Overview

- Probability density estimation
- Instance-based techniques
  - histogram
  - kernel-based methods
  - nearest neighbor
- Midterm

Probability density estimation

Continuous: probability density function

\[ P(x) \geq 0, \forall x \]
\[ \int_{-\infty}^{\infty} P(x) \, dx = 1 \]

Discrete: probability mass function

\[ P(x) \geq 0, \forall x \]
\[ \sum_{x \in X} P(x) = 1 \]

Probability density estimation

Uses of probabilistic models

1. Classification
2. Classification with missing attributes
3. Learning with missing data
4. Clustering
5. Prediction
6. Diagnosis
7. Temporal reasoning
8. Making decisions

1. Classification

- Plant-eating robot
- Three classes of plant
  - nutritious, poisonous, empty calories
- Four attributes
  - rough bark, smooth bark, no bark
  - thorns or no thorns
  - leaf shape: broad or narrow
  - leaf arrangement: alternate, opposite, whorled
1. Classification

- We have a joint probability function:
  \[ P(C, X_1, X_2, X_3, X_4) \]
- New instance: smooth bark, thorns, narrow leaves with alternate leaf arrangement
- What should we do?

\[
\begin{align*}
P(C \mid X_1, X_2, X_3, X_4) &= \frac{P(C, X_1, X_2, X_3, X_4)}{P(X_1, X_2, X_3, X_4)} \\
&= \frac{P(C, X_1, X_2, X_1, X_4)}{\sum_c P(c, X_1, X_2, X_3, X_4)} \\
&= \frac{P(C, X_1, X_2, X_3, X_4)}{\sum_c P(c, X_1, X_2, X_3, X_4)} \\
&\propto P(C, X_1, X_2, X_3, X_4)
\end{align*}
\]

Choose C that maximizes this

2. Classification with missing attributes

- Plant-eating robot
- New instance has:
  - indeterminate bark (image processing fails)
  - thorns
  - broad leaves
  - whorled arrangement
- Now what do we do?

\[
\begin{align*}
P(C, X_2, X_3, X_4) &= \sum_{X_1} P(C, X_1, X_2, X_3, X_4) \\
P(C \mid X_2, X_3, X_4) &= \frac{P(C, X_2, X_3, X_4)}{\sum_c P(c, X_2, X_3, X_4)} \\
&= \frac{\sum_{X_1} P(C, X_1, X_2, X_3, X_4)}{\sum_c \sum_{X_1} P(c, X_1, X_2, X_3, X_4)} \\
&\propto \sum_{X_1} P(C, X_1, X_2, X_3, X_4)
\end{align*}
\]
2. Classification with missing attributes

- Known variables: \(X_1, \ldots, X_k\)
- Unknown variables: \(X_{k+1}, \ldots, X_n\)

\[
P(C | \text{known}) \propto \sum_{\text{unknown}} P(C, \text{known, unknown})
\]

- Sum for each unknown variable, \(X_{k+1}, \ldots, X_n\)

\[
P(C | X_1, \ldots, X_k) = \sum_{X_{k+1}} \ldots \sum_{X_n} P(C | X_1, \ldots, X_k, X_{k+1}, X_n)
\]

- Number of terms exponential in number of missing attributes
- Bayesian networks will make this tractable

3. Learning with missing data

- So far, we’ve assumed complete training data
- Training data may be incomplete
  - missing labels
  - missing attributes
- Expectation-maximization algorithm allows us to learn with missing data

4. Clustering

- Attribute values given, but no class labels
- Goal: group data into clusters of similar instances

- Algorithms:
  - hierarchical agglomerative clustering
  - k-means
  - autoclass

5. Prediction

- Reasoning from causes to effects
  - e.g. given diseases, what is prognosis
- Natural to encode \(P(\text{effects} | \text{cause})\)
- Bayesian networks will allow us to do this reasoning

6. Diagnosis

- Reasoning from effects to causes
  - e.g. given symptoms, what are diseases?
- Use Bayes rule
  \[
P(\text{causes} | \text{effects}) \propto P(\text{effects} | \text{causes})P(\text{causes})
\]
- Bayesian networks will allow us to do this reasoning

7. Temporal reasoning

- Systems that change over time
- Systems that produce sequences of data
- Transition and observation models
  - like POMDPs
- Hidden Markov models are a framework for reasoning about temporal processes
8. Making decisions

- Probability is a component of expected utility
  - probabilistic model of action results
  - observation model
- Maximizing expected utility to make decisions incorporates probabilities

Probabilistic techniques

- Unifying framework for many tasks
- Basic to modern AI

Two probabilistic approaches

- Discriminative approach:
  learn $P(C|X_1,\ldots,X_n)$
  - can be represented many ways, e.g. neural net
- Generative approach:
  learn $P(C)$ and $P(X_1,\ldots,X_n|C)$
  - use Bayes rule to derive $P(C|X_1,\ldots,X_n)$
  - use independence assumptions to make representing probabilities easier

Advantages of each

- Discriminative
  - don’t need to estimate $P(X_1,\ldots,X_n|C)$, which can be difficult
- Generative
  - natural to think about class generating attributes
    - disease generates symptoms
    - digit generates pixel image
  - works well with missing data
- We will focus on generative in this class

Overview

- Probability density estimation
- Instance-based techniques
  - histogram
  - kernel-based methods
  - nearest neighbor

Instance-based techniques

- Store all the instances you’ve seen
- Regression: find similar instances, and combine their outputs to make prediction
- Classification: find similar instances and classify the way they do
- Probability density estimation: density estimate is frequency of similar instances
Terminology

- "Instance-based methods": models based on training instances previously seen
- "Non-parametric methods": make no assumptions about particular parameterized form of models

Histograms

Consider n-dimensional bounded attribute space

Probability of a bin

What’s the probability that a new point will fall in a particular bin?

\[
P(\text{top left}) = \frac{N_{\text{top left}}}{N} = \frac{5}{9} = 0.56
\]
Probability of a bin

\[ P(\text{bottom left}) = \frac{N_{\text{bottom left}}}{N} = \frac{2}{9} = 0.22 \]

Probability density

We need: \( \int p(x) \, dx = 1 \)

\[ p(x) = \frac{p(\text{bin}_x)}{\text{Width}(\text{bin}_x)} \]

Density estimate

\[ p(x) = \frac{p(\text{bin}_x)}{V(\text{bin}_x)} = \frac{N_x}{NV(\text{bin}_x)} \]

Density estimate

Assume uniform density: \( V(\text{bin}_x) = \frac{1}{4} \)
Density estimate

\[ p(x) = \frac{N_x}{N V(bin_x)} = \frac{5}{9 \times 0.25} = 2.24 \]

Classification

Choose most common class in bin.

Justification

- Learn a separate histogram for each class
- This gives class-conditional probability density
  - \( N_{cx} = \) number of instances in \( bin_x \) with class \( c \)
  - \( N_c = \) number of instances with class \( c \)
  - \( P(x | c) = \frac{N_{cx}}{N_c V(bin_x)} \)
- Obtain class prior \( P(c) = \frac{N_c}{N} \)

Producing the classifier

\[ h(x) = \arg \max_c P(c | x) \]
\[ = \arg \max_c \frac{p(x | c)P(c)}{p(x)} \]
\[ = \arg \max_c \frac{p(x | c)P(c)}{p(x)} \]
\[ = \arg \max_c \frac{N_{cx}}{N_c V(bin_x)} \]
\[ = \arg \max_c N_{cx} \]

Inductive bias of histograms

- What is inductive bias of histograms?
  - locality: points close to each other have similar class, density
  - close to each other: in same bin

Histogram bin size

- small bins
  - can fit fine details of distribution
  - jagged estimator
    - abrupt changes at bin boundaries
  - subject to noise
    - small amount of data in each bin
    - might overfit

- large bins
  - cannot model details of distribution
  - groups distant points together
  - smooth estimator
  - less subject to noise
  - more points in each bin
  - might underfit
Extreme case: small bins
- Tiny bins, each containing at most one point
- Very jagged estimator
- Completely unable to generalize

Extreme case: large bins
- Only one bin
- All points have same density
- Does not model any pattern in data

Bin size is “smoothing parameter”

Increasing the dimension
- Attribute space has \( n \) dimensions
- Each dimension divided into \( d \) bins
- Number of bins: \( d^n \)
- What goes wrong?
  - almost all bins will be empty
  - useless for generalization

VC dimension
- What is VC dimension of histogram with \( d^n \) bins?
- Can shatter \( d^n \) points by having one per bin
- Cannot shatter \( d^n+1 \) points
  - at last two will share a bin
- So VC dimension is \( d^n \)
  - lower bound on sample complexity linear in \( \text{VC}(H) \)
  - so number of samples required exponential in \( n \)

Bottom line
- Amount of training data required is exponential in dimensionality of input space
- Curse of dimensionality
- Generally a problem for instance-based methods
Advantages of histograms

- Very simple
- Very fast classification
- Very fast learning
- Easy to understand and depict graphically
- Easy online updates (increment bin count)
- No prior assumptions about form of model

Disadvantages of histograms

- Curse of dimensionality
- Need to choose bin size
- Discontinuous density function
- Fixed bin size over input space
  - some bins may have many points: underfit
  - other may have few points: subject to noise

Overview

- Probability density estimation
- Instance-based techniques
  - histogram
  - kernel-based methods
  - nearest neighbor

Kernel-based methods

- Idea: have each point create its own bin
- Use \( p(x) = \frac{N_x}{NV(bin_x)} \)
- Estimate \( N_x \) by kernel function \( H(x) \)
- So \( p(x) = \frac{H(x)}{NV(bin_x)} \)
- Different meaning of kernel than in SVMs!

Example: hyper-cube kernel

- Hyper-cube with length \( h \)
- Max norm: \( \|x - z\|_\infty \)
  - maximum distance in any dimension
- \( H(x) = \left| \left\{ x^i : \|x^i - x\|_\infty \leq \frac{h}{2} \right\} \right| \)
  - the number of points inside hypercube centered at \( x \)

Example: hyper-cube

- \( h = \frac{1}{2} \)
- \( H(x) = 3 \)
- \( V(bin_x) = \frac{1}{4} \)
- \( N = 9 \)

- \( p(x) = \frac{N_x}{NV(bin_x)} = \frac{3}{9(0.25)} = 1.33 \)
Example: hyper-cube

- \( H(x) = \left\{ x^i : \|x^i - x\|_2 \leq \frac{h}{2} \right\} \)

- Continuous or discontinuous?
  - discontinuous

Soft bins

- Points are partially in bin, depending on distance from center
- All training instances used to compute density
- Instances weighted by distance from point

- Gaussian kernel: 
  \[
  H(x) = \sum_{i=1}^{N} \exp\left( -\frac{\|x - x^i\|^2}{2h^2} \right)
  \]

Gaussian kernel

- \( H(x) = w_1 + w_2 \)

Gaussian kernel density estimate

- For soft kernel, volume is integral of kernel function over the space
- For Gaussian kernel: 
  \[
  V = \frac{1}{(2\pi h^2)^d}
  \]

- So density estimate:
  \[
  p(x) = \frac{1}{N(2\pi h^2)^d} \sum_{i=1}^{N} \exp\left( -\frac{\|x - x^i\|^2}{2h^2} \right)
  \]

Bimodal density function

Gaussian kernel estimator does not assume that data normally distributed.
Kernel-based classification

- Class-specific kernel $H_c(x)$
  - counts only points with class $c$
  - hypercube: $H_c(x) = \left\{ x' : \| x' - x \|_2 \leq \frac{h}{2}, c' = c \right\}$
  - Gaussian: $H_c(x) = \sum_{i=1}^{N} \exp \left( -\frac{\| x - x_i \|^2}{2h^2} \right) \delta_{c,c}$
- Given a point $x$, choose class that maximizes $H_c(x)$

Hypercube kernel classification

Gaussian kernel classification

Weighted sum of positive points is greater than negative points

Smoothing parameter

- Parameter $h$ is a smoothing parameter
  - hypercube: size of bin
  - Gaussian: variance of Gaussian bump
- As with number of histogram bins
  - $h$ too large, may underfit
  - $h$ too small, may overfit

Effect of $h$

Curse of dimensionality: hard bins

- As dimension increase, volume of bin is $h^n$
- In high dimensions, almost all bins are empty
- Impossible to generalize
**Curse of dimensionality: soft bins**

- All training points are in bins
- In high dimensions, all points are far away from point we want to classify
  - have low weight
  - not good guide for classifying
- In high dimensional space, randomly generated points are approximately equidistant
  - almost all points classified the same way

**Kernel-based advantages**

- Very fast learning (simply store instances)
- Easy online updates (store new instance)
- No prior assumptions about form of model
- With enough data, we can fit any distribution
- Complexity of estimator grows with amount of data
- For soft kernels, continuous density estimator

**Kernel-based disadvantages**

- Curse of dimensionality
- Need to choose $h$
- Expensive space needs (need to store all instances in memory)
- Slow density estimation or classification
  - hard bins: find all points in kernel
  - soft bins: need to sum over all points
- Fixed bin size

**Overview**

- Probability density estimation
- Instance-based techniques
  - histogram
  - kernel-based methods
    - nearest neighbor

**k-Nearest neighbors**

- Idea: fixed number of neighbors determines bin size
- Use: $p(x) = \frac{N_k}{NV(bin_x)}$
- $N_k$ is fixed ($k$)
- Compute how large volume must be to enclose $k$ neighbors

**Nearest neighbor “density estimator”**

\[
p(x) = \frac{k}{N\pi r^2} = \frac{3}{8(\pi(0.2^2))} = 3.0
\]

Grow sphere until it includes $k$ points
$r = 0.2$
$V_x = \pi r^2$

Not really probability density: doesn’t integrate to 1
Nearest-neighbor classification

To classify, take majority vote of k nearest neighbors.

Smoothing parameter

- Parameter k is smoothing parameter
  - like h for histogram and kernel methods
- Small k
  - jagged estimator, susceptible to noise, may overfit
- Large k
  - smooth estimator, robust to noise, may underfit

Nearest neighbor advantages

- Very simple and intuitive
- Very fast learning and online updates
- No prior assumptions about form of model
- With enough data, can fit any hypothesis
- Complexity of classifier grows with amount of data
- Bin sizes adjust to density of points

Nearest neighbor disadvantages

- Curse of dimensionality
  - with large spaces, neighbors are all far away
- Need to choose k
- Expensive space needs (need to store all instances in memory)
- Slow classification (need to find k neighbors)
  - but clever data structures can help
- Not a true density estimator

Instance-based summary

- Histogram
  - low memory needs
  - discontinuous density estimator
- Kernel-based
  - with smooth bins, continuous
- Nearest neighbor
  - variable bin sizes

Instance-based summary

- Pros
  - simple to implement
  - complexity of model grows with amount of data
  - no assumptions about form of model
- Cons
  - smoothing parameter
  - curse of dimensionality
Curse of dimensionality

- Applies in high dimensions
- Sometimes effective dimensionality is lower
  - points lie on lower-dimensional surface

Dimensionality reduction techniques