $X_{J^*}$, irrespectively of the distribution of $X$. In what follows, we will let $\beta_{jk}$ denote the coefficient corresponding to $X_k$ in the linear model of $X_j$ based on $X_{J^*}$. We define:

- **D**: a $p \times p$ diagonal matrix with $d_j$ along its diagonal and,

- **B**: a $p \times p$ matrix with zeros along its diagonal and off-diagonal terms given by $\beta_{jk}$. 
Relating regression coefficients with inverse covariance matrix

Using (1) for \( j = 1, \ldots, p \) yields:

\[
\Sigma^{-1} = D^{-2}(I_p - B)
\]

Since \( \Sigma^{-1} \) is symmetric, (2) implies that the following constraints hold:

\[
d_k^2 \beta_{jk} = d_j^2 \beta_{kj}, \quad \text{for} \quad j, k = 1, \ldots, p.
\]

It follows that the precision matrix has the same off-diagonal sparsity patterns as \( B \).
Gaussian Graphical Model

- Gaussian inverse covariance characterizes conditional independencies (and hence graphical model)

\[ \Theta_{st}^* = 0 \iff X_s \perp X_t \mid X_{V \backslash \{s,t\}} \iff \text{Non-edge } (s, t) \]
L1 penalized \(-\log\) Gaussian Likelihood

Given \(n\) iid observations of \(X\) with

\[
X = (X_1, \ldots, X_p) \in \mathbb{R}^p, \text{ mean zero, } \Sigma^* = EX'X, \Theta^* := \Sigma^*-1
\]

Yuan and Lin (06):

\[
\hat{\Theta} := \arg\max_{\Theta > 0} \left[ -\text{loglik}_{\text{Gaussian}}(\Theta) + \lambda_n \|\Theta\|_{1,\text{off}} \right]
\]

Banerjee et al (08), Friedman et al (07):

fast algorithms based on block descent.

An earlier work was Meinshause and Buhlmann (2006) who estimated the sparsity pattern of the precision matrix via Lasso by regressing one variable at a time against the rest. Rocha et al (2008) integrated the many Lasso problems into one via pseudo-likelihood (SPLICE).
Example: voting clusters via sparse graphical model estimation

Banerjee, El Ghaoui, d’Aspremont (08)
Nuclear norm penalty to induce low-rank in matrix estimation

The nuclear norm of a matrix is the sum of its singular values and it is the same as trace norm for symmetric matrices.

It has been used as a surrogate for rank in the rank minimization problem to complete a matrix (e.g. Netflix) with missing entries and noise.

See lectures by Smola and Mairal and Yu’s V4 modeling lecture where the nuclear norm is used with data.
AdaBoost

L2Boosting is an extension of AdaBoost for binary classification to regression with the L2 loss function. AdaBoost was invented by Freund, Yoav; Schapire, Robert E. (1996). *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*.

It often improved the classification accuracy of a simple classifier such as stumps by 30% or more via repeatedly application of the same classifier to modified data and then a combination the resulting classifiers. The iteration is stopped early by CV or test data.

It was observed that for many data sets even after the training error hit zero, the generalization error kept improving. Is AdaBoost resistant to overfitting?
Recall that L2Boosting is an extension of AdaBoost for binary classification to regression with the L2 loss function. AdaBoost was invented by Freund, Yoav; Schapire, Robert E. (1995). *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*.

It often improved the classification accuracy of a simple classifier such as stumps by 30% or more via repeatedly application of the same classifier to modified data and then a combination the resulting classifiers. The iteration is stopped early by CV or test data.

It was observed that for many data sets even after the training error hit zero, the generalization error kept improving. Is AdaBoost resistant to overfitting?
AdaBoost for a given base (weak) learner with early stopping via test or CV

For data \((x_i, y_i)_{i=1}^{n}\), to obtain a classification function \(F\) given a base learner that is better than random guessing, update as follows:

\[ F_0(x) = 0 \quad f_0(x) \quad \text{is the classifier learned from original data using the base learner.} \]

For \(m=0, 1, \ldots\), let \(err_m\) be the error rate of \(f_m(x)\)

\[ F_{m+1}(x) = F_m(x) + \frac{1}{2} \log \frac{1 - err_m}{err_m} f_m(x) \]

Sample data with weights:

\[ w_{m+1}(i) \propto w_m(i) \exp(\log \frac{1 - err_m}{err_m}) I[y_i \neq f_m(x_i)] \]

Apply the base learner to the sampled data to get \(f_{m+1}(x)\)
AdaBoost was much better understood statistically after the gradient descent point of view of it was developed by Breiman, Manson et al and Friedman et al – all in 2000.

It could be viewed as taking gradient descent on an exponential loss function of the margin yF(x):

$$\exp (-yF(x)),$$

which is an upper bound on the 0-1 loss function of classification. The $-\log$ likelihood function of the logistic model can also been seen as a different upper bound.
Different upper bounds of the 0-1 loss function
(Friedman et al, 2000, ADDITIVE LOGISTIC REGRESSION: A STATISTICAL VIEW OF BOOSTING, Annals of Stats, p. 337-407)
AdaBoost is an ensemble method

Other ensemble methods include bagging, random forests (RF), model averaging, …

Such methods average different models/classifiers to come up with the final model/classifier.

See Hansen’s lecture on CART (decision trees) and RF…
Ridge regression method: L2 penalized LS

Ridge regression is an old method, but still very useful. Tikhonov (1943) used it for the first time and this method is also called Tikhonov regularization.

Ridge estimator is defined to minimize $L^2$ penalized LS:

$$||Y - X\beta||^2 + \lambda||\beta||^2,$$

where $\lambda$ is the tuning parameter to trade off the fidelity to data and stability imposed by the penalty term. The penalty term is very stable because it doesn’t depend on data! When $\lambda$ is large, this term dominates and we have very stable solutions close to $\beta = 0$, however, it does not have much to do with data. By choosing $\lambda$ appropriately, an appropriate amount of Ridge’s stability translates into good statistical properties of the Ridge estimator.
Ridge regression method: L2 penalized LS

It is easy to see that the Ridge estimator takes the form

$$\hat{\beta}^{(r)} = (X'X + \lambda I)^{-1} X'Y.$$ 

In the orthogonal design case, we can work out the ridge estimator as

$$\beta_i^{(r)} = \frac{1}{1+\lambda} X'_j Y,$$

where $X_j$ is the jth column of $X$. It can be shown that the MSE or prediction error of Ridge for some $\lambda > 0$ is better than OLS ($\lambda = 0$).

For fixed $\lambda$, Ridge is also a MAP (maximum a posteriori) estimator with an iid Gaussian prior on $\beta$.

When the predictors are highly correlated, inverting $X'X$ could be unstable, but adding the ridge term $\lambda I$ stabilizes this inversion.
Ridge regression method: L2 penalized LS

Suppose we fit only the constant predictor, or we assume $Y_1, \ldots, Y_n$ are iid $N(\mu, \sigma^2)$.

The OLS is $\bar{Y}$ with a variance or MSE $\sigma^2/n$.

The ridge estimator shrinks the OLS towards zero:

$$\tilde{\beta} = \frac{n}{n + \lambda} \bar{Y} = \alpha \bar{Y},$$

where $\alpha = n/(n + \lambda) \in (0, 1)$. Hence

$$MSE(\tilde{\beta}) = (1 - \alpha)^2 \mu^2 + \alpha^2 \sigma^2.$$ 

The above MSE is smaller than that of the OLS iff

$$SNR = \frac{\mu^2}{\sigma^2} < \frac{1 + \alpha}{1 - \alpha},$$

which can always be made true if we choose appropriately alpha in $(0, 1)$. 

June 24, 2013
Ridge covariance estimation

\[ X'X + \lambda I \] can also be viewed as a regularized covariance estimation. This is the method that we used to reconstruct the movies based on encoding models and fMRI signals.

We also tried sparse graphical method for precision matrix estimation and the ridge method worked better.
Back to reconstructing movies...

Using e-L2boosting or Lasso, for each voxel, we have a good linear prediction rule for fMRI responses corresponding to any movie clips.

What else do we have?

A large movie clip database from movie trailers, YouTube, etc…

An “external” memory to approximately “replace” our internal memory?
Given an observed fMRI vector $Y$ over selected “informative” voxels

Lasso-model

clip 1  \longrightarrow \text{predicted fMRI vector } Y_1

clip 2  \longrightarrow \text{predicted fMRI vector } Y_2

clip 3  \longrightarrow \text{predicted fMRI vector } Y_3

\vdots

Rank clip 1, clip 2, clip 3, … depending on how close $Y_1, \ldots, Y_3$… are to $Y$ in terms of a “weighted LS metric.”
Decoding details

- **Task A**: Identify the stimulus the subject has seen

- **Main strategy**:
  1. Fit encoding models
  2. Choose $r=2700$ best voxels (based on CV)
  3. For each candidate stimulus:
     1. Generate prediction of brain activity
     2. Give score by how well it fits observed activity

**Notation**

- $I_{m_i}, I_{m_k}$ represent images.
- $I_{m_i}, i \in 1...n_t$ training images, $I_{m_k}, k = 1...N$ candidates.
- $Y_i \in \mathbb{R}^r$ (averaged) observed responses to $I_{m_i}$
- $f(I_{m_i}) \in \mathbb{R}^p$ is the vector of features,
- $\beta, (\hat{\beta}) \in \mathbb{R}^{p \times r}$ (fitted) linear coefficients.
Simplest solution – square distance

- In other words, for a fixed response, we want to find an input that gives the prediction through the encoding model that is closest to the response.

- It is natural to use square distance

\[ d_2(\hat{Y}, Y) = \| \hat{Y} - Y \|^2 \]

**Problem:** Best voxels tend to estimate the same thing. The same signal (part of the image) dominates the reconstruction. Need to down-weigh voxel-groups
Gaussian model (second order)

Given an observed fMRI response vector, we postulate a regression model with its linear response function estimated on training data and e-L2Boosting or Lasso:

\[ Y = f(Im)\beta + w, \quad w \sim N(0, \Sigma_w) \]

Then the likelihood function for the kth image in the validation set, given the observed fMRI vector Y, is

\[ \propto \exp[-(Y - f(Im_k)\beta)'\Sigma_w^{-1}(Y - f(Im_k)\beta)] \]

We’d like to pick the image k in validation set that is closest to the observed fMRI signal in a metric defined by the inverse covariance (precision or concentration) matrix.

\[ \hat{k} = \arg \min_{k \leq N} (Y - f(Im_k)\beta)'\Sigma_w^{-1}(Y - f(Im_k)\beta) \]
Estimating $\Sigma_\omega^{-1}$

- Take $\beta = \hat{\beta}$, so $\omega$ represents observed residuals
- $\hat{\Sigma}_\omega$ estimated from residuals (of data set aside)
  \[
  \hat{\Sigma}_\omega = \hat{\text{Cov}}(Y_i - f(Im_i)\hat{\beta})
  \]

- Since $r \geq n$ we need some regularization to invert
  - Pseudo-Inverse

- L1-constraints on $\hat{\Sigma}_\omega^{-1}$ (Banerjee et al 2009) Modeling assumptions can be introduced by L1 Penalty

- Ridge Inverse $(\hat{\Sigma}_\omega + \lambda M)^{-1}$ (Ledoit and Wolf 2003) Modeling assumptions by $M$
Results on a simpler image identification data set from the Gallant Lab

![Image identification rates](chart.png)

- **Test**: (2 Repeats)
  - Corr: 0.67
  - kinv: 0.82
  - Glasso: 0.85
  - Ridge: 0.88

- **Validation**: (13 Repeats)
  - Corr: 0.85
  - kinv: 0.925
  - Glasso: 0.96
  - Ridge: 0.975
Reconstruction of movies
Nishimoto, Vu, Naselaris, Benjamini, Yu, and Gallant (2011)

1. Download N videos/images from the internet to construct an empirical prior (e.g. movie trailers, youtube videos, …)

2. Using residuals from fitted encoding models, we take into account correlation between voxels through a ridge regularized covariance estimate \( \hat{\Sigma}_\omega \)

3. Given an fMRI vector \( Y \) on voxels, for some image feature \( g(Im) \), we can calculate its posterior mean:

\[
\sum_{k \leq N} g(Im_k) \exp\left[ -(Y - f(Im_k)\beta)'\hat{\Sigma}_w^{-1}(Y - f(Im_k)\beta) \right]
\]

\( g \)'s represent red, green, blue values at each pixel, frame. Recover color based on the prior, not part of the model.
Some issues with reconstruction

- Weighing images by prior does not work well:

  the posterior weights are very similar, or not differentiating, possibly because in 26K-dim space, everything is quite far apart.

- Instead, we used equal weights for top 100 images -- the ordering of posterior weights is informative.
Reconstruction algorithm on one slide

A. Record data

B. Fit encoding models (p=26220, n=7200; 20-35K voxels)

B. Sample prior and rank fit based on posterior (using 2000+ chosen voxels)

B. Pick top 100 ranks

B. Average in color space

Averaging in color-scale is quite rough
Analyzing static image data: comparing Lasso and ridge, nonlinear sparse model and power transform for prediction

Episode 1: linear
- pre-selection by correlation
- localization

Fitting details:
- Regularization parameter selected by 5-fold cross validation
- e-L2 boosting applied to all 10,000+ features
  -- L2 boosting is the method of choice in Gallant Lab
- Other methods applied to 500 features pre-selected by correlation
Other methods

\[ \tilde{\beta} = \text{argmin}_b \frac{1}{n} \| Y - X b \|^2 + \frac{1}{2} \alpha \| \Sigma^{k/2} b \|^2 \]

\[ \tilde{\beta} = (\hat{\Sigma} + \alpha \Sigma^k)^{-1} X^T Y / n \]

- k = 1: semi OLS (theoretically better than OLS)
- k = 0: ridge
- k = -1: semi OLS (inverse)

k =1,-1: semi-supervised because of the use of population cov. \( \Sigma \), which is estimated from the whole image data base.
## Validation correlation

<table>
<thead>
<tr>
<th>Voxel</th>
<th>OLS</th>
<th>L2 boost</th>
<th>Semi OLS</th>
<th>Ridge</th>
<th>Semi OLS inverse (k=-1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.434</td>
<td>0.738</td>
<td>0.572</td>
<td>0.775</td>
<td>0.773</td>
</tr>
<tr>
<td>2</td>
<td>0.228</td>
<td>0.731</td>
<td>0.518</td>
<td>0.725</td>
<td>0.731</td>
</tr>
<tr>
<td>3</td>
<td>0.320</td>
<td>0.754</td>
<td>0.607</td>
<td>0.741</td>
<td>0.754</td>
</tr>
<tr>
<td>4</td>
<td>0.303</td>
<td>0.764</td>
<td>0.549</td>
<td>0.742</td>
<td>0.737</td>
</tr>
<tr>
<td>5</td>
<td>0.357</td>
<td>0.742</td>
<td>0.544</td>
<td>0.763</td>
<td>0.763</td>
</tr>
<tr>
<td>6</td>
<td>0.325</td>
<td>0.696</td>
<td>0.520</td>
<td>0.726</td>
<td>0.734</td>
</tr>
<tr>
<td>7</td>
<td>0.427</td>
<td>0.739</td>
<td>0.579</td>
<td>0.728</td>
<td>0.736</td>
</tr>
<tr>
<td>8</td>
<td>0.512</td>
<td>0.741</td>
<td>0.608</td>
<td>0.747</td>
<td>0.743</td>
</tr>
<tr>
<td>9</td>
<td>0.247</td>
<td>0.746</td>
<td>0.571</td>
<td>0.754</td>
<td>0.753</td>
</tr>
<tr>
<td>10</td>
<td>0.458</td>
<td>0.751</td>
<td>0.606</td>
<td>0.761</td>
<td>0.758</td>
</tr>
<tr>
<td>11</td>
<td>0.288</td>
<td>0.778</td>
<td>0.610</td>
<td>0.801</td>
<td>0.801</td>
</tr>
</tbody>
</table>
Comparison of the feature locations

Semi methods

e-L2boost
Our Story (cont)

Episode 2: nonlinear via iV-SpAM (machine learning stage)

- Additive Models (Hastie and Tibshirani, 1990):
  \[ Y_i = \sum_{j=1}^{p} f_j(X_{ij}) + \varepsilon_i, i = 1, \ldots, n \]

- Sparse:
  \[ f_j \equiv 0 \quad \text{for most } j \]

- High dimensional: \( p \gg n \)

SpAM (Sparse Additive Models)

SpAM V1 encoding model (1331 voxels from V1)  
(Ravikumar, Vu, Gallant Lab, BY, NIPS08, AoAS11)

For each voxel,

- Start with 10921+ complex cells (features)
- Pre-selection of 500 complex cells via correlation
- Fitting of SpAM to 500 complex cells with AIC stopping
- Pooling of SpAM fitted complex cells according to location and frequency to form pooled complex cells
- Fitting SpAM to 500 complex cells and pooled complex cells with AIC stopping
Prediction performance ($R^2$)

inset region

median improvement 12%.
Nonlinearities

Compressive effect (finite energy supply)

Common nonlinearity for each voxel?
Power transformation of features

Episode 3: nonlinearity via power transformations (classical stage)

Plot a predictor vs. the response:

The feature is skewed
The response is Gaussian.

After transformation (4’th root):

Get better correlation!
Make it Gaussian

- Histograms of Gabor wavelets, by spatial frequency.

**Transformations**

<table>
<thead>
<tr>
<th>Frequency</th>
<th>None</th>
<th>Square root</th>
<th>4’th root</th>
<th>Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1.png" alt="Graph" /></td>
<td><img src="image2.png" alt="Graph" /></td>
<td><img src="image3.png" alt="Graph" /></td>
<td><img src="image4.png" alt="Graph" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image5.png" alt="Graph" /></td>
<td><img src="image6.png" alt="Graph" /></td>
<td><img src="image7.png" alt="Graph" /></td>
<td><img src="image8.png" alt="Graph" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="image9.png" alt="Graph" /></td>
<td><img src="image10.png" alt="Graph" /></td>
<td><img src="image11.png" alt="Graph" /></td>
<td><img src="image12.png" alt="Graph" /></td>
</tr>
<tr>
<td>8</td>
<td><img src="image13.png" alt="Graph" /></td>
<td><img src="image14.png" alt="Graph" /></td>
<td><img src="image15.png" alt="Graph" /></td>
<td><img src="image16.png" alt="Graph" /></td>
</tr>
<tr>
<td>16</td>
<td><img src="image17.png" alt="Graph" /></td>
<td><img src="image18.png" alt="Graph" /></td>
<td><img src="image19.png" alt="Graph" /></td>
<td><img src="image20.png" alt="Graph" /></td>
</tr>
<tr>
<td>32</td>
<td><img src="image21.png" alt="Graph" /></td>
<td><img src="image22.png" alt="Graph" /></td>
<td><img src="image23.png" alt="Graph" /></td>
<td><img src="image24.png" alt="Graph" /></td>
</tr>
</tbody>
</table>
Gaussianization of features improves prediction

- Gaussianized features used to train (using L2 epsilon-boosting)
  
  - $F_T = F^{1/2}$ if $F$ in levels 2,3 (4 x 4 and 8 x 8)
  
  - $F_T = F^{1/4}$ if $F$ in levels 4,5 (16 x 16 and 32 x 32)