Simple Variable Selection LASSO: Sparse Regression

Machine Learning – CSE546
Carlos Guestrin
University of Washington

October 7, 2013

Sparsity

- Vector **w** is sparse, if many entries are zero:
- Very useful for many tasks, e.g.,
  - **Efficiency**: If size(**w**) = 100B, each prediction is expensive:
    - If part of an online system, too slow
    - If **w** is sparse, prediction computation only depends on number of non-zeros
  - **Interpretability**: What are the relevant dimension to make a prediction?
    - E.g., what are the parts of the brain associated with particular words?
- But computationally intractable to perform "all subsets" regression

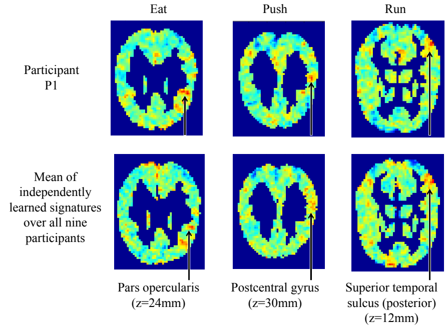


Figure from Tom Mitchell

Simple greedy model selection algorithm

- Pick a dictionary of features
  - e.g., polynomials for linear regression
- Greedy heuristic:
  - Start from empty (or simple) set of features $F_0 = \varnothing$
  - Run learning algorithm for current set of features F_t
    - Obtain h_t
  - Select **next best feature** $\mathbf{X_i^*}$
    - e.g., X_j that results in lowest training error learner when learning with $F_t + \{X_j\}$
  - $F_{t+1} \leftarrow F_t + \{X_i^*\}$
  - Recurse

Greedy model selection

- Applicable in many settings:
  - Linear regression: Selecting basis functions
  - Naïve Bayes: Selecting (independent) features $P(X_i|Y)$
  - Logistic regression: Selecting features (basis functions)
  - Decision trees: Selecting leaves to expand
- Only a heuristic!
  - But, sometimes you can prove something cool about it
    - e.g., [Krause & Guestrin '05]: Near-optimal in some settings that include Naïve Bayes
- There are many more elaborate methods out there

When do we stop???

- Greedy heuristic:
  - …
  - Select **next best feature X_i^***
    - e.g., X_j that results in lowest training error learner when learning with $F_t + \{X_j\}$
  - $F_{t+1} \leftarrow F_t + \{X_i^*\}$
  - Recurse

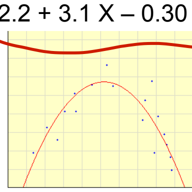
When do you stop???

- When training error is low enough?
- When test set error is low enough?
-

Regularization in Linear Regression

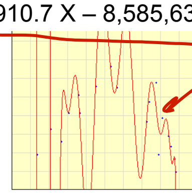
- Overfitting usually leads to very large parameter choices, e.g.:

$-2.2 + 3.1\,X - 0.30\,X^2$



$-1.1 + 4{,}700{,}910.7\,X - 8{,}585{,}638.4\,X^2 + \ldots$

penalty for large weights



overfitting

- ***Regularized*** or ***penalized*** regression aims to impose a "complexity" penalty by penalizing large weights
  - "Shrinkage" method

L_2 regularization → penalizes towards smoother functions

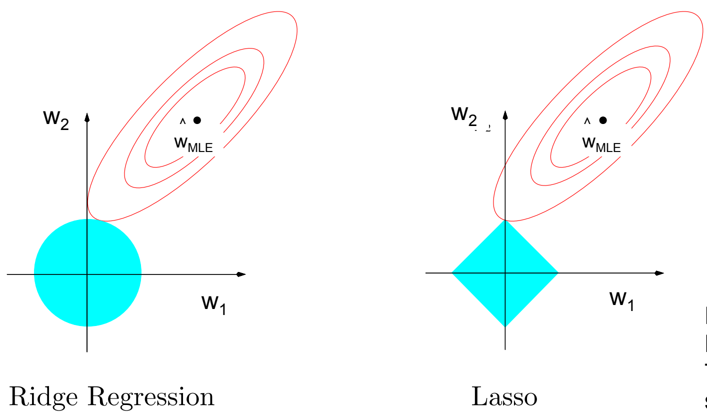
Variable Selection by Regularization

- Ridge regression: Penalizes large weights
- What if we want to perform "feature selection"?
  - E.g., Which regions of the brain are important for word prediction?
  - Can't simply choose features with largest coefficients in ridge solution
- Try new penalty: Penalize non-zero weights
  - Regularization penalty:
  - Leads to sparse solutions
  - Just like ridge regression, solution is indexed by a continuous param λ
  - This simple approach has changed statistics, machine learning & electrical engineering

LASSO Regression

- **LASSO:** least absolute shrinkage and selection operator
- New objective:

Geometric Intuition for Sparsity



From Rob Tibshirani slides

Optimizing the LASSO Objective

- LASSO solution:

$$\hat{\mathbf{w}}_{LASSO} = \arg\min_{w} \sum_{j=1}^{N} \left(t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

Coordinate Descent

- Given a function F
  - Want to find minimum
- Often, hard to find minimum for all coordinates, but easy for one coordinate
- Coordinate descent:
- How do we pick next coordinate?
- Super useful approach for *many* problems
  - Converges to optimum in some cases, such as LASSO

Optimizing LASSO Objective One Coordinate at a Time

$$\sum_{j=1}^{N}\left(t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j))\right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

- Taking the derivative:
  - Residual sum of squares (RSS):

$$\frac{\partial}{\partial w_\ell} RSS(\mathbf{w}) = -2 \sum_{j=1}^{N} h_\ell(x_j) \left(t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j))\right)$$

  - Penalty term:

Subgradients of Convex Functions

- Gradients lower bound convex functions:

- Gradients are unique at **w** iff function differentiable at **w**

- Subgradients: Generalize gradients to non-differentiable points:
  - Any plane that lower bounds function:

Taking the Subgradient

$$\sum_{j=1}^{N}\left(t(x_j)-(w_0+\sum_{i=1}^{k}w_ih_i(x_j))\right)^2+\lambda\sum_{i=1}^{k}|w_i|$$

- Gradient of RSS term:

$$\frac{\partial}{\partial w_\ell}RSS(\mathbf{w}) = a_\ell w_\ell - c_\ell$$

$$a_\ell = 2\sum_{j=1}^{N}(h_\ell(\mathbf{x}_j))^2$$

$$c_\ell = 2\sum_{j=1}^{N}h_\ell(\mathbf{x}_j)\left(t(\mathbf{x}_j)-(w_0+\sum_{i\neq\ell}w_ih_i(\mathbf{x}_j))\right)$$

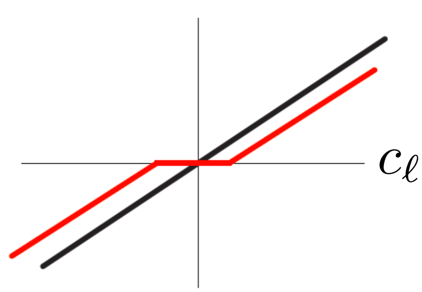
  - If no penalty:
- Subgradient of full objective:

Setting Subgradient to 0

$$\partial_{w_\ell} F(\mathbf{w}) = \begin{cases} a_\ell w_\ell - c_\ell - \lambda & w_\ell < 0 \\ [-c_\ell - \lambda, -c_\ell + \lambda] & w_\ell = 0 \\ a_\ell w_\ell - c_\ell + \lambda & w_\ell > 0 \end{cases}$$

Soft Thresholding

$$\hat{w}_\ell = \begin{cases} (c_\ell + \lambda)/a_\ell & c_\ell < -\lambda \\ 0 & c_\ell \in [-\lambda, \lambda] \\ (c_\ell - \lambda)/a_\ell & c_\ell > \lambda \end{cases}$$

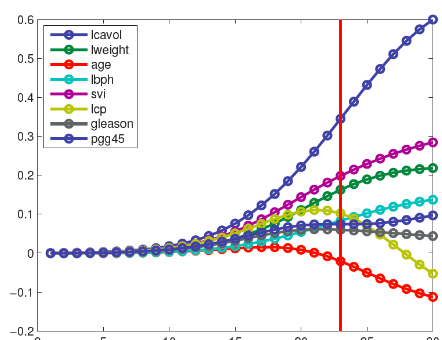


From
Kevin Murphy
textbook

Coordinate Descent for LASSO (aka Shooting Algorithm)

- Repeat until convergence
  - Pick a coordinate *l* at (random or sequentially)
    - Set:
$$\hat{w}_\ell = \begin{cases} (c_\ell + \lambda)/a_\ell & c_\ell < -\lambda \\ 0 & c_\ell \in [-\lambda, \lambda] \\ (c_\ell - \lambda)/a_\ell & c_\ell > \lambda \end{cases}$$
    - Where:
$$a_\ell = 2\sum_{j=1}^{N}(h_\ell(\mathbf{x}_j))^2$$
$$c_\ell = 2\sum_{j=1}^{N} h_\ell(\mathbf{x}_j)\left(t(\mathbf{x}_j) - (w_0 + \sum_{i\neq\ell} w_i h_i(\mathbf{x}_j))\right)$$
  - For convergence rates, see Shalev-Shwartz and Tewari 2009
- Other common technique = LARS
  - Least angle regression and shrinkage, Efron et al. 2004

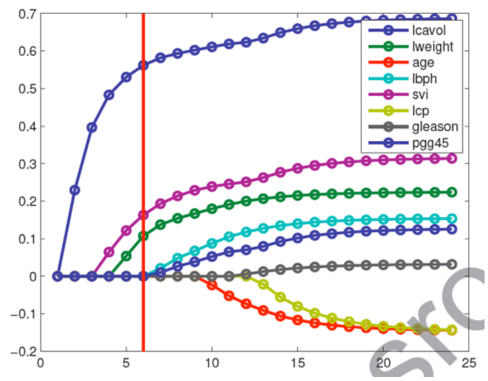
Recall: *Ridge Coefficient Path*



From
Kevin Murphy
textbook

- Typical approach: select λ using cross validation

Now: *LASSO Coefficient Path*



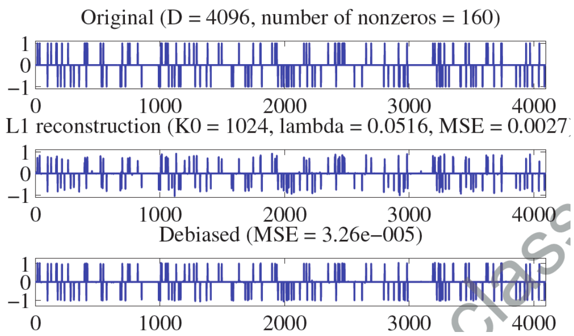
From
Kevin Murphy
textbook

LASSO Example

Term	Least Squares	Ridge	Lasso
Intercept	2.465	2.452	2.468
lcavol	0.680	0.420	0.533
lweight	0.263	0.238	0.169
age	−0.141	−0.046	
lbph	0.210	0.162	0.002
svi	0.305	0.227	0.094
lcp	−0.288	0.000	
gleason	−0.021	0.040	
pgg45	0.267	0.133	

From
Rob
Tibshirani
slides

Debiasing



From Kevin Murphy textbook

What you need to know

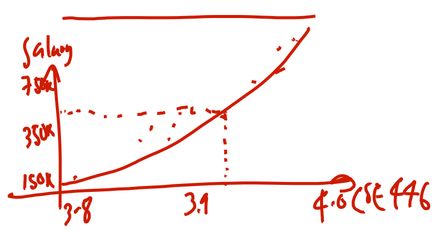
- Variable Selection: find a sparse solution to learning problem
- L_1 regularization is one way to do variable selection
  - Applies beyond regressions
  - Hundreds of other approaches out there
- LASSO objective non-differentiable, but convex ➔ Use subgradient
- No closed-form solution for minimization ➔ Use coordinate descent
- Shooting algorithm is very simple approach for solving LASSO

Classification
Logistic Regression

Machine Learning – CSE546

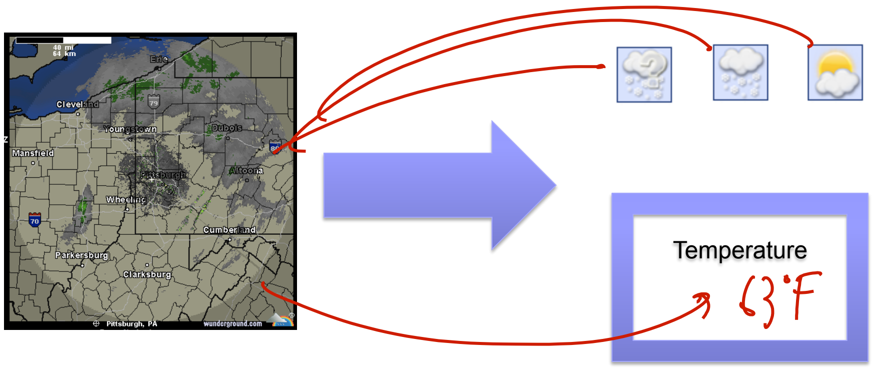
Carlos Guestrin
University of Washington

October 7, 2013



THUS FAR, REGRESSION: PREDICT A CONTINUOUS VALUE GIVEN SOME INPUTS

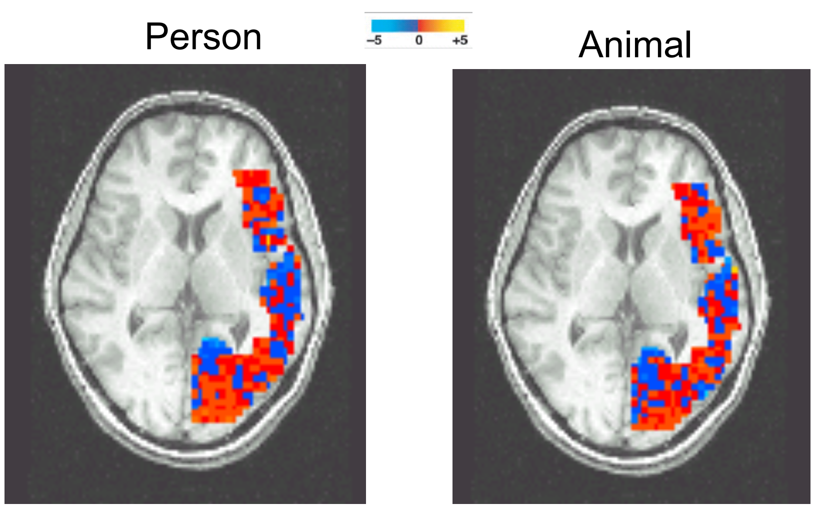
Weather prediction revisted



Reading Your Brain, Simple Example

[Mitchell et al.]

Pairwise classification accuracy: 85%



Classification

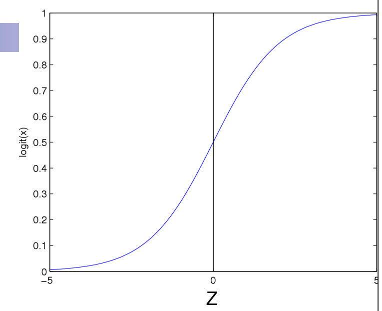
- **Learn**: h:**X** $\mapsto$ Y
  - **X** – features
  - Y – target classes
- Conditional probability: P(Y|**X**)
- Suppose you know P(Y|**X**) exactly, how should you classify?
  - Bayes optimal classifier:
- **How do we estimate P(Y|X)?**

Link Functions

- Estimating P(Y|**X**): Why not use standard linear regression?
- Combing regression and probability?
  - Need a mapping from real values to [0,1]
  - A link function!

Logistic Regression

Logistic function (or Sigmoid): $\frac{1}{1+exp(-z)}$



- Learn P(Y|**X**) directly
  - Assume a particular functional form for link function
  - Sigmoid applied to a linear function of the input features:

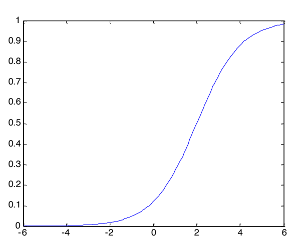
$$P(Y=0|X,W) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

Features can be discrete or continuous!

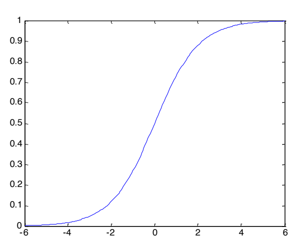
Understanding the sigmoid

$$g(w_0+\sum_i w_i x_i) = \frac{1}{1+e^{w_0+\sum_i w_i x_i}}$$

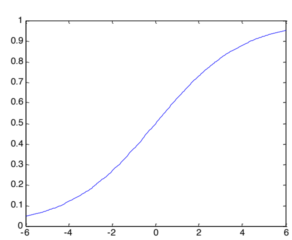
w_0=-2, w_1=-1



w_0=0, w_1=-1

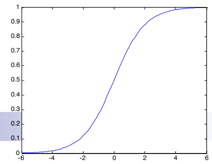


w_0=0, w_1=-0.5



Logistic Regression – a Linear classifier

$$\frac{1}{1+exp(-z)}$$



$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1+e^{w_0+\sum_i w_i x_i}}$$

Very convenient!

$$P(Y=0 \,|X=<X_1,...X_n>) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$P(Y=1 \,|X=<X_1,...X_n>) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$\frac{P(Y=1\,|X)}{P(Y=0\,|X)} = exp(w_0+\sum_i w_i X_i)$$

linear classification rule!

implies

$$\ln \frac{P(Y=1\,|X)}{P(Y=0\,|X)} = w_0+\sum_i w_i X_i$$