CS-GY 6923: Lecture 3
Regularization + Bayesian Perspective

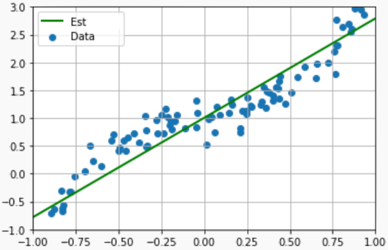
NYU Tandon School of Engineering, Prof. Christopher Musco

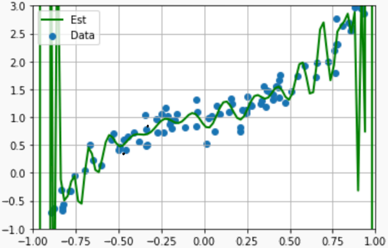
Model selection:

- 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 $L_{test}\left(f^{(1)}_{\boldsymbol{\theta}^*_1}\right), \ldots, L_{test}\left(f^{(1)}_{\boldsymbol{\theta}^*_q}\right)$ on test data.
- Select mode with lowest test lost.

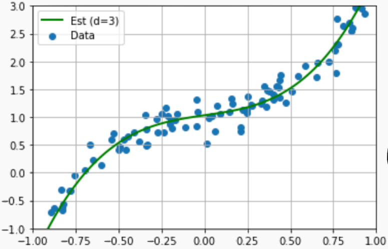
Can we used for arbitrary sets of models. Often used when you are not sure how “complex” your model should be for the data, and want to find the sweet spot between a good fit, and not overfitting.

β_1^*





β_{20}^*



β_3^*

β^*

0

$d \times 1$

Underfit, overfit, just right.

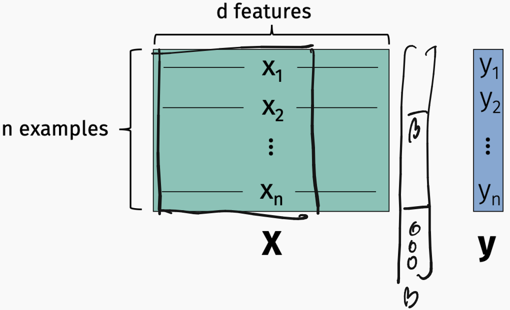
OVER-PARAMETERIZED MODELS

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

$\bar{X}\bar{\beta} = y \qquad \bar{\beta} = \bar{X}^{-1} y$

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



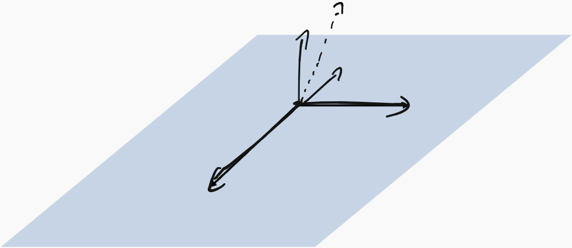
Can (almost) always find β so that $\mathbf{X}\beta = \mathbf{y}$ exactly.

Condition when this true:

As long as there are n linearly independent features.

Claim: For <u>almost all</u> 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.

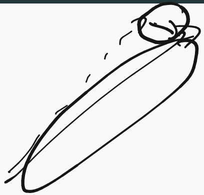
$n = 3$



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

$$\begin{bmatrix} 1 & X_1 & X_2 \\ 1 & & \\ 1 & & \\ 1 & & \\ 1 & & \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$$

$$\underbrace{\beta_1}_{y_{10}} x^{(1)} + \underline{\beta_2} x^{(2)} \approx y \qquad 10$$



- 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 if will be a sign of overfitting, as in the polynomial regression example.

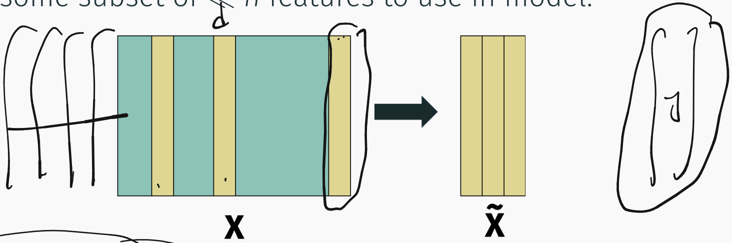
$$\|y\|_2^2$$

$$R^2 = 1 - \frac{\|X\beta - y\|_2^2}{\|y\|_2^2}$$

$$= 0 \qquad = 1$$

FEATURE SELECTION

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



$s_1, \ldots s_d$

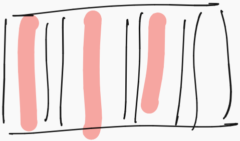
Filter method: Compute some metric for each feature, and select features with highest score.

- 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?)

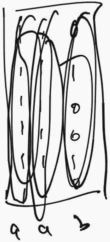
$q = 2 \ \ldots\ 10$

$$\frac{(d)(d-1)\ldots(d-q+1)}{q!} \approx d^q$$



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

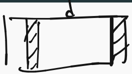
X



$$y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

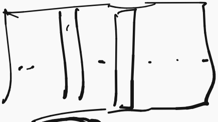
$a > b$

$$\beta = \begin{bmatrix} 100 \\ 10 \\ .5 \\ 200 \end{bmatrix}$$



$d + d-1 + \ldots + q$

$= 100 - 90$ $\quad O(d^2)$



Faster approach: Greedily select q features.

Stepwise Regression: $d + d-1 + \ldots d - q \approx qd$

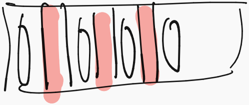
- **Forward:** Step 1: pick single feature that gives lowest loss. Step k: pick feature that when combined with previous $k-1$ chosen features gives lowest loss. $O(d(d-q))$
- **Backward:** Start with all of the features. Greedily eliminate those which have least impact on model performance.

$d-3$

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

$d \cdot (d-q)$

$\approx O(d)$



$d-3$ 4-variable regression problem?

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

$\beta^* = \min_{\beta} L(\beta)$

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

positive constant

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): $Reg(\beta) = \lambda \|\beta\|_1 = \sum_{i=1}^{d} |\beta_i|$
- Elastic net: $Reg(\beta) = \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$

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

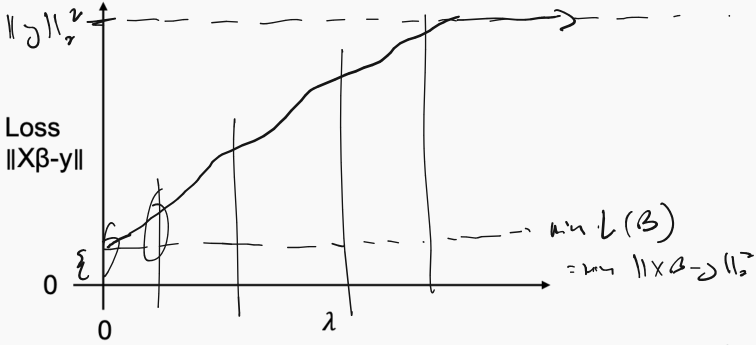
$\|y\|_2^2 := 100000$

$X\beta \to 0$

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

- As $\lambda \to \infty$, we expect $\|\beta\|_2^2 \to 0$ and $\|X\beta - y\|_2^2 \to \|y\|_2^2$.
- By choosing different values of λ we have models of varying levels of model fit.

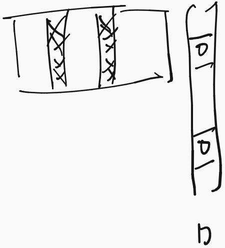
$\lambda := 1000000000$

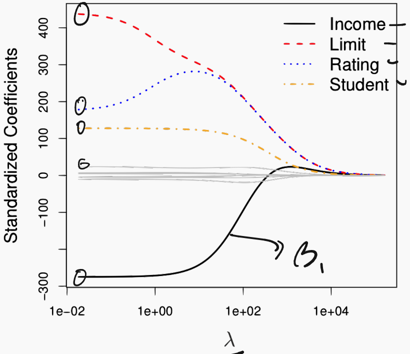


RIDGE REGULARIZATION

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

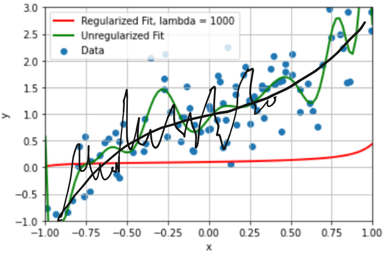
- As $\lambda \to \infty$, we expect $\|\beta\|_2^2 \to 0$ and $\|\mathbf{X}\beta - \mathbf{y}\|_2^2 \to \|\mathbf{y}\|_2^2$.
- Feature selection attempts to set many coordinates in β to 0. Regularization encourages coordinates to be small.

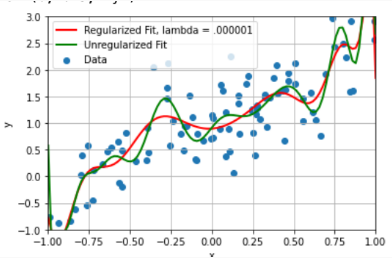


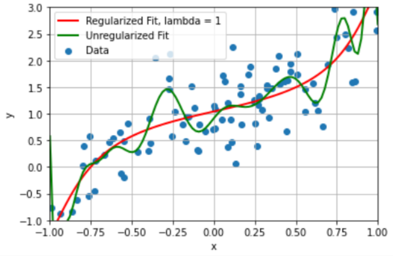


(Can be viewed as a “soft” version of feature selection.)

Fit degree 20 polynomial with varying leves of regularization.







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

$$\underline{\nabla L_R(\beta) = 0}$$

$$\nabla(f+g) = \nabla f + \nabla g$$

$$\underbrace{\nabla \|X\beta - y\|_2^2}_{} + \nabla\left(\lambda \|\beta\|_2^2\right)$$

$$\cancel{2} X^T(X\beta - y) + \cancel{2}\lambda\beta = 0$$

(d×d identity)

$$X^T X\beta - X^T y + \lambda I \beta$$

$$= (X^T X + \lambda I)\beta - X^T y = 0$$

$$(X^T X + \lambda I)\beta = X^T y \qquad \boxed{\beta^* = (X^T X + \lambda I)^{-1} X^T y}$$

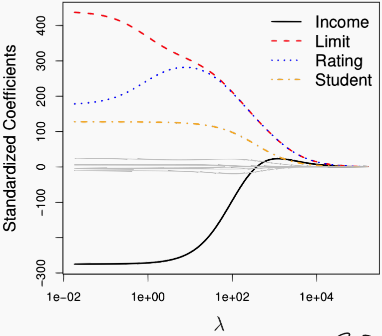
How do we minimize: $L_R(\boldsymbol{\beta}) = \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2$?

$$\beta^* = (X^T X + \lambda I)^{-1} \underline{X^T y} \qquad \frac{1}{\lambda} X^T y$$

$$\begin{bmatrix}\lambda & & & \\ & \lambda & & \\ & & \ddots & \\ & & & \lambda\end{bmatrix} \rightarrow \lambda \begin{bmatrix}1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1\end{bmatrix}$$

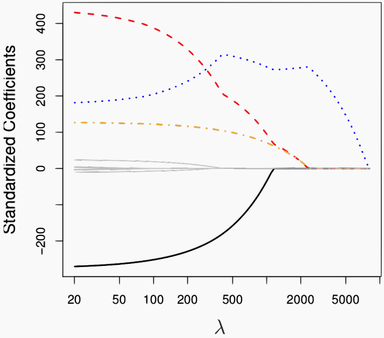
Lasso regularization: $\min_{\beta} \|\mathbf{X}\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_1$.

- 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.



1 2 20

Ridge regularization



Lasso Regularization

Why Lasso encourages sparsity is a long story that was only understand relatively recently. Major topic in the field of compressed sensing and sparse recovery.

Pros:

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

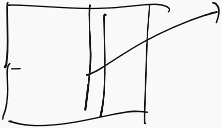
Cons:

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

$\lambda = 10, 20, 30, \ldots$ $\frac{1}{16}$ $\frac{1}{8}$ $\frac{1}{4}$ . . . 16

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 $[\lambda_1, \ldots, \lambda_q] = 1.5^{[-4,-3,-2,-1,-0,1,2,3,4]}$. 2 1.1
- Regularization methods are <u>not invariant</u> to data scaling. Typically when using regularization we mean center and scale columns to have unit variance.



$$\frac{(x_i - \bar{x}_i)}{\|x_i - \bar{x}_i\|_2}$$

Break Until 3:25.

THE BAYESIAN/PROBABILISTIC MODELING PERSPECTIVE

- **Data Examples:** $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$
- **Target:** $y_1, \ldots, y_n \in \{0, 1, 2, \ldots, q-1\}$ 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.

- 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 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, interesting justifications for tools like regularization. will also lead to natural approaches 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

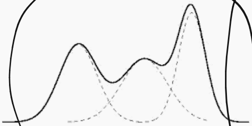
probabilistic model

that plausibly could have generated our data.[2]

- 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).

[2] "Data" here includes both the predictors $x_1, \ldots, x_n$ and targets $y_1, \ldots, y_n$.

Typically we try to keep things simple!



✓

X

X

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?

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:

$x_i \sim \mathcal{N}(1000, \sigma^2)$ option 1

$x_i \sim \text{Unif}(300, \ldots, 10000]$ option 2

$y_i = \beta_1 x_i + \beta_0 + \mathcal{N}(0, \sigma^2)$

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

Description: For n undergraduate **students**: each $x_i \in \{1, 2, 3, 4\}$ indicating class year. Each $y_i \in \{0, 1\}$ with zero indicating the student has not taken machine learning, one indicating they have.

Model:

$x_i \sim \text{Unif}(1, 2, 3, 4)$

$y_i \sim \text{Ber}(p(x_i))$

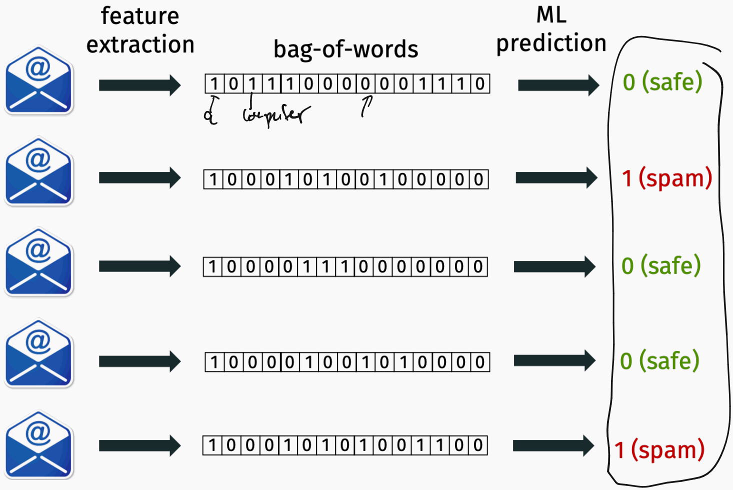
0 w/p $1-p$

1 w/p p.

$p(1) = .1 \quad p(2) = .2 \quad p(3) = .7 \quad p(4) = .99$

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).



Both target labels and data vectors are binary.

$$[1 \; 0 \; 0 \; 0 \; 1 \; 0 \; 0 \; 0 \; 0 \; 0 \; 0 \; 0 \; 0]$$

$p_1 \qquad p_j \qquad p_d$

Let's create a probabilistic model that generates emails.

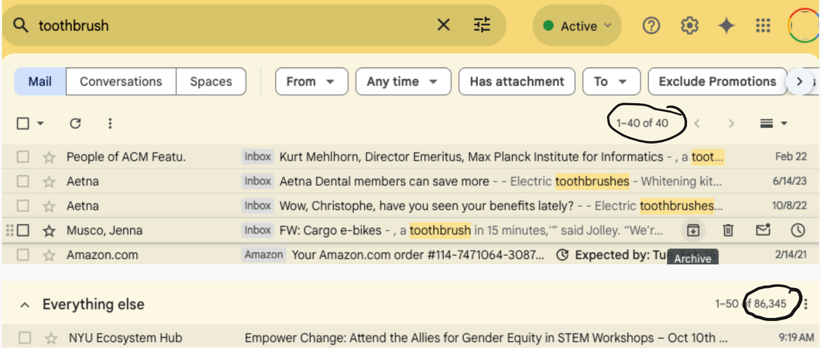
Observation: Since bag-of-words features don't care about word order, our model does not need to either.

- 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 $1 - p_i$.

$$p_{\text{the}} = .9 \qquad p_{\text{calendar}} = .2 \qquad p_{\text{toothbrush}} = .01$$

EMAIL MODEL

Model training: Find parameters $p_1, \ldots, p_d$ that best fit our training data.



In this case, set p_i to empirical word frequency of word i.

$p_{toothbrush} = 40/86k$

How can we make this model richer to generate both spam and non-spam email?

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

Not Spam

$$p_{\text{won}} = .01$$
$$p_{\$} = .1$$
$$p_{student} = .1$$

Spam

$$p_{\text{won}} = .1$$
$$p_{\$} = .2$$
$$p_{student} = .01$$

PROBABILISTIC MODEL FOR EMAIL

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

$C_0 = 95\%$

- Set $y = 0$ with probability c_0, $y = 1$ with probability $c_1 = 1 - c_0$.
  - c_0 is probability an email is not spam (e.g. 99%).
  - c_1 is probability an email is spam (e.g. 1%).
- If $y = 0$, for each i, set $x_i = 1$ with prob. $p_i^{(0)}$.
- If $y = 1$, for each i, set $x_i = 1$ with prob. $p_i^{(1)}$.

X

Unknown model parameters:

- c_0, c_1,
- $p_1^{(0)}, p_2^{(0)}, \ldots p_d^{(0)}$, one for each of the d vocabulary words.

  non. spam prob.
- $p_1^{(1)}, p_2^{(1)}, \ldots p_d^{(1)}$, one for each of the d vocabulary words.

  spam prob.

How would you estimate these parameters?

Reasonable way to set parameters:

- Set c_0 and c_1 to the empirical fraction of not spam/spam emails.
- For each word i, set $p_i^{(0)}$ to the empirical probability word i appears in a non-spam email.
- For each word i, set $p_i^{(1)}$ to the empirical probability word i appears in a spam email.

DONE WITH MODELING

ON TO PREDICTION

$p(A \cap B)$ \land

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

$$p(A \mid B) = \frac{p(A,B)}{p(B)}$$

$$p(A \mid B)\,p(B) = p(A,B)$$

$Pr = \Pr = P$

Two random events are **independent** if:

$Pr(A \mid B) = \Pr(A)$, or equivalently, $Pr(B \mid A) = \Pr(B)$

Equivalent characterization:

$$\Pr(A, B) = P(A) \cdot P(B).$$

$P(\cancel{A \mid B}) = \frac{P(A, B)}{P(B)}$

$P(A)$

$P(A)P(B) = P(A, B)$

B: event that dice role is even.

Note that when we write something like $p(A \mid B)$, A and B are **random events** not **random variables**.

We will sometimes (informally) write $p(X \mid B)$, where X is a random variable. In this case, $p(X \mid B)$ is understood to be a probability density/mass function.

E.g., suppose X is a dice role that takes values $1, \ldots, 6$. Then $p(X \mid B)$ is a function from $\{1, \ldots, 6\} \rightarrow [0, 1]$ whose i^{th} value equals $p(X = i \mid B)$. $A = \{X = i\}$ is a proper random event.

$p(X=1 \mid B)$

$$p(X \mid B) = \underset{1}{\underline{0}}\ \underset{2}{\underline{1/3}}\ \underset{3}{\underline{0}}\ \underset{4}{\underline{1/3}}\ \underset{5}{\underline{0}}\ \underset{6}{\underline{1/3}}$$

$$\left(P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \right)$$

$$\cancel{p(A \mid B) = \frac{p(A \mid B)\,p(A)}{p(B)}}$$

Proof:

$$P(A|B) = \frac{P(A,B)}{P(B)} \qquad P(B|A) = \frac{P(A,B)}{P(A)}$$

$$P(B)\,P(A|B) = P(A)\,P(B|A)$$

$$\left(P(A|B) = \frac{P(A)\,P(B|A)}{P(B)} \right)$$

Bayes Rule.

CLASSIFICATION RULE

$[0\ 0\ 1\ 0\ 0\ 0]$

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.

$Pr(y = 0 \mid w = [0\ 0\ 1\ 0\ 0\ 0])$

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.

$Pr(w = [0 0 1 0 0 0 0] \mid y = 0)$

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.

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 c_0, c_1.
  45% 5%
- $p(\mathbf{w} \mid y = 0) = ?$ $p(\mathbf{w} \mid y = 1) = ?$

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

Recall that, under our model, index i is 1 with probability $p_i^{(0)}$ if we are not spam, and 1 with probability $p_i^{(1)}$ if we are spam .

$$p(\mathbf{w} = [0, 1, 1, 0, 0, 0, 1, 0] \mid y = 0) =$$

$$(1 - p_1^{(0)})\, p_2^{(0)}\, p_3^{(0)}\, (1 - p_4^{(0)}) \ldots \ldots (1 - p_m^{(0)})$$

$$p(\mathbf{w} = [0, 1, 1, 0, 0, 0, 1, 0] \mid y = 1) =$$

$$(1 - p_1^{(1)})\, p_2^{(1)}\, p_3^{(1)} \ldots \ldots$$

Final Naive Bayes Classifier

Training/Modeling: Use existing data to compute:

- Prior class probabilities $c_0 = p(y = 0), c_1 = p(y = 1)$
- For all i:
  - $p_i^{(0)} = p(w_i = 1 \mid y = 0)$ and $(1 - p_i^{(0)}) = p(w_i = 0 \mid y = 0)$
  - $p_i^{(1)} = p(w_i = 1 \mid y = 1)$ and $(1 - p_i^{(1)}) = p(w_i = 0 \mid y = 1)$

Prediction:

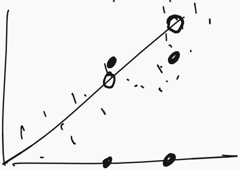
- For new input $\mathbf{w} \in \{0, 1\}^d$:
  - Compute $p(\mathbf{w} \mid y = 0) = \prod_{i=1}^{d} p(w_i \mid y = 0)$
  - Compute $p(\mathbf{w} \mid y = 1) = \prod_{i=1}^{d} p(w_i \mid y = 1)$
- Return

$$\arg\max \left[p(\mathbf{w} \mid y = 0) \cdot \overset{.95}{p(y = 0)},\ p(\mathbf{w} \mid y = 1) \cdot \overset{.05}{p(y = 1)} \right].$$

OTHER APPLICATIONS OF

THE BAYESIAN PERSPECTIVE

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

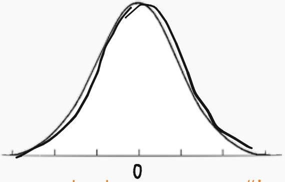


Example: Linear Regression.

Probabilistic model: For some "true" set of parameters $\boldsymbol{\beta}_{\text{true}}$,

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

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



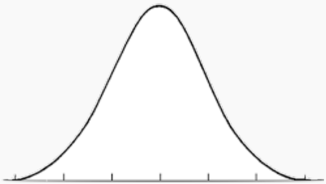
$$Pr(\eta = z) \sim e^{-z^2}$$

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

GAUSSIAN DISTRIBUTION REFRESHER

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

Parameterized by mean μ and variance σ^2.



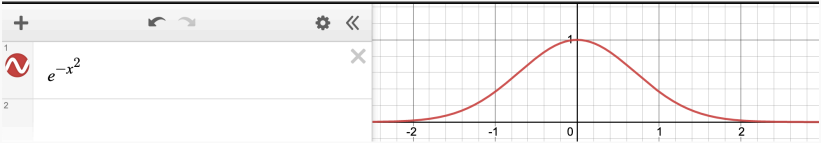
η is a continuous random variable, so it has a probability density function $p(\eta)$ with $\int_{-\infty}^{\infty} p(\eta)d\eta = 1$

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

$e^{-\frac{\eta^2}{2\sigma^2}}$

e^{-x^2}

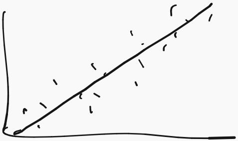
The important thing to remember is that the the PDF falls off exponentially as we move further from the mean.



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

BAYESIAN REGRESSION

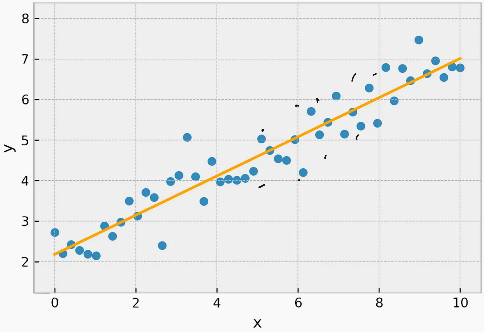
Example: Linear Regression.



Probabilistic model:

$$y = \langle \mathbf{x}, \underline{\beta_{\text{true}}} \rangle + \eta$$

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



How should we find the unknown parameters β for our model?

$p(X, y \mid \beta)$

Also use a Bayesian approach!

First thought: choose β to maximize:

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

But 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:

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

This is called the maximum likelihood estimate.

$\Pr(X, y \mid \beta) = \Pr(y \mid \beta)$

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:

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

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.

$$\Pr(\mathbf{y} \mid \boldsymbol{\beta}) \sim \prod_{i=1}^{n} \Pr(y_i \mid \beta) = \prod_{i=1}^{n} \Pr(\eta_i = y_i - \langle x_i, \beta \rangle)$$

$$= \prod_{i=1}^{n} e^{-(y_i - \langle x_i, \beta \rangle)^2 / 2\sigma^2}$$

LOG LIKELIHOOD

Easier to work with the log likelihood:

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

Conclusion: Choose $\boldsymbol{\beta}$ to minimize:

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

This is a completely different justification for minimizing squared loss!

Minimizing the ℓ_2 loss is "optimal" when you assume your data follows a linear model with i.i.d. Gaussian noise (with <u>any</u> fixed variance).