Regression

Mark Craven and David Page
Computer Sciences 760
Spring 2018

www.biostat.wisc.edu/~craven/cs760

Goals for the lecture

you should understand the following concepts

• linear regression
• RMSE, MAE, and R-square
• ridge regression (L2 penalty)
• Lagrange multipliers
• convex functions and sets
• lasso (L1 penalty): least absolute shrinkage and selection operator
• lasso by proximal method (ISTA)
• lasso by coordinate descent
• logistic regression and penalized logistic regression
Linear Regression

- Linear regression assumes that the relation between the expected value of dependent variable Y and the value of independent variable(s) X, is linear.

Ordinary Least Square (OLS)

- For single variable assume the data is given by
  \[ y_i = \alpha + \beta x_i + \varepsilon_i \]
  where \( \varepsilon_i \) are Gaussian noises which are independent and have mean 0 and variance \( \sigma^2 \)
Many assumptions… Some major ones:

• Linear relationship
  – Can partially address by taking square, cube, exponential, square root, or logarithm of x’s or y
  – If modify y, also modifies variance…

• Homoscedasticity (same variance)

• Independence of input features

Other Practicalities

• Might want all features to be distributed as standard normal (Gaussian with mean 0 and standard deviation 1: subtract mean and then divide by standard deviation
  – Simplifies notation, e.g., three slides from now
  – Makes coefficients comparable

• Another option for last sub-bullet: Force values into [0,1] by subtracting Min value and then dividing by Max – Min

• Might pre-compute “interaction terms,” e.g., \( x_i x_j \): new features to capture non-linearities just like \( x^2 \). (So much for third assumption 2 slides ago…)
Ordinary Least Square (OLS)

- Goal: Minimize the objective function:
  \[ \text{error} = \sqrt{\sum_i (h(x_i) - y_i)^2} \text{ or } \sum_i |h(x_i) - y_i| \]

- Solution:
  \[ y = \alpha + \beta x + \epsilon \]

  \[ \hat{\beta} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}, \hat{\alpha} = \bar{y} - \hat{\beta}\bar{x} \]

Some Terminology You May Hear

- Squared error is one loss function

- Loss function is a real valued function associating a cost with an outcome (a prediction and actual value pair)

- Empirical risk is average loss over training data set

- Empirical Risk Minimization (ERM) is a general principle of finding the model in our language with lowest empirical risk
Using Linear Algebra

• As we go to more variables, notation more complex

• Use matrix representation and operations, assume all features standardized (standard normal), and assume an additional constant 1 feature

• Given data matrix $X$ with label vector $Y$

• Find vector of coefficients $\beta$ to minimize:

  • $||X\beta - Y||_2^2$

Multivariate Linear Regression

• Write Matrix $X$ and $Y$ as:

  \[
  X = \begin{pmatrix}
      \\
  \end{pmatrix}
  \quad Y = \begin{pmatrix}
      \\
  \end{pmatrix}
  \]

• Solution:

  \[
  \hat{\beta} = (X^TX)^{-1}X^TY
  \]
Evaluation Metrics for Numeric Prediction

- Root mean squared error (RMSE)
- Mean absolute error (MAE) – average error
- R-square (R-squared)
- Historically all were computed on training data, and possibly adjusted after, but really should cross-validate

R-square(d)

- Formulation 1:
  \[
  R^2 = 1 - \frac{\sum_i (y_i - h(x_i))^2}{\sum_i (y_i - \bar{y})^2}
  \]

- Formulation 2: square of Pearson correlation coefficient r. Recall for x, y:

  \[
  r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}
  \]
Some Observations

• R-square of 0 means you have no model, R-square of 1 implies perfect model (loosely, explains all variation)

• These two formulations agree when performed on the training set

• The do not agree when we do cross-validation, in general, because mean of training set is different from mean of each fold

• Should do CV and use first formulation, but can be negative!

Great things about OLS regression

• Closed-form solution: fast!!

• Works well when given a small number of carefully chosen variables (say < 50)

• Works well even if some assumptions not fully satisfied

• Models are understandable

• Method is understood by non-stats/ML folks
Estimation of the Warfarin Dose with Clinical and Pharmacogenetic Data

International Warfarin Pharmacogenetics Consortium (IWPC)

New England Journal of Medicine, February 19, 2009, vol. 360, no. 8


Initial dosing (warfarin package insert)

“The dosing of COUMADIN must be individualized according to patient’s sensitivity to the drug as indicated by the PT/INR..... It is recommended that COUMADIN therapy be initiated with a dose of 2 to 5 mg per day with dosage adjustments based on the results of PT/INR determinations. The lower initiation doses should be considered for patients with certain genetic variations in CYP2C9 and VKORC1 enzymes as well as for elderly and/or debilitated patients....”

Age, height and weight

Weekly dose by age
Statistical Analysis

Derivation Cohort
- 4,043 patients with a stable dose of warfarin and target INR of 2-3 mg/week
- Used for developing dose prediction models

Validation Cohort
- 1,009 patients (20% of dataset)
- Used for testing final selected model

Analysis group did not have access to validation set until after the final model was selected
Numerical modeling methods used

- Support vector regression
- Regression trees
- Model trees
- Multivariate adaptive regression splines
- Least-angle regression
- Lasso
- Logarithmic and square-root transformations
- Direct prediction of dose

Least-squares linear regression modeling method was best according to criterion yielding the lowest mean absolute error

- Predicted the square root of the dose
- Incorporated both genetic and clinical data

IWPC pharmacogenetic dosing algorithm

**The output of this algorithm must be squared to compute weekly dose in mg**

^All references to VKORC1 refer to genotype for rs9923231

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6044</td>
<td></td>
</tr>
<tr>
<td>-0.2614</td>
<td>Age in decades</td>
</tr>
<tr>
<td>0.0087</td>
<td>Height in cm</td>
</tr>
<tr>
<td>0.0128</td>
<td>Weight in kg</td>
</tr>
<tr>
<td>-0.8677</td>
<td>VKORC1 A/G</td>
</tr>
<tr>
<td>-1.6974</td>
<td>VKORC1 A/A</td>
</tr>
<tr>
<td>-0.4854</td>
<td>VKORC1 genotype unknown</td>
</tr>
<tr>
<td>-0.5211</td>
<td>CYP2C9 *1/*2</td>
</tr>
<tr>
<td>-0.9357</td>
<td>CYP2C9 *1/*3</td>
</tr>
<tr>
<td>-1.0616</td>
<td>CYP2C9 *2/*2</td>
</tr>
<tr>
<td>-1.9206</td>
<td>CYP2C9 *2/*3</td>
</tr>
<tr>
<td>-2.3312</td>
<td>CYP2C9 *3/*3</td>
</tr>
<tr>
<td>-0.2188</td>
<td>CYP2C9 genotype unknown</td>
</tr>
<tr>
<td>-0.1092</td>
<td>Asian race</td>
</tr>
<tr>
<td>-0.2760</td>
<td>Black or African American</td>
</tr>
<tr>
<td>-0.1032</td>
<td>Missing or Mixed race</td>
</tr>
<tr>
<td>1.1816</td>
<td>Enzyme inducer status</td>
</tr>
<tr>
<td>-0.5503</td>
<td>Amiodarone status</td>
</tr>
</tbody>
</table>

= Square root of weekly warfarin dose **
IWPC clinical dosing algorithm

**The output of this algorithm must be squared to compute weekly dose in mg**

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0376</td>
<td></td>
</tr>
<tr>
<td>-0.2546</td>
<td>Age in decades</td>
</tr>
<tr>
<td>+0.0118</td>
<td>Height in cm</td>
</tr>
<tr>
<td>+0.0134</td>
<td>Weight in kg</td>
</tr>
<tr>
<td>-0.6752</td>
<td>Asian race</td>
</tr>
<tr>
<td>+0.4060</td>
<td>Black or African American</td>
</tr>
<tr>
<td>+0.0443</td>
<td>Missing or Mixed race</td>
</tr>
<tr>
<td>+1.2790</td>
<td>Enzyme inducer status</td>
</tr>
<tr>
<td>-0.3693</td>
<td>Amiodarone status</td>
</tr>
<tr>
<td></td>
<td>Square root of weekly warfarin dose**</td>
</tr>
</tbody>
</table>

Model comparisons

![Mean Absolute Error by Model](image)
Regularized Regression

- Regression is prone to overfitting, especially when:
  - there are a large number of features or
  - It’s fit with high order polynomial features

- Regularization helps combat overfitting by having a simpler model. It is used when we want to have:
  - less variation in the different weights or
  - smaller weights overall or
  - only a few non-zero weights (and thus features kept)

- Regularization is accomplished by adding a penalty term to the target function that is being optimized

- Two widely-used types – L₂ and L₁ regularization.

\[ L_2 \text{ regularization in linear regression} \]

- What if hundreds or thousands of variables?

- Big risk of overfitting

- Force simpler model, often defined as smaller and “more regular” (less varying) coefficients... small Euclidean norm

- Like limiting maximum decision tree size or depth

- \[ \arg\min \|X\beta - Y\|_2^2 \text{ such that } \|\beta\|_2 < s \]

- Constrained optimization problem... can’t just set derivative (gradient) with respect to \(\beta\) to 0 and solve as did for OLS
Lagrange Multipliers

To maximize \( f(\mathbf{x}) \) such that \( g(\mathbf{x}) < s \)
instead maximize: \( f(\mathbf{x}) + \lambda (g(\mathbf{x}) - s) \)
- \( \lambda \) is Lagrange multiplier
- Resulting optimization task is unconstrained

- To find \( \mathbf{\beta} \) to minimize \( ||\mathbf{X}\mathbf{\beta} - \mathbf{Y}||_2^2 \) s.t. \( ||\mathbf{\beta}||_2 < s \):
- find \( \mathbf{\beta} \) to minimize \( ||\mathbf{X}\mathbf{\beta} - \mathbf{Y}||_2^2 + \lambda (||\mathbf{\beta}||_2 - s) \)
- In practice since we tune hyperparameter \( \lambda \),
s doesn’t matter, so problem becomes:
find \( \mathbf{\beta} \) to minimize \( ||\mathbf{X}\mathbf{\beta} - \mathbf{Y}||_2^2 + \lambda ||\mathbf{\beta}||_2 \)

\( L_2 \) regularization in linear regression

- Called “ridge regression”
- Still has a closed-form solution, so even though continuous
differentiable and convex, don’t need gradient descent
- Setting gradient with respect to \( \mathbf{\beta} \), from previous slide, to
0 and solving we get:

\[ \mathbf{\beta} = (\mathbf{X}^\top \mathbf{X} - \lambda I)^{-1} \mathbf{X}^\top \mathbf{Y} \]
Simple Lagrange Multipliers Example  
(Thanks Wikipedia!) 

Minimize $f(x, y) = x + y$ such that $x^2 + y^2 = 1$  

Note that constraint is: $g(x, y) = x^2 + y^2 - 1$ 

$$L(x, y, \lambda) = f(x, y) + \lambda \cdot g(x, y)$$
$$= x + y + \lambda(x^2 + y^2 - 1).$$

Now we can calculate the gradient:

$$\nabla_{x,y,\lambda} L(x, y, \lambda) = \left( \frac{\partial L}{\partial x}, \frac{\partial L}{\partial y}, \frac{\partial L}{\partial \lambda} \right)$$
$$= \left(1 + 2\lambda x, 1 + 2\lambda y, x^2 + y^2 - 1\right)$$

$$\nabla_{x,y,\lambda} L(x, y, \lambda) = 0 \iff \begin{cases} 
1 + 2\lambda x = 0 \\
1 + 2\lambda y = 0 \\
x^2 + y^2 - 1 = 0 
\end{cases}$$

Read as AND

The first two equations yield

$$x = y = -\frac{1}{2\lambda}, \quad \lambda \neq 0.$$ 

By substituting into the last equation we have:

$$\frac{1}{4\lambda^2} + \frac{1}{4\lambda^2} - 1 = 0,$$

so

$$\lambda = \pm \frac{1}{\sqrt{2}}.$$

Can work out that the constrained maximum is $\sqrt{2}$

which implies that the stationary points of $L$ are

$$\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, -\frac{1}{\sqrt{2}}\right), \quad \left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}, \frac{1}{\sqrt{2}}\right).$$
Logistic Regression: Motivation

- Linear regression was used to fit a linear model to the feature space in order to predict continuous response.
- Suppose response is binary; predict positive if linear function exceeds some value: step function.
- But also want to produce a probability that a feature will take a particular value given other features.

\[ P(Y = 1 \mid X) \]

- So, extend linear regression for classification; no closed-form solution anymore, so need to do gradient descent.

Logistic (Sigmoid) Function

- To exhibit the relation between a dependent and an independent variable, we could use a step function. But it is not differentiable.
- We need a continuous and differentiable function: **Logistic function**

\[ y = \frac{1}{1 + e^{-cx}} \]

\[ = \frac{1}{1 + e^{-wx}} \]

\( wx \rightarrow \) a linear function of the feature vector \( x \)
The Algorithmic Approach

- Instead of squared error, want to minimize probability (according to model) of incorrect class

- So error $E$ is $1 –$ probability of correct class

- Probability of data according to model is likelihood of model; probability of correct class is *conditional likelihood* (more on likelihood in Bayes nets)

- No closed form solution for $w$; we will have to rely on *gradient descent* to minimize $E$ (more in neural nets)

Gradient descent in weight space

gradient descent is an iterative process aimed at finding a minimum in the error surface

on each iteration
- current weights define a point in this space
- find direction in which error surface descends most steeply
- take a step (i.e. update weights) in that direction
Gradient descent in weight space

Calculate the gradient of $E$: 
\[ \nabla E(w) = \left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right] \]

take a step in the opposite direction 
\[ \Delta w = -\eta \nabla E(w) \]
\[ \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \]

Logistic Regression Algorithm

- The conditional log likelihood is given by 
  \[ l(w) = \ln(\prod_j P(y_j | x_{j(i)}, w)) \text{, where } j \rightarrow j^{th} \text{ sample} \]
- Need to find 'w' that maximizes the conditional log likelihood 
  \[ \arg \max_w \ln(\prod_j P(y_j | x_{j(i)}, w)) \]
- Can use gradient ascent 
  \[ w_i^{\text{new}} = w_i + \eta \frac{\partial l(w)}{\partial w_i} \text{, where } \eta \rightarrow \text{learning rate parameter, } i \rightarrow i^{th} \text{ feature} \]
- The derivative comes out to: 
  \[ \frac{\partial l(w)}{\partial w_i} = \sum_j x_{i(j)} \left( y_{i(j)} - P(y_{i(j)} = 1 | x_{j(i)}, w) \right) \]
- This gives us the gradient ascent rule: 
  \[ w_i^{\text{new}} = w_i + \eta \sum_j x_{i(j)} \left( y_{i(j)} - P(y_{i(j)} = 1 | x_{j(i)}, w) \right) \]
More on Gradient Descent

• Gradient descent yields an optimal solution if the minimization problem is convex

• Can compute gradient at once over all examples (batch) or compute from one example at a time (stochastic gradient descent, where stochastic part is next example randomly chosen)

Convexity (from Bubeck, 2015)

Definition 1.1 (Convex sets and convex functions). A set $\mathcal{X} \subset \mathbb{R}^n$ is said to be convex if it contains all of its segments, that is
\[
\forall (x, y, \gamma) \in \mathcal{X} \times \mathcal{X} \times [0, 1], \quad (1 - \gamma)x + \gamma y \in \mathcal{X}.
\]
A function $f: \mathcal{X} \to \mathbb{R}$ is said to be convex if it always lies below its chords, that is
\[
\forall (x, y, \gamma) \in \mathcal{X} \times \mathcal{X} \times [0, 1], \quad f((1 - \gamma)x + \gamma y) \leq (1 - \gamma)f(x) + \gamma f(y).
\]

We are interested in algorithms that take as input a convex set $\mathcal{X}$ and a convex function $f$ and output an approximate minimum of $f$ over $\mathcal{X}$. We write compactly the problem of finding the minimum of $f$ over $\mathcal{X}$ as
\[
\min f(x) \\
\text{s.t. } x \in \mathcal{X}.
\]
We introduce now the key notion of subgradients.

**Definition 1.2 (Subgradients).** Let $\mathcal{X} \subseteq \mathbb{R}^n$, and $f : \mathcal{X} \rightarrow \mathbb{R}$. Then $g \in \mathbb{R}^n$ is a subgradient of $f$ at $x \in \mathcal{X}$ if for any $y \in \mathcal{X}$ one has
\[
f(x) \neq f(y) \quad \Leftrightarrow \quad g \in \partial f(x).
\]
The set of subgradients of $f$ at $x$ is denoted $\partial f(x)$.

To put it differently, for any $x \in \mathcal{X}$ and $g \in \partial f(x)$, $f$ is above the linear function $y \mapsto f(x) + g \cdot (y - x)$.

The next result shows (essentially) that a convex function always admits subgradients.

**Proposition 1.1 (Existence of subgradients).** Let $\mathcal{X} \subseteq \mathbb{R}^n$ be convex, and $f : \mathcal{X} \rightarrow \mathbb{R}$. If $x \in \mathcal{X}$, $\partial f(x)$ is non-empty. Conversely, if $f$ is convex then for any $x \in \text{int}(\mathcal{X})$, $\partial f(x)$ is non-empty.

Furthermore, if $f$ is convex and differentiable at $x$, then $\nabla f(x) \in \partial f(x)$.

Before going to the proof we recall the definition of the epigraph of a function $f$:
\[
\text{epi}(f) = \{(x, t) \in \mathcal{X} \times \mathbb{R} : t \geq f(x)\}.
\]
It is obvious that a function is convex if and only if its epigraph is a convex set.

Epigraph (Bubeck, 2015)
• Show for all real $a < b$ and $0 \leq c \leq 1$,
  $f(ca + (1-c)b) \leq c f(a) + (1-c) f(b)$ for following:
  
  • $f(x) = |x|$
  • $f(x) = x^2$
  • Not so for $f(x) = x^3$

• In general $x$ could be a vector $x$

• For gradient descent, also want $f(x)$ to be continuous differentiable

• For $|x|$ we need proximal methods, subgradient methods, or coordinate descent

---

**Comments on basic logistic regression**

• Logistic Regression is a linear classifier

• Logistic Regression optimized by using conditional likelihood
  • no closed-form solution
  • Error function is continuous differentiable – can always compute gradient
  • convex -> find global optimum with gradient ascent
L1 regularization

• L1 regularization uses 1-norm of the weight vector in the penalty term as shown:
  \[ \lambda ||w||_1 \]
  
i.e., \[ \lambda \sum_i |w_i| \] where ‘i’ represents the \( i \)th feature
• Also called ‘Lasso’ penalty.
• Gradient ascent is no longer feasible since L1 norm is not differentiable.

LASSO: Penalty as a Constraint

Add penalty as a constraint to OBJ function:

Find \( \hat{\alpha} \) and \( \hat{\beta} \)

To minimize \( \sum_i (h(x_i) - y_i)^2 \)

Such that \( \sum_j |\hat{\beta}_j|^{\text{error}} \leq s \) \( (s \text{ is a constant}) \)
LASSO: Penalty as a Term in OBJ

Add penalty as a term to OBJ function to be minimized:

\[
\sum_i (h(x_i) - y_i)^2 + \lambda \sum_j |\hat{\beta}_j|
\]

(\(\lambda\) is penalty parameter)

Obtained by taking Langrangian. Even for linear regression, no closed-form solution. Ordinary gradient ascent also does not work because no derivative. Fastest methods now FISTA and (faster) coordinate descent.

Proximal Methods

- \(f(x) = g(x) + h(x)\)
  - \(g\) is convex, differentiable
  - \(h\) is convex and decomposable, but not differentiable
  - Example: \(g\) is squared error, \(h\) is lasso penalty – sum of absolute value terms, one per coefficient (so one per feature)
  - Find \(\beta\) to minimize \(\|X\beta - Y\|_2^2 + \lambda \|\beta\|_1\)

Proximal Operator: Soft-Thresholding

\[ S_\lambda(x) = \begin{cases} 
  x_i - \lambda & \text{if } x_i > \lambda \\
  0 & \text{if } -\lambda \leq x_i \leq \lambda \\
  x_i + \lambda & \text{if } x_i < -\lambda 
\end{cases} \]

for all \( i \)

We typically apply this to coefficient vector \( \beta \).

Iterative Shrinkage-Thresholding Algorithm (ISTA)

- Initialize \( \hat{\beta} \); let \( \eta \) be learning rate
- Repeat until convergence
  - Make a gradient step of:
    \[ \beta \leftarrow S_\lambda(\beta - \eta X^T(X\beta - y)) \]
Coordinate Descent

- *Fastest* current method for lasso-penalized linear or logistic regression
- *Simple* idea: adjust one feature at a time, and special-case it near 0 where gradient not defined (where absolute value’s effect changes)
- Can take features in a cycle in any order, or randomly pick next feature (analogous to Gibbs Sampling)
- To “special-case it near 0” just apply soft-thresholding everywhere

Coordinate Descent Algorithm

- Initialize coefficients
- Cycle over features until convergence:
  - For each example $i$ and feature $j$, compute “partial residual”:
    \[ r_{ij} = y_i - \sum_{k \neq j} x_{ik} \beta_k \]
  - Compute least-squares coefficients of these residuals (as we did in OLS regression):
    \[ \beta_j^* = \frac{1}{N} \sum_{i=1}^{N} x_{ij} r_{ij} \]
  - Update $\beta_j$ by soft-thresholding, where for any term $T$, “$T_+$” denotes $\min(0,A)$:
    \[ \beta_j \leftarrow S_\lambda(\beta_j^*) \]
Comments on penalized regression

- L2-penalized regression also called “ridge regression”
- Can combine L1 and L2 penalties: “elastic net”
- L1-penalized regression is especially active area of research
  - group lasso
  - fused lasso
  - others

\[ L_2 \text{ regularization in logistic regression} \]

- \( L_2 \) regularization uses 2-norm of the weight vector in the penalty term as shown:
  \[ \lambda \| w \|_2^2 \]
  i.e., \( \lambda \sum_i w_i^2 \) where ‘i’ represents the \( i^{th} \) feature
- ‘\( \lambda \)’ is the regularization parameter used to control the weights. It governs how big the penalty is relative to fitting the data well.
- When the penalty term is added to the objective function, the gradient ascent algorithm’s update changes to:
  \[ w_i^{\text{new}} := w_i + \eta \sum_i x_{i(j)} \left( y_{(j)} - P \left( y_{(j)} = 1 \mid x_{(j)} w \right) \right) - \eta \lambda w_i \]
  where, ‘\( \eta \)’ is the learning rate parameter
More comments on regularization

- Linear and logistic regression prone to overfitting

- Regularization helps combat overfitting by adding a penalty term to the target function being optimized

- L1 regularization often preferred since it produces sparse models. It can drive certain co-efficients(weights) to zero, performing feature selection in effect

- L2 regularization drives towards smaller and simpler weight vectors but cannot perform feature selection like L1 regularization

- Few uses of OLS these days... e.g., Warfarin Dosing (NEJM 2009)... just 30 carefully hand-selected features