Linear Models

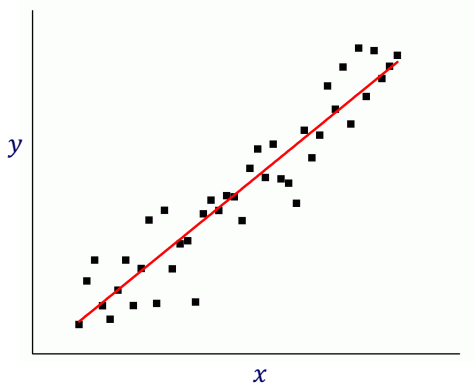
Mark Craven and David Page
Computer Sciences 760
Spring 2019

Goals for the lecture

you should understand the following concepts

- linear regression
- Ordinary Least Square (OLS) regression
- residual sum of squares loss
- logistic regression (again!)
- Ridge regression
- Lasso regression
- L_1 and L_2 penalties
- feature selection
- coordinate descent
- RMSE, MAE, and R-square
- locally weighted regression

Linear regression



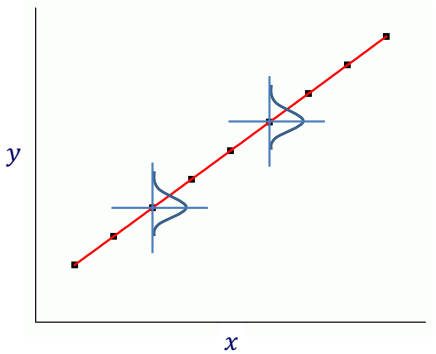
Linear regression assumes that the relationship between the expected value of the dependent variable y and the value of the dependent variable(s) x is linear

Ordinary Least Square (OLS) regression

For a single variable, assume the data is given by:

$$y^{(i)} = \alpha + \beta x^{(i)} + \varepsilon^{(i)}$$

where $\varepsilon^{(i)}$ represent noise from a Gaussian which are independent and have mean 0 and variance σ^2



Ordinary Least Square (OLS) regression

Goal: minimize the loss function

$$E(\beta) = \sum_i \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$ residual sum of squares (RSS)

linear model's prediction: $\alpha + \beta x^{(i)}$

Solution:

$$\hat{\beta} = \frac{\sum_i \left(x^{(i)} - \bar{x}\right)\left(y^{(i)} - \bar{y}\right)}{\sum_i (x^{(i)} - \bar{x})^2}, \qquad \hat{\alpha} = \bar{y} - \hat{\beta}\,\bar{x}$$

Multivariate linear regression

$$h\left(\boldsymbol{x}^{(i)}\right) = \beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} + \cdots + \beta_n x_n^{(i)}$$

$$E(\boldsymbol{\beta}) = \sum_i \left(y^{(i)} - \beta_0 - \sum_j \beta_j x_j^{(i)}\right)^2$$

- as we go to more variables
  - Use matrix representation and operations
  - assume all features standardized (standard normal)
  - assume an additional constant 1 feature as we did with neural nets
- given data matrix $\boldsymbol{X}$ with label vector $\boldsymbol{y}$
- find vector of coefficients $\widehat{\boldsymbol{\beta}}$ to minimize:

$$\left\|\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \boldsymbol{y}\right\|_2^2$$

Multivariate linear regression

- Write matrix as:

$$\boldsymbol{X} = \begin{pmatrix} x_1^{(1)} & \cdots & x_n^{(1)}, 1 \\ \vdots & & \vdots \\ x_1^{(m)} & \cdots & x_n^{(m)}, 1 \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

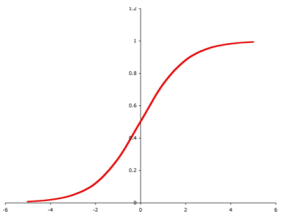
- Solution:

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

Can we extend this linear approach to binary classification?

let all $y^{(i)} \in [0,1]$

$$h\left(\boldsymbol{x}^{(i)}\right) = \frac{1}{1 + e^{-\left(\beta_0 + \sum_j \beta_j x_j^{(i)}\right)}}$$



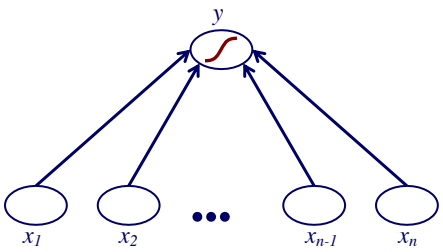
minimize the loss function:

$$E(\boldsymbol{\beta}) = \sum_i -y^{(i)} \ln\left(h\left(\boldsymbol{x}^{(i)}\right)\right) - \left(1 - y^{(i)}\right) \ln\left(1 - h\left(\boldsymbol{x}^{(i)}\right)\right)$$

Have we seen this before?

Logistic regression

Yes, this is logistic regression!



- a single layer neural net with a sigmoid in which the weights are trained to minimize cross entropy
- there is not a closed-form solution to the weights in this case, so we use gradient descent to minimize the loss function

Regularized regression

- regression (linear or logistic) is prone to overfitting when the number of features is large
- there are a number of versions of regression that accomplish regularization by including a penalty term in the objective function to "shrink" the weights

Ridge regression

- in *ridge regression* we add a penalty term to the loss function that penalizes weights, shrinking them towards zero

$$E(\boldsymbol{\beta}) = \sum_i \left(y^{(i)} - \beta_0 - \sum_j \beta_j x_j^{(i)} \right)^2 + \lambda \sum_{j=1}^{n} \beta_j^2$$

- λ is a hyperparameter we need to set; it determines the tradeoff between minimizing error and regularization
- the penalty term is also called L_2 regularization because it is based on the L_2 norm of the weight vector

$$\|(\boldsymbol{\beta})\|_2^2 = \sum_{j=1}^{n} \beta_j^2$$

Ridge regression

- still has a closed form solution

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X} - \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{y}$$

- or can solve via gradient descent

Lasso regression

- in *Lasso regression* we add a penalty term to the loss function that penalizes weights, shrinking them towards zero

$$E(\boldsymbol{\beta}) = \sum_i \left(y^{(i)} - \beta_0 - \sum_j \beta_j x_j^{(i)} \right)^2 + \lambda \sum_{j=1}^{n} |\beta_j|$$

- the penalty term is also called L_1 regularization because it is based on the L_1 norm of the weight vector

$$\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{n} |\beta_j|$$

Lasso regression

- no closed form solution
- can't use gradient descent because derivative is not defined at 0 in absolute value
- instead, use proximal method like coordinate descent

Coordinate descent

repeat until convergence

for $j = 1 \ldots n$

$$\rho_j = \sum_i x_j^{(i)} \left(y^{(i)} - \sum_{k \neq j} \beta_k \, x_k^{(i)} \right)$$

$$z_j = \sum_i \left(x_j^{(i)} \right)^2$$

$$\beta_j = \frac{1}{z_j} S(\rho_j, \lambda)$$

where S is a soft thresholding function

$$S(\rho, \lambda) = \begin{cases} \rho - \lambda & \text{if } \rho > \lambda \\ 0 & \text{if } -\lambda \leq \rho \leq \lambda \\ \rho + \lambda & \text{if } \rho < -\lambda \end{cases}$$

An alternative formulation

- Both ridge and lasso can be equivalently formulated as constrained optimization problems:

ridge

$$\min_{\boldsymbol{\beta}} \left\{ \sum_i \left(y^{(i)} - \beta_0 - \sum_j \beta_j x_j^{(i)} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{n} \beta_j^2 < s$$

lasso

$$\min_{\boldsymbol{\beta}} \left\{ \sum_i \left(y^{(i)} - \beta_0 - \sum_j \beta_j x_j^{(i)} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{n} |\beta_j| < s$$

Ridge regression and the Lasso

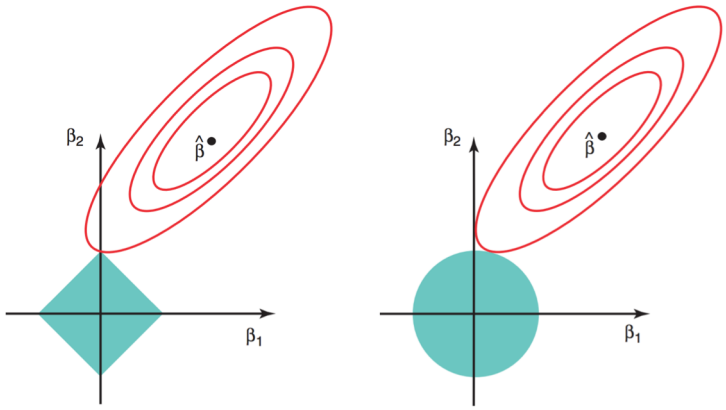


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

Figure from James et al.

Figure from Hastie et al., The Elements of Statistical Learning, 2008

Regularization via shrinkage

- Lasso (L_1) tends to make many weights 0, inherently performing *feature selection*
- Ridge regression (L_2) shrinks weights but isn't as biased towards selecting features
- L_1 and L_2 penalties can be used with other learning methods (deep neural nets, SVMs, etc.)
- both can help avoid overfitting by reducing variance
- there are many variants with somewhat different biases
  - elastic net: includes L_1 and L_2 penalties
  - group lasso: bias towards selecting defined groups of features
  - fused lasso: bias towards selecting "adjacent" features in a defined chain
  - etc.

Evaluating regression models

There are several commonly used measures to assess the predictive accuracy of regression models

mean absolute error

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}\left|y^{(i)} - h\left(\boldsymbol{x}^{(i)}\right)\right|$$

root mean squared error

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{m}\left(y^{(i)} - h(\boldsymbol{x}^{(i)})\right)^2}{m}}$$

R^2: proportional improvement in prediction from the regression model, compared to the mean model

$$R^2 = 1 - \frac{\sum_{i=1}^{m}\left(y^{(i)} - h\left(\boldsymbol{x}^{(i)}\right)\right)^2}{\sum_{i=1}^{m}(y^{(i)} - \bar{y})^2}$$

Why bother studying linear models when we can use extremely deep recurrent convolutional neural networks with residual connections, attention mechanisms and embedded input representations?

- linear models typically provide interpretable models; this is important in many domains
- simple methods often work well (see following example)
- my advice: always try a simple method first, at least as a baseline

npj | Digital Medicine

www.nature.com/npjdigitalmed

ARTICLE OPEN

Scalable and accurate deep learning with electronic health records

Alvin Rajkomar[1,2], Eyal Oren[1], Kai Chen[1], Andrew M. Dai[1], Nissan Hajaj[1], Michaela Hardt[1], Peter J. Liu[1], Xiaobing Liu[1], Jake Marcus[1], Mimi Sun[1], Patrik Sundberg[1], Hector Yee[1], Kun Zhang[1], Yi Zhang[1], Gerardo Flores[1], Gavin E. Duggan[1], Jamie Irvine[1], Quoc Le[1], Kurt Litsch[1], Alexander Mossin[1], Justin Tansuwan[1], De Wang[1], James Wexler[1], Jimbo Wilson[1], Dana Ludwig[2], Samuel L. Volchenboum[3], Katherine Chou[1], Michael Pearson[1], Srinivasan Madabushi[1], Nigam H. Shah[4], Atul J. Butte[2], Michael D. Howell[1], Claire Cui[1], Greg S. Corrado[1] and Jeffrey Dean[1]

Predictive modeling with electronic health record (EHR) data is anticipated to drive personalized medicine and improve healthcare quality. Constructing predictive statistical models typically requires extraction of curated predictor variables from normalized EHR data, a labor-intensive process that discards the vast majority of information in each patient's record. We propose a representation of patients' entire raw EHR records based on the Fast Healthcare Interoperability Resources (FHIR) format. We demonstrate that deep learning methods using this representation are capable of accurately predicting multiple medical events from multiple centers without site-specific data harmonization. We validated our approach using de-identified EHR data from two US academic medical centers with 216,221 adult patients hospitalized for at least 24 h. In the sequential format we propose, this volume of EHR data unrolled into a total of 46,864,534,945 data points, including clinical notes. Deep learning models achieved high accuracy for tasks such as predicting: in-hospital mortality (area under the receiver operator curve [AUROC] across sites 0.93–0.94), 30-day unplanned readmission (AUROC 0.75–0.76), prolonged length of stay (AUROC 0.85–0.86), and all of a patient's final discharge diagnoses (frequency-weighted AUROC 0.90). These models outperformed traditional, clinically-used predictive models in all cases. We believe that this approach can be used to create accurate and scalable predictions for a variety of clinical scenarios. In a case study of a particular prediction, we demonstrate that neural networks can be used to identify relevant information from the patient's chart.

npj Digital Medicine (2018)1:18 ; doi:10.1038/s41746-018-0029-1

ARTICLE OPEN

Scalable and accurate deep learning with electronic health records

Supplemental Table 1: Prediction accuracy of each task of deep learning model compared to baselines

	Hospital A	Hospital B
Inpatient Mortality, AUROC[1](95% CI)		
Deep learning 24 hours after admission	**0.95**(0.94-0.96)	**0.93**(0.92-0.94)
Full feature enhanced baseline at 24 hours after admission	0.93 (0.92-0.95)	0.91 (0.89-0.92)
Full feature simple baseline at 24 hours after admission	0.93 (0.91-0.94)	0.90 (0.88-0.92)
Baseline (aEWS[2]) at 24 hours after admission	0.85 (0.81-0.89)	0.86 (0.83-0.88)
30-day Readmission, AUROC (95% CI)		
Deep learning at discharge	**0.77**(0.75-0.78)	**0.76**(0.75-0.77)
Full feature enhanced baseline at discharge	0.75 (0.73-0.76)	0.75 (0.74-0.76)
Full feature simple baseline at discharge	0.74 (0.73-0.76)	0.73 (0.72-0.74)
Baseline (mHOSPITAL[3]) at discharge	0.70 (0.68-0.72)	0.68 (0.67-0.69)
Length of Stay at least 7 days AUROC (95% CI)		
Deep learning 24 hours after admission	**0.86**(0.86-0.87)	**0.85**(0.85-0.86)
Full feature enhanced baseline at 24 hours after admission	0.85 (0.84-0.85)	0.83 (0.83-0.84)
Full feature simple baseline at 24 hours after admission	0.83 (0.82-0.84)	0.81 (0.80-0.82)
Baseline (mLiu[4]) at 24 hours after admission	0.76 (0.75-0.77)	0.74 (0.73-0.75)

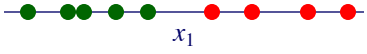
"baseline" models are regularized logistic regression; nearly as good as deep networks for all tasks

Linear models can be combined with other ML approaches

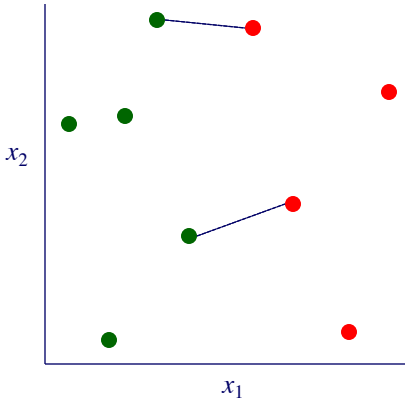
- the CART algorithm can learn linear functions to predict values at the leaves of regression trees
- *locally weighted regression* is an approach that combines linear regression with a nearest-neighbor method

Irrelevant features in instance-based learning

here's a case in which there is one relevant feature x_1 and a 1-NN rule classifies each instance correctly



consider the effect of an irrelevant feature x_2 on distances and nearest neighbors



Locally weighted regression

- one way around this limitation of instance-based methods is to weight features differently
- *locally weighted regression* is one nearest-neighbor variant that does this

prediction task

- **given**: an instance $\boldsymbol{x}^{(q)}$ to make a prediction for
- find the k training-set instances that are most similar to $\boldsymbol{x}^{(q)}$
- return the value

$$h\left(\boldsymbol{x}^{(q)}\right) = \beta_0^{(q)} + \beta_1^{(q)} x_1^{(q)} + \beta_2^{(q)} x_2^{(q)} + \cdots + \beta_n^{(q)} x_n^{(q)}$$

Locally weighted regression

prediction/learning task

- find the weights $\beta_i^{(q)}$ for each $\boldsymbol{x}^{(q)}$ by minimizing

$$E\left(\boldsymbol{\beta}^{(q)}\right) = \sum_{i=1}^{k} \left(h\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

- this is done at prediction time, specifically for $\boldsymbol{x}^{(q)}$