YOU'VE BEEN DOING STATISTICS ALL ALONG

ML IS:

DOGS	CATS	TURTLES
3	4	3
1	9	6
2	2	7
5	1	2
3	3	1
6	5	4
4	8	9
7	1	4

```
X = df[['dogs', 'cats']]
y = df['turtles']

ŷ = model.predict(X)
```

SO, HOW DO WE DO THIS?

```
import sklearn
```

THIS PRESENTATION STARTS AND ENDS HERE.

YOUR HEAD IS CURRENTLY ABOVE WATER: WE'RE GOING TO DIVE INTO THE POOL, TOUCH THE BOTTOM, THEN COME BACK TO THE SURFACE.

LINEAR REGRESSION

```
y = weights^T * x
```

$$\hat{y} = \theta^T x$$

$\hat{y}$ IS COMPUTED AS A FUNCTION OF BOTH x AND θ. WE'D LIKE THIS VALUE TO BE CLOSE TO THE TRUE y.

$$\mathcal{L} = \sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2$$

FOR CLARITY, NOW IN SEMI-OBNOXIOUS PYTHON:

```
error = guess - true
loss = (error**2).sum()
```

I DO MACHINE LEARNING; I SHOULD PROBABLY LEARN STATISTICS, THOUGH, TOO?

-- ME AT SOME POINT; HOPEFULLY NONE OF YOU

- `df['dogs']` IS A "RANDOM VARIABLE." IT DOESN'T DEPEND ON ANYTHING.
- `df['cats']` IS A "RANDOM VARIABLE." IT DOESN'T DEPEND ON ANYTHING.

- `df['turtles']` IS A "RANDOM VARIABLE." IT DEPENDS ON `df['dogs']` AND `df['cats']`.

GOAL: CHANGE `weights` SO AS TO MINIMIZE OUR ERROR FUNCTION.

PROBABILITY DISTRIBUTIONS

THIS IS A PROBABILITY DISTRIBUTION. IT IS A LOOKUP TABLE FOR THE LIKELIHOOD OF OBSERVING EACH UNIQUE OUTCOME OF A RANDOM VARIABLE.

```
cats = {2: .17, 5: .24, 3: .11, 9: .48}
```

TRIVIALLY, THE VALUES SHOULD SUM TO 1.

TYPICALLY, PROBABILITY DISTRIBUTIONS TAKE PARAMETERS WHICH CONTROL THEIR SHAPE. LET'S DEFINE A FUNCTION TO ILLUSTRATE THIS:

```
def prob_distribution_cats(λ):
    return distribution

In [1]: prob_distribution_cats(λ=5)
Out[1]: {2: .37, 5: .14, 3: .13, 9: .36}

In [2]: prob_distribution_cats(λ=84)
Out[2]: {2: .09, 5: .32, 3: .17, 9: .42}
```

THE REALIZED VALUES OF A RANDOM VARIABLE ARE DICTATED BY ITS PROBABILITY DISTRIBUTION.

- WHAT PROBABILITY DISTRIBUTIONS DICTATE THE VALUES OF OUR RANDOM VARIABLES?

LET'S START BY WRITING THE COMPONENTS OF OUR MODEL AS DRAWS FROM A DISTRIBUTION.

$$P(\theta), P(X), P(y|X;\theta)$$

WHICH PROBABILITY DISTRIBUTIONS DESCRIBE OUR DATA? LET'S START WITH y AND ASSUME IT IS DISTRIBUTED NORMALLY, I.E.

$$y \sim \mathcal{N}(\mu, \sigma^2)$$

WHERE $\mathcal{N}$ GIVES THE NORMAL DISTRIBUTION.

NATURALLY, WE'LL ASSUME THAT $\mu = \theta^T x$, WHERE σ DESCRIBES SOME "IRREDUCIBLE ERROR" IN OUR ESTIMATE OF y.

- IRREDUCIBLE ERROR MEANS: y REALLY DEPENDS ON SOME OTHER INPUT - E.G. `df['zebras']` - THAT WE HAVEN'T INCLUDED IN OUR MODEL.

PROBABILITY DENSITY FUNCTIONS

THE VALUES OF y ARE DISTRIBUTED AS $y \sim \mathcal{N}(\theta^T x, \sigma^2)$.

THE PROBABILITY OF DRAWING A SPECIFIC VALUE OF $y^{(i)}$ GIVEN $x^{(i)}$ AND θ IS GIVEN BY THE NORMAL DENSITY FUNCTION.

$$P(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(- \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right)$$

-- CARL FRIEDRICH GAUSS, 1809

- **GOAL: CHANGE `weights` SO AS TO MINIMIZE OUR ERROR FUNCTION.**

WHICH SHOULD WE CHOOSE?

MAXIMUM LIKELIHOOD ESTIMATION

A MOROCCAN WALKS INTO A BAR. HE'S WEARING A FOOTBALL JERSEY THAT'S MISSING A SLEEVE. HE HAS A BLACK EYE, AND BLOOD ON HIS JEANS. HOW DID HE MOST LIKELY SPEND HIS DAY?

1. AT HOME, READING A BOOK.
2. TRAINING FOR A BICYCLE RACE.
3. AT THE WAC VS. RAJA GAME DRINKING CASABLANCA BEERS WITH HIS FRIENDS - ALL OF WHOM ARE MMA FIGHTERS AND DESPISE THE OTHER TEAM.

WHICH `weights` MAXIMIZE THE LIKELIHOOD OF HAVING OBSERVED THE y THAT WE DID?

THIS IS CALLED THE **MAXIMUM LIKELIHOOD ESTIMATE**. TO COMPUTE IT, WE SIMPLY PICK THE `weights` THAT MAXIMIZE $P(y^{(i)}|x^{(i)};\theta)$ FROM ABOVE. HOWEVER, WE'RE NOT JUST CONCERNED ABOUT ONE OUTCOME $y^{(i)}$; INSTEAD, WE CARE ABOUT THEM ALL.

ASSUMING THAT $y^{(i)}$ VALUES ARE INDEPENDENT OF ONE ANOTHER, WE CAN WRITE THEIR JOINT **LIKELIHOOD AS FOLLOWS:**

$$P(y|x;\theta) = \prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta)$$

THERE IS NOTHING SCARY ABOUT THIS PRODUCT**: WHEREAS THE LONE TERM GIVES THE LIKELIHOOD OF ONE DATA POINT, THE PRODUCT GIVES THE LIKELIHOOD OF HAVING OBSERVED ALL DATA POINTS.**

SINCE PROBABILITIES ARE NUMBERS IN $[0, 1]$, THE PRODUCT OF A BUNCH OF PROBABILITIES GETS VERY SMALL, VERY QUICK. FOR THIS REASON, WE OFTEN TAKE THE NATURAL LOGARITHM.

$$\begin{aligned}
\log P(y|x;\theta) &= \log \prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta) \\
&= \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)};\theta) \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} + \sum_{i=1}^{m} \log \exp\left(-\frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y^{(i)}-\theta^T x^{(i)})^2 \\
&= C_1 - C_2 \sum_{i=1}^{m} (y^{(i)}-\theta^T x^{(i)})^2
\end{aligned}$$

MAXIMIZING THE LOG-LIKELIHOOD OF OUR DATA WITH RESPECT TO $\boldsymbol{\theta}$, I.E. `weights`, IS EQUIVALENT TO MAXIMIZING THE NEGATIVE MEAN SQUARED ERROR BETWEEN $\boldsymbol{y}$ AND $\hat{\boldsymbol{y}}$.

MOST OPTIMIZATION ROUTINES MINIMIZE.

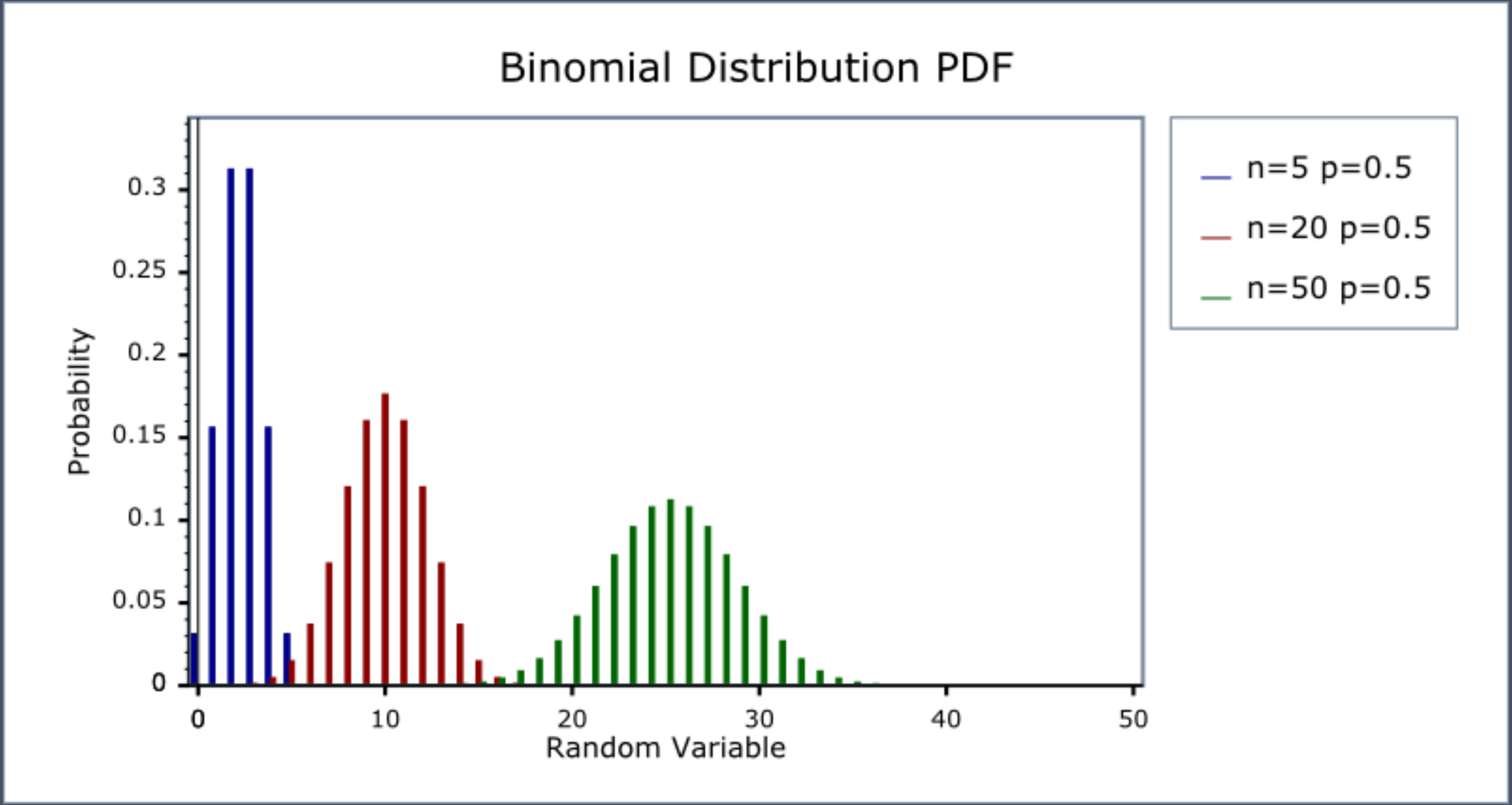
MINIMIZING THE NEGATIVE LOG-LIKELIHOOD OF OUR DATA WITH RESPECT TO θ, I.E. `weights`, IS EQUIVALENT TO MINIMIZING THE MEAN SQUARED ERROR BETWEEN y AND $\hat{y}$.

LOGISTIC REGRESSION

$$p = \frac{1}{1 + e^{-\theta^T x}}$$

$$\mathcal{L} = -\sum_{i=1}^{m} y^{(i)} \log p^{(i)} + (1 - y^{(i)}) \log (1 - p^{(i)})$$

y IS A BINARY RANDOM VARIABLE: IT IS A THING THAT TAKES VALUES IN $\{0, 1\}$.


Binomial Distribution PDF
Probability
0.3
0.25
0.2
0.15
0.1
0.05
0
0
10
20
30
40
50
Random Variable
n=5 p=0.5
n=20 p=0.5
n=50 p=0.5

$$P(y|x;\theta) = \prod_{i=1}^{m} (p^{(i)})^{y^{(i)}} (1 - p^{(i)})^{1-y^{(i)}}$$

IF THIS LOOKS CONFUSING:

- DISREGARD THE LEFT SIDE
- ASK YOURSELF: WHAT IS THE PROBABILITY OF OBSERVING THE FOLLOWING SPECIFIC SEQUENCE OF COIN FLIPS, WHERE $P(\text{heads}) = .7$:

$$\begin{aligned} P(\text{heads}, \text{tails}, \text{heads}, \text{heads}) &= (.7^1 * .3^0)(.7^0 * .3^1)(.7^1 * .3^0)(.7^1 * .3^0) \\ &= .7 * .3 * .7 * .7 \\ &= .102899 \end{aligned}$$

NEGATIVE LOG-LIKELIHOOD:

$$\begin{aligned}
-\log P(y|x;\theta) &= -\log \prod_{i=1}^{m} (p^{(i)})^{y^{(i)}} (1-p^{(i)})^{1-y^{(i)}} \\
&= -\sum_{i=1}^{m} \log \left((p^{(i)})^{y^{(i)}} (1-p^{(i)})^{1-y^{(i)}} \right) \\
&= -\sum_{i=1}^{m} \log (p^{(i)})^{y^{(i)}} + \log (1-p^{(i)})^{1-y^{(i)}} \\
&= -\sum_{i=1}^{m} y^{(i)} \log (p^{(i)}) + (1-y^{(i)}) \log (1-p^{(i)})
\end{aligned}$$

THIS WILL LOOK FAMILIAR AS WELL.

REMEMBER THAT USING BAYES' THEOREM DOESN'T MAKE YOU A BAYESIAN. QUANTIFYING UNCERTAINTY WITH PROBABILITY MAKES YOU A BAYESIAN.

-- MICHAEL BETANCOURT (@BETANALPHA)

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} = \frac{P(X,y)}{P(X)}$$

CLASSIFICATION

DOGS	CATS	HEADACHE
3	4	HIGH
1	9	HIGH
2	2	LOW
5	1	MEDIUM
3	3	LOW
6	5	HIGH
4	8	LOW
7	1	MEDIUM

DISCRIMINATIVE MODELS JUMP DIRECTLY TO ESTIMATING $P(y|X)$ WITHOUT COMPUTING $P(X|y), P(y)$ AND $P(X)$.

IN EFFECT, THEY CAN DISTINGUISH DARIJA FROM FRENCH, BUT CAN'T SPEAK EITHER.

GENERATIVE MODELS COMPUTE $P(X|y), P(y)$ AND $P(X)$. THEN ESTIMATE $P(y|X)$ VIA BAYES' THEOREM.

GENERATIVE MODELS SPEAK FLUENT DARIJA AND FRENCH. THEY CAN DISTINGUISH THE TWO BECAUSE DUH, THEY SPEAK EACH ONE AND KNOW THEIR DIFFERENCES.

GENERATIVE MODELS

THE JOINT DISTRIBUTION

GENERATIVE MODELS START BY MODELING THE JOINT DISTRIBUTION **OF OUR DATA. I.E.** $P(X, y) = P(X|y)P(y)$.

Let's try to understand this with an example. Consider the following 4 data points: $(x, y) = \{(0,0), (0,0), (1,0), (1,1)\}$

For above data, $p(x, y)$ will be following:

	$y = 0$	$y = 1$
$x = 0$	$1/2$	0
$x = 1$	$1/4$	$1/4$

while $p(y|x)$ will be following:

	$y = 0$	$y = 1$
$x = 0$	1	0
$x = 1$	$1/2$	$1/2$

NAIVE BAYES

$$P_\theta(x,y) = P_\theta(x|y)P_\theta(y) = P_\theta(y)\prod_{i=1}^{m} P_\theta(x_i|y)$$

- ESTIMATE $\hat{\theta}$ AS BEFORE
- TO MAKE A PREDICTION:

```
np.argmax([P(x, y) for y in ['low', 'medium', 'high']])
```

FINALLY, TO GENERATE LIKELY DATA GIVEN A CLASS,

1. DRAW A CLASS FROM $P(y)$
2. DRAW DATA FROM $P(x|y)$

WHAT I CANNOT CREATE, I DO NOT UNDERSTAND.

-- RICHARD FEYNMAN

DISCRIMINATIVE MODELS

DEFINE 3 SEPARATE MODELS. DO THE SAME LINEAR COMBINATION. BIGGEST NUMBER WINS.

$$\hat{y}_{\text{low}} = \theta_{\text{low}}^T x$$

$$\hat{y}_{\text{medium}} = \theta_{\text{medium}}^T x$$

$$\hat{y}_{\text{high}} = \theta_{\text{high}}^T x$$

- **THESE ARE** PROPORTIONAL **TO THE JOINT DISTRIBUTION OF THE RESPECTIVE CLASS AND THE DATA OBSERVED.**

$$P(y|x) = \frac{P(y,x)}{P(x)} = \frac{e^{\hat{y}}}{\sum_{y} e^{\hat{y}}} = \frac{e^{\left(\sum_{i} w_i x_i\right)_{\hat{y}}}}{\sum_{y} e^{\left(\sum_{i} w_i x_i\right)_{\hat{y}}}}$$

e. BECAUSE THE NUMERATOR NEEDS TO BE BIGGER THAN THE DENOMINATOR.

WE DON'T COMPUTE THE TRUE $P(y, x)$: OUR MODEL WILL NOT LEARN THE TRUE DISTRIBUTION OF DATA WITHIN EACH CLASS.

$$P(y|x) = \frac{P(y, x)}{P(x)} = \frac{\tilde{P}(y, x)}{\text{normalizer}}$$

```
linear_regression(loss=mean_squared_error).fit()
```

MAXIMIZE THE LIKELIHOOD OF THE NORMALLY-DISTRIBUTED RESPONSE VARIABLE WITH RESPECT TO SOME SET OF WEIGHTS (AND FIXED DATA).

```
logistic_regression(loss=log_loss).fit()
```

MAXIMIZE THE LIKELIHOOD OF THE BINOMIALLY-DISTRIBUTED RESPONSE VARIABLE WITH RESPECT TO SOME SET OF WEIGHTS (AND FIXED DATA).

```
naive_bayes(loss=negative_log_joint_likelihood).predict()
```

COMPUTE THE JOINT PROBABILITY $P(x, y)$ OF THE DATA AND RESPONSE VARIABLES, THEN TAKE THE ARGMAX.

```
neural_network(loss=categorical_cross_entropy).predict()
```

COMPUTE THE CONDITIONAL DISTRIBUTION $P(y|x)$ OF THE RESPONSE VARIABLES GIVEN THE DATA. THEREIN, AN UNNORMALIZED **JOINT PROBABILITY IS COMPUTED - NOT THE REAL THING.**

MACHINE LEARNING LIBRARIES LIKE SCIKIT-LEARN ARE A TERRIFIC RESOURCE (AND IT IS ONLY IN RARE CIRCUMSTANCE THAT YOU SHOULD HAND-ROLL A MODEL YOURSELF).

THIS SAID, PLEASE DO NOTE:

WHEN YOU'RE CALLING `.fit()` AND `.predict()`, YOU'VE BEEN DOING STATISTICS ALL ALONG.

RESOURCES

- STANFORD UNIVERSITY CS229
- STACKOVERFLOW DISCUSSION ON DISCRIMINATIVE VS. GENERATIVE MODELS
- CROSSVALIDATED DISCUSSION ON DISCRIMINATIVE VS. GENERATIVE MODELS
- ON DISCRIMINATIVE VS. GENERATIVE CLASSIFIERS: A COMPARISON OF LOGISTIC REGRESSION AND NAIVE BAYES

GITHUB.COM/CAVAUNPEU/STATISTICS-ALL-ALONG