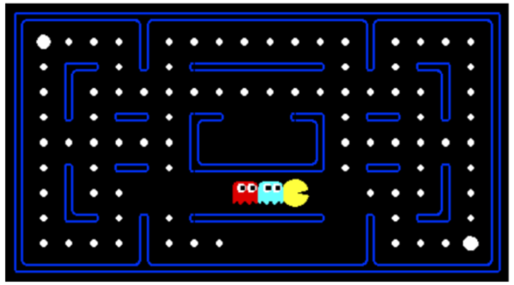
Machine Learning

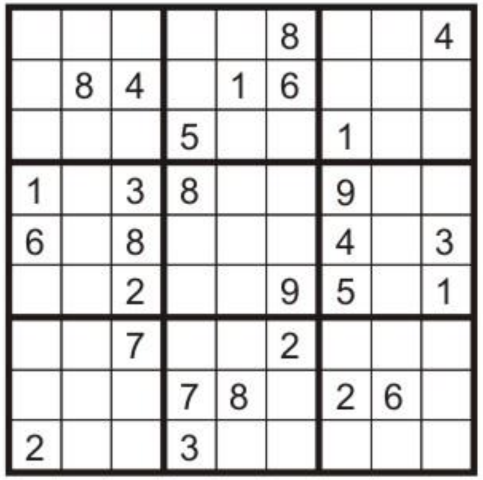
Chen-Yu Wei

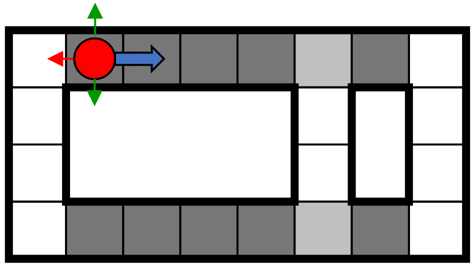
What we have studied so far…

How are these “rules” or “model of the world” obtained?

- Given the **rule of a game** (and/or **behavior model of the opponents**), find the optimal solution (search, search in multi-player games)
- Given the **relation among variables**, find a satisfied solution (constraint satisfaction)
- Given the **relation among variables**, find the probability of certain events, or the most probable events (Bayes nets, HMM)
- Given a **knowledge base**, infer some facts (logic)

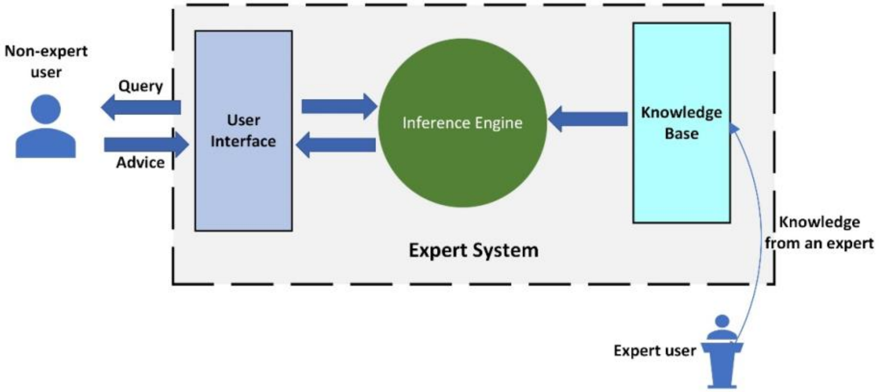






Knowledge base

If **has_hair**, then **mammal**.
If **mammal** and **has_hooves**, then **ungulate**.
If **has_feathers**, then **bird**.



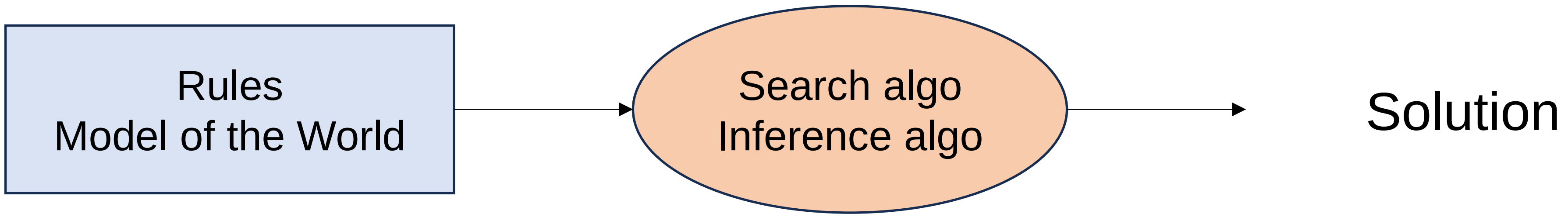
Rules or Model of the World

- In games designed by human, we simply have the ground-truth rules
  - Pacman
  - Chess, Go
  - Sudoku

- Some are rules set by human based on their observations/knowledge of the world
  - Knowledge base in expert systems

- Some have appeared magically so far
  - The behavior model of the ghosts in Pacman
  - Probability tables in Bayes nets, HMM

Machine Learning



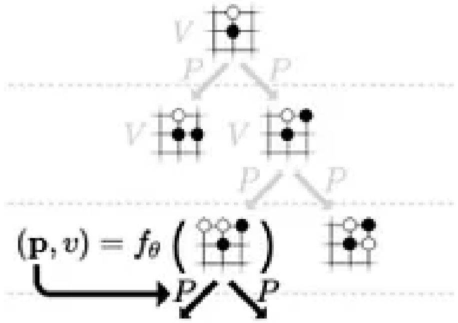
What we have taken for granted

We will discuss how to let machine **learn** these models through **data** collected from the world

Machine Learning

Machine Learning

In some cases, even when the world model is designed by human and known, we still want to perform machine learning



Evaluation function / Heuristic function

Depth-limited search

Guide the search in games with large branching factors

Low-level rules (known)

Machine learning from simulations

High-level rules (learned)

Naïve Bayes

Learning Simple Bayesian Networks

Suppose we have a set of data:

(Y, X) =

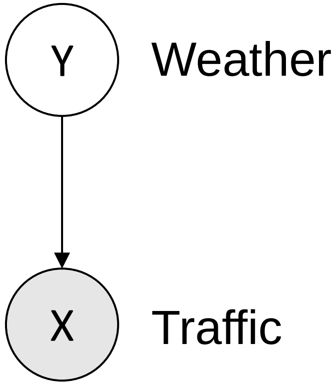
{ (sun, F) , (rain, F) , (rain, T) , (rain, T)
(sun, T) , (sun, F) , (sun, F) , (sun, T) }

How should we build the BN model?

Y	P(Y)
Sun	5/8
Rain	3/8

Y	X	P(X | Y)
Sun	T	2/5
Sun	F	3/5
Rain	T	2/3
Rain	F	1/3

training



P(Y)

P(X | Y)

P(Y/X)

If now we observe X (traffic) = T, how to infer the Y (weather) distribution?

How did we obtain the parameters?

Why do we model $\mathrm{P}(\mathrm{X}=\mathrm{T} \mid \mathrm{Y}=\mathrm{sun})$ as $\frac{\#\,(\mathrm{Y=sun, X=T})}{\#(\mathrm{Y=sun})}$ in the dataset?

Maximum Likelihood Estimation (MLE)

can be used in training any BNs with finite domains

set of all possible models

$\subset \mathbb{R}^6$

Likelihood

Pick $\underset{\boldsymbol{M}}{\operatorname{argmax}} \prod_{i=1}^{n} P_{\boldsymbol{M}}(x_i, y_i)$

Best explains the data (?)

--- has some drawbacks (discussed later)

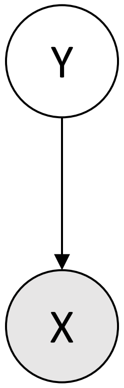
Approximate inference

We have the model, and thus the exact value of P(Y|X) is available. But because the exact computation is expensive, we approximate it with samples drawn from the model.

Model learning

We do not have the model, and try to build it from data drawn from the nature.

Dealing with High-Dimensional Observation

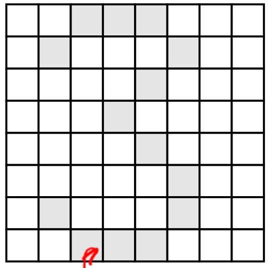


Digit

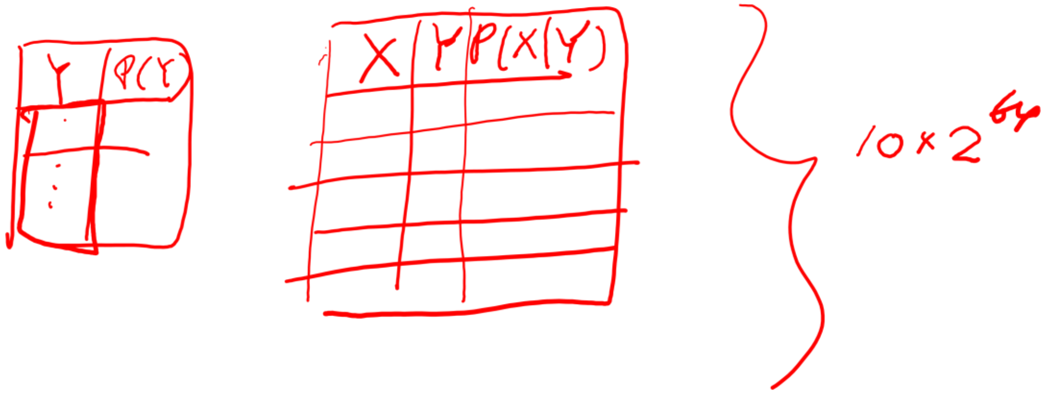
8x8 Bitmap

$Y \in \{0, 1, 2, \ldots, 9\}$

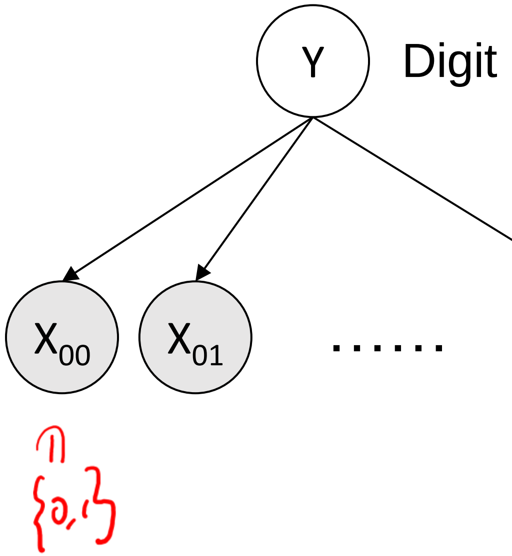
X:



Number of parameters in this model?

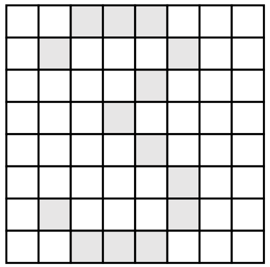


Dealing with High-Dimensional Observation



$Y \in \{0, 1, 2, \ldots, 9\}$

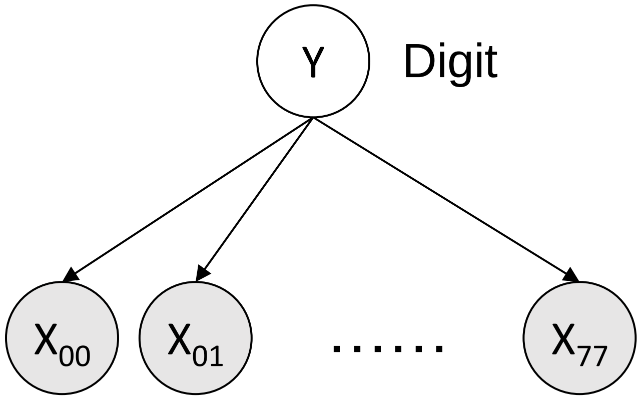
X:



Number of parameters in this model?

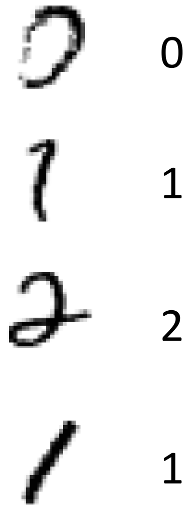
$10 \times 2 \times 64$

Dealing with High-Dimensional Observation



Training:

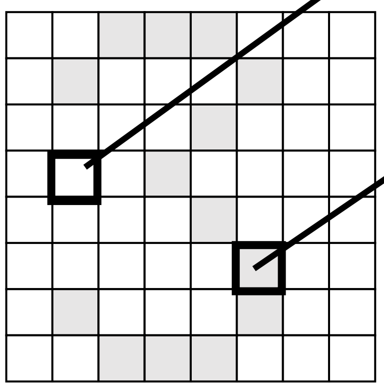
1) Get dataset



2) Match model with empirical frequency

P(Y)

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(X_{31}=on \mid Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

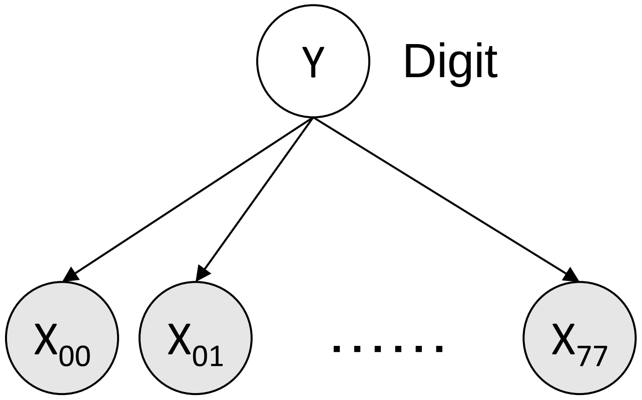
$P(X_{55}=on \mid Y)$

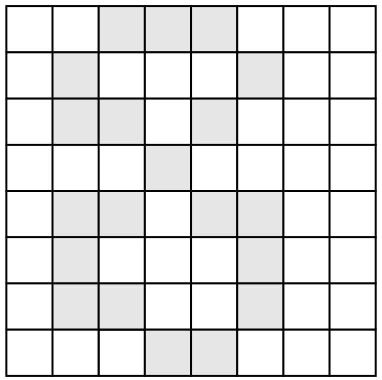
1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Dealing with High-Dimensional Observation

Inference:

After training, now given a bitmap, decide its likelihood to be each digit



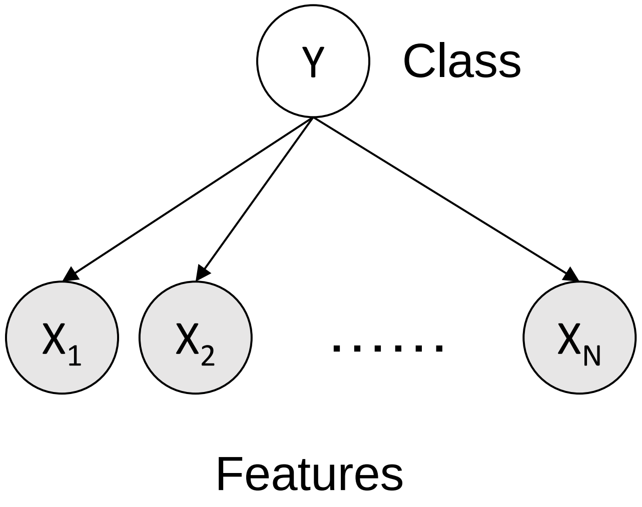


$P(Y \mid x_{00}, x_{01}, \ldots, x_{77})$

$\propto p(Y, x_{00}, x_{01}, \ldots, x_{77})$

$= P(Y)\, P(x_{00} \mid Y)\, P(x_{01} \mid Y) \cdots P(x_{77} \mid Y)$

General Naïve Bayes Model



Finite domains for Y and X_i

Training:

1) Get dataset consisting of $(X, Y) = (X_1, \ldots, X_N, Y)$ pairs
2) Train model $P(Y)$, $P(X_i \mid Y)$ with maximum likelihood estimation (= empirical frequency)

(more options discussed later)

Inference:

Given x,

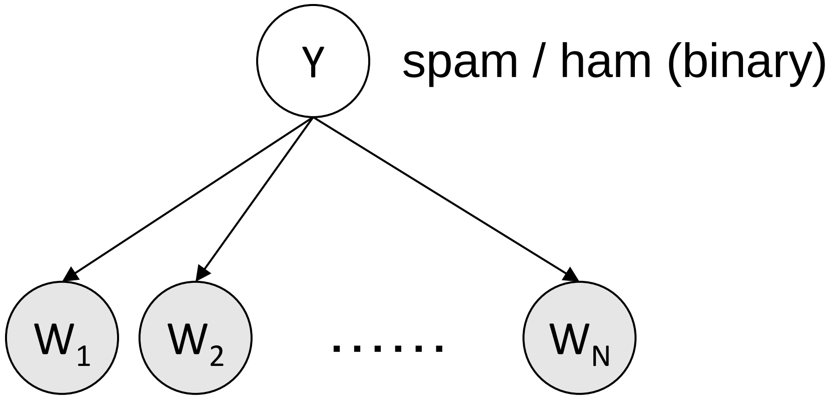
Infer $P(Y \mid x) \propto P(Y)\, P(x_1 \mid Y)\, P(x_2 \mid Y) \ldots P(x_N \mid Y)$

Example: Spam Filtering

Training data:

Collection of emails, labeled spam or ham

Model (**bag-of-word**):



Special assumption (not in the digit example):
$P(W_i \mid Y)$ is identical for every i

→ This is why it is called bag-of-world (word ordering does not matter)

✗

> Dear Sir.
>
> First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

✗

> TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.
>
> 99 MILLION EMAIL ADDRESSES
> FOR ONLY $99

✓

> Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Example: Spam Filtering

- Model: $P(Y, W_1 \ldots W_n) = P(Y) \prod_i P(W_i|Y)$

- What are the parameters?

$P(Y)$

ham	0.66
spam	0.33

$P(W|\text{spam})$

the	0.0156
to	0.0153
and	0.0115
of	0.0095
you	0.0093
a	0.0086
with	0.0080
from	0.0075
...	

$P(W|\text{ham})$

the	0.0210
to	0.0133
of	0.0119
2002	0.0110
with	0.0108
from	0.0107
and	0.0105
a	0.0100
...	

Spam Example

$$\log\left(P(Y) \prod_i P(w_i|Y) \right) = \log P(Y) + \sum_i \log(w_i|Y)$$

Word	P(w|spam)	P(w|ham)	Tot Spam	Tot Ham
(prior)	0.33333	0.66666	-1.1	-0.4
Gary	0.00002	0.00021	-11.8	-8.9
would	0.00069	0.00084	-19.1	-16.0
you	0.00881	0.00304	-23.8	-21.8
like	0.00086	0.00083	-30.9	-28.9
to	0.01517	0.01339	-35.1	-33.2
lose	0.00008	0.00002	-44.5	-44.0
weight	0.00016	0.00002	-53.3	-55.0
while	0.00027	0.00027	-61.5	-63.2
you	0.00881	0.00304	-66.2	-69.0
sleep	0.00006	0.00001	-76.0	-80.5

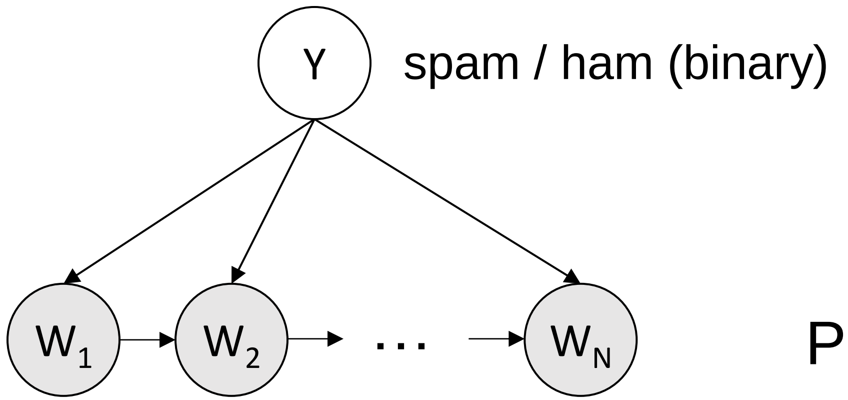
P(spam | w) = 98.9

Another Possible Model

May slightly improve accuracy

e.g., Earn money vs. Earn degree

with the price of larger model (usually requires more data to train)



$P(W_i \mid Y, W_{i-1})$

Overfitting and Generalization

If using Maximum Likelihood Estimation…

For a new bitmap:

$p(c|x) \propto p(x, c) = p(c)\, p(x_{00}|c)\, p(x_{01}|c) \cdots p(x_{77}|c)$

$P(\text{features}, C = 2)$

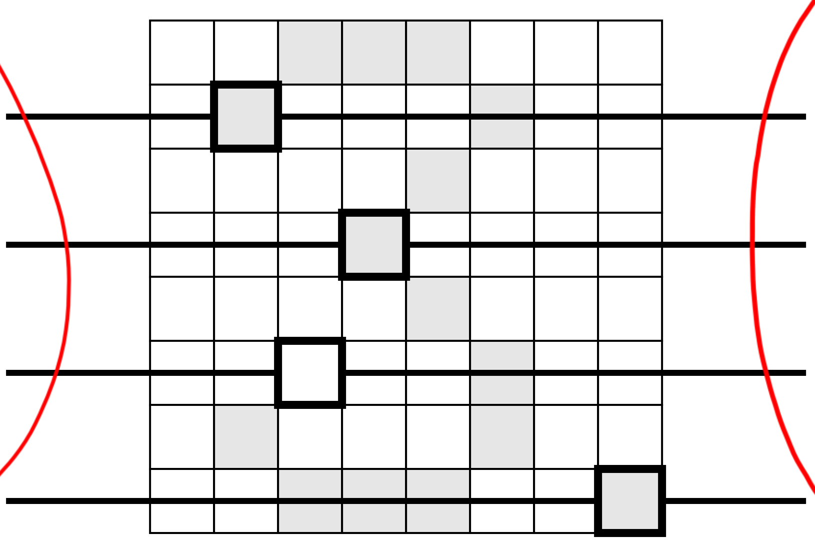
$P(C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.8$

$P(\text{on}|C = 2) = 0.1$

$P(\text{off}|C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.01$



$P(\text{features}, C = 3)$

$P(C = 3) = 0.1$

$P(\text{on}|C = 3) = 0.8$

$P(\text{on}|C = 3) = 0.9$

$P(\text{off}|C = 3) = 0.7$

$P(\text{on}|C = 3) = 0.0$

2 wins!!

If using Maximum Likelihood Estimation…

Prediction determined by *relative* probabilities:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

```
south-west : inf
nation     : inf
morally    : inf
nicely     : inf
extent     : inf
seriously  : inf
...
```

$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

```
screens    : inf
minute     : inf
guaranteed : inf
$205.00    : inf
delivery   : inf
signature  : inf
...
```

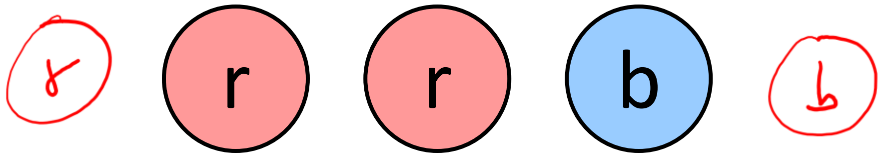
Overfitting

- Relative frequency parameters will overfit the training data!
  - Just because we never saw a 3 with pixel (15,15) on during training doesn't mean we won't see it at test time
  - Unlikely that every occurrence of "minute" is 100% spam
  - Unlikely that every occurrence of "seriously" is 100% ham
  - What about all the words that don't occur in the training set at all?
  - In general, we can't give unseen events zero probability

- For Naïve Bayes we use smoothing to address this issue
  - A special case of the general concept of "regularization"

Laplace Smoothing

- Laplace's estimate:
  - Pretend you saw every outcome k extra times



$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - What's Laplace with k = 0?
  - k is the strength of the prior

$$P_{LAP,0}(X) = \left\langle \frac{2}{3}, \frac{1}{3} \right\rangle$$

$$P_{LAP,1}(X) = \left\langle \frac{3}{5}, \frac{2}{5} \right\rangle$$

$$P_{LAP,100}(X) = \left\langle \frac{102}{203}, \frac{101}{203} \right\rangle$$

- Laplace for conditionals:
  - Smooth each condition independently:

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$

Real NB: Smoothing

- For real classification problems, smoothing is critical
- New odds ratios:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

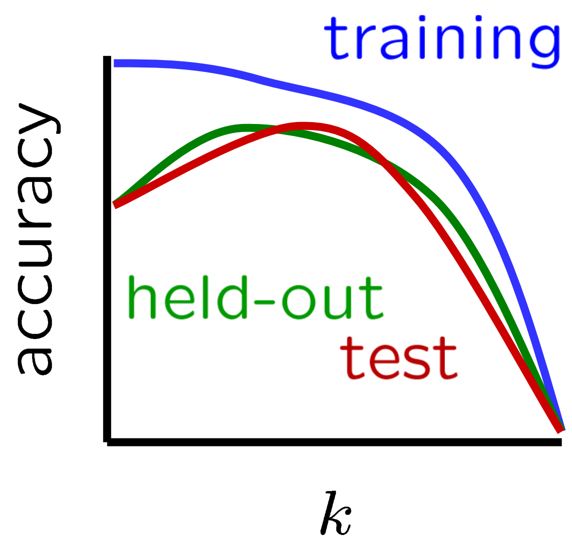
```
helvetica : 11.4
seems     : 10.8
group     : 10.2
ago       :  8.4
areas     :  8.3
...
```

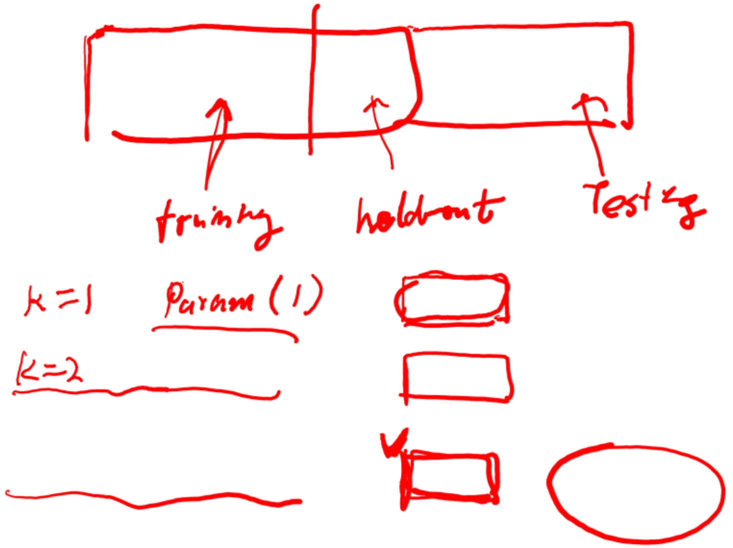
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

```
verdana : 28.8
Credit  : 28.4
ORDER   : 27.2
<FONT>  : 26.9
money   : 26.5
...
```

Tuning on Held-Out Data

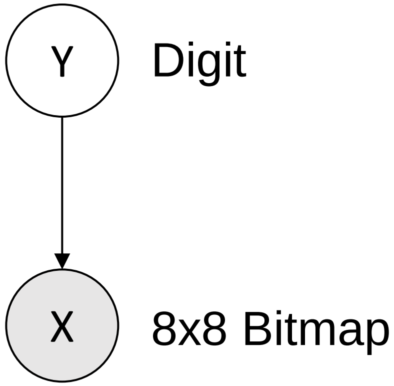
- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(X|Y), P(Y)
  - Hyperparameters: e.g. the amount / type of smoothing to do, k, α
- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
  - For each value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data





Logistic Regression

Two Ways to Model Digit Classification



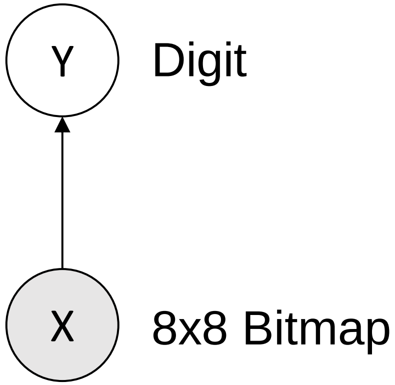
Modeling P(X|Y) and P(Y)

Inference: $P(Y|X) \propto P(Y)P(X|Y)$

More “causal”, modeling how the data is generated

Generative Model

(allows data generation)

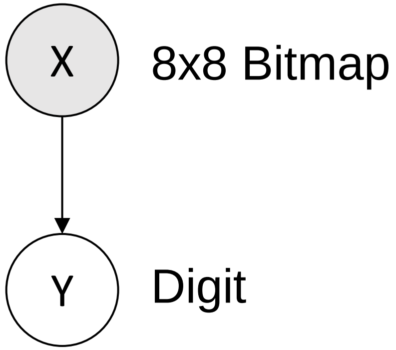


Modeling P(Y|X)

Inference: P(Y|X)

More direct, focusing on the classification task but not how the data is generated

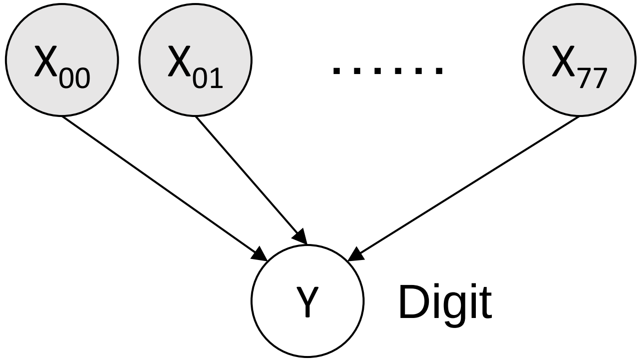
Discriminative Model



Like in Naïve Bayes, we cannot afford to model P(Y|X) in the most general way

$$P(Y|X) = P(Y \mid X_{00}, X_{01}, \ldots, X_{77})$$

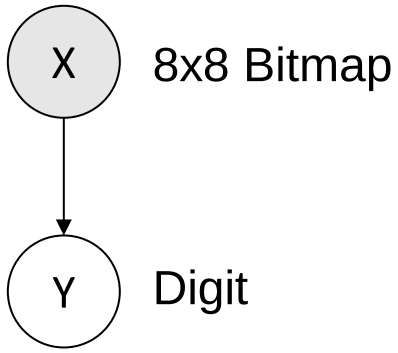
Involves $2^{64} \times 10$ parameters

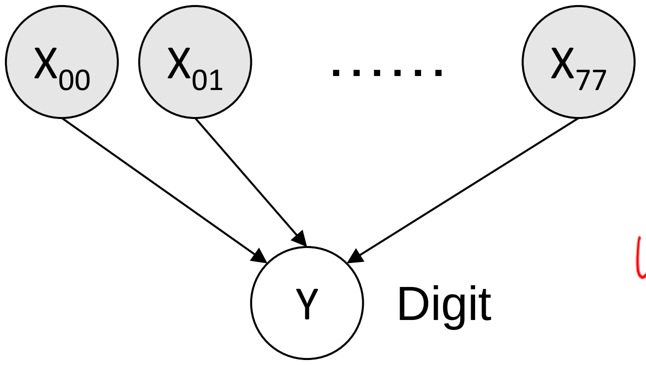


We will again make some “**assumptions**” on P(Y|X) to make the problem tractable.

These assumptions by no means model the true world, but suffice for our classification task.

Logistic Regression





Assumption:

$$P(Y = y \mid X = x) = \frac{\exp(f_w(x,y))}{\sum_{y'} \exp(f_w(x,y'))}$$

$f_w(x, y) \in \mathbb{R}$ is a function defined through parameter w that assigns a score for any (x, y) that indicates how much x and y matches each other.

$$f_w(x, y) = w_{00}^{(y)} x_{00} + w_{01}^{(y)} x_{01} + \cdots + w_{77}^{(y)} x_{77}$$

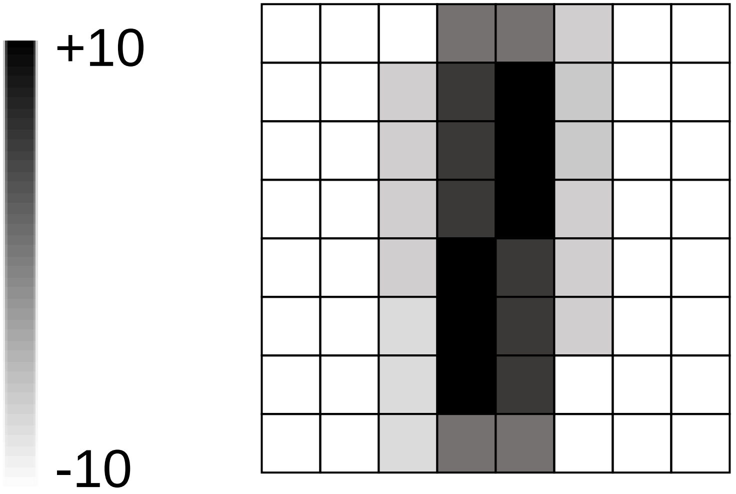
$$= w^{(y)} \cdot x$$

Determining w will determine the whole P(Y|X)

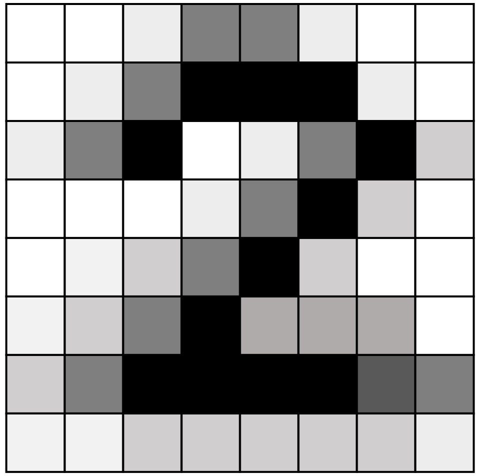
Logistic Regression

$P_w(Y|x)$

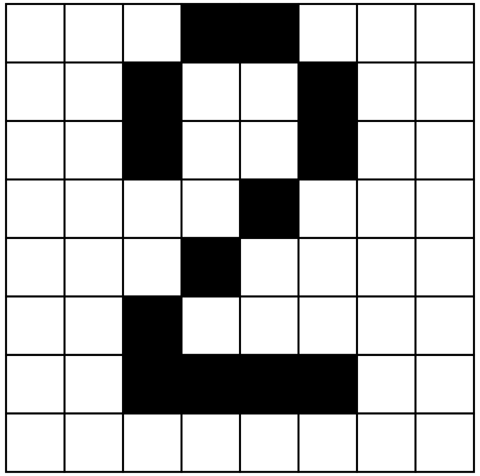
A good w may look like:



$w^{(1)} \in \mathbb{R}^{64}$



$w^{(2)} \in \mathbb{R}^{64}$



$x \in \{0,1\}^{64}$

$$f_w(x,2) = w^{(2)} \cdot x > w^{(1)} \cdot x = f_w(x,1)$$

$$\Rightarrow \; f_w(x,2) > f_w(x,1)$$

$$\Rightarrow \; P_w(2|x) > P_w(1|x)$$

Logistic Regression

Given a set of data $(x_1, y_1), \ldots, (x_n, y_n)$, how can we find a good w?

Maximum Likelihood Estimation (MLE):

Find the w that maximizes

$$\prod_{i=1}^{n} P_w(y_i|x_i) = \prod_{i=1}^{n} \frac{\exp(f_w(x_i, y_i))}{\sum_{y'} \exp(f_w(x_i, y'))} = \prod_{i=1}^{n} \frac{\exp\left(w^{(y_i)} \cdot x_i\right)}{\sum_{y'} \exp\left(w^{(y')} \cdot x_i\right)}$$

This is equivalent to **minimizing**

$$-\log\left(\prod_{i=1}^{n} P_w(y_i|x_i)\right)^{\frac{1}{n}} = \frac{1}{n}\sum_{i=1}^{n} -\log P_w(y_i|x_i) = \frac{1}{n}\sum_{i=1}^{n} \log\left(\sum_{y'} \exp\left(w^{(y')} \cdot x_i - w^{(y_i)} \cdot x_i\right)\right)$$

$L_i(w)$

Logistic loss

Logistic Regression

Example: Suppose that feature dimension = 2 and #Classes = 3

$w^{(1)}$ = [0.7, -0.1]

$w^{(2)}$ = [0.3, -0.4]

$w^{(3)}$ = [-0.9, 0.6]

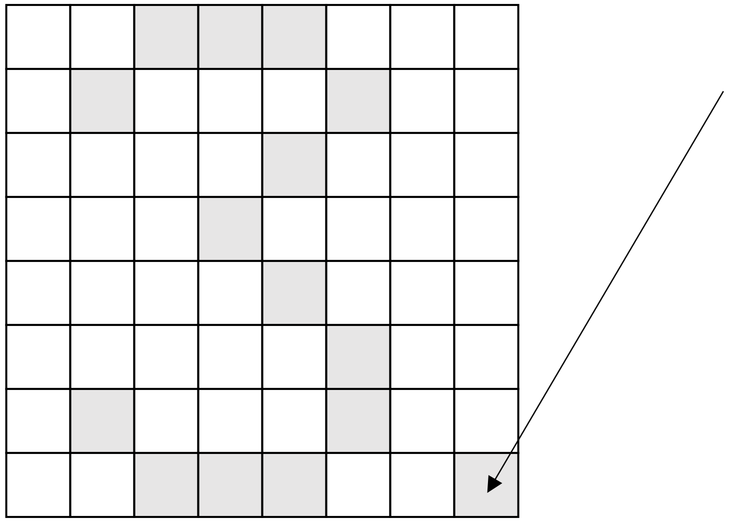
What is the logistic loss of w on the sample (x, y) = ([0, 1], 3) ?

$$\text{loss}(w) = \log\left(\sum_{y'=1}^{3} \exp\left(f_w(x,y') - f_w(x,y)\right)\right)$$

$$= \log\Big(\underbrace{\exp(-0.1 - 0.6)}_{y'=1} + \underbrace{\exp(-0.4 - 0.6)}_{y'=2} + \underbrace{\exp(0)}_{y'=3}\Big)$$

Overfitting in Logistic Regression

Similar to Naïve Bayes + MLE, Logistic Regression + MLE may **overfit** and give **too extreme** distribution that only aligns with the training data



Classified as 2!

Assume that in the training data, pixel (7,0) has ever been ON only when y=2

Then MLE would give $w_{70}^{(2)} = \infty$

$(w_{70}^{(2)} \cdot x)$

⇒ Every sample with x_{70}= ON will be classified as 2

Logistic Regression with Regularization

Minimize $\frac{1}{n}\sum_{i=1}^{n} -\log P_w(y_i|x_i)$ Subject to $\|w^{(y)}\| \leq R$ for all y

or

Minimize $\frac{1}{n}\sum_{i=1}^{n} -\log P_w(y_i|x_i) + \lambda \sum_{y} \|w^{(y)}\|^2$

Hyperparameters

Smaller $\|w^{(y)}\|$ will lead to less extreme P(Y|X)

Optimization Procedure

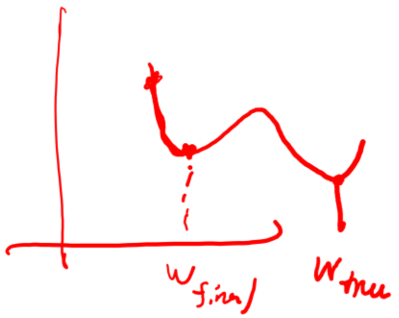
How to Find the Minimizer?

$$\underset{w}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} -\log P_w(y_i|x_i) \quad = \quad \underset{w}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \log \left(\sum_{y'} \exp\left(w^{(y')} \cdot x_i - w^{(y_i)} \cdot x_i \right) \right)$$

Unlike in Naïve Bayes where the optimal model has a “closed-form” solution (i.e., just counting the frequency), here, there is no closed form solution for the optimal w.

We will use **Gradient Descent (GD)** or **Stochastic Gradient Descent (SGD)** to find an approximate optimal solution of w.

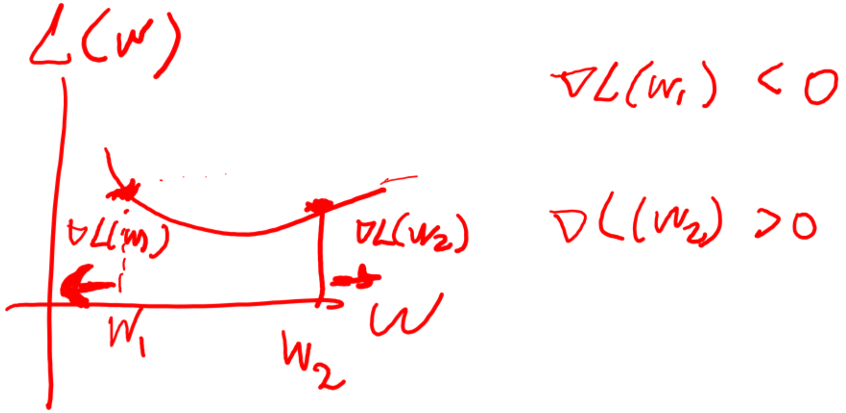
Gradient Descent



In general, if we want to find

$$\underset{w}{\operatorname{argmin}}\; L(w)$$

for some loss function L, we can run the following iterative procedure:



Gradient Descent

Randomly initialize w_0

For $t = 1, 2, \ldots$

$$w_t = w_{t-1} - \eta \nabla L(w_{t-1})$$

hyperparameter

$\eta > 0$ is called the "step size" or the "learning rate"

Exercise

When #Classes=2, the logistic loss can be fully specified by $w = (w^{(1)} - w^{(-1)})/2$

When #Classes=2, the logistic loss can be written as

$$L_i(w) = -\log\left(1 + \exp(-y_i\, w \cdot x_i)\right)$$

where x_i is the feature, and $y_i \in \{-1,1\}$ is the label

$$\nabla L_i(w) = ?$$

$$\nabla\left(-\log\left(1+\exp(-y_i\, w\cdot x_i)\right)\right) = -\frac{1}{1+\exp(-y_i\, w\cdot x_i)}\nabla\left(1+\exp(-y_i\, w\cdot x_i)\right)$$

$$= -\frac{1}{1+\exp(-y_i\, w\cdot x_i)}\cdot\exp(-y_i\, w\cdot x_i)\,\nabla(-y_i\, w\cdot x_i)$$

$$= -\frac{1}{1+\exp(-y_i\, w\cdot x_i)}\exp(-y_i\, w\cdot x_i)\times(-y_i\, x_i)$$

Gradient Descent

If we have n samples, then we would like to find

$$\underset{w}{\operatorname{argmin}}\ L(w) = \underset{w}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} L_i(w) = \underset{w}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} -\log P_w(y_i|x_i)$$

Gradient Descent

Randomly initialize w_0

For $t = 1, 2, \ldots$

$$w_t = w_{t-1} - \eta \nabla L(w_{t-1})$$

$$= \frac{1}{n}\sum_{i=1}^{n} \nabla L_i(w)$$

Per-round complexity =

$n \times$ (complexity of calculating the gradient of logistic loss)

Stochastic Gradient Descent

If we have n samples, then we would like to find

$$\underset{w}{\operatorname{argmin}}\ L(w) = \underset{w}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} L_i(w) = \underset{w}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} -\log P_w(y_i|x_i)$$

Stochastic Gradient Descent

Randomly initialize w_0

For $t = 1, 2, \ldots$

Sample $i \sim \mathrm{Unif}\{1, 2, \ldots, n\}$

$w_t = w_{t-1} - \eta \nabla L_i(w_{t-1})$

Per-round complexity = (complexity of calculating the gradient of logistic loss)

or let $i = (t \bmod n)$ if the dataset is sufficiently shuffled

Because of uniform sampling, $\mathbb{E}_i[\nabla L_i(w)] = \frac{1}{n}\sum_{i=1}^{n} \nabla L_i(w) = \nabla L(w)$ for any w.

Stochastic Gradient Descent with Mini-batch

If we have n samples, then we would like to find

$$\operatorname*{argmin}_w \; L(w) = \operatorname*{argmin}_w \frac{1}{n}\sum_{i=1}^{n} L_i(w) = \operatorname*{argmin}_w \frac{1}{n}\sum_{i=1}^{n} -\log P_w(y_i|x_i)$$

Stochastic Gradient Descent with Minibatch (Less noisy than SGD without minibatch)

Randomly initialize w_0

For $t = 1, 2, \ldots$

Sample a set $B_i \subset \{1, 2, \ldots, n\}$ with size $|B_i| = b$

$$w_t = w_{t-1} - \eta \cdot \frac{1}{b}\sum_{i \in B_i} \nabla L_i(w_{t-1})$$

The gradient of different samples in a minibatch can be computed parallelly with GPUs

or forming the mini-batches following the order 1, 2, .., n if the dataset is sufficiently shuffled.

Implicit Regularization by GD/SGD

- If we set $w_0 \approx 0$ and let η to be small enough (and don't train too long), then the final $\|w\|$ will not be too large.
- In this case, we don't really need to add constraint $\|w\| \leq R$ or add penalty $\lambda\|w\|^2$

Recap: Logistic Regression for Classification

- Get dataset consisting of (X, Y) pairs:

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) \in \mathbb{R}^d \times \{1, 2, \dots, C\}$$

- Write out the **objective function / loss function**:

$$\frac{1}{n}\sum_{i=1}^{n} -\log P_w(y_i|x_i) = \frac{1}{n}\sum_{i=1}^{n} \log\left(\sum_{y'} \exp\left(w^{(y')} \cdot x_i - w^{(y_i)} \cdot x_i\right)\right)$$

- Use stochastic gradient descent (usually with minibatch) to minimize the loss
- Output the final w for inference

Classification without Probabilistic Modeling

- Often times, we don't care about **P(Y|X=x)**, i.e., the probability/likelihood of being each class. Instead, we just want to know **argmax P(Y|X=x)**, i.e. which class has the largest likelihood.

- There are several classic classification algorithms that do not go through probabilistic modeling, such as Support Vector Machine (SVM) and Perceptron algorithm.