# CS 760: Machine Learning
# **Linear regression**

University of Wisconsin-Madison

# Outline

- **Recap of logistic regression**
  - setup, MLE, multiclass extension

- **Linear regression**
  - setup, normal equations, metrics

- **Regularization**
  - motivation, tuning, interpretations

# Outline

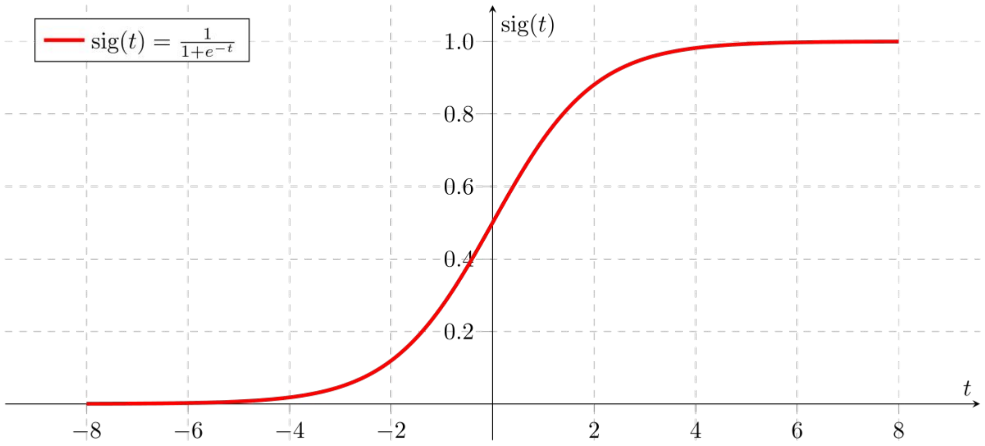
- **Recap of logistic regression**
  - setup, MLE, multiclass extension
- **Linear regression**
  - setup, normal equations, metrics
- **Regularization**
  - motivation, tuning, interpretations

# Linear Classification

- Let's think probabilistically and learn $P_\theta(y|x)$

- How?
  - Specify the conditional distribution $P_\theta(y|x)$
  - Use **MLE** to derive a loss
  - Run gradient descent (or related optimization algorithm)

  - Leads to logistic regression



# Likelihood Function

- Captures the probability of seeing some data as a function of model parameters:

$$\mathcal{L}(\theta; X) = P_\theta(X)$$

  - If data is iid, we have $\mathcal{L}(\theta; X) = \prod_j p_\theta(x_j)$

- Often more convenient to work with the log likelihood
  - Log is a monotonic + strictly increasing function

# Maximum Likelihood

- For some set of data, find the parameters that maximize the likelihood / log-likelihood

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta; X)$$

- Example: suppose we have n samples from a Bernoulli distribution

$$P_{\theta}(X = x) = \begin{cases} \theta & x = 1 \\ 1 - \theta & x = 0 \end{cases}$$

Then,

$$\mathcal{L}(\theta; X) = \prod_{i=1}^{n} P(X = x_i) = \theta^k (1 - \theta)^{n-k}$$

# **Maximum Likelihood:** Example

- Want to maximize likelihood w.r.t. Θ

$$\mathcal{L}(\theta; X) = \prod_{i=1}^{n} P(X = x_i) = \theta^k (1-\theta)^{n-k}$$

- Differentiate (use product rule) and set to 0. Get

$$\theta^{h-1}(1-\theta)^{n-h-1}(h - n\theta) = 0$$

- So: ML estimate is $\hat{\theta} = \frac{h}{n}$

# **ML:** Conditional Likelihood

- Similar idea, but now using conditional probabilities:

$$\mathcal{L}(\theta; Y, X) = p_\theta(Y|X)$$

  - If data is iid, we have

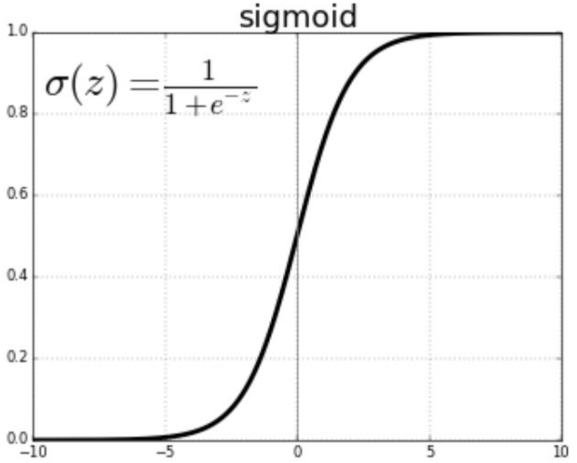
$$\mathcal{L}(\theta; Y, X) = \prod_j p_\theta(y_j|x_j)$$

  - Now we can apply this to linear classification: yields **logistic regression.**

# **Logistic Regression**: Conditional Distribution

- Notation: $$\sigma(z) = \frac{1}{1+\exp(-z)} = \frac{\exp(z)}{1+\exp(z)}$$

**Sigmoid**



- **Conditional Distribution**:

$$P_\theta(y=1|x) = \sigma(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$$

# **Logistic Regression**: Loss

- Conditional MLE:

$$\log \text{ likelihood}(\theta | x^{(i)}, y^{(i)}) = \log P_\theta(y^{(i)} | x^{(i)})$$

- So:

$$\min_\theta \ell(f_\theta) = \min_\theta -\frac{1}{n} \sum_{i=1}^{n} \log P_\theta(y^{(i)} | x^{(i)})$$

Or,

$$\min_\theta -\frac{1}{n} \sum_{y^{(i)}=1} \log \sigma(\theta^T x^{(i)}) - \frac{1}{n} \sum_{y^{(i)}=0} \log(1 - \sigma(\theta^T x^{(i)}))$$

# **Logistic Regression**: Sigmoid Properties

- **Bounded**:

$$\sigma(z) = \frac{1}{1+\exp(-z)} \in (0,1)$$

- **Symmetric**:

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

- **Gradient**:

$$\sigma'(z) = \frac{\exp(-z)}{(1+\exp(-z))^2} = \sigma(z)(1-(\sigma(z))$$

# **Logistic regression**: Summary

- **Logistic regression = sigmoid conditional distribution + MLE**

- More precisely:
  - Give training data iid from some distribution D,
  - **Train**:

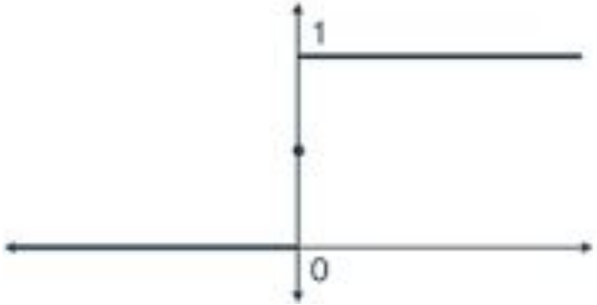
$$\min_{\theta} \ell(f_\theta) = \min_{\theta} -\frac{1}{n}\sum_{i=1}^{n} \log P_\theta(y^{(i)}|x^{(i)})$$

  - **Test**: output label probabilities

$$P_\theta(y=1|x) = \sigma(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$$

# **Logistic Regression**: Comparisons

step function



- Recall the first attempt:

$$\ell(f_\theta) = \frac{1}{m}\sum_{i=1}^{m} 1\{\text{step}(f_\theta(x^{(i)}) \neq y^{(i)})$$

- **Difficult to optimize!!**
- Another way: run least squares, ignore that y is 0 or 1:

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n} (f_\theta(x^{(j)}) - y^{(j)})^2$$

# **Logistic Regression**: Comparisons

- Downside: not robust to “outliers”

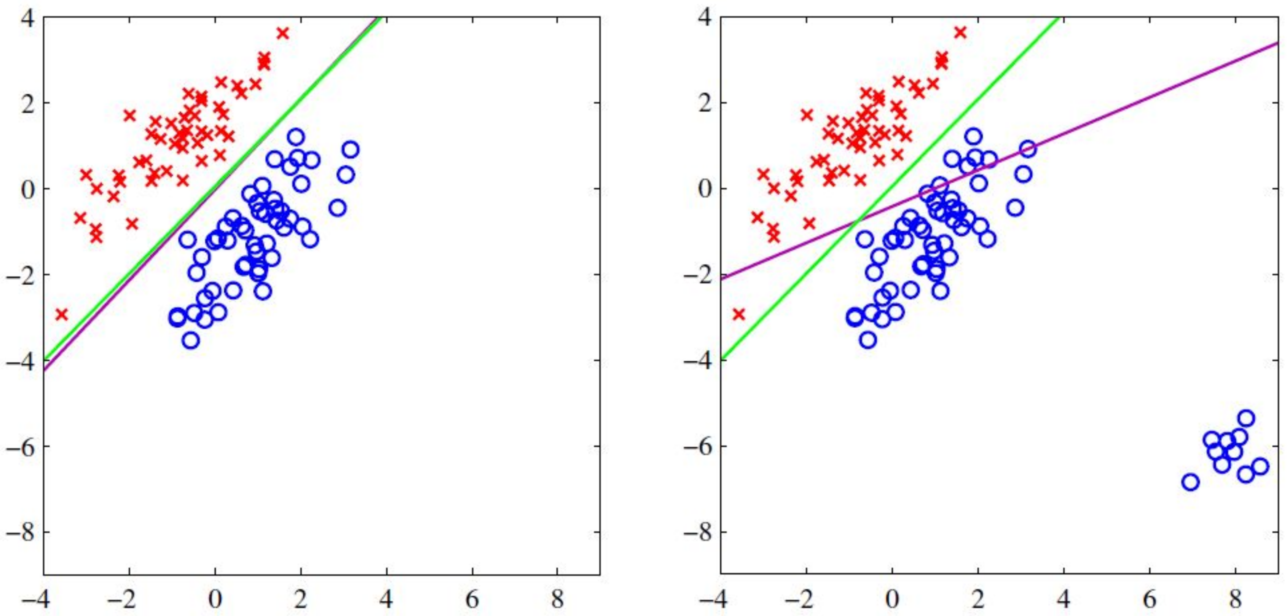


**Figure 4.4** The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve), which is discussed later in Section 4.3.2. The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

Figure: *Pattern Recognition and Machine Learning*, Bishop

# **Logistic Regression**: Beyond Binary

- We started with this conditional distribution:

$$P_\theta(y = 1|x) = \sigma(\theta^T x) = \frac{1}{1 + \exp(-\theta^T x)}$$

- Now let's try to extend it.
  - Can no longer just use one $\theta^T x$
  - But we can try multiple…

# **Logistic Regression**: Beyond Binary

- Let's set, for y in 1,2,…,k

$$P_\theta(y = i|x) = \frac{\exp((\theta^i)^T x)}{\sum_{j=1}^{k} \exp((\theta^j)^T x)}$$

- Note: we have several weight vectors now (1 per class).
- To train, same as before (just more weight vectors).

$$\min_\theta -\frac{1}{n}\sum_{i=1}^{n} \log P_\theta(y^{(i)}|x^{(i)})$$

# **Cross-Entropy** Loss

- Let's define $q^{(i)}$ as the one-hot vector for the ith datapoint.
- Next, let's let $p^{(i)} = P_\theta(y|x^{(i)})$ be our prediction

**Note: only 1 term non-zero.**

- Our loss terms can be written

$$-\log p(y^{(i)}|x^{(i)}) = -\underbrace{\sum_{j=1}^{k} q_j^{(i)} \log p(y = j|x^{(i)})}$$

**Should look familiar…**

- This is the "cross-entropy" $H(q^{(i)}, p^{(i)})$

# **Cross-Entropy** Loss

- This is the “cross-entropy”

$$H(q^{(i)}, p^{(i)}) = \mathbb{E}_{q^{(i)}}[\log p^{(i)}]$$

- What are we doing when we minimize the cross-entropy?
- Recall KL divergence,

$$D(q^{(i)}||p^{(i)}) = \underbrace{\mathbb{E}_{q^{(i)}}[\log p^{(i)}]}_{\text{Cross-entropy}} - \underbrace{\mathbb{E}_{q^{(i)}}[\log q^{(i)}]}_{\substack{\text{Entropy } H(q^{(i)}) \\ \text{(fixed)}}}$$

- Matching distributions!

# Softmax

- We wrote

$$P_\theta(y = i|x) = \frac{\exp((\theta^i)^T x)}{\sum_{j=1}^{k} \exp((\theta^j)^T x)}$$

- This operation is called softmax.
  - Converts a vector into a probability vector (note normalization).
  - If one component in the vector **a** is **dominant**, softmax(**a**) is close to one-hot vector

# Break & quiz

Q:  Why **can we** work with the log-likelihood rather than the likelihood?

Q:  Why **do we want to** work with the log-likelihood rather than the likelihood?
Give two reasons

Q:  Why **can we** work with the log-likelihood rather than the likelihood?

The logarithm is a monotonically increasing function and so any parameter $\theta$ that maximizes the log-likelihood also maximizes the likelihood

Q:  Why **do we want to** work with the log-likelihood rather than the likelihood?
Give two reasons

Q: Why **can we** work with the log-likelihood rather than the likelihood?

The logarithm is a monotonically increasing function and so any parameter $\theta$ that maximizes the log-likelihood also maximizes the likelihood

Q: Why **do we want to** work with the log-likelihood rather than the likelihood?
Give two reasons

1. easier to work with mathematically because it converts products to sums
2. easier to optimize numerically

# Outline

- **Recap of logistic regression**
  - setup, MLE, multiclass extension

- **Linear regression**
  - setup, normal equations, metrics

- **Regularization**
  - motivation, tuning, interpretations

# **Linear Regression**: Setup

- **Training/learning**: given

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

- Find $f_\theta(x) = \theta^T x = \sum_{i=0} \theta_i x_i$ that minimizes

**Note**: set $x_0$ = 1

Hypothesis Class

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n} \underbrace{(f_\theta(x^{(j)}) - y^{(j)})^2}_{\text{Loss function (how far are we)?}}$$

# A word on x

The x feature vector does not need to be “raw features”.

If your raw feature vector is v, x can be ANY computable vector derived from v

e.g. x=(v1, v1^2, v2, v3, v2*v3, exp(v4),...).

Linear regression on x produced nonlinear function on v.

# **Linear Regression**: Notation

- **Matrix notation**: set *X* to have *j*th row be $(x^{(j)})^T$
  - And *y* to be the vector $[y^{(1)}, \ldots, y^{(n)}]^T$

- Can re-write the loss function as

$$\ell(f_\theta) = \frac{1}{n} \sum_{j=1}^{n} (f_\theta(x^{(j)}) - y^{(j)})^2 = \frac{1}{n} \|X\theta - y\|_2^2$$

# **Linear Regression**: Fitting

- **Set gradient to 0 w.r.t. the weight,**

$$\nabla \ell(f_\theta) = \nabla \frac{1}{n} \|X\theta - y\|_2^2 = 0$$

$$\implies \nabla[(X\theta - y)^T (X\theta - y)] = 0$$

$$\implies \nabla[\theta^T X^T X\theta - 2\theta^T X^T y + y^T y] = 0$$

$$\implies 2X^T X\theta - 2X^T y = 0$$

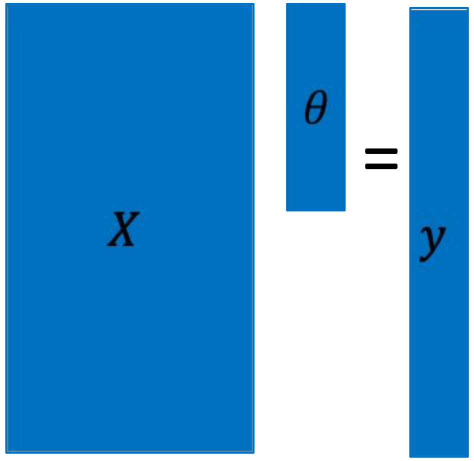
$$\implies \theta = (X^T X)^{-1} X^T y$$

**(assume $X^T X$ is invertible)**

# **Linear Regression**: Minimizer

- Let's study this solution algebraically

- If $X$ is invertible, just solve $X\theta = y$ and get $\theta = X^{-1}y$
- But typically $X$ is a tall matrix



Normal equation: $\theta = (X^T X)^{-1} X^T y$

usually no longer "exact": MSE>0

# **Evaluation**: Metrics

- MSE/RMSE (mean-square error + root version)

- MAE (mean average error)

- R-squared (more on this next)

- Usually, compute on training data… (but should do cross validation!)

# R-squared

- Several ways to define it, one way:

$$R^2 = 1 - \frac{\sum_j (y^{(j)} - f_\theta(x^{(j)}))^2}{\sum_j (y^{(j)} - \bar{y})^2}$$

Empirical mean of labels

- Intuition: how much of the variance in y is predictable by x

For OLS, always in [0,1 (best)]. Can be negative for other models.

# Outline

- **Recap of logistic regression**
    - setup, MLE, multiclass extension

- **Linear regression**
    - setup, normal equations, metrics

- **Regularization**
    - motivation, tuning, interpretations

# High-dimensional linear regression

Data matrix $X$ is $n \times d$

- number of data points $n$
- number of features $d$

If $n > d$ and X has full column rank then $X^\top X$ is invertible

**But what if $d \gg n$ ?**

- e.g. a training set of $n$ =1K documents, each represented as a **bag-of-words** vector ($X_{[j,i]} = \# \, times \, word \, i \, is \, in \, document \, j$) with vocabulary size $d$ =10K
- now $X^\top X$ will not be invertible

# Solution: **Regularization**

- Same setup, new loss (**Ridge regression**):

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_2^2$$

regularization parameter

  - Conveniently, still has a closed form solution

$$\theta = (X^T X + \lambda n I)^{-1} X^T y$$

  - **Goals:**
    - solves the problem of $X^\top X$ not being invertible
    - results in a $\theta$ with small norm, which is often less likely to overfit

# Alternative regularization: **LASSO**

- Another type of regularization:

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \underset{\text{regularization parameter}}{\lambda}\|\theta\|_1$$

  - unlike the $\ell_2$-norm, regularizing by the $\ell_1$-norm is known to encourage a sparse $\theta$
    - theoretical understanding of this phenomenon exists under assumptions on $X$ and $y$ (**compressed sensing**)
    - useful for both regularization and **feature selection**

# Choosing the regularization strength $\lambda$

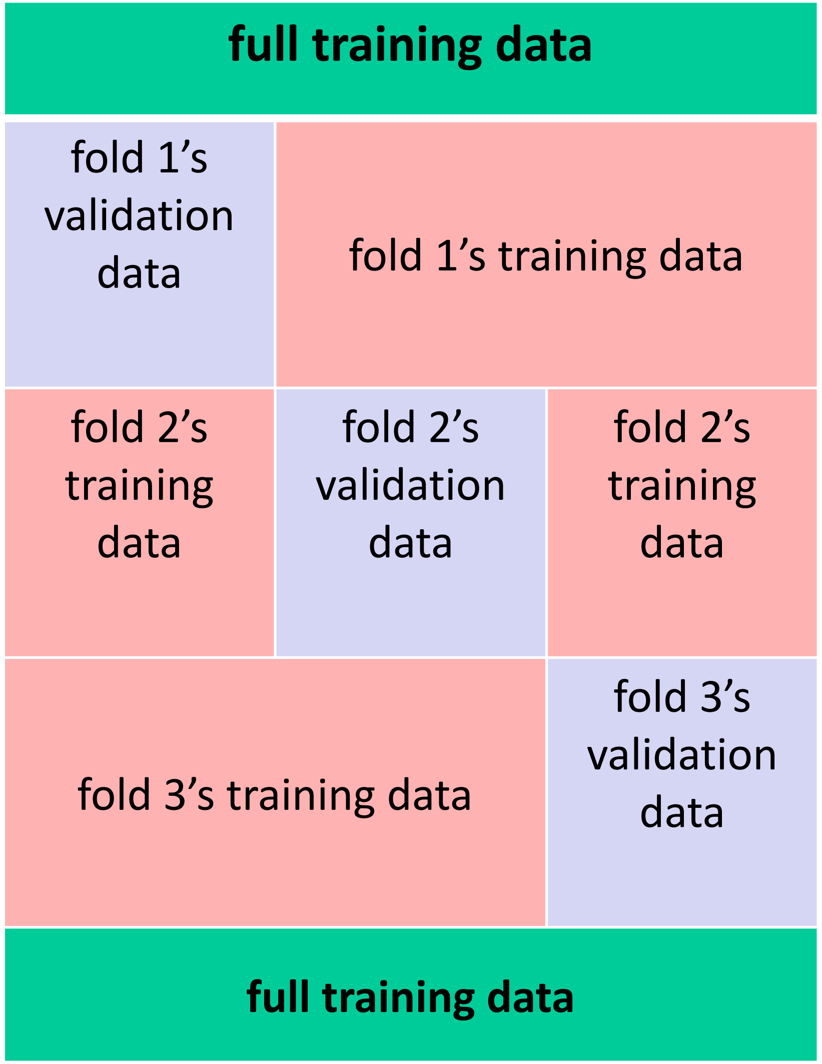
For prediction: use cross-validation!

- split dataset into $k$ train-validation folds

- for each candidate $\lambda$:
  - compute average across folds $i = 1, \ldots, k$ of
    - MSE (or other metric) of $\theta_{\lambda,i}$ on fold i's validation data
    - $\theta_{\lambda,i}$ minimizes Ridge/LASSO with parameter $\lambda$ on fold i's training data

- retrain on the full training data with the optimal candidate $\lambda$

Tip: use geometric sequence e.g. 10^{-5,...,5}



**full training data**

fold 1's validation data | fold 1's training data

fold 2's training data | fold 2's validation data | fold 2's training data

fold 3's training data | fold 3's validation data

**full training data**

# Other things you can do with regularization

-

- combine $\ell_1$ and $\ell_2$ regularization (Elastic Net)

- feature selection: determine which features of your model are important

- regularize classifiers like logistic regression (just add a norm penalty to the MLE objective)

# Probabilistic interpretation

the ordinary least squares (OLS) estimator $\theta = (X^T X)^{-1} X^T y$ estimator is the MLE of a Gaussian probabilistic model:

- $y^{(i)} \sim N(\theta^\top x^{(i)}, \sigma^2)$
- assume variance $\sigma^2$ is known

Ridge regression and LASSO are **MAP estimators of the same probabilistic model** with different priors for $\theta$

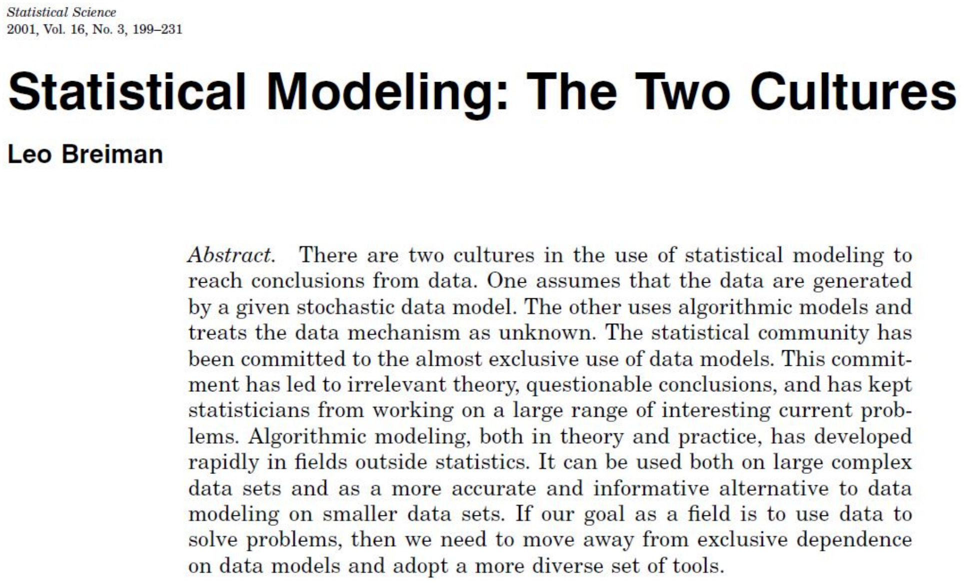
- Ridge regression: $\theta \sim N(0_d, \tau^2 I_d)$
- LASSO: $\theta \sim \mathrm{Laplace}(0_d, \tau)$
- in both cases $\tau$ depends on $\sigma^2$ and $\lambda$

# Regularized least squares in statistics

- LASSO / Elastic Net often used to pick out relevant and irrelevant features

- numerous tools to identify the “true” data model $\theta$
    - AIC / BIC
    - statistical tests
    - LARS / regularization paths

- see today’s reading to learn about the **two cultures of statistical modeling:**
    - data modeling (“98% of all statisticians”)
    - algorithmic modeling (most of ML today)



*Statistical Science*
2001, Vol. 16, No. 3, 199–231

## Statistical Modeling: The Two Cultures

**Leo Breiman**

*Abstract.* There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown. The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems. Algorithmic modeling, both in theory and practice, has developed rapidly in fields outside statistics. It can be used both on large complex data sets and as a more accurate and informative alternative to data modeling on smaller data sets. If our goal as a field is to use data to solve problems, then we need to move away from exclusive dependence on data models and adopt a more diverse set of tools.

# Break & quiz

Q:  Suppose you find that your linear regression model is under fitting the data. In such situation which of the following options would you consider?

A. *Add more variables*

B. *Start introducing polynomial degree variables*

C. *Use L1 regularization*

D. *Use L2 regularization*

1. A, B, C
2. A, B, D
3. A, B
4. A, B, C, D

Q: Suppose you find that your linear regression model is under fitting the data. In such situation which of the following options would you consider?

A. *Add more variables*

B. *Start introducing polynomial degree variables*

C. *Use L1 regularization*

D. *Use L2 regularization*

1. A, B, C
2. A, B, D
3. [✓ A, B]
4. A, B, C, D

In case of under fitting, you need to induce more variables in variable space or you can add some polynomial degree variables to make the model more complex to be able to fit the data better. No regularization methods should be used because regularization is used in case of overfitting.

# Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Misha Khodak, Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, Alex Smola, Fred Sala, Tengyang Xie