

# CS-GY 6923: Lecture 3

## Regularization + Bayesian Perspective

---

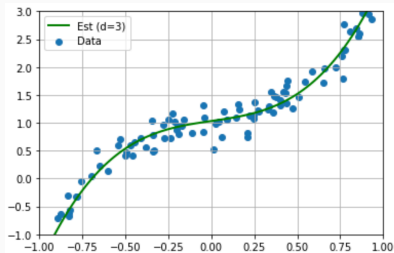
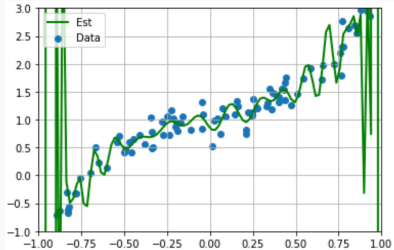
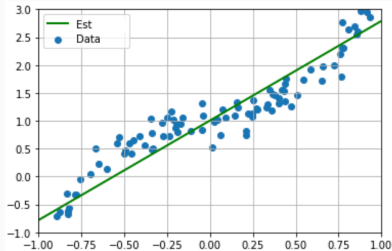
NYU Tandon School of Engineering, Prof. Christopher Musco

## Model selection:

- Train models  $f^{(1)}_{\theta_1}, \dots, f^{(q)}_{\theta_q}$  independently on training data to find optimal parameters  $\theta_1^*, \dots, \theta_q^*$ .
- Check loss  $L_{\text{test}}(f_1, \theta_1^*), \dots, L_{\text{test}}(f_q, \theta_q^*)$  on test data.
- Select mode with lowest test loss.

Can we 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 the sweet spot between a good fit, and not overfitting.

## LAST CLASS + LAB



Underfit, overfit, just right.

## COMMENT ON NUMERICAL ISSUES

In the lab we had you use `numpy.polynomial.polynomial`. Last class, however, we discussed how we could use multiple linear regression instead. If our point to fit at are  $x_1, \dots, x_n \in [-1, 1]$ , would construct the data matrix:

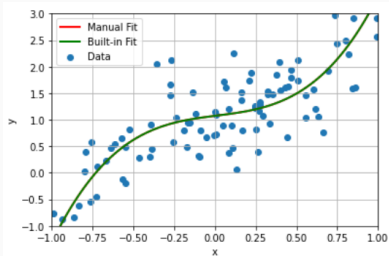
$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

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

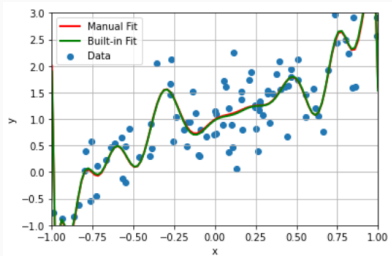
# COMMENT ON NUMERICAL ISSUES

```
# 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

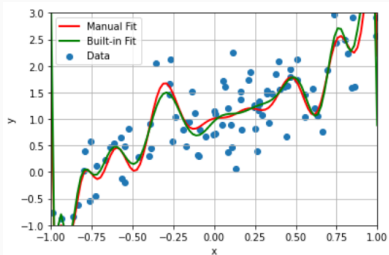


Degree 22

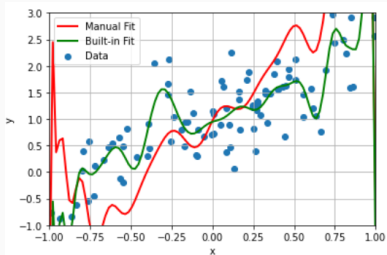
# COMMENT ON NUMERICAL ISSUES

```
# 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

## COMMENT 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
```

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

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

## REGULARIZATION

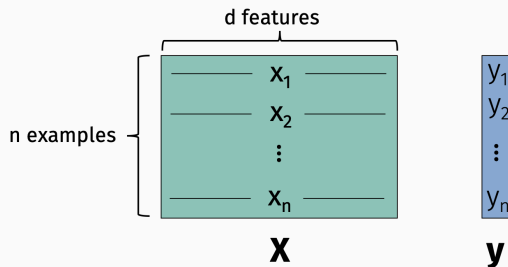
## OVER-PARAMETERIZED MODELS

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

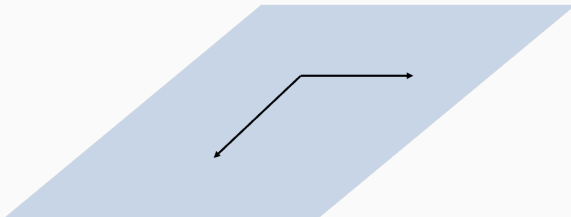
**Example:** Linear regression model where  $d \geq n$ .



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

## HIGH DIMENSIONAL LINEAR MODELS

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



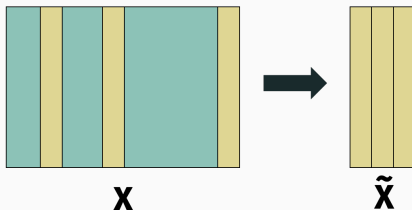
I.e., can find some coefficients so that  
$$\beta_1 \mathbf{x}^{(1)} + \dots + \beta_q \mathbf{x}^{(q)} = \mathbf{X}\boldsymbol{\beta} = \mathbf{y}.$$

## 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  $X$  is used in single variate regression.

Any potential limitations of this approach?

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

**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.

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

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

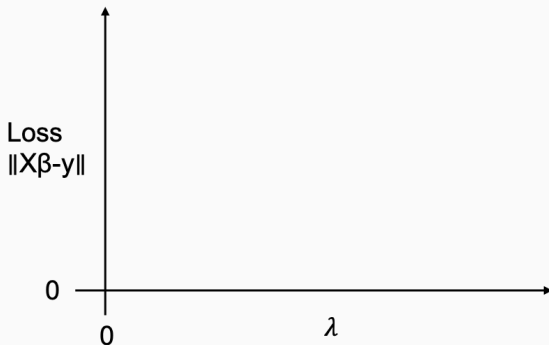
- Ridge regression ( $\ell_2$ ):  $\text{Reg}(\beta) = \lambda \|\beta\|_2^2$
- LASSO (least absolute shrinkage and selection operator) ( $\ell_1$ ):  $\text{Reg}(\beta) = \lambda \|\beta\|_1$
- Elastic net:  $\text{Reg}(\beta) = \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$

Note that  $\arg \min_{\beta} [L(\beta) + \text{Reg}(\beta)] \neq \arg \min_{\beta} [L(\beta)]$

## RIDGE REGULARIZATION: PERSPECTIVE 1

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

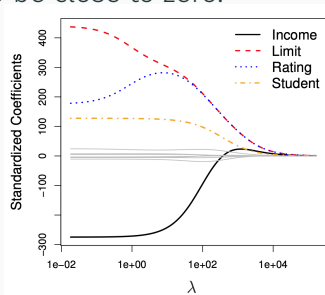
- As  $\lambda \rightarrow \infty$ , we expect  $\|\beta\|_2^2 \rightarrow 0$  and  $\|X\beta - y\|_2^2 \rightarrow \|y\|_2^2$ .
- By choosing different values of  $\lambda$  we have models of varying accuracy/complexity.



## RIDGE REGULARIZATION:PERSPECTIVE 2

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

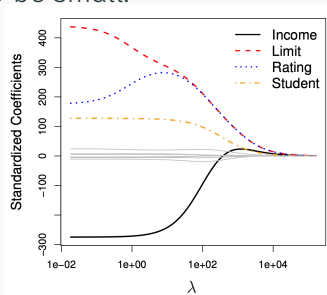
- As  $\lambda \rightarrow \infty$ , we expect  $\|\beta\|_2^2 \rightarrow 0$  and  $\|\mathbf{X}\beta - \mathbf{y}\|_2^2 \rightarrow \|\mathbf{y}\|_2^2$ .
- Feature selection methods attempt to set many coordinates in  $\beta$  to 0. Ridge regularizations encourages coordinates to be close to zero.



# RIDGE REGULARIZATION

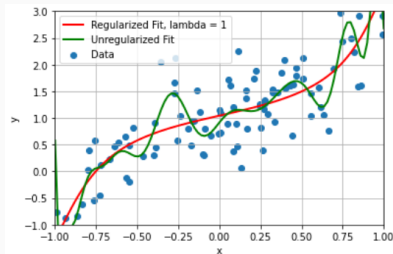
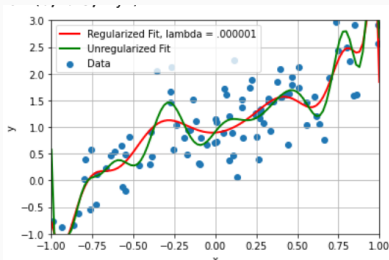
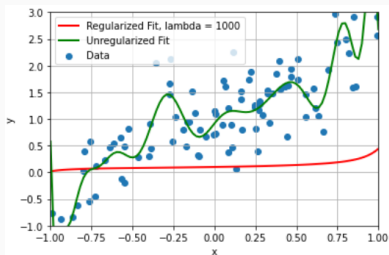
Ridge regression:  $\min_{\beta} \|X\beta - y\|_2^2 + \lambda \|\beta\|_2^2$ .

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



# POLYNOMIAL EXAMPLES

Fit degree 20 polynomial with varying levels of regularization.

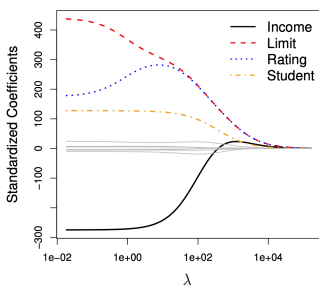


How do we minimize:  $L_R(\beta) = \|\mathbf{X}\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_2^2$ ?

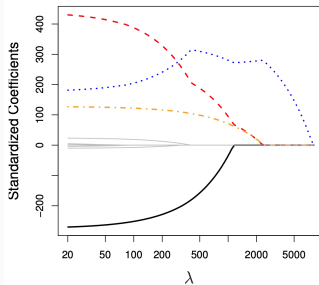
# LASSO REGULARIZATION

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

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



Ridge 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.

## Notes:

- Model selection/cross validation used to choose optimal scaling  $\lambda$  on  $\lambda\|\beta\|_2^2$  or  $\lambda\|\beta\|_1$ .
- Often grid search for best parameters is performed in “log space”. E.g. consider  $[\lambda_1, \dots, \lambda_q] = 1.5^{[-4, -3, -2, -1, -0, 1, 2, 3, 4]}$ .
- 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

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

---

<sup>1</sup>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 Recognition: 100s of classes.
- ImageNet challenge: 21,000 classes.

Running example today: **Email Spam 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 new an interesting justifications for tools like regularization. Will also give us an approach for generative ML.

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

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

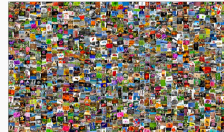
## probabilistic model

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!



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

- 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?

**Dataset:**  $(x_1, y_1), \dots, (x_n, y_n)$

**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:**

**Dataset:**  $(x_1, y_1), \dots, (x_n, y_n)$

**Description:** For  $n$  **students:** each

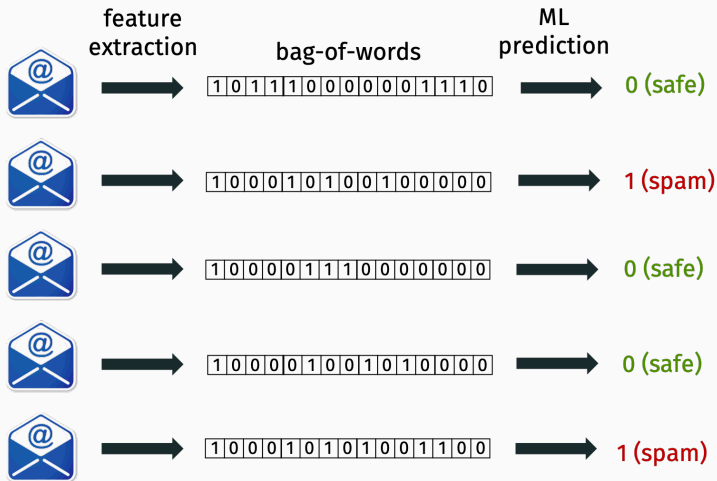
$x_i \in \{\text{Fresh.}, \text{Soph.}, \text{Jun.}, \text{Sen.}\}$  indicating class year. Each  $y_i \in \{0, 1\}$  with zero indicating the student has not taken machine learning, one indicating they have.

**Model:**

### 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.

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

The screenshot shows an email client interface with a search bar at the top containing the text "toothbrush". Below the search bar, there are several filter buttons: "Mail", "Messages", "Spaces", "From", "Any time", "Has attachment", "To", and "Exclude Promos". The "Mail" button is selected. Below the filters, there is a list of 13 email results. Each result is displayed in a row with a checkbox, a star icon, the sender's name, the email subject, and the date. The word "toothbrush" is highlighted in yellow in several of the email subjects.

Sender	Subject	Date
Aetna	Inbox Wow, Christophe, have you seen your benefits lately? - - Electric ...	10/8/22
Musco, Jenna	Inbox FW: Cargo e-bikes - , a toothbrush in 15 minutes," said Jolley. "W...	5/31/22
Amazon.com	Amazon Your Amazon.com order #114-7471064-3087458 - Amazon.com...	2/14/21
Christopher .. Jenna 3	Inbox first 2 week baby thoughts - keep our toothbrushes in the cabine...	12/29/20
NYU Office of Publi.	Inbox NYU Responds - View in Browser	4/9/20
Walgreens.com	10X everyday points = all treat, no trick! - To ensure delivery to your inbo...	10/16/19
Real Simple	Wayfair's Having a HUGE Sale Right Now—Up to 70% Off - -----...	3/17/19
Amazon.com	Amazon Your Amazon.com order - Amazon.com Order Confirmation ww...	2/3/19
Amazon.com	What's new with Alexa? - change my toothbrush in three months." - "Ale...	1/12/19
Amazon.com	Amazon Your Amazon.com order has shipped (#113-2545543-7095419) - ,	1/2/19
Debbie Goodwin	[csail-related] Holiday Giving to begin next week - Toiletries (toothbrush...	11/21/18
Real Simple	Our Favorite Cooling Bed Sheets Are \$20 Off for Real Simple Readers - -R	9/23/18
erica, Dani., Char. 3	Inbox Last Reminder re: routes - anything (toothbrush, sunblock etc.) w...	8/21/18

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

$$p_{\text{won}} =$$

$$p_{\S} =$$

$$p_{\text{student}} =$$

Spam

$$p_{\text{won}} =$$

$$p_{\S} =$$

$$p_{\text{student}} =$$

**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:**

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

**How would you estimate these parameters?**

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

$$p(x|y) =$$

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

Proof:

## CLASSIFICATION RULE

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

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.

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

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

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} \quad (2)$$

- **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:

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

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

- 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  $p_0, p_1$ .
- $p(\mathbf{w} \mid y = 0) = ?$   $p(\mathbf{w} \mid y = 1) = ?$

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:
  - $p_{i0} = p(x_i = 1 | y = 0)$  and  $(1 - p_{i0}) = p(x_i = 0 | y = 0)$
  - $p_{i1} = p(x_i = 1 | y = 1)$  and  $(1 - p_{i1}) = p(x_i = 0 | y = 1)$

**Prediction:**

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

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

OTHER APPLICATIONS OF  
THE BAYESIAN PERSPECTIVE

# BAYESIAN REGRESSION

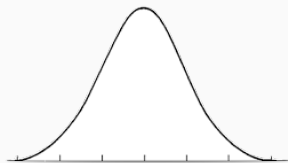
The Bayesian view offers an interesting alternative perspective on many machine learning techniques.

**Example:** Linear Regression.

**Probabilistic model:**

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

where the  $\eta$  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 same thing:** Normal distribution, Gaussian distribution, bell curve.

Parameterized by **mean  $\mu$**  and **variance  $\sigma^2$** .

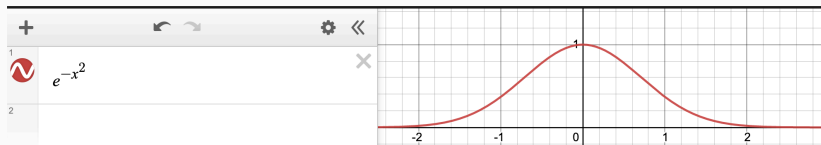


$\eta$  is a continuous random variable, so it has a probability density function  $p(\eta)$  with  $\int_{-\infty}^{\infty} p(\eta) d\eta = 1$

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

## 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.

**Example:** Linear Regression.

**Probabilistic model:**

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

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

## How should we select $\beta$ for our model?

Also use a Bayesian approach!

Choose  $\beta$  to maximize:

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

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

Choose  $\beta$  to maximize just the likelihood:

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

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

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  $\Pr(\mathbf{X}, \mathbf{y} \mid \beta)$  simplifies to maximizing

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

Data:

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

**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  $\eta_1, \dots, \eta_n$  are independent.

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

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

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

**Conclusion:** Choose  $\beta$  to minimize:

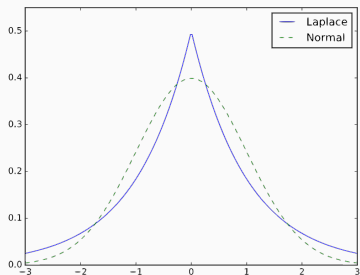
$$\sum_{i=1}^n (y_i - \langle \mathbf{x}_i, \beta \rangle)^2 = \|\mathbf{y} - \mathbf{X}\beta\|_2^2.$$

**This is a completely different justification for squared loss!**

Minimizing the  $\ell_2$  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  $\eta$  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  $\ell_1$  loss.

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

Return to **Maximum a posteriori (MAP estimation)**:

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

Assume values in  $\beta = [\beta_1, \dots, \beta_d]$  come from some distribution.

- **Common model:** Each  $\beta_i$  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.

**Goal:** choose  $\beta$  to maximize:

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

- We can still ignore the “evidence” term  $\Pr(X, y)$  since it is a constant that does not depend on  $\beta$ .
- $\Pr(\beta) = \Pr(\beta_1) \cdot \Pr(\beta_2) \cdot \dots \cdot \Pr(\beta_d)$
- $\Pr(\beta) \sim$

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

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

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

**Completely different justification for ridge regularization!**

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