work with Data, instead of model

1.Learning—>classification:
INPUT(X) — extraction —>
(Eg.3D has x, y, z, color…)
Features—Machine learning—>
predicted OUTPUT(Y)

2.ML:
train data(some domain):
attributes, class belongs to
(learning balanced way: learn and
able to generate not matching)

tune parameter: not fearure parameter,
it optimalize ML model and training process

Eg.
Spam filter
SET UP:get a large collection, labeled each
Note: someone has to hand label all this data!
Want to learn to predict labels of new , future emails
INPUT: an email
FEATURE(attributes): spam/ham feature
OUTPUT: Spam/ham

Eg.
Digit Recognition

- Input: images / pixel grids
- Output: a digit 0-9

- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images

- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …

other classification also work with ML
object recognition, Medical diagnosis..

General Naïve Bayes

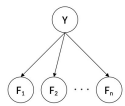
- A general Naive Bayes model: Y -> class, F -> Features

$$P(Y, F_1 \dots F_n) = P(Y) \prod_i P(F_i|Y)$$

|Y| parameters

|Y| x |F|^n values

n x |F| x |Y| parameters



- We only have to specify how each feature depends on the class
- Total number of parameters is *linear* in n
- Model is very simplistic, but often works anyway

Example: Naïve Bayes for Spam Filter

- Step 4.2: Inference
- Instantiate features (evidence):
  - F_1 = yes
  - F_2 = 1
- Compute joint probabilities:
  - P(Y = spam, F_1 = yes, F_2 = 1) = P(Y = spam) P(F_1 = yes | spam) P(F_2 = 1 | spam) = 0.4 * 0.1 * 0.12 = 0.0048
  - P(Y = ham, F_1 = yes, F_2 = 1) = P(Y = ham) P(F_1 = yes | ham) P(F_2 = 1 | ham) = 0.6 * 0.7 * 0.07 = 0.0294
- Normalize:
  - P(Y = spam | F_1 = yes, F_2 = 1) = 0.0048 / (0.0048+0.0294) = 0.14
  - P(Y = ham | F_1 = yes, F_2 = 1) = 0.0294 / (0.0048+0.0294) = 0.86
- Classification result:
  - 14% chance the email is spam. 86% chance it's ham.
  - Or, if you don't need probabilities, note that 0.0294 > 0.0048 and guess ham.

Y: The label (spam or ham)

Y	P(Y)
ham	0.6
spam	0.4

F_1: do I know the sender?

F_1	Y	P(F_1|Y)
yes	ham	0.7
no	ham	0.3
yes	spam	0.1
no	spam	0.9

F_2: # of occurrences of FREE

F_2	Y	P(F_2|Y)
0	ham	0.85
1	ham	0.07
2	ham	0.08
0	spam	0.75
1	spam	0.12
2	spam	0.13

there maybe lots of words, Ignor Less Informative Words

Inference for Naïve Bayes

- Goal: compute posterior distribution over label variable Y
  - Step 1: get joint probability of label and evidence for each label

$$P(Y, f_1 \dots f_n) = \begin{bmatrix} P(y_1, f_1 \dots f_n) \\ P(y_2, f_1 \dots f_n) \\ \vdots \\ P(y_k, f_1 \dots f_n) \end{bmatrix} \Rightarrow \begin{bmatrix} P(y_1) \prod_i P(f_i|y_1) \\ P(y_2) \prod_i P(f_i|y_2) \\ \vdots \\ P(y_k) \prod_i P(f_i|y_k) \end{bmatrix}$$

Vectors

$$P(f_1 \dots f_n)$$

  - Step 2: sum to get probability of evidence
  - Step 3: normalize by dividing Step 1 by Step 2

$$P(Y|f_1 \dots f_n)$$

Eg. NB

Model based Classification

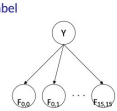
- Model-based approach
  - Build a model (e.g. Bayes' net) where both the label and features are random variables
  - Instantiate any observed features
  - Query for the distribution of the label conditioned on the features
- Challenges
  - What structure should the BN have?
  - How should we learn its parameters?

Naïve Bayes for Digits

- Naïve Bayes: Assume all features are independent effects of the label
- Simple digit recognition version:
  - One feature (variable) F_{ij} for each grid position <i,j>
  - Feature values are on / off based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow (F_{0,0}=0 \; F_{0,1}=0 \; F_{0,2}=1 \; F_{0,3}=1 \; F_{0,4}=0 \; \dots F_{15,15}=0)$$

  - Here: lots of features, each is binary valued
- Naïve Bayes model: $P(Y|F_{0,0} \dots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$
- What do we need to learn?



Naïve Bayes Model

output Variable Predicted

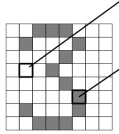
- Random variables in this Bayes' net:
  - Y = The label
  - F_1, F_2, …, F_n = The n features
- Probability tables in this Bayes' net:
  - P(Y) = Probability of each label occurring, given no information about the features. Sometimes called the *prior*.
  - P(F_i|Y) = One table per feature. Probability distribution over a feature, given the label.

P(Y) baseline

Naïve Bayes for Digits: Parameters

$P(Y)$

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



$P(F_{3,1} = on|Y)$

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

$P(F_{5,5} = on|Y)$

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

General Naïve Bayes

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

Posterior · Likelihood · Prior · Prior

这张幻灯片介绍了使用朴素贝叶斯分类器所需的组件以及这些组件的来源。

1. **推理方法**：这部分我们刚刚讨论过。使用朴素贝叶斯分类器进行推理时，我们首先需要一组概率值：类别的先验概率 $P(Y)$ 和条件概率 $P(F_i|Y)$。然后，我们使用标准的推理方法来计算给定所有特征 $F_1, F_2, ..., F_n$ 后，类别 Y 的后验概率 $P(Y|F_1, ..., F_n)$。这里并没有什么新颖的内容，这是朴素贝叶斯方法的标准操作。
2. **局部条件概率表的估计**：我们需要两种类型的概率估计值：
   - 类别的先验概率 $P(Y)$，也就是在观察任何特征之前，每个类别可能出现的概率。
   - 每个特征给定类别的条件概率 $P(F_i|Y)$，也就是当我们知道某个类别时，每个特征出现的概率。

important

这些概率值被统称为模型的参数（这里用符号 θ 表示）。

在之前的讨论中，我们可能会认为这些概率值凭空出现，但实际上它们通常是通过对训练数据进行统计而获得的。例如，通过计算训练数据中每个类别出现的频率来估计先验概率，以及在每个类别下，不同特征出现的频率来估计条件概率。

ERM：no classifier

这张幻灯片主要讨论了如何评估机器学习模型的质量，以及“经验风险最小化”（Empirical Risk Minimization, ERM）的概念。

1. **模型质量评估**：在机器学习中，我们通常想要评估模型的质量，确保它在未知数据上表现良好。理想情况下，我们希望模型在真实的测试数据分布上表现最好。
2. **经验风险最小化**：ERM是机器学习的一个基本原则，它的目标是找到在训练数据集上表现最佳的模型。由于我们通常不知道数据的真实分布，ERM策略就是选择在实际训练集上“看起来”最好的模型。将寻找最佳模型的过程表述为一个优化问题，目标是最小化训练集上的损失函数，这个损失函数衡量的是模型预测和实际值之间的误差。
3. **主要担忧：过拟合**：
   - 过拟合是指模型在训练数据上表现非常好，但在未见过的数据上表现不佳。这通常发生在模型过于复杂，以至于它学习了训练数据中的噪声，而不仅仅是底层的数据生成分布。
   - 为了缓解过拟合，增加训练数据是一个有效的策略，因为更多的数据可以减少采样的方差，并使训练数据更好地代表真实的测试数据分布。
   - 另一个策略是限制模型假设的复杂度，比如通过正则化技术，这可以鼓励模型简化它的结构，减少对训练数据中噪声的学习。

Important Concepts

- How do we check that we're not overfitting during training?
- Split training data into 3 different sets:
  - Training set
  - Held out set (more on this later)
  - Test set
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - Compute accuracy of test set
  - Very important: never "peek" at the test set!
- Evaluation (many metrics possible, e.g. accuracy)
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well
  - We'll investigate overfitting and generalization formally in a few lectures

Training Data | Held-Out Data | Test Data

We don't want to be too specific, but smooth and regularize the estimate

Parameter Estimation

- Estimating the distribution of a random variable
- *Elicitation:* ask a human (why is this hard?)
- *Empirically:* use training data (learning!)
  - E.g.: for each outcome x, look at the *empirical rate* of that value:

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

count(x) → Count r

2r+1b

$P_{\text{ML}}(r) = 2/3$

- This is the estimate that maximizes the *likelihood of the data*

$$L(x, \theta) = \prod_i P_\theta(x_i)$$

Leading to Parameter Estimate (Distribution of Variable)

Elicitation: ask human

Empirical way: use training data

Eg.

- Example: The parameter θ is the true fraction of red beans in the jar. You don't know θ but would like to estimate it.
- Collecting training data: You randomly pull out 3 beans:

r r b

- Estimating θ using counts, you guess 2/3 of beans in the jar are red.
- Can we mathematically show that using counts is the "right" way to estimate θ?

right answer?

proof:

Parameter Estimation with Maximum Likelihood

- **Maximum likelihood estimation:** Choose the θ value that maximizes the probability of the observation
  - In other words, choose the θ value that maximizes P(observation | θ)
  - For our problem:

P(observation | θ)

= P(randomly selected 2 red and 1 blue | θ of beans are red)

= P(red | θ) * P(red | θ) * P(blue | θ)

= $\theta^2 (1-\theta)$

maximize

  - We want to compute:

$$\arg\max_\theta \theta^2 (1-\theta)$$

- We want to compute:

$$\arg\max_\theta \theta^2 (1-\theta)$$

If need, math way

- Set derivative to 0, and solve!
  - Common issue: The likelihood (expression we're maxing) is the product of a lot of probabilities. This can lead to complicated derivatives.
  - Solution: Maximize the log-likelihood instead. Useful fact:

$$\arg\max_\theta f(\theta) = \arg\max_\theta \ln f(\theta)$$

Maximum Likelihood?

- Relative frequencies are the maximum likelihood estimates

Best · Features · Not ML itself

$$\theta_{ML} = \arg\max_\theta P(\mathbf{X}|\theta) = \arg\max_\theta \prod_i P_\theta(X_i) \Rightarrow P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

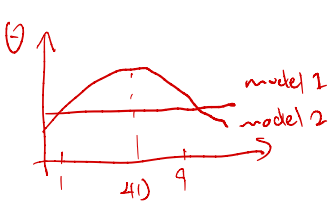
- Another way showing it (semi proof) is as follows:

$$\arg\max_\theta P(\theta|\mathbf{X}) = \arg\max_\theta \prod_i P_\theta(X_i)$$

by Bayes rule

$$= \arg\max_\theta P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X}) \Rightarrow \text{????}$$

$$= \arg\max_\theta P(\mathbf{X}|\theta)P(\theta)$$



Parameter Estimation with Maximum Likelihood

$\underset{\theta}{\operatorname{argmax}}\, \theta^2(1-\theta)$	Find θ that maximizes likelihood
$= \underset{\theta}{\operatorname{argmax}} \ln\left(\theta^2(1-\theta)\right)$	Find θ that maximizes log-likelihood (will be the same θ)
$\frac{d}{d\theta} \ln\left(\theta^2(1-\theta)\right) = 0$	Set derivative to 0
$\frac{d}{d\theta}\left[\ln(\theta^2) + \ln(1-\theta)\right] = 0$	Logarithm rule: products become sums
$\frac{d}{d\theta}\left[2\ln(\theta) + \ln(1-\theta)\right] = 0$	Logarithm rule: exponentiation becomes multiplication
$\frac{d}{d\theta} 2\ln(\theta) + \frac{d}{d\theta}\ln(1-\theta) = 0$	Now we can derive each term of the original product
$\frac{2}{\theta} - \frac{1}{1-\theta} = 0$	separately Reminder: Derivative of ln(θ) is 1/θ
$\theta = \frac{2}{3}$	Use algebra to solve for θ. If we used arbitrary red and blue counts r and b instead of r=2 and b=1, we'd get θ = r / (r+b), the count estimate.

Laplace Smoothing

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

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

$$P_{LAP,0}(X) =$$

$$P_{LAP,1}(X) =$$

$$P_{LAP,100}(X) =$$

  - What's Laplace with k = 0?
  - k is the strength of the prior
- Laplace for conditionals:
  - Smoothing each condition independently:

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

- $c(x)$ 是观测到的事件x的次数。
- N 是总的观测次数。
- $|X|$ 是所有不同事件的数量。
- k 是平滑参数，也称作先验的强度。

在拉普拉斯平滑中，如果 $k=0$，则没有平滑效果。我们得到的是事件的经验概率（直接的频率概率）。当 $k>0$ 时，k 越大，平滑效果越明显，也就是说对概率估计的影响越大。例如，$k=1$ 时通常被称为加一平滑（因为我们给每个事件加上1），这在许多情况下可以避免零概率问题，并且提供了一个比较温和的平滑效果。而 $k=100$ 会给予先验一个非常强的权重，可能会过度平滑，使得估计概率偏离实际观察的数据。

Error

- Need more features– words aren't enough!
  - Have you emailed the sender before?
  - Have 1K other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?
- Can add these information sources as new variables in the NB model