CS-GY 6923: Lecture 3
Regularization + Bayesian Perspective

NYU Tandon School of Engineering, Akbar Rafiey
Slides by Prof. Christopher Musco

Course news

- The first written HW has been posted, due October 01.
- Reminder: Lab 02 due September 24.

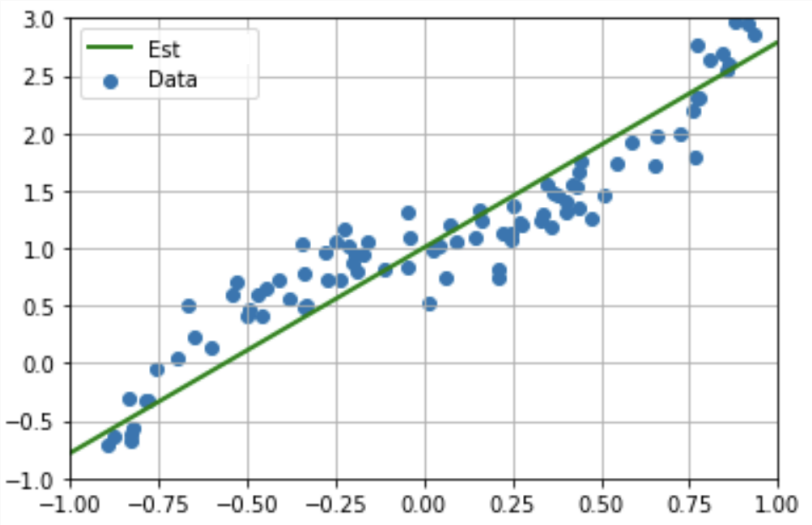
Recap

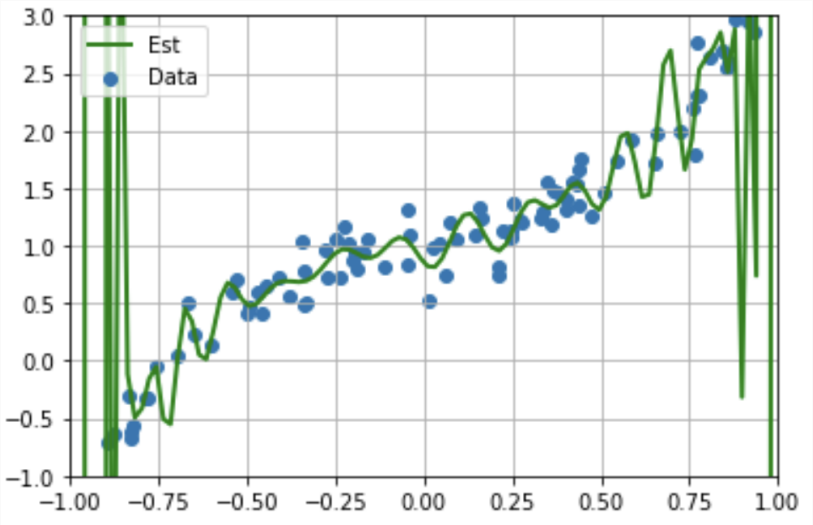
Model selection:

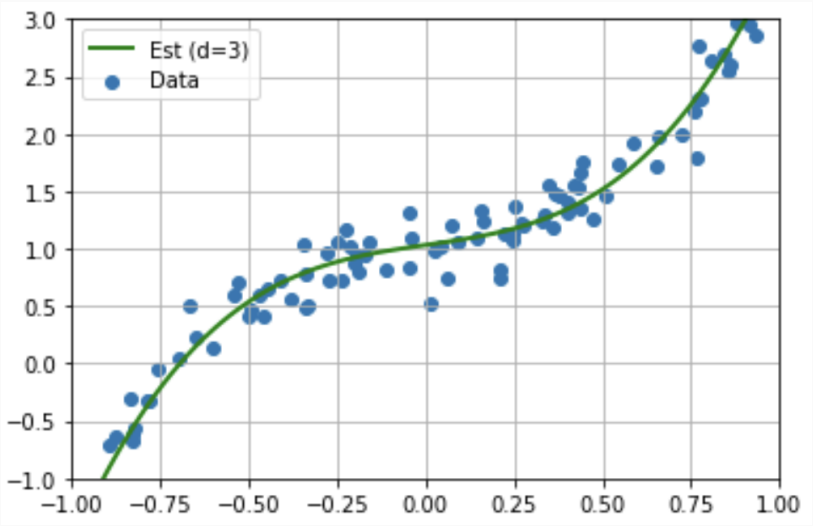
- Train models $f^{(1)}_{\boldsymbol{\theta}_1}, \ldots, f^{(q)}_{\boldsymbol{\theta}_q}$ independently on training data to find optimal parameters $\boldsymbol{\theta}^*_1, \ldots, \boldsymbol{\theta}^*_q$.
- Check loss $L_{test}(f^{(1)}, \boldsymbol{\theta}^*_1), \ldots, L_{test}(f^{(q)}, \boldsymbol{\theta}^*_q)$ on test data.
- Select model with lowest test lost.

Can be used for arbitrary sets of models. Often used when you are not sure how "complex" your model should be for the data, and want to find a sweet spot between a good fit, and not overfitting.

Recap







Underfit, overfit, just right.

Quick aside on numerical issues

In the demo 4 we had you use `numpy.polynomial.polynomial`. However, as we discussed early, we can use multiple linear regression instead by constructing the data matrix:

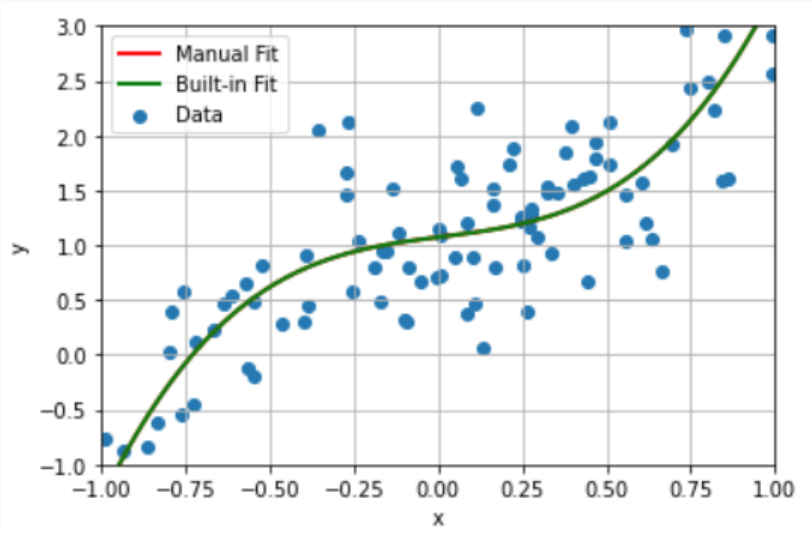
$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^1 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

Then find polynomial coefficents as $\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.

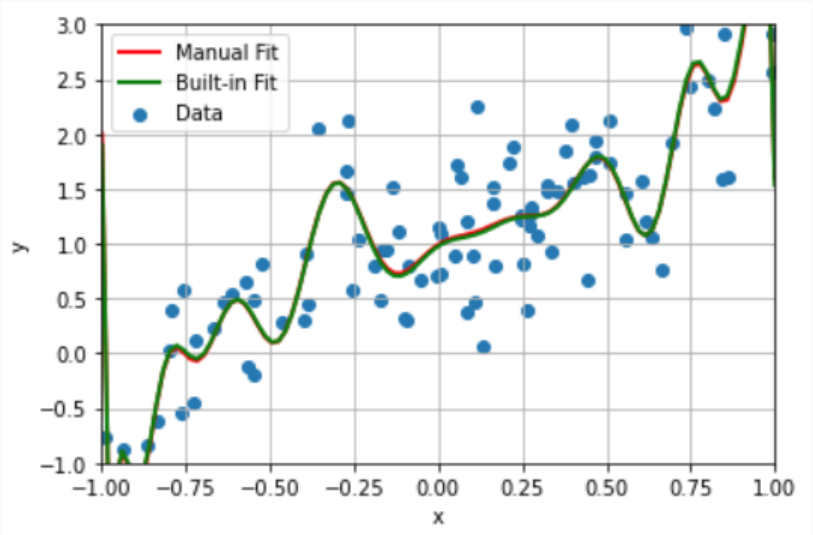
Quick aside on numerical issues

```
# built in function
beta_hat = poly.polyfit(xdat,ydat,d)

# manual fit using naive multivariate regression
X = np.zeros([len(xdat),d+1])
for i in range(d+1):
  X[:,i] = xdat**i
my_beta = np.linalg.inv(np.transpose(X)@X)@np.transpose(X)@ydat
```



Degree 3

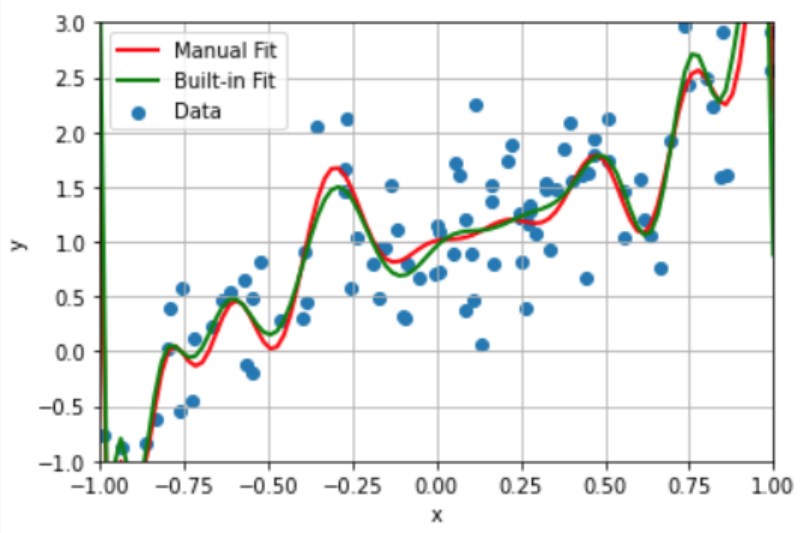


Degree 22

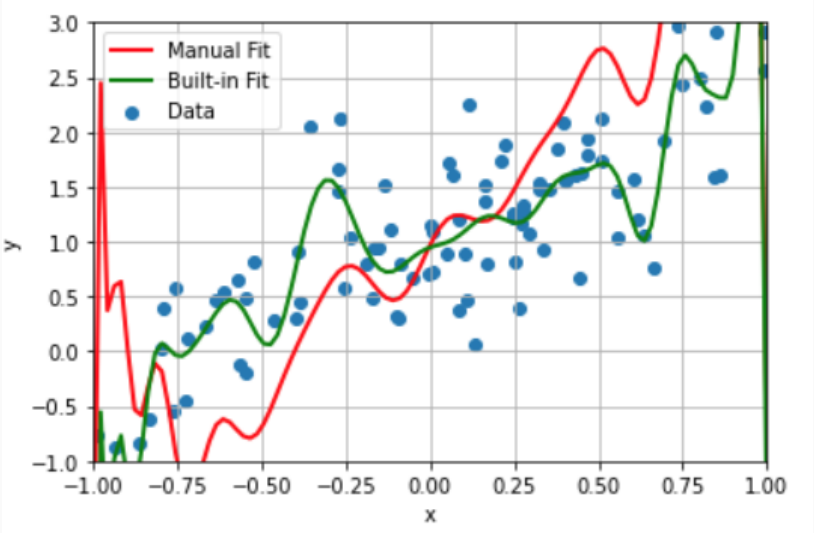
Quick aside on numerical issues

```
# built in function
beta_hat = poly.polyfit(xdat,ydat,d)

# manual fit using naive multivariate regression
X = np.zeros([len(xdat),d+1])
for i in range(d+1):
  X[:,i] = xdat**i
my_beta = np.linalg.inv(np.transpose(X)@X)@np.transpose(X)@ydat
```



Degree 23



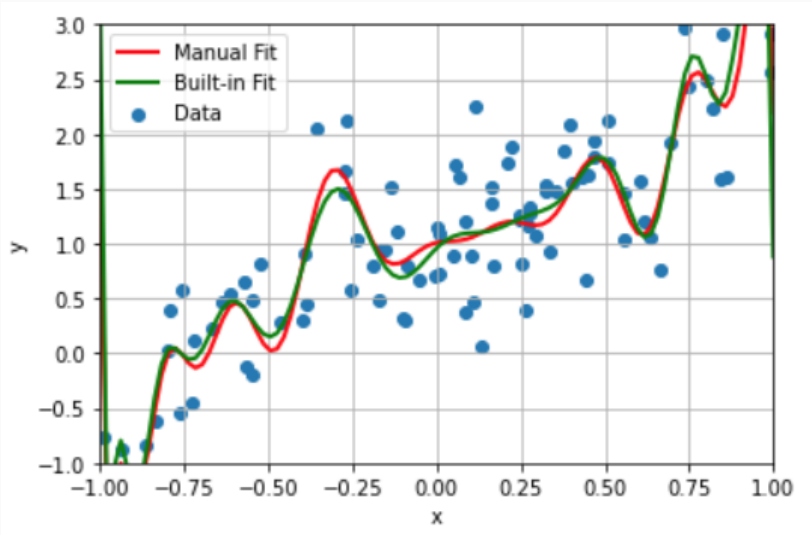
Degree 30

What do you think is the reason?

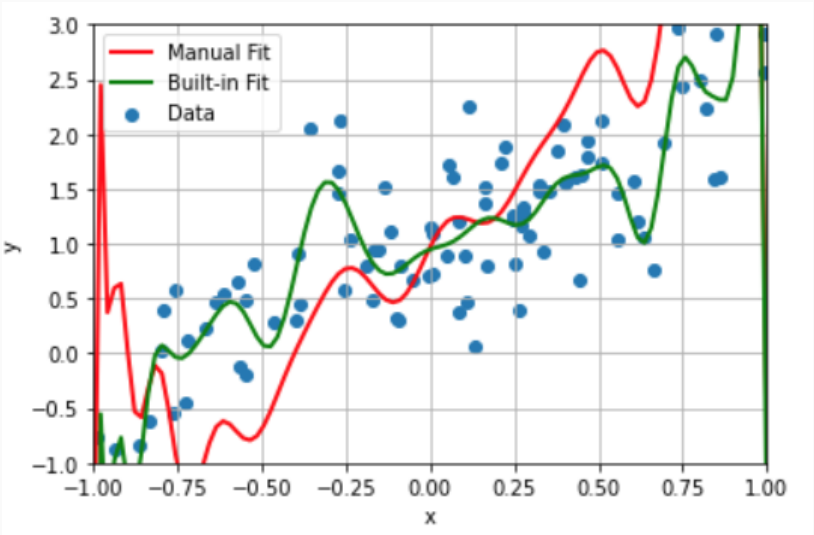
Quick aside on numerical issues

```
# built in function
beta_hat = poly.polyfit(xdat,ydat,d)

# manual fit using naive multivariate regression
X = np.zeros([len(xdat),d+1])
for i in range(d+1):
  X[:,i] = xdat**i
my_beta = np.linalg.inv(np.transpose(X)@X)@np.transpose(X)@ydat
```



Degree 23



Degree 30

Has to do with numerical round-off error. (Scipy still uses linear regression, but with extra "tricks" to avoid numerical issues.)

Quick aside on numerical issues

- Your computer can easily deal with both very large and very small numbers. Underflow and overflow are extremely unlikely to be issues in floating point arithmetic.
- The issue is when you compute using numbers of very differing magnitude.

Quick aside on numerical issues

- Your computer can easily deal with both very large and very small numbers. Underflow and overflow are extremely unlikely to be issues in floating point arithmetic.
- The issue is when you compute using numbers of very differing magnitude.

```
print(.3*10**-34 + 10**-36 - 10**-36)
```

```
3e-35
```

```
print(.3*10**34 + 10**36)
```

```
1.003e+36
```

```
print(.3*10**-34 + 10 - 10)
```

```
0.0
```

Quick aside on numerical issues

Recall that we chose each $x_i \in [-1, 1]$ uniformly at random.

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

Regularization

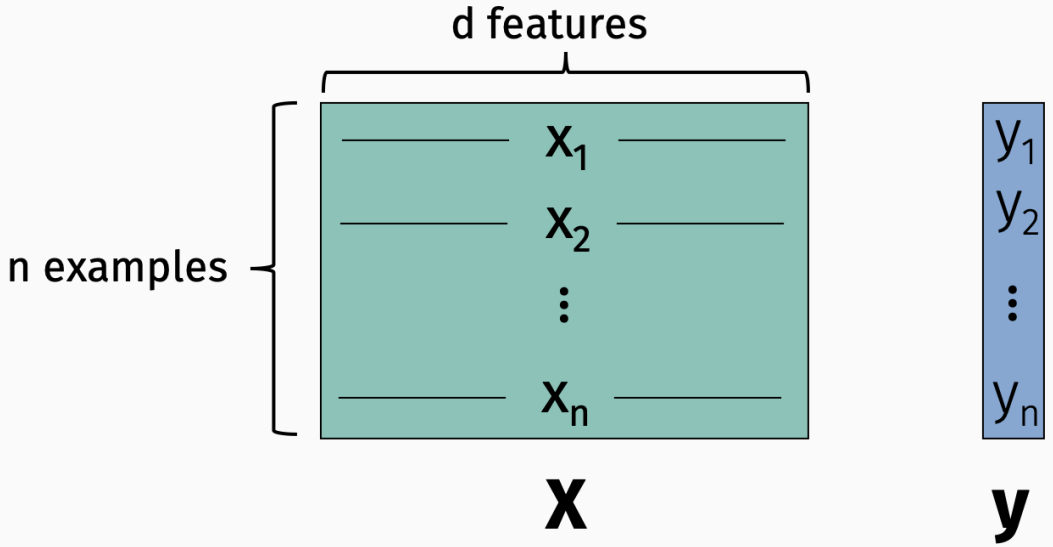
Over-parameterized models

In the model selection examples we discussed last class, we had full control over the complexity of the model: could range from underfitting to overfitting.

In practice, you often don't have this freedom. Even the most basic model might lead to overfitting.

Over-parameterized models

Example: Linear regression model where $d \geq n$.

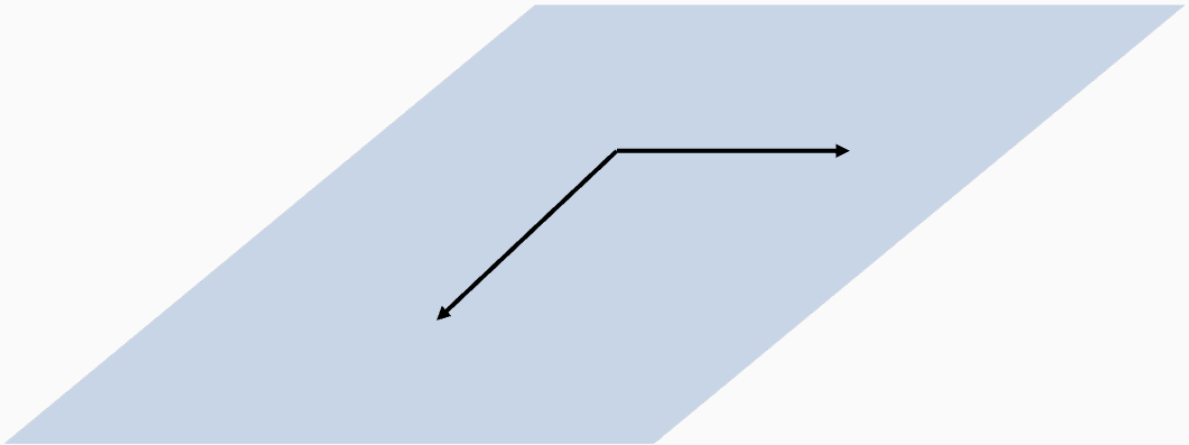


Can (almost) always find β so that $\mathbf{X}\beta = \mathbf{y}$ exactly.

Why?

High dimensional linear models

Claim: For almost all sets of n, length n vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, we can write any vector $\mathbf{y}$ as a linear combination of these vectors.



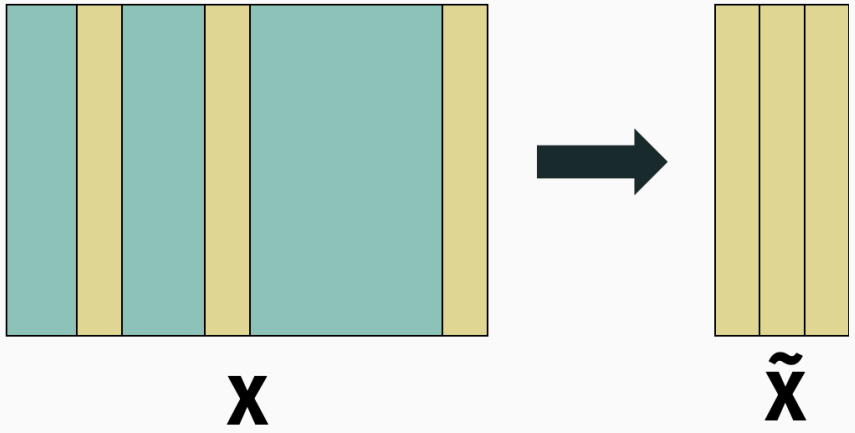
I.e., can find some coefficients so that
$\beta_1 \mathbf{x}^{(1)} + \ldots + \beta_q \mathbf{x}^{(q)} = \mathbf{X}\boldsymbol{\beta} = \mathbf{y}$.

Zero train loss

- We will discuss some models later in the class where zero training loss is not necessarily a bad sign: k-nearest neighbors, some neural nets.
- Typically however it will be a sign of overfitting, as in the polynomial regression example.

Feature selection

Select some subset of $\ll n$ features to use in model:



Filter method: Compute some metric for each feature, and select features with highest score.

- Example: compute loss or R^2 value when each feature in $\mathbf{X}$ is used in single variate regression.

Any potential limitations of this approach?

Feature selection

Exhaustive approach: Pick best subset of q features. Train $\binom{d}{q}$ models.

Feature selection

Faster approach: Greedily select q features.

Stepwise Regression:

- **Forward:** Step 1: pick single feature that gives lowest loss. Step k: pick feature that when combined with previous $k-1$ chosen features gives lowest loss.
- **Backward:** Start with all of the features. Greedily eliminate those which have least impact on model performance.

Feature selection deserves more than two slides, but we won't go into too much more detail!

Alternative approach

Regularization: Discourage overfitting by adding a regularization penalty to the loss minimization problem.

$$\min_{\beta} \left[L(\beta) + Reg(\beta) \right].$$

Alternative approach

Regularization: Discourage overfitting by adding a regularization penalty to the loss minimization problem.

$$\min_{\beta} \left[L(\beta) + Reg(\beta) \right].$$

Example: Least squares regression: $L(\beta) = \|\mathbf{X}\beta - \mathbf{y}\|_2^2$.

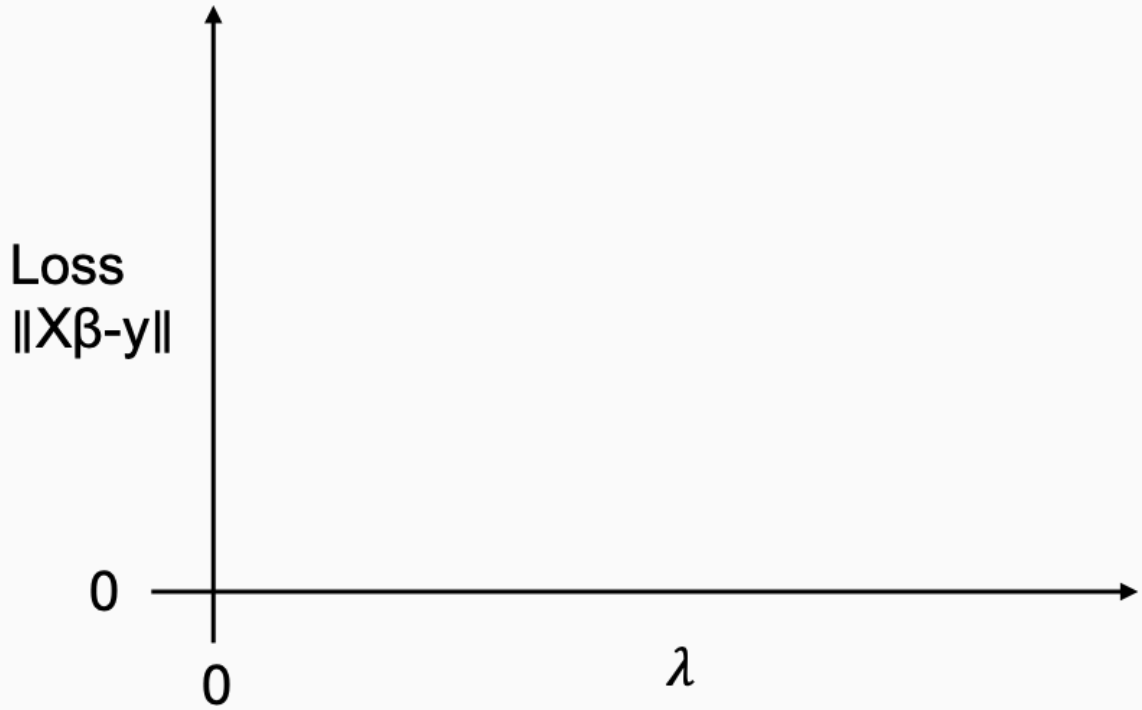
- Ridge regression (ℓ_2): $Reg(\beta) = \lambda \|\beta\|_2^2$
- LASSO (least absolute shrinkage and selection operator) (ℓ_1): $Reg(\beta) = \lambda \|\beta\|_1$
- Elastic net: $Reg(\beta) = \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$

Note that $\arg\min_\beta \left[L(\beta) + Reg(\beta) \right] \neq \arg\min_\beta \left[L(\beta) \right]$

Ridge regularization: perspective 1

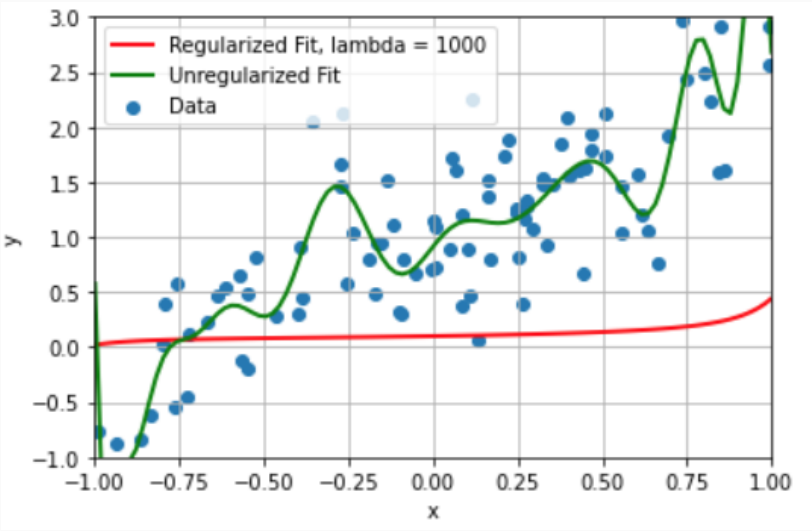
$$\text{Ridge regression: } \min_{\boldsymbol{\beta}} \left(\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right).$$

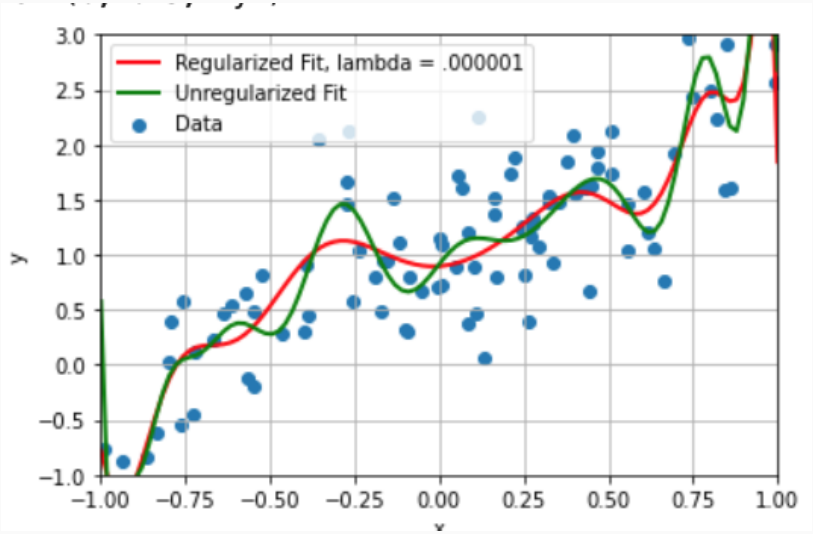
- As $\lambda \to \infty$, we expect $\|\boldsymbol{\beta}\|_2^2 \to 0$ and $\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 \to \|\mathbf{y}\|_2^2$.
- By choosing different values of λ we have models of varying accuracy/complexity.

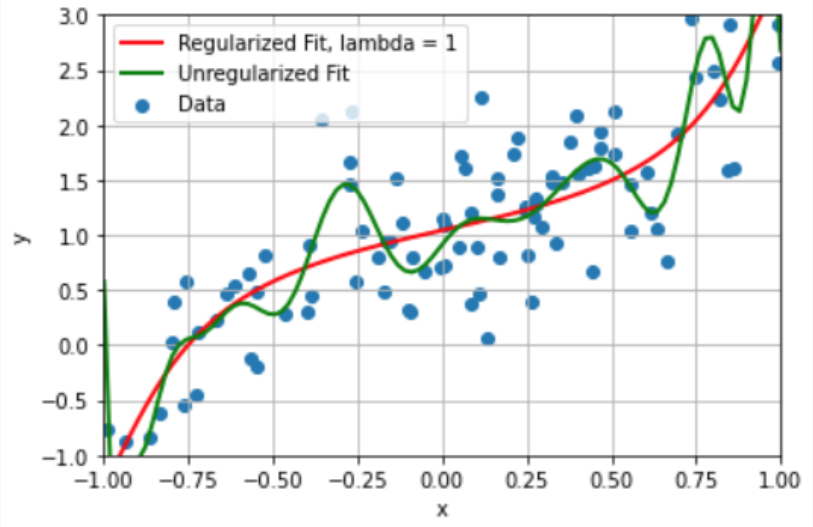


Polynomial examples

Fit degree 20 polynomial with varying levels of regularization.



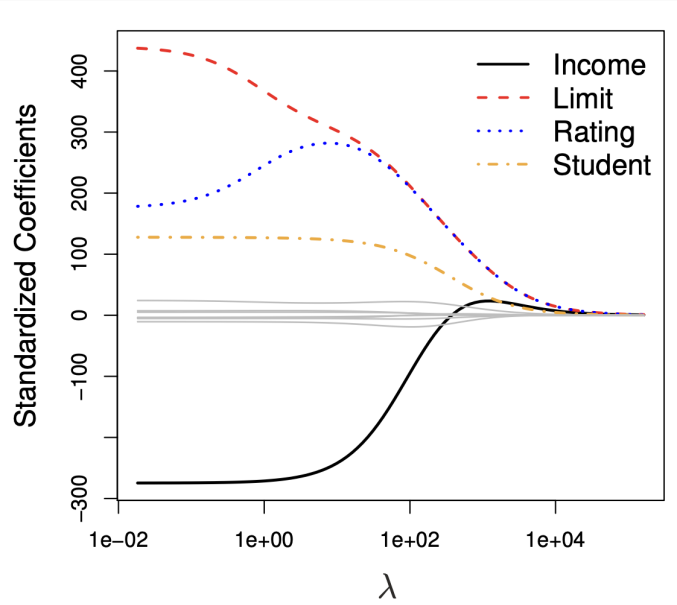




Ridge regularization: perspective 2

$$\text{Ridge regression: } \min_{\boldsymbol{\beta}} \left(\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right).$$

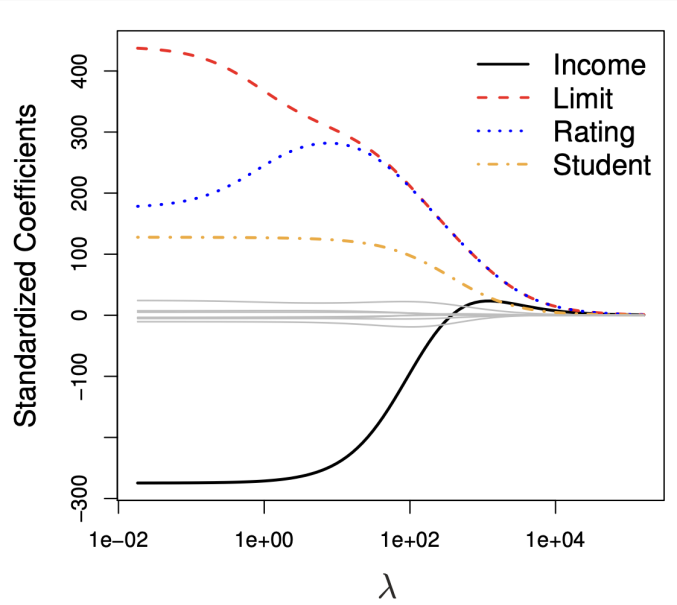
- As $\lambda \to \infty$, we expect $\|\boldsymbol{\beta}\|_2^2 \to 0$ and $\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 \to \|\mathbf{y}\|_2^2$.
- Feature selection methods attempt to set many coordinates in $\boldsymbol{\beta}$ to 0. Ridge regularizations encourages coordinates to be close to zero.



Ridge regularization

Ridge regression: $\min_\beta \|\mathbf{X}\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_2^2$.

- As $\lambda \to \infty$, we expect $\|\beta\|_2^2 \to 0$ and $\|\mathbf{X}\beta - \mathbf{y}\|_2^2 \to \|\mathbf{y}\|_2^2$.
- Feature selection methods attempt to set many coordinates in β to 0. Ridge regularizations encourages coordinates to be small.



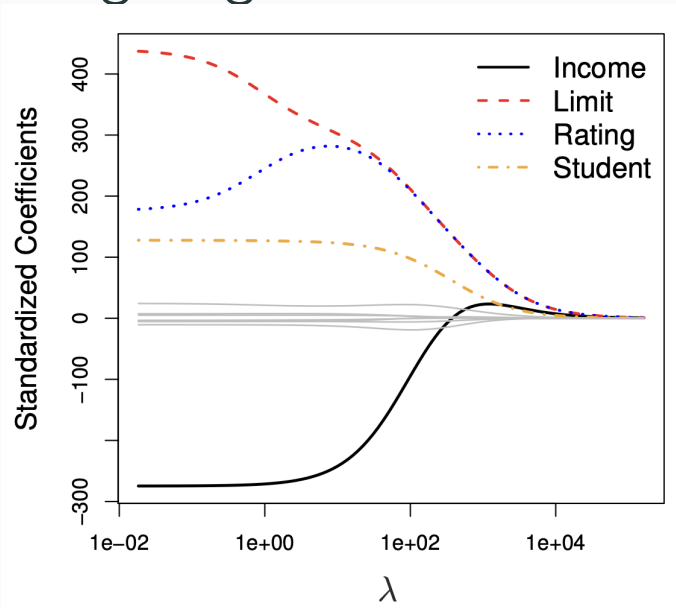
Ridge regularization

How do we minimize: $L_R(\boldsymbol{\beta}) = \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2$?

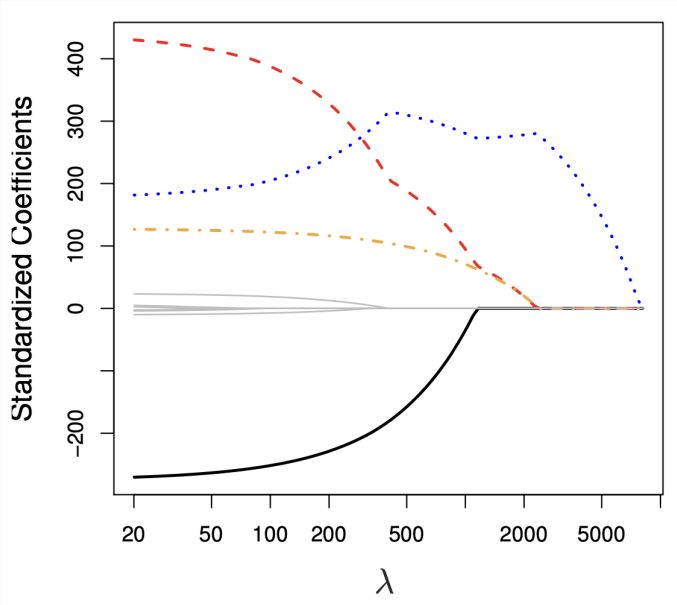
Lasso regularization

Lasso regularization: $\min_\beta \|\mathbf{X}\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_1$.

- As $\lambda \to \infty$, we expect $\|\beta\|_1 \to 0$ and $\|\mathbf{X}\beta - \mathbf{y}\|_2^2 \to \|\mathbf{y}\|_2^2$.
- Typically encourages subset of β_i's to go to zero, in contrast to ridge regularization.



Ridge regularization



Lasso Regularization

Lasso regularization

Pros:

- Simpler, more interpretable model.
- More intuitive reduction in model order.

Cons:

- No closed form solution because $\|\beta\|_1$ is not differentiable.
- Can be solved with iterative methods, but generally not as quickly as ridge regression.

Regularization

Notes:

- Model selection/cross validation used to choose optimal scaling λ on $\lambda\|\boldsymbol{\beta}\|_2^2$ or $\lambda\|\boldsymbol{\beta}\|_1$.
- Often grid search for best parameters is performed in "log space". E.g. consider $[\lambda_1, \ldots, \lambda_q] = 1.5^{[-4,-3,-2,-1,-0,1,2,3,4]}$.
- Regularization methods are not invariant to data scaling. Typically when using regularization we mean center and scale columns to have unit variance.

The bayesian/probabilistic modeling perspective

Classification setup

- **Data Examples:** $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$
- **Target:** $y_1, \ldots, y_n \in \{0, 2, \ldots, q-1\}$ when there are q classes.
  - Binary Classification: $q = 2$, so each $y_i \in \{0, 1\}$.
  - Multi-class Classification: $q > 2$. [1]

[1] Note that there is also multi-label classification where each data example may belong to more than one class.

Classification examples

- Medical diagnosis from MRI: 2 classes.
- MNIST digits: 10 classes.
- Full Optical Character Regonition: 100s of classes.
- ImageNet challenge: 21,000 classes.

Running example today: Email Spam Classification.

Classification

Classification can (and often is) solved using the same **loss-minimization framework** we saw for regression.

We won't see that today! We're going to use classification as a window into another way of thinking about machine learning.

Will give an interesting new justifications for tools like regularization. Will also give us an approach for generative ML.

Rest of Today: ML from a **Probabilistic Modeling/Bayesian Perspective**.

Probabilistic modeling

In a Bayesian or Probabilistic approach to machine learning we always start by conjecturing a

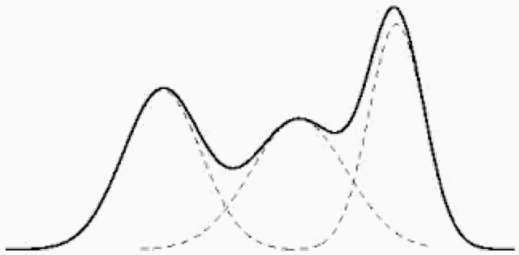
probabilistic model

that plausibly could have generated our data.

- The model guides how we make predictions.
- The model typically has unknown parameters $\vec{\theta}$ and we try to find the most reasonable parameters based on *observed* data (more on this later in lecture).

Probabilistic modeling

Typically we try to keep things simple!



✓

X

X

Probabilistic modeling

Exercise: Come up with a probabilistic model for the following data set $(x_1, y_1), \ldots, (x_n, y_n)$.

- For n **NYC apartments**: each x_i is the size of the apartment in square feet. Each y_i is the monthly rent in dollars.

What are the unknown parameters of your model. What would be a guess for their values? How would you confirm or refine this guess using data?

Probabilistic modeling

Dataset: $(x_1, y_1), \ldots, (x_n, y_n)$

Description: For n **NYC apartments**: each x_i is the size of the apartment in square feet. Each y_i is the monthly rent in dollars.

Model:

Probabilistic modeling

Dataset: $(x_1, y_1), \ldots, (x_n, y_n)$

Description: For n **students**: each $x_i \in \{Fresh., Soph., Jun., Sen.\}$ indicating class year. Each $y_i \in \{0, 1\}$ with zero indicating the student has not taken machine learning, one indicating they have.

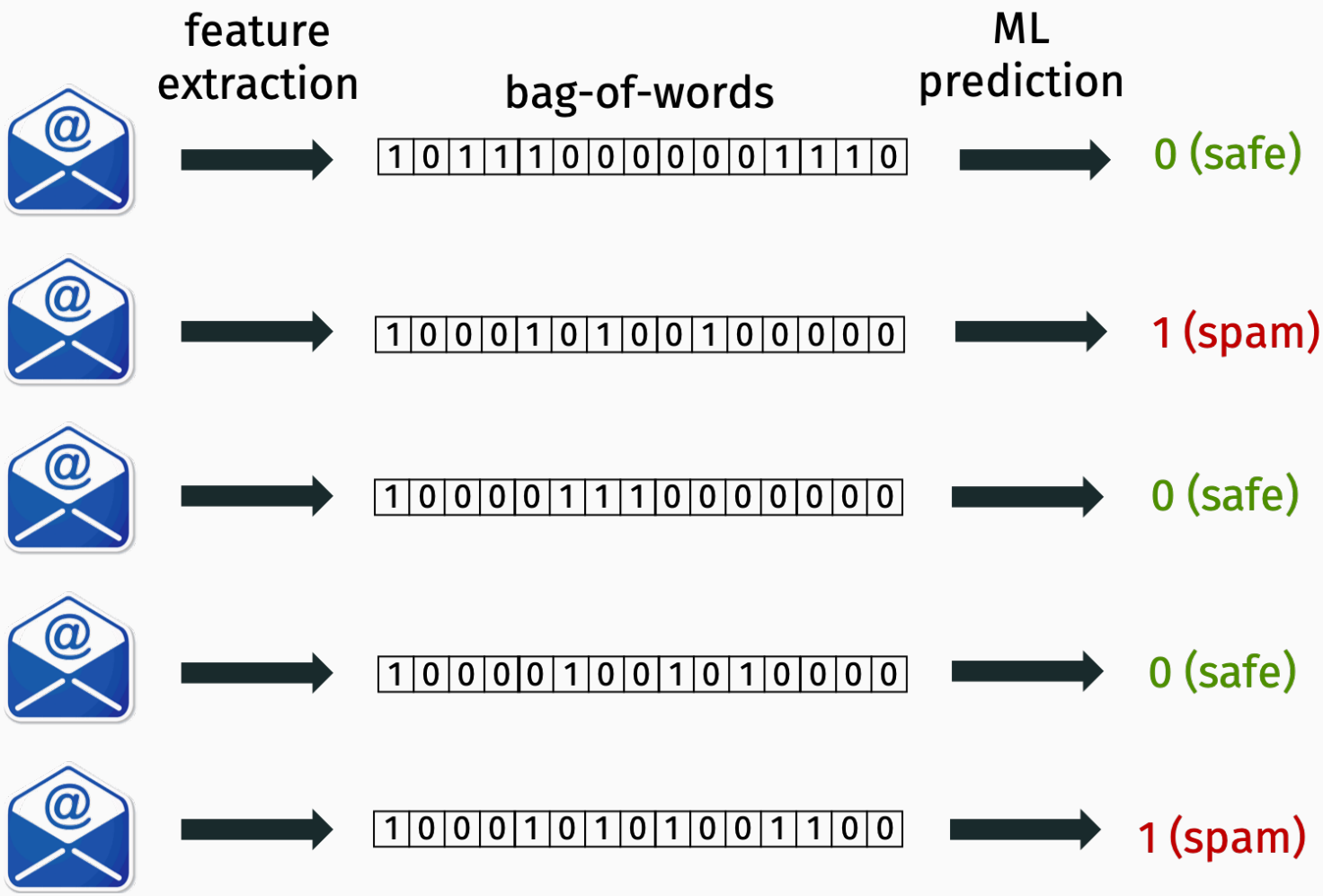
Model:

Naive bayes classifier

Goal:

- Build a probabilistic model for a binary classification problem.
- Estimate parameters of the model.
- From the model derive a classification rule for future predictions (the **Naive Bayes Classifier**).

Spam prediction



Both target labels and data vectors are binary.

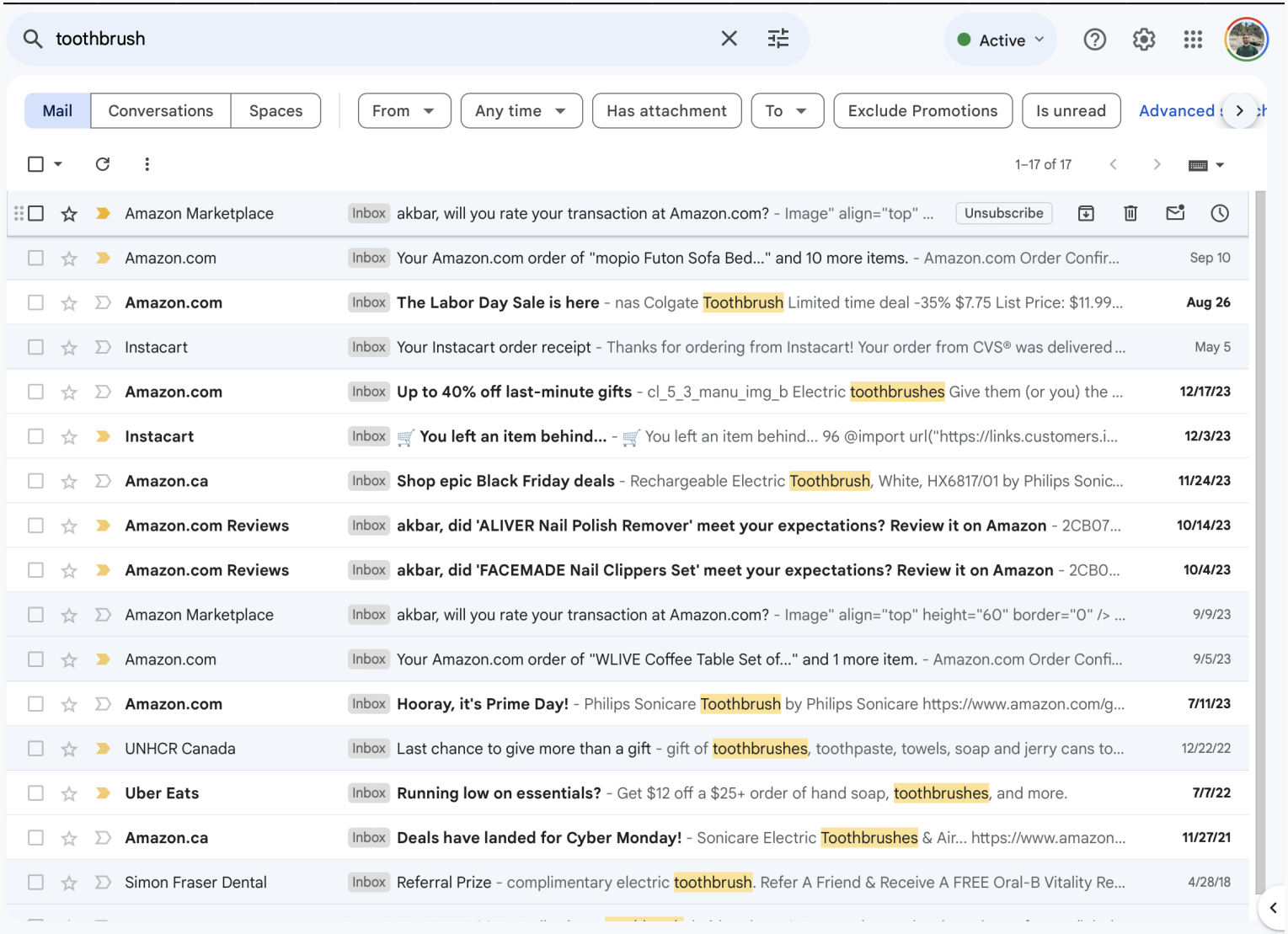
Email model

Let's create a model that generates spam and non-spam emails.
Observation: Since bag-of-words features don't care about word order, our model does not need to either.

- Common approach: assign a probability $p_i \in [0, 1]$ to word i. Set $\mathbf{x}_i = 1$ with probability p_i, $\mathbf{x}_i = 0$ with probability $1 - p_i$.

$p_{\text{the}} =$ $p_{\text{calendar}} =$ $p_{\text{toothbrush}} =$

Email model


toothbrush
Active
Mail
Conversations
Spaces
From
Any time
Has attachment
To
Exclude Promotions
Is unread
Advanced
1–17 of 17
Amazon Marketplace Inbox akbar, will you rate your transaction at Amazon.com? - Image" align="top" ... Unsubscribe
Amazon.com Inbox Your Amazon.com order of "mopio Futon Sofa Bed..." and 10 more items. - Amazon.com Order Confir... Sep 10
Amazon.com Inbox The Labor Day Sale is here - nas Colgate Toothbrush Limited time deal -35% $7.75 List Price: $11.99... Aug 26
Instacart Inbox Your Instacart order receipt - Thanks for ordering from Instacart! Your order from CVS® was delivered ... May 5
Amazon.com Inbox Up to 40% off last-minute gifts - cl_5_3_manu_img_b Electric toothbrushes Give them (or you) the ... 12/17/23
Instacart Inbox 🛒 You left an item behind... - 🛒 You left an item behind... 96 @import url("https://links.customers.i... 12/3/23
Amazon.ca Inbox Shop epic Black Friday deals - Rechargeable Electric Toothbrush, White, HX6817/01 by Philips Sonic... 11/24/23
Amazon.com Reviews Inbox akbar, did 'ALIVER Nail Polish Remover' meet your expectations? Review it on Amazon - 2CB07... 10/14/23
Amazon.com Reviews Inbox akbar, did 'FACEMADE Nail Clippers Set' meet your expectations? Review it on Amazon - 2CB0... 10/4/23
Amazon Marketplace Inbox akbar, will you rate your transaction at Amazon.com? - Image" align="top" height="60" border="0" /> ... 9/9/23
Amazon.com Inbox Your Amazon.com order of "WLIVE Coffee Table Set of..." and 1 more item. - Amazon.com Order Confi... 9/5/23
Amazon.com Inbox Hooray, it's Prime Day! - Philips Sonicare Toothbrush by Philips Sonicare https://www.amazon.com/g... 7/11/23
UNHCR Canada Inbox Last chance to give more than a gift - gift of toothbrushes, toothpaste, towels, soap and jerry cans to... 12/22/22
Uber Eats Inbox Running low on essentials? - Get $12 off a $25+ order of hand soap, toothbrushes, and more. 7/7/22
Amazon.ca Inbox Deals have landed for Cyber Monday! - Sonicare Electric Toothbrushes & Air... https://www.amazon... 11/27/21
Simon Fraser Dental Inbox Referral Prize - complimentary electric toothbrush. Refer A Friend & Receive A FREE Oral-B Vitality Re... 4/28/18

Email model

How can we make this model richer when we take spam into account?

- Different words tend to be more or less frequent in spam or regular emails.

Not Spam	**Spam**
$p_{\text{won}} =$	$p_{\text{won}} =$
$p_{\$} =$	$p_{\$} =$
$p_{student} =$	$p_{student} =$

Probabilistic model for email

Probabilistic model for (bag-of-words, label) pair $(\mathbf{x}, y)$:

- Set $y = 0$ with probability p_0, $y = 1$ with probability $p_1 = 1 - p_0$.
  - p_0 is probability an email is not spam (e.g. 99%).
  - p_1 is probability an email is spam (e.g. 1%).
- If $y = 0$, for each i, set $x_i = 1$ with prob. p_{i0}.
- If $y = 1$, for each i, set $x_i = 1$ with prob. p_{i1}.

Unknown model parameters:

- p_0, p_1,
- $p_{10}, p_{20}, \ldots p_{d0}$, one for each of the d vocabulary words.
- $p_{11}, p_{21}, \ldots p_{d1}$, one for each of the d vocabulary words.

How would you estimate these parameters?

Parameter estimation

Reasonable way to set parameters:

- Set p_0 and p_1 to the empirical fraction of not spam/spam emails.
- For each word i, set p_{i0} to the empirical probability word i appears in a non-spam email.
- For each word i, set p_{i1} to the empirical probability word i appears in a spam email.

Estimating these parameters from previous data examples is the only "training" we will do.

Done with modeling

on to prediction

Probability review

- **Probability:** p(x) – the probability event x happens.
- **Joint probability:** p(x,y) – the probability that event x and event y happen.
- **Conditional Probability** $p(x \mid y)$ – the probability x happens given that y happens.

$$p(x|y) =$$

Bayes theorem/rule

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Proof:

Classification rule

Given unlabeled input $(\mathbf{w}, \_\_\_)$, choose the label $y \in \{0, 1\}$ which is most likely given the data. Recall $\mathbf{w} = [0, 0, 1, \ldots, 1, 0]$.

Classification rule: maximum a posterior (MAP) estimate.

Step 1. Compute:

- $p(y = 0 \mid \mathbf{w})$: prob. $y = 0$ given observed data vector $\mathbf{w}$.
- $p(y = 1 \mid \mathbf{w})$: prob. $y = 1$ given observed data vector $\mathbf{w}$.

Step 2. Output: 0 or 1 depending on which probability is larger.

$p(y = 0 \mid \mathbf{w})$ and $p(y = 1 \mid \mathbf{w})$ are called **posterior** probabilities.

Evaluating the posterior

How to compute the posterior? **Bayes rule!**

$$p(y = 0 \mid \mathbf{w}) = \frac{p(\mathbf{w} \mid y = 0)p(y = 0)}{p(\mathbf{w})} \tag{1}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} \tag{2}$$

- **Prior:** Probability in class 0 prior to seeing any data.
- **Posterior:** Probability in class 0 after seeing the data.

Evaluating the posterior

Goal is to determine which is larger:

$$p(y = 0 \mid \mathbf{w}) = \frac{p(\mathbf{w} \mid y = 0)p(y = 0)}{p(\mathbf{w})} \qquad \text{vs.}$$

$$p(y = 1 \mid \mathbf{w}) = \frac{p(\mathbf{w} \mid y = 1)p(y = 1)}{p(\mathbf{w})}$$

- We can ignore the evidence $p(\mathbf{w})$ since it is the same for both sides!
- $p(y = 0)$ and $p(y = 1)$ already known (computed from training data). These are our computed parameters p_0, p_1.
- $p(\mathbf{w} \mid y = 0) = ?$ $p(\mathbf{w} \mid y = 1) = ?$

Evaluating the posterior

Consider the example $\mathbf{w} = [0, 1, 1, 0, 0, 0, 1, 0]$.

Recall that, under our model, index i is 1 with probability p_{i0} if we are not spam, and 1 with probability p_{i1} if we are spam .

$$p(\mathbf{w} \mid y = 0) =$$

$$p(\mathbf{w} \mid y = 1) =$$

Final Naive Bayes Classifier

Training/Modeling: Use existing data to compute:

- $p_0 = p(y = 0), p_1 = p(y = 1)$
- For all i compute:
  - $p_{i0} = p(x_i = 1 \mid y = 0)$ and $(1 - p_{i0}) = p(x_i = 0 \mid y = 0)$
  - $p_{i1} = p(x_i = 1 \mid y = 1)$ and $(1 - p_{i1}) = p(x_i = 0 \mid y = 1)$

Prediction:

- For new input $\mathbf{w}$:
  - Compute $p(\mathbf{w} \mid y = 0) = \prod_i p(w_i \mid y = 0)$
  - Compute $p(\mathbf{w} \mid y = 1) = \prod_i p(w_i \mid y = 1)$
- Return

$$\arg\max \left[p\left(\mathbf{w} \mid y = 0\right) \cdot p\left(y = 0\right), p\left(\mathbf{w} \mid y = 1\right) \cdot p\left(y = 1\right) \right].$$

Other applications of
the bayesian perspective

Bayesian regression

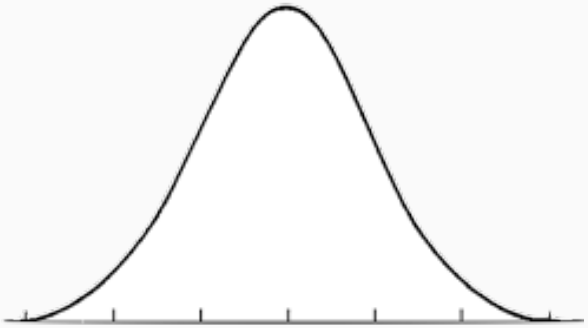
The Bayesian view offers an interesting alternative perspective on many machine learning techniques.

Example: Linear Regression.

Probabilistic model:

$$y = \langle \mathbf{x}, \boldsymbol{\beta} \rangle + \eta$$

where the η drawn from $N(0, \sigma^2)$ is **random Gaussian noise**.



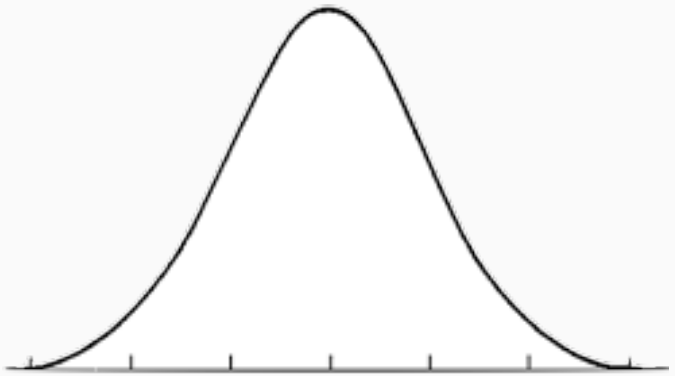
$$Pr(\eta = z) \sim$$

The symbol $\sim$ means "is proportional to".

Gaussian distribution refresher

Names for the same thing: Normal distribution, Gaussian distribution, bell curve.

Parameterized by mean μ and variance σ^2.

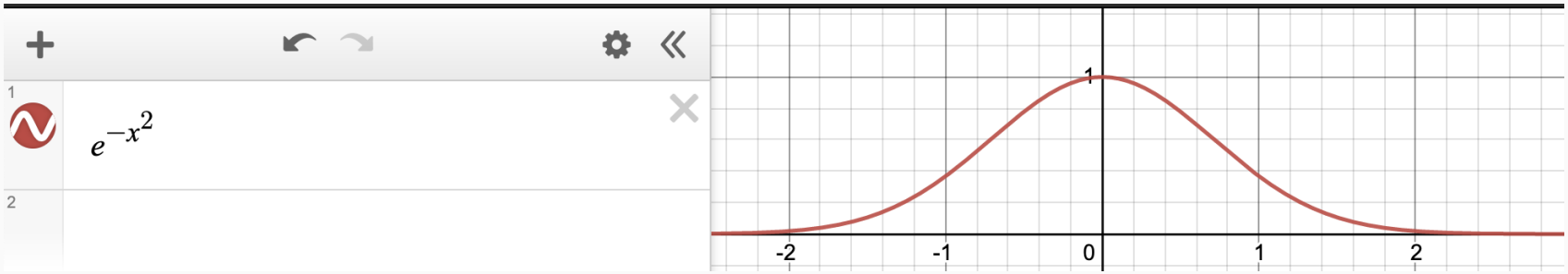


η is a continuous random variable, so it has a probability density function $p(\eta)$ with $\int_{-\infty}^{\infty} p(\eta)d\eta = 1$

$$p(\eta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{\eta-\mu}{\sigma})^2}$$

Gaussian distribution refresher

The important thing to remember is that the the PDF falls off exponentially as we move further from the mean.



The normalizing constant in front 1/2, etc. don't matter so much.

Quick check

Example: Linear Regression.

Probabilistic model:

$$y = \langle \mathbf{x}, \boldsymbol{\beta} \rangle + \eta$$

where the η drawn from $N(0, \sigma^2)$ is **random Gaussian noise**. The noise is independent for different inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

Bayesian regression

How should we select β for our model?

Also use a Bayesian approach!

Choose β to maximize:

$$\text{posterior} = \Pr(\beta \mid \mathbf{X}, \mathbf{y}) = \frac{\Pr(\mathbf{X}, \mathbf{y} \mid \beta)\Pr(\beta)}{\Pr(\mathbf{X}, \mathbf{y})} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}.$$

In this case, we don't have a prior – no values of β are inherently more likely than others.

Choose β to maximize just the likelihood:

$$\frac{\Pr(\mathbf{X}, \mathbf{y} \mid \beta)\cancel{\Pr(\beta)}}{\cancel{\Pr(\mathbf{X}, \mathbf{y})}} = \frac{\text{likelihood} \times \cancel{\text{prior}}}{\cancel{\text{evidence}}}.$$

This is called the **maximum likelihood estimate**.

Fixed design linear regression

Often we think of $\mathbf{X}$ as fixed and deterministic, and only $\mathbf{y}$ is generated at random in the model. This is called the fixed design setting. Can also consider a randomized design setting, but it is slightly more complicated.

In the fixed design setting our task of maximizing $\Pr(\mathbf{X}, \mathbf{y} \mid \boldsymbol{\beta})$ simplifies to maximizing

$$\max_{\boldsymbol{\beta}} \Pr(\mathbf{y} \mid \boldsymbol{\beta})$$

Maximum likelihood estimate

Data:

$$\mathbf{X} = \begin{bmatrix} — & \mathbf{x}_1 & — \\ — & \mathbf{x}_2 & — \\ & \vdots & \\ — & \mathbf{x}_n & — \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Model: $y_i = \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle + \eta_i$ where $p(\eta_i = z) \sim e^{-z^2/2\sigma^2}$ and $\eta_1, \ldots, \eta_n$ are independent.

$$\Pr(\mathbf{y} \mid \boldsymbol{\beta}) \sim$$

Log likelihood

Easier to work with the **log likelihood**:

$$\begin{aligned}
\arg\max_{\beta} \Pr(\mathbf{X}, \mathbf{y} \mid \beta) &= \arg\max_{\beta} \prod_{i=1}^{n} e^{-(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 / 2\sigma^2} \\
&= \arg\max_{\beta} \ \log \left(\prod_{i=1}^{n} e^{-(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 / 2\sigma^2} \right) \\
&= \arg\max_{\beta} \sum_{i=1}^{n} -(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 / 2\sigma^2 \\
&= \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \langle \mathbf{x}_i, \beta \rangle)^2 .
\end{aligned}$$

Maximum likelihood estimator

Conclusion: Choose β to minimize:

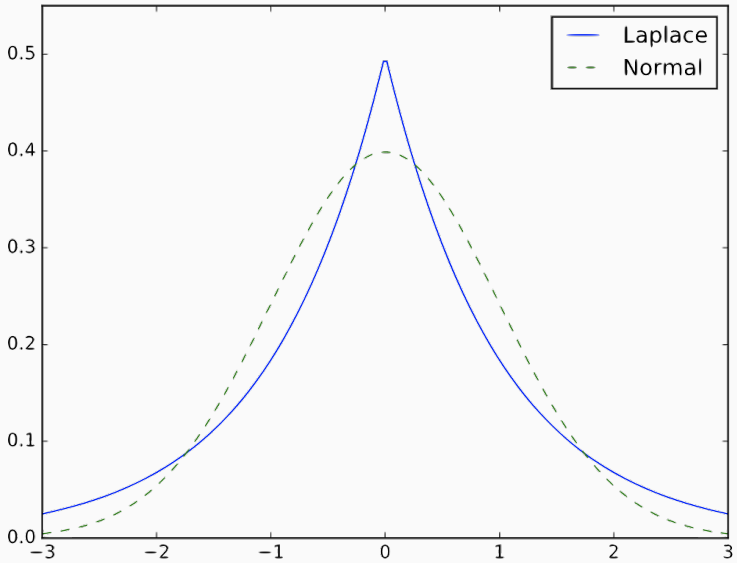
$$\sum_{i=1}^{n}(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 = \|\mathbf{y} - \mathbf{X}\beta\|_2^2.$$

This is a completely different justification for squared loss!

Minimizing the ℓ_2 loss is optimal in a certain sense when you assume your data follows a linear model with i.i.d. Gaussian noise.

Bayesian regression

If we had modeled our noise η as Laplace noise, we would have found that minimizing $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_1$ was optimal.



$$Pr(\eta = z) \sim$$

Laplace noise has "heavier tails", meaning that it results in more outliers.

This is a completely different justification for ℓ_1 loss.

Bayesian regularization

We can add another layer of probabilistic modeling by also assuming $\boldsymbol{\beta}$ is random and comes from some distribution, which encodes our prior belief on what the parameters are.

Maximum a posteriori (MAP estimation):

$$\Pr(\boldsymbol{\beta} \mid \mathbf{X}, \mathbf{y}) = \frac{\Pr(\mathbf{X}, \mathbf{y} \mid \boldsymbol{\beta}) \Pr(\boldsymbol{\beta})}{\Pr(\mathbf{X}, \mathbf{y})}.$$

Assume values in $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_d]$ come from some distribution.

- **Common model:** Each β_i drawn from $N(0, \gamma^2)$, i.e. normally distributed, independent.
- Encodes a belief that we are unlikely to see models with very large coefficients.

Bayesian regularization

Goal: choose $\boldsymbol{\beta}$ to maximize:

$$\Pr(\boldsymbol{\beta} \mid \mathbf{X}, \mathbf{y}) = \frac{\Pr(\mathbf{X}, \mathbf{y} \mid \boldsymbol{\beta}) \Pr(\boldsymbol{\beta})}{\Pr(\mathbf{X}, \mathbf{y})}.$$

- We can still ignore the "evidence" term $\Pr(\mathbf{X}, \mathbf{y})$ since it is a constant that does not depend on $\boldsymbol{\beta}$.
- $\Pr(\boldsymbol{\beta}) = \Pr(\beta_1) \cdot \Pr(\beta_2) \cdot \ldots \cdot \Pr(\beta_d)$
- $\Pr(\boldsymbol{\beta}) \sim$

Bayesian regularization

Easier to work with the **log likelihood**:

$$\begin{aligned}
&\arg\max_{\beta} \Pr(\mathbf{X}, \mathbf{y} \mid \beta) \cdot \Pr(\beta) \\
&= \arg\max_{\beta} \prod_{i=1}^{n} e^{-(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 / 2\sigma^2} \cdot \prod_{i=1}^{n} e^{-(\beta_i)^2 / 2\gamma^2} \\
&= \arg\max_{\beta} \sum_{i=1}^{n} -(y_i - \langle \mathbf{x}_i, \beta \rangle)^2 / 2\sigma^2 + \sum_{i=1}^{d} -(\beta_i)^2 / 2\gamma^2 \\
&= \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \langle \mathbf{x}_i, \beta \rangle)^2 + \frac{\sigma^2}{\gamma^2} \sum_{i=1}^{d} (\beta_i)^2 / \sigma^2.
\end{aligned}$$

Choose β to minimize $\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \frac{\sigma^2}{\gamma^2}\|\beta\|_2^2$.

Completely different justification for ridge regularization!

Bayesian regularization

Test your intuition: What modeling assumption justifies LASSO regularization: $\min \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$?