CS 188: Artificial Intelligence
Learning I:
Agent Testing Today!

Instructors: Stuart Russell and Dawn Song
University of California, Berkeley
Policy Iteration

- Basic idea: make the implied policy in $U$ explicit, compute its long-term implications for value
- Repeat until no change in policy:
  - Step 1: Policy evaluation: calculate value $U^{\pi_k}$ for current policy $\pi_k$
  - Step 2: Policy improvement: extract the new implied policy $\pi_{k+1}$ from $U^{\pi_k}$
- It’s still optimal!
- Can converge (much) faster under some conditions
Policy Evaluation
Fixed Policies

- Expectimax trees max over all actions to compute the optimal values.
- If we fixed some policy $\pi(s)$, then the tree would be simpler – only one action per state.
  - ... though the tree’s value would depend on which policy we fixed.
Another basic operation: compute the utility of a state $s$ under a fixed (generally non-optimal) policy.

Define the utility of a state $s$, under a fixed policy $\pi$:

$$U^\pi(s) = \text{expected total discounted rewards starting in } s \text{ and following } \pi$$

Recursive relation (one-step look-ahead / Bellman equation):

$$U_i(s) = \sum_{s'} P(s' | s, \pi_i(s)) [R(s, \pi_i(s), s') + \gamma U_i(s')]$$

$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s' | s, a) [R(s, a, s') + \gamma U(s')] \quad \text{(Bellman equation)}$$
How do we calculate the U’s for a fixed policy \( \pi \)?

Idea 1: Turn recursive Bellman equations into updates (like value iteration)

\[ U_0^\pi(s) = 0 \]

\[
U_{i+1}(s) \leftarrow \sum_{s'} P(s' | s, \pi_i(s)) [R(s, \pi_i(s), s') + \gamma U_i(s')] 
\]

Efficiency: \( O(S^2) \) per iteration

Idea 2: Without the maxes, the Bellman equations are just a linear system

- Solve with Matlab (or your favorite linear system solver)
Policy Iteration
Policy Iteration

- **Evaluation**: For fixed current policy $\pi$, find values with policy evaluation:
  - Iterate until values converge:

  $$U_{i+1}(s) \leftarrow \sum_{s'} P(s' | s, \pi_i(s))[R(s, \pi_i(s), s') + \gamma U_i(s')]$$

- **Improvement**: For fixed values, get a better policy using policy extraction:
  - One-step look-ahead:

  $$\pi_{i+1}(s) = \arg \max_a \sum_{s'} P(s' | s, a) \left[ R(s, a, s') + \gamma U_i^{\pi_i(s')}(s') \right]$$
Comparison

- Both value iteration and policy iteration compute the same thing (all optimal values)

- In value iteration:
  - Every iteration updates both the values and (implicitly) the policy
  - We don’t track the policy, but taking the max over actions implicitly re-computes it

- In policy iteration:
  - We do several passes that update utilities with fixed policy (each pass is fast because we consider only one action, not all of them)
    - Policy evaluation reveals long-term effects of policy, unlike local value updates
  - After the policy is evaluated (looking at those long-term effects), a new policy is chosen (slow like a value iteration pass)
    - The new policy will be better (or we’re done)

- Both are dynamic programs for solving MDPs

- In fact, any fair sequence of value and/or policy updates on any states will converge to an optimal solution!
So you want to....
- Compute optimal values: use value iteration or policy iteration
- Compute values for a particular policy: use policy evaluation
- Turn your values into a policy: use policy extraction (one-step lookahead)

These all look the same!
- They basically are – they are all variations of Bellman updates
- They all use one-step lookahead expectimax fragments
- They differ only in whether we plug in a fixed policy or max over actions
The Bellman Equations

How to be optimal:
Step 1: Take correct first action
Step 2: Keep being optimal
AlphaGo & AlphaStar: Winning over World Champions

Source: David Silver
Deep Learning Powering Everyday Products

[Images of smart speakers from pcmag.com and theverge.com]
Lectures on learning

- **Learning**: a process for improving the performance of an agent through experience
- Learning I (today):
  - The general idea: generalization from experience
  - Supervised learning: classification and regression
- Learning II: neural networks and deep learning
- Reinforcement learning: learning complex V and Q functions
Supervised learning

- **To learn an unknown target function** $f$
- **Input:** a *training set* of *labeled examples* $(x_j,y_j)$ where $y_j = f(x_j)$
  - E.g., $x_j$ is an image, $f(x_j)$ is the label “giraffe”
  - E.g., $x_j$ is a seismic signal, $f(x_j)$ is the label “explosion”
- **Output:** *hypothesis* $h$ that is “close” to $f$, i.e., predicts well on unseen examples ("test set")
- Many possible hypothesis families for $h$
  - Linear models, logistic regression, neural networks, decision trees, examples (nearest-neighbor), grammars, kernelized separators, etc etc
- **Classification** = learning $f$ with discrete output value
- **Regression** = learning $f$ with real-valued output value
Inductive Learning (Science)

- Simplest form: learn a function from examples
  - A target function: \( g \)
  - Examples: input-output pairs \((x, g(x))\)
  - E.g. \( x \) is an email and \( g(x) \) is spam / ham
  - E.g. \( x \) is a house and \( g(x) \) is its selling price

- Problem:
  - Given a hypothesis space \( H \)
  - Given a training set of examples \( x_i \)
  - Find a hypothesis \( h(x) \) such that \( h \sim g \)

- Includes:
  - Classification (outputs = class labels)
  - Regression (outputs = real numbers)
Classification example: Object recognition

\[ x = \begin{array}{cccccc}
\text{giraffe} & \text{giraffe} & \text{giraffe} & \text{llama} & \text{llama} & \text{llama}
\end{array}\]

\[ f(x) = \begin{array}{cccccc}
\text{giraffe} & \text{giraffe} & \text{giraffe} & \text{llama} & \text{llama} & \text{llama}
\end{array}\]

\[ X = \text{giraffe} \]
\[ f(x) = ? \]
Example: Spam Filter

- **Input:** an email
- **Output:** spam/ham

**Setup:**
- Get a large collection of example emails, each labeled “spam” or “ham” (by hand)
- Learn to predict labels of new incoming emails
- Classifiers reject 200 billion spam emails per day

**Features:** The attributes used to make the ham / spam decision
- Words: FREE!
- Text Patterns: $dd, CAPS
- Non-text: SenderInContacts, AnchorLinkMismatch
- ...

---

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential 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, I know 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: Digit Recognition

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

**Setup:**
- MNIST data set of 60K collection hand-labeled images
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new digit images

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

- Medical diagnosis
  - input: symptoms
  - output: disease
- Automatic essay grading
  - input: document
  - output: grades
- Fraud detection
  - input: account activity
  - output: fraud / no fraud
- Email routing
  - input: customer complaint email
  - output: which department needs to ignore this email
- Fruit and vegetable inspection
  - input: image (or gas analysis)
  - output: moldy or OK
- ... many more
Regression example: Curve fitting
Regression example: Curve fitting
Regression example: Curve fitting
Regression example: Curve fitting
Regression example: Curve fitting
Basic questions

- Which hypothesis space $H$ to choose?
- How to measure degree of fit?
- How to trade off degree of fit vs. complexity?
  - “Ockham’s razor”
- How do we find a good $h$?
- How do we know if a good $h$ will predict well?
Training and Testing
A few important points about learning

- **Data:** labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set

- **Features:** attribute-value pairs which characterize each x

- **Experimentation cycle**
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of test set
  - Very important: never “peek” at the test set!

- **Evaluation**
  - 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
  - Underfitting: fits the training set poorly

- **Diagram:**
  - Training Data
  - Held-Out Data (validation set)
  - Test Data
A few important points about learning

- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
    - Why?
    - 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

- What are examples of hyperparameters?
Supervised Learning

- **Classification** = learning $f$ with discrete output value
- **Regression** = learning $f$ with real-valued output value
Linear Regression

Hypothesis family: Linear functions
(x, y=f(x)), x: house size, y: house price
Linear regression = fitting a straight line/hyperplane

Prediction: \( h_w(x) = w_0 + w_1 x \)

Berkeley house prices, 2009
Prediction error

Error on one instance: $y - h_w(x)$
Find w

- Define loss function
- Find $w^*$ to minimize loss function
Least squares: Minimizing squared error

- L2 loss function: sum of squared errors over all examples
  - Loss = ____________________________
- We want the weights \( \mathbf{w}^* \) that minimize loss
- At \( \mathbf{w}^* \) the derivatives of loss w.r.t. each weight are zero:
  - \( \frac{\partial \text{Loss}}{\partial w_0} = ____________________________ \)
  - \( \frac{\partial \text{Loss}}{\partial w_1} = ____________________________ \)
- Exact solutions for \( N \) examples:
  - \( w_1 = \frac{[N \sum_j x_j y_j - (\sum_j x_j)(\sum_j y_j)]}{[N \sum_j x_j^2 - (\sum_j x_j)^2]} \) and \( w_0 = \frac{1}{N} [\sum_j y_j - w_1 \sum_j x_j] \)
- For the general case where \( x \) is an \( n \)-dimensional vector
  - \( \mathbf{X} \) is the data matrix (all the data, one example per row); \( \mathbf{y} \) is the column of labels
  - \( \mathbf{w}^* = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \)
Least squares: Minimizing squared error

- L2 loss function: sum of squared errors over all examples
  - Loss = $\sum_j (y_j - h_w(x_j))^2 = \sum_j (y_j - (w_0 + w_1 x_j))^2$
- We want the weights $w^*$ that minimize loss
- At $w^*$ the derivatives of loss w.r.t. each weight are zero:
  - $\frac{\partial \text{Loss}}{\partial w_0} = -2 \sum_j (y_j - (w_0 + w_1 x_j)) = 0$
  - $\frac{\partial \text{Loss}}{\partial w_1} = -2 \sum_j (y_j - (w_0 + w_1 x_j)) x_j = 0$
- Exact solutions for $N$ examples:
  - $w_1 = \left[ N \Sigma_j x_j y_j - (\Sigma_j x_j)(\Sigma_j y_j) \right] / \left[ N \Sigma_j x_j^2 - (\Sigma_j x_j)^2 \right]$ and $w_0 = 1/N [\Sigma_j y_j - w_1 \Sigma_j x_j]$
- For the general case where $x$ is an $n$-dimensional vector
  - $X$ is the data matrix (all the data, one example per row); $y$ is the column of labels
  - $w^* = (X^T X)^{-1} X^T y$
Regression vs Classification

- Linear regression when output is binary, $y \in \{-1, 1\}$
  - $h_w(x) = w_0 + w_1 x$

- Linear classification
  - Used with discrete output values
  - Threshold a linear function
  - $h_w(x) = 1$, if $w_0 + w_1 x \geq 0$
  - $h_w(x) = -1$, if $w_0 + w_1 x < 0$
  - $w$: weight vector
  - Activation function $g$
Threshold perceptron as linear classifier
A **threshold perceptron** is a single unit that outputs

- $y = h_w(x) = 1$ when $w.x \geq 0$
- $y = -1$ when $w.x < 0$

In the input vector space

- Examples are points $x$
- The equation $w.x = 0$ defines a **hyperplane**
- One side corresponds to $y = 1$
- Other corresponds to $y = -1$
$x$ classified into positive class  

$\text{ }$  

$x$ classified into negative class
Dear Stuart,

I’m leaving Macrosoft to return to academia. The money is great here but I prefer to be free to do my own research; and I really love teaching undergrads!

Do I need to finish my BA first before applying?

Best wishes

Bill

\[
\begin{align*}
  w_0 & : -3 \\
  w_{\text{free}} & : 4 \\
  w_{\text{money}} & : 2
\end{align*}
\]

\[
\begin{align*}
  x_0 & : 1 \\
  x_{\text{free}} & : 1 \\
  x_{\text{money}} & : 1
\end{align*}
\]

\[
w.x = -3x_1 + 4x_1 + 2x_1 = 3
\]
Weight Updates

Need a different solution than before given the characteristic of perceptron
Perceptron learning rule

- If true $y^* \neq h_w(x)$ (an error), adjust the weights
- If $w.x < 0$ but the output should be $y^*=1$
  - This is called a **false negative**
  - Should *increase* weights on **positive** inputs
  - Should *decrease* weights on **negative** inputs
- If $w.x > 0$ but the output should be $y^*=-1$
  - This is called a **false positive**
  - Should *decrease* weights on **positive** inputs
  - Should *increase* weights on **negative** inputs
Perceptron Learning Rule

- Start with weights = 0
- For each training instance:
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. $y^*$ is true label.

$$w = w + y^* x$$

$$y = h_w(x) = 1 \text{ when } w \cdot x \geq 0$$
$$= -1 \text{ when } w \cdot x < 0$$

Mis-classifying $x$ with old $w$

Updating $w$

Updated classification of $x$
Dear Stuart, I wanted to let you know that I have decided to leave Macrosoft and return to academia. The money is great here but I prefer to be free to pursue more interesting research and I really love teaching undergraduates! Do I need to finish my BA first before applying?

Best wishes
Bill

\[
\begin{align*}
\mathbf{w} &\leftarrow \mathbf{w} + \alpha \, y^* \, \mathbf{x} \\
\alpha &= 0.5
\end{align*}
\]

\[
\begin{align*}
\mathbf{w} &\leftarrow (-3,4,2) + 0.5 \,(0 - 1) \,(1,1,1) \\
&= (-3.5,3.5,1.5)
\end{align*}
\]
Perceptron convergence theorem

- A learning problem is **linearly separable** iff there is some hyperplane exactly separating positive from negative examples.

- Convergence: if the training data are linearly **separable**, perceptron learning applied repeatedly to the training set will eventually converge to a perfect separator.
Example: Earthquakes vs nuclear explosions

63 examples, 657 updates required
A learning problem is **linearly separable** iff there is some hyperplane exactly separating +ve from –ve examples.

Convergence: if the training data are separable, perceptron learning applied repeatedly to the training set will eventually converge to a perfect separator.

Convergence: if the training data are **non-separable**, perceptron learning will converge to a minimum-error solution provided the learning rate $\alpha$ is decayed appropriately (e.g., $\alpha=1/t$).
Perceptron learning with fixed $\alpha$

71 examples, 100,000 updates
fixed $\alpha = 0.2$, no convergence
Perceptron learning with decaying $\alpha$

71 examples, 100,000 updates
decaying $\alpha = 1000/(1000 + t)$, near-convergence
Non-Separable Case

Even the best linear boundary makes at least one mistake
Other Linear Classifiers

- Perceptron is perfectly happy as long as it separates the training data.

- **Logistic Regression**
  \[
  g_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-x}}
  \]

- **Support Vector Machines (SVM)**
  - Maximize margin between boundary and nearest points.
Perceptrons hopeless for XOR function

(a) $x_1 \text{ and } x_2$

(b) $x_1 \text{ or } x_2$

(c) $x_1 \text{ xor } x_2$
Basic questions

- Which hypothesis space $H$ to choose?
- How to measure degree of fit?
- How to trade off degree of fit vs. complexity?
  - “Ockham’s razor”
- How do we find a good $h$?
- How do we know if a good $h$ will predict well?
Classical stats/ML: Minimize loss function

- Which hypothesis space $H$ to choose?
  - *E.g.*, linear combinations of features: $h_w(x) = w^T x$
- How to measure degree of fit?
  - *Loss function*, *e.g.*, squared error $\sum_j (y_j - w^T x)^2$
- How to trade off degree of fit vs. complexity?
  - *Regularization*: complexity penalty, *e.g.*, $||w||^2$
- How do we find a good $h$?
  - *Optimization* (*closed-form, numerical*); *discrete search*
- How do we know if a good $h$ will predict well?
  - *Try it and see* (*cross-validation, bootstrap, etc.*)
Probabilistic: Max. likelihood, max. a priori

- Which hypothesis space $H$ to choose?
  - *Probability model* $P(y \mid x, h)$, e.g., $Y \sim N(w^T x, \sigma^2)$

- How to measure degree of fit?
  - *Data likelihood* $\Pi_j P(y_j \mid x_j, h)$

- How to trade off degree of fit vs. complexity?
  - *Regularization or prior*: $\arg\max_h P(h) \Pi_j P(y_j \mid x_j, h)$ (*Max a Priori*)

- How do we find a good $h$?
  - *Optimization (closed-form, numerical); discrete search*

- How do we know if a good $h$ will predict well?
  - *Empirical process theory* (generalizes Chebyshev, CLT, PAC...);
  - *Key assumption is (i)id*
Bayesian: Computing posterior over H

- Which hypothesis space $H$ to choose?
  - *All hypotheses with nonzero a priori probability*
- How to measure degree of fit?
  - *Data probability, as for MLE/MAP*
- How to trade off degree of fit vs. complexity?
  - *Use prior, as for MAP*
- How do we find a good $h$?
  - *Don’t! Bayes predictor* $P(y|x,D) = \sum_h P(y|x,h) P(D|h) P(h)$
- How do we know if a good $h$ will predict well?
  - *Silly question! Bayesian prediction is optimal!!*
Parameter Estimation
Maximum Likelihood Parameter Estimation

- **Estimating the distribution of a random variable**
  - E.g., here is a coin; what is the probability $\theta$ of heads?

- **Evidence** $\mathbf{x} = x_1, \ldots, x_N$
  - E.g., three independent coin tosses $X_1=$heads, $X_2=$heads, $X_3=$tails

- **Likelihood:** probability of the evidence $P(x_1, \ldots, x_N ; \theta)$
  - E.g., $P(X_1=$heads, $X_2=$heads, $X_3=$tails ; $\theta) =$ ___________

- **Maximum likelihood:** What value $\theta_{ML}$ maximizes the likelihood?

- **Log likelihood:** $L(\mathbf{x}; \theta) = \log P(\mathbf{x}; \theta)$
  - E.g., $L(\mathbf{x}; \theta) =$ ___________

- $\theta_{ML}$ also maximizes the log likelihood and it’s easier to differentiate

- $\partial L/\partial \theta =$ ___________

- $\theta_{ML} =$ ________

- For $h$ heads and $t$ tails, $\theta_{ML} =$ __________
Maximum Likelihood Parameter Estimation

- Estimating the distribution of a random variable
  - E.g., here is a coin; what is the probability $\theta$ of heads?

- Evidence $\mathbf{x} = x_1, \ldots, x_N$
  - E.g., three independent coin tosses $X_1=\text{heads}, X_2=\text{heads}, X_3=\text{tails}$

- Likelihood: probability of the evidence $P(x_1, \ldots, x_N; \theta)$
  - E.g., $P(X_1=\text{heads}, X_2=\text{heads}, X_3=\text{tails}; \theta) = \theta^2(1-\theta)$

- Maximum likelihood: What value $\theta_{\text{ML}}$ maximizes the likelihood?

- Log likelihood: $L(\mathbf{x}; \theta) = \log P(\mathbf{x}; \theta)$
  - E.g., $L(\mathbf{x}; \theta) = 2 \log \theta + \log(1-\theta)$

- $\theta_{\text{ML}}$ also maximizes the log likelihood and it’s easier to differentiate
  - $\frac{\partial L}{\partial \theta} = 2/\theta - 1/(1-\theta) = 0$
  - $\theta_{\text{ML}} = 2/3$

- For $h$ heads and $t$ tails, $\theta_{\text{ML}} = h/(h+t)$
Laplace Smoothing

- Suppose we see three heads: is a $\theta_{ML} = 0$ a reasonable estimate?
- Laplace smoothing with strength $\alpha$:
  - Pretend you saw every outcome $\alpha$ times before starting
  - $\theta_{Lap} = (h+\alpha)/[(h+\alpha) + (t+\alpha)]$
  - $= (3+\alpha)/(3+2\alpha)$
  - In general, for a K-valued variable:
    - $\theta_k = (N_k+\alpha) / \Sigma_k(N_k+\alpha) = (N_k+\alpha) / (N + K\alpha)$
    - For $\alpha \gg N$, $\theta_k$ tends to $1/K$ (uniform prior)
    - For $\alpha \ll N$, $\theta_k$ tends to $N_k/N$ (ML estimate)
Probabilistic Classification
Example: Spam Filter

- **Input:** an email
- **Output:** spam/ham

**Setup:**
- Get a large collection of example emails, each labeled “spam” or “ham”
- Note: someone has to hand label all this data!
- Want to learn to predict labels of new, future emails

**Features:** The attributes used to make the ham / spam decision

- Words: FREE!
- Text Patterns: $dd, CAPS
- Non-text: SenderInContacts
- ...

---

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential 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, I know 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.
Bayes net model for ham/spam

- Class $C$ of a document is spam or ham, with prior $P(C)$
- **Bag-of-words** model: Each word $W_i$ in the document is generated independently from a class-specific distribution $P(W_i | C)$ over words
- This is an example of a *naïve Bayes* model

$$P(C, W_1, ..., W_n) = P(C) \prod_i P(W_i | C)$$
A Naïve Bayes model is a polytree, so solvable in linear time.

To compute posterior distribution for class C given a document:

\[
P(C \mid w_1, \ldots, w_n) = \alpha \ P(C, w_1, \ldots, w_n) = \alpha \ P(C) \ \prod_i P(W_i \mid C)
\]

I.e., multiply \( n+1 \) numbers, for each value of \( C \), then normalize.
Computing the class probabilities

\[
\alpha[e^{-76.0},e^{-80.5}] = [0.989,0.011]
\]

| Word | P(w|spam) | P(w|ham) | Cum LogSpam | Cum LogHam |
|------|-----------|----------|-------------|------------|
| Word | 0.33333   | 0.66666  | -1.1        | -0.4       |
We need to estimate the following parameters:

- \( P(C) = [\theta_C, 1-\theta_C] \), the prior over classes
  - ML estimate: relative frequencies in training set
- \( P(W_i | C) \), the distribution for each word position given the class
  - For the bag-of-words model, this is the same for all positions
  - Parameters are \( \theta_{k|c} = P(W_i=k | C=c) \) for each class \( c \) and each dictionary entry \( k \)
  - E.g., \( \theta_{"you"|\text{spam}} = 0.00881 \quad \theta_{"you"|\text{ham}} = 0.00304 \)
  - Estimated by measuring frequency of occurrence in ham and spam
  - Need Laplace smoothing! Many dictionary words may not appear in training set
Recap: Maximum Likelihood Estimation

- Learning = Bayesian updating of a probability distribution over \( H \)
- Prior \( P(H) \), training data \( X \)

- Maximum likelihood estimation

\[
\theta_{ML} = \arg \max_{\theta} P(X|\theta) = \arg \max_{\theta} \prod_{i} P_{\theta}(X_{i})
\]

- Maximum conditional likelihood estimation

\[
\theta^* = \arg \max_{\theta} P(Y|X, \theta) = \arg \max_{\theta} \prod_{i} P_{\theta}(y_{i}|x_{i})
\]

- How to solve for \( \theta_{ML} \)?
Recap: Naïve Bayes

- **Naïve Bayes model:**
  - Attributes conditionally independent of each other, given the class
  - Assuming there are 2 classes, n Boolean attributes $X_i$, how many parameters are in this Naïve Bayes model?
    \[
    \theta = P(C = true), \theta_{i1} = P(X_i = true | C = true), \theta_{i2} = P(X_i = true | C = false)
    \]
  - Maximum likelihood parameter estimation
  - Inference: what’s the class given attribute values $(x_1, ..., x_n)$?
    \[
    P(C | x_1, ..., x_n) = \alpha P(C) \prod_i P(x_i | C)
    \]
Bayesian learning

- Learning = Bayesian updating of a probability distribution over $H$
- Prior $P(H)$, training data $X=x_1,...,x_N$
- Given the data so far, each hypothesis has a posterior probability:
  - $P(h_k|X) = \alpha P(X|h_k)P(h_k) = \alpha \times$ Likelihood $\times$ Prior
- Predictions use a likelihood-weighted average over the hypotheses:
  - $P(x_{N+1}|X) = \Sigma_k P(x_{N+1}|X,h_k)P(h_k|X) = \Sigma_k P(x_{N+1}|h_k)P(h_k|X)$
- No need to pick one best-guess hypothesis!
  - Drawback: $\Sigma_k$ may be expensive/impossible for large/infinite $H$
Example: Surprise Candy Co.

- Suppose there are five kinds of bags of candies, no labels!!
  - 10% are h1: 100% cherry candies
  - 20% are h2: 75% cherry candies + 25% lime candies
  - 40% are h3: 50% cherry candies + 50% lime candies
  - 20% are h4: 25% cherry candies + 75% lime candies
  - 10% are h5: 100% lime candies

- Then we observe candies drawn from some bag: 🍋🍋🍋🍋🍋
- What kind of bag is it?
- What flavour will the next candy be?
Posterior probability of hypotheses

- Prior over $h_1, ..., h_5$: $<0.1, 0.2, 0.4, 0.2, 0.1>$
- $P(h_k | X) = \alpha P(X | h_k) P(h_k)$
  - $P(h_1 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h_1) P(h_1)$ = _________________
  - $P(h_2 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h_2) P(h_2)$ = _________________
  - $P(h_3 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h_3) P(h_3)$ = _________________
  - $P(h_4 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h_4) P(h_4)$ = _________________
  - $P(h_5 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h_5) P(h_5)$ = _________________
- $\alpha$ = _________________

$h_1$: 100% cherry,
$h_2$: 75% cherry + 25% lime,
$h_3$: 50% cherry + 50% lime,
$h_4$: 25% cherry + 75% lime,
$h_5$: 100% lime.
Posterior probability of hypotheses

\[ P(h_k | X) = \alpha P(X | h_k)P(h_k) \]

- \( P(h1 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h1) P(h1) = \alpha \cdot 0.0 \cdot 0.1 = 0 \)
- \( P(h2 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h2) P(h2) = \alpha \cdot 0.25 \cdot 0.2 = 0.000195 \alpha \)
- \( P(h3 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h3) P(h3) = \alpha \cdot 0.5 \cdot 0.4 = 0.0125 \alpha \)
- \( P(h4 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h4) P(h4) = \alpha \cdot 0.75 \cdot 0.2 = 0.0475 \alpha \)
- \( P(h5 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h5) P(h5) = \alpha \cdot 1.0 \cdot 0.1 = 0.1 \alpha \)

\( \alpha = \frac{1}{(0 + 0.000195 + 0.0125 + 0.0475 + 0.1)} = 6.2424 \)

- \( P(h1 | 5 \text{ limes}) = 0 \)
- \( P(h2 | 5 \text{ limes}) = 0.00122 \)
- \( P(h3 | 5 \text{ limes}) = 0.07803 \)
- \( P(h4 | 5 \text{ limes}) = 0.29650 \)
- \( P(h5 | 5 \text{ limes}) = 0.62424 \)
Posterior probability of hypotheses

\[ P(h_1 \mid d) \]
\[ P(h_2 \mid d) \]
\[ P(h_3 \mid d) \]
\[ P(h_4 \mid d) \]
\[ P(h_5 \mid d) \]
Prediction probability

\[ P(x_{N+1} | X) = \sum_k P(x_{N+1} | h_k)P(h_k | X) \]

\[ P(\text{lime on 6 | 5 limes}) \]
  \[= P(\text{lime on 6 | h1})P(h1 | 5 \text{ limes}) \]
  \[+ P(\text{lime on 6 | h2})P(h2 | 5 \text{ limes}) \]
  \[+ P(\text{lime on 6 | h3})P(h3 | 5 \text{ limes}) \]
  \[+ P(\text{lime on 6 | h4})P(h4 | 5 \text{ limes}) \]
  \[+ P(\text{lime on 6 | h5})P(h5 | 5 \text{ limes}) \]
  \[= 0 \times 0 + 0.25 \times 0.00122 + 0.5 \times 0.07830 + 0.75 \times 0.29650 + 1.0 \times 0.62424 \]
  \[= 0.88607 \]
Prediction probability

\[ P(\text{next candy is lime} \mid d) \]

Number of samples in \( d \)