Machine Learning
10-701/15-781, Fall 2012

Linear Regression and Sparsity

Eric Xing

Reading:
© Eric Xing @ CMU, 2006-2012

Lecture 5, September 24, 2012

Now you’ve moved to Pittsburgh!!
And you want to find the most reasonably priced apartment satisfying your needs:
square-ft., # of bedroom, distance to campus …

<table>
<thead>
<tr>
<th>Living area (ft²)</th>
<th># bedroom</th>
<th>Rent ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>230</td>
<td>1</td>
<td>600</td>
</tr>
<tr>
<td>506</td>
<td>2</td>
<td>1000</td>
</tr>
<tr>
<td>433</td>
<td>2</td>
<td>1100</td>
</tr>
<tr>
<td>109</td>
<td>1</td>
<td>500</td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>270</td>
<td>1.5</td>
<td>?</td>
</tr>
</tbody>
</table>

© Eric Xing @ CMU, 2006-2012
The learning problem

- Features:
  - Living area, distance to campus, # bedroom ...
  - Denote as $x = [x^1, x^2, ... x^k]$
- Target:
  - Rent
  - Denoted as $y$
- Training set:

$$
X = \begin{bmatrix}
- x_1 & - & x_1^2 & ... & x_1^k \\
- x_2 & - & x_2^2 & ... & x_2^k \\
... & ... & ... & ... & ...
\end{bmatrix}
$$

$$
Y = \begin{bmatrix}
y_1 \\
y_2 \\
... \\
y_n
\end{bmatrix}
$$

Our goal:

Linear Regression

- Assume that $Y$ (target) is a linear function of $X$ (features):
  - e.g.:
    $$
    \hat{y} = \theta_0 + \theta_1 x^1 + \theta_2 x^2
    $$
  - let's assume a vacuous "feature" $x^0 = 1$ (this is the intercept term, why?), and define the feature vector to be:

$$
\begin{bmatrix}
x^0 \\
x_1 \\
... \\
x_n
\end{bmatrix}
$$

- then we have the following general representation of the linear function:

- Our goal is to pick the optimal $\theta$. How!
  - We seek $\theta$ that minimize the following cost function:

$$
J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (\hat{y}_i(x_i) - y_i)^2
$$
The Least-Mean-Square (LMS) method

- The Cost Function:
  \[ J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (x_i^T \theta - y_i)^2 \]

- Consider a gradient descent algorithm:
  \[ \theta_j^{t+1} = \theta_j^t - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \]

Now we have the following descent rule:

\[ \theta_j^{t+1} = \theta_j^t + \alpha \sum_{i=1}^{n} (y_i - x_i^T \theta^t)x_i^j \]

- For a single training point, we have:
  - This is known as the LMS update rule, or the Widrow-Hoff learning rule
  - This is actually a "stochastic", "coordinate" descent algorithm
  - This can be used as a on-line algorithm
Claim: when the step size $\alpha$ satisfies certain condition, and when certain other technical conditions are satisfied, LMS will converge to an "optimal region".

\[
\theta^{i+1} = \theta^i + \alpha(y_i - \hat{x}_i^T \theta^i)\hat{x}_i
\]

Note that:

\[
\nabla_{\theta} J = \left[ \frac{\partial}{\partial \theta_1} J, \ldots, \frac{\partial}{\partial \theta_k} J \right]^T = -\sum_{i=1}^{n} (y_n - x_n^T \theta) x_n
\]

\[
\theta^{i+1} = \theta^i + \alpha \sum_{i=1}^{n} (y_n - x_n^T \theta^i) x_n
\]

This is as a batch gradient descent algorithm.
The normal equations

- Write the cost function in matrix form:
  \[ J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (x_i^T \theta - y_i)^2 \]
  \[ = \frac{1}{2} (X\theta - \bar{y})^T (X\theta - \bar{y}) \]
  \[ = \frac{1}{2} (\theta^T X^T X \theta - \theta^T X^T \bar{y} - \bar{y}^T X \theta + \bar{y}^T \bar{y}) \]

- To minimize \( J(\theta) \), take derivative and set to zero:
  \[ \nabla_{\theta} J = \frac{1}{2} \left( \nabla \text{tr} (\theta^T X^T X \theta - \theta^T X^T \bar{y} - \bar{y}^T X \theta + \bar{y}^T \bar{y}) \right) \]
  \[ = \frac{1}{2} \left( \nabla \text{tr} (\theta^T X^T X \theta) - \nabla \text{tr} (\theta^T X^T \bar{y}) \right) \]
  \[ = \frac{1}{2} \left( X^T X \theta + X^T y \theta - 2 \bar{y}^T \bar{y} \right) \]
  \[ = X^T X \theta - X^T \bar{y} = 0 \]

\[ \Rightarrow \quad X^T X \theta = X^T \bar{y} \]

Some matrix derivatives

- For \( f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R} \), define:
  \[ \nabla_A f(A) = \begin{bmatrix}
  \frac{\partial}{\partial A_{11}} f & \cdots & \frac{\partial}{\partial A_{1n}} f \\
  \vdots & \ddots & \vdots \\
  \frac{\partial}{\partial A_{m1}} f & \cdots & \frac{\partial}{\partial A_{mn}} f
  \end{bmatrix} \]

- Trace:
  \[ \text{tr} A = \sum_{i=1}^{n} A_{ii}, \quad \text{tr} a = a, \quad \text{tr} ABC = \text{tr} CAB = \text{tr} BCA \]

- Some fact of matrix derivatives (without proof)
  \[ \nabla_A \text{tr} AB = B^T, \quad \nabla_A \text{tr} ABA^T C = CAB + C^T AB^T, \quad \nabla_A [A] = A (A^{-1})^T \]
Comments on the normal equation

- In most situations of practical interest, the number of data points $N$ is larger than the dimensionality $k$ of the input space and the matrix $X$ is of full column rank. If this condition holds, then it is easy to verify that $X^TX$ is necessarily invertible.

- The assumption that $X^TX$ is invertible implies that it is positive definite, thus at the critical point we have found is a minimum.

- What if $X$ has less than full column rank? $\rightarrow$ regularization (later).

Direct and Iterative methods

- Direct methods: we can achieve the solution in a single step by solving the normal equation
  - Using Gaussian elimination or QR decomposition, we converge in a finite number of steps
  - It can be infeasible when data are streaming in in real time, or of very large amount

- Iterative methods: stochastic or steepest gradient
  - Converging in a limiting sense
  - But more attractive in large practical problems
  - Caution is needed for deciding the learning rate $\alpha$
Convergence rate

- **Theorem:** the steepest descent equation algorithm converge to the minimum of the cost characterized by normal equation:

\[
\theta(\infty) = (X^T X)^{-1} X^T y
\]

If

\[
0 < \alpha < 2/\lambda_{\text{max}}[X^T X]
\]

- A formal analysis of LMS need more math-mussels; in practice, one can use a small \(\alpha\), or gradually decrease \(\alpha\).

A Summary:

- **LMS update rule**

\[
\theta_{j+1} = \theta_j + \alpha(y_n - x_n^T \theta) x_{n,j}
\]

- **Pros:** on-line, low per-step cost, fast convergence and perhaps less prone to local optimum
- **Cons:** convergence to optimum not always guaranteed

- **Steepest descent**

\[
\theta_{r+1} = \theta_r + \alpha \sum_{i=1}^{n}(y_n - x_n^T \theta_i) x_i
\]

- **Pros:** easy to implement, conceptually clean, guaranteed convergence
- **Cons:** batch, often slow converging

- **Normal equations**

\[
\theta^* = (X^T X)^{-1} X^T \tilde{y}
\]

- **Pros:** a single-shot algorithm! Easiest to implement.
- **Cons:** need to compute pseudo-inverse \((X^T X)^{-1}\), expensive, numerical issues (e.g., matrix is singular ...), although there are ways to get around this ...
**Geometric Interpretation of LMS**

- The predictions on the training data are:
  \[ \hat{y} = X\theta^* = X(X^TX)^{-1}X^T\bar{y} \]

- Note that
  \[ \hat{y} - \bar{y} = (X(X^TX)^{-1}X^T - I)\bar{y} \]

and
  \[ X^T(\hat{y} - \bar{y}) = X^T(X(X^TX)^{-1}X^T - I)\bar{y} \]
  \[ = (X^T X(X^TX)^{-1}X^T - X^T)\bar{y} \]
  \[ = 0 \]

\( \hat{y} \) is the orthogonal projection of \( \bar{y} \) into the space spanned by the columns of \( X \)

**Probabilistic Interpretation of LMS**

- Let us assume that the target variable and the inputs are related by the equation:
  \[ y_i = \theta^T x_i + \epsilon_i \]

  where \( \epsilon \) is an error term of unmodeled effects or random noise.

- Now assume that \( \epsilon \) follows a Gaussian \( \mathcal{N}(0,\sigma) \), then we have:
  \[ p(y_i | x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(y_i - \theta^T x_i)^2}{2\sigma^2}\right) \]

- By independence assumption:
  \[ L(\theta) = \prod_{i=1}^{n} p(y_i | x_i; \theta) = \left(\frac{1}{\sqrt{2\pi\sigma}}\right)^n \exp\left(-\frac{\sum_{i=1}^{n} (y_i - \theta^T x_i)^2}{2\sigma^2}\right) \]
Probabilistic Interpretation of LMS, cont.

- Hence the log-likelihood is:

\[ l(\theta) = n \log \frac{1}{\sqrt{2\pi\sigma}} - \frac{1}{\sigma^2} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2 \]

- Do you recognize the last term?

Yes it is:

\[ J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (x_i^T \theta - y_i)^2 \]

- Thus under independence assumption, LMS is equivalent to MLE of \( \theta \! \)

Case study: predicting gene expression

The genetic picture

causal SNPs

CGTACTGTACAATT

a univariate phenotype:

i.e., the expression intensity of a gene
### Association Mapping as Regression

<table>
<thead>
<tr>
<th>Phenotype (BMI)</th>
<th>Genotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual 1</td>
<td>2.5</td>
</tr>
<tr>
<td>ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Individual 2</td>
<td>4.8</td>
</tr>
<tr>
<td>ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Individual N</td>
<td>4.7</td>
</tr>
<tr>
<td>ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Benign SNPs

Causal SNP

\[ y_i = \sum_{j=1}^{J} x_{ij} \beta_j \]

SNPs with large \(|\beta_j|\) are relevant

© Eric Xing @ CMU, 2006-2012
Experimental setup

- Asthma dataset
  - 543 individuals, genotyped at 34 SNPs
  - Diploid data was transformed into 0/1 (for homozygotes) or 2 (for heterozygotes)
  - $X=543 \times 34$ matrix
  - $Y=$Phenotype variable (continuous)
- A single phenotype was used for regression

- Implementation details
  - Iterative methods: Batch update and online update implemented.
  - For both methods, step size $\alpha$ is chosen to be a small fixed value ($10^{-6}$). This choice is based on the data used for experiments.
  - Both methods are only run to a maximum of 2000 epochs or until the change in training MSE is less than $10^{-4}$

Convergence Curves

- For the batch method, the training MSE is initially large due to uninformed initialization
- In the online update, $N$ updates for every epoch reduces MSE to a much smaller value.
The Learned Coefficients

Multivariate Regression for Trait Association Analysis

<table>
<thead>
<tr>
<th>Trait</th>
<th>Genotype</th>
<th>Association Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>TGAACCATGAACTA</td>
<td>x</td>
</tr>
</tbody>
</table>

\[ y = X \times \beta \]
Multivariate Regression for Trait Association Analysis

<table>
<thead>
<tr>
<th>Trait</th>
<th>Genotype</th>
<th>Association Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>T GAAC CAT GAAG TA</td>
<td>x</td>
</tr>
</tbody>
</table>

\[ \beta^* = \arg \min_{\beta} (y - X\beta)^T(y - X\beta) \]

Many non-zero associations: Which SNPs are truly significant?

Sparsity

- One common assumption to make **sparsity**.

- **Makes biological sense**: each phenotype is likely to be associated with a small number of SNPs, rather than all the SNPs.

- **Makes statistical sense**: Learning is now feasible in high dimensions with small sample size
Sparsity: In a mathematical sense

- Consider least squares linear regression problem:
  \[ \hat{\beta} = \arg\min_\beta \| Y - X\beta \|^2 \]
  subject to:
  \[ \sum_{j=1}^p \mathbb{I}[|\beta_j| > 0] \leq C \]

- But this is not convex!!! Many local optima, computationally intractable.

L1 Regularization (LASSO)
(Tibshirani, 1996)

- A convex relaxation.
  \[
  \hat{\beta} = \arg\min_\beta \| Y - X\beta \|^2 \quad \hat{\beta} = \arg\min_\beta \| Y - X\beta \|^2 + \lambda \| \beta \|_1
  
  \text{subject to:}
  \sum_{j=1}^p |\beta_j| \leq C
  
  - Still enforces sparsity!\]
Lasso for Reducing False Positives

<table>
<thead>
<tr>
<th>Trait</th>
<th>Genotype</th>
<th>Association Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>TGAACTGAAGTA</td>
<td>x</td>
</tr>
</tbody>
</table>

+ \sum_{\beta} = J_{\beta} \\
\beta^* = \arg \min_{\beta} (y - X\beta)^T(y - X\beta) + \lambda \sum_{\beta} \\

Many zero associations (sparse results), but what if there are multiple related traits?

Ridge Regression vs Lasso

\[
\min_{\beta} (X\beta - Y)^T (X\beta - Y) + \lambda \text{pen}(\beta) = \min_{\beta} J(\beta) + \lambda \text{pen}(\beta)
\]

<table>
<thead>
<tr>
<th>Ridge Regression:</th>
<th>Lasso:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pen(\beta) = |\beta|_2^2</td>
<td>pen(\beta) = |\beta|_1</td>
</tr>
</tbody>
</table>

\beta$s with constant l2 norm (level sets of J(\beta)) \\
\beta$s with constant l1 norm

Lasso (l1 penalty) results in sparse solutions – vector with more zero coordinates
Good for high-dimensional problems – don’t have to store all coordinates!
Bayesian Interpretation

- Treat the distribution parameters \( \theta \) also as a random variable
- The \textit{a posteriori} distribution of \( \theta \) after seeing the data is:

\[
p(\theta | D) = \frac{p(D | \theta) p(\theta)}{p(D)} = \frac{p(D | \theta) p(\theta)}{\int p(D | \theta) p(\theta) d\theta}
\]

This is Bayes Rule

\[
\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}
\]


The prior \( p(.) \) encodes our prior knowledge about the domain

Regularized Least Squares and MAP

What if \((X^T X)\) is not invertible?

\[
\hat{\beta}_{\text{MAP}} = \arg \max_{\beta} \log p\{\{X_i, Y_i\}_{i=1}^{n}|\beta, \sigma^2\} + \log p(\beta)
\]

log likelihood log prior

1) Gaussian Prior

\[
\beta \sim \mathcal{N}(0, \tau^2 I) \quad p(\beta) \propto e^{-\beta^T \beta / 2\tau^2}
\]

\[
\hat{\beta}_{\text{MAP}} = \arg \min_{\beta} \sum_{i=1}^{n} (Y_i - X_i \beta)^2 + \lambda \| \beta \|^2
\]

Closed form: HW

Prior belief that \( \beta \) is Gaussian with zero-mean biases solution to “small” \( \beta \)
**Regularized Least Squares and MAP**

What if \((X'X)\) is not invertible?

\[
\hat{\beta}_{\text{MAP}} = \arg \max_{\beta} \log p(\{(X_i, Y_i)\}_{i=1}^{n}|\beta, \sigma^2) + \log p(\beta)
\]

\[\text{log likelihood} \quad \text{log prior}\]

II) Laplace Prior

\[\beta_i \sim \text{Laplace}(0, t)\]

\[p(\beta) \propto e^{-|\beta|/t}\]

\[\hat{\beta}_{\text{MAP}} = \arg \min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda||\beta||_1\]

Closed form: HW

Prior belief that \(\beta\) is Laplace with zero-mean biases solution to "small" \(\beta\)

**Take home message**

- Gradient descent
  - On-line
  - Batch
- Normal equations
- Geometric interpretation of LMS
- Probabilistic interpretation of LMS, and equivalence of LMS and MLE under certain assumption (what?)
- Sparsity:
  - Approach: ridge vs. lasso regression
  - Interpretation: regularized regression versus Bayesian regression
  - Algorithm: convex optimization (we did not discuss this)
- LR does not mean fitting linear relations, but linear combination or basis functions (that can be non-linear)
- Weighting points by importance versus by fitness
Advanced Material: Beyond basic LR

- LR with non-linear basis functions
- Locally weighted linear regression
- Regression trees and Multilinear Interpolation

We will discuss this in next class after we set the state right!
(if we've got time 😊)

Non-linear functions:
LR with non-linear basis functions

- LR does not mean we can only deal with linear relationships
- We are free to design (non-linear) features under LR

\[ y = \theta_0 + \sum_{j=1}^{\infty} \theta_j \phi(x) = \theta^T \phi(x) \]

where the \( \phi_j(x) \) are fixed basis functions (and we define \( \phi_0(x) = 1 \)).

- Example: polynomial regression:
  \[ \phi(x) := [1, x, x^2, x^3] \]
- We will be concerned with estimating (distributions over) the weights \( \theta \) and choosing the model order \( M \).

Basis functions

- There are many basis functions, e.g.:
  - Polynomial  \( \phi_j(x) = x^{j-1} \)
  - Radial basis functions  \( \phi_j(x) = \exp \left( -\frac{(x - \mu_j)^2}{2\sigma^2} \right) \)
  - Sigmoidal  \( \phi_j(x) = \sigma \left( \frac{x - \mu_j}{s} \right) \)
  - Splines, Fourier, Wavelets, etc
1D and 2D RBFs

- 1D RBF

\[ y^{\text{true}} = \beta_1 \phi_1(x) + \beta_2 \phi_2(x) + \beta_3 \phi_3(x) \]

- After fit:

\[ y^{\text{fit}} = 2\phi_1(x) + 0.05\phi_2(x) + 0.5\phi_3(x) \]

Good and Bad RBFs

- A good 2D RBF

- Two bad 2D RBFs
Overfitting and underfitting

\[ y = \theta_0 + \theta_1 x \]
\[ y = \theta_0 + \theta_1 x + \theta_2 x^2 \]
\[ y = \sum_{j=0}^{5} \theta_j x^j \]

Bias and variance

- We define the bias of a model to be the expected generalization error even if we were to fit it to a very (say, infinitely) large training set.
- By fitting "spurious" patterns in the training set, we might again obtain a model with large generalization error. In this case, we say the model has large variance.
Locally weighted linear regression

- The algorithm:
  Instead of minimizing $J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (x_i^T \theta - y_i)^2$
  now we fit $\theta$ to minimize $J(\theta) = \frac{1}{2} \sum_{i=1}^{n} w_i (x_i^T \theta - y_i)^2$
  Where do $w_i$'s come from?  $w_i = \exp\left(-\frac{(x_i - x)^2}{2\tau^2}\right)$
  - where $x$ is the query point for which we'd like to know its corresponding $y$

→ Essentially we put higher weights on (errors on) training examples that are close to the query point (than those that are further away from the query)

Parametric vs. non-parametric

- Locally weighted linear regression is the second example we are running into of a non-parametric algorithm. (what is the first?)

- The (unweighted) linear regression algorithm that we saw earlier is known as a parametric learning algorithm
  - because it has a fixed, finite number of parameters (the $\theta$), which are fit to the data;
  - Once we’ve fit the $\theta$ and stored them away, we no longer need to keep the training data around to make future predictions.
  - In contrast, to make predictions using locally weighted linear regression, we need to keep the entire training set around.

- The term "non-parametric" (roughly) refers to the fact that the amount of stuff we need to keep in order to represent the hypothesis grows linearly with the size of the training set.
Robust Regression

- The best fit from a quadratic regression
- But this is probably better …

How can we do this?

LOESS-based Robust Regression

- Remember what we do in "locally weighted linear regression"?
  → we "score" each point for its impotence

- Now we score each point according to its "fitness"
Robust regression

- For $k = 1$ to $R$...
  - Let $(x_k, y_k)$ be the $k$th datapoint
  - Let $y_{k}^{est}$ be predicted value of $y_k$
  - Let $w_k$ be a weight for data point $k$ that is large if the data point fits well and small if it fits badly:
    $$w_k = \varphi((y_k - y_{k}^{est})^2)$$
  - Then redo the regression using weighted data points.
  - Repeat whole thing until converged!

Robust regression—probabilistic interpretation

- What regular regression does:
  
  Assume $y_k$ was originally generated using the following recipe:
  
  $$y_k = \theta^T x_k + \mathcal{N}(0, \sigma^2)$$

  Computational task is to find the Maximum Likelihood estimation of $\theta$
Robust regression—probabilistic interpretation

- What LOESS robust regression does:

  Assume $y_k$ was originally generated using the following recipe:

  \[
  \text{with probability } p: \quad y_k = \theta^T x_k + \mathcal{N}(0, \sigma^2)
  \]

  but otherwise

  \[
  y_k \sim \mathcal{N}(\mu, \sigma_{\text{huge}}^2)
  \]

  Computational task is to find the Maximum Likelihood estimates of $\theta$, $p$, $\mu$ and $\sigma_{\text{huge}}$.

- The algorithm you saw with iterative reweighting/refitting does this computation for us. Later you will find that it is an instance of the famous E.M. algorithm.

Regression Tree

- Decision tree for regression

<table>
<thead>
<tr>
<th>Gender</th>
<th>Rich?</th>
<th>Num. Children</th>
<th># travel per yr.</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>No</td>
<td>2</td>
<td>5</td>
<td>38</td>
</tr>
<tr>
<td>M</td>
<td>No</td>
<td>0</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>M</td>
<td>Yes</td>
<td>1</td>
<td>0</td>
<td>72</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

Predicted age=39

Predicted age=36
A conceptual picture

- Assuming regular regression trees, can you sketch a graph of the fitted function $y^*(x)$ over this diagram?

How about this one?

- Multilinear Interpolation
  - We wanted to create a continuous and piecewise linear fit to the data
Take home message

- Gradient descent
  - On-line
  - Batch
- Normal equations
- Geometric interpretation of LMS
- Probabilistic interpretation of LMS, and equivalence of LMS and MLE under certain assumption (what?)
- Sparsity:
  - Approach: ridge vs. lasso regression
  - Interpretation: regularized regression versus Bayesian regression
  - Algorithm: convex optimization (we did not discuss this)
- LR does not mean fitting linear relations, but linear combination or basis functions (that can be non-linear)
- Weighting points by importance versus by fitness