Some figures from Wikipedia/Google image search
Administrivia

• Reminder: HW comments and minor corrections on Slack
• HW3 coming soon
• *Anonymous* survey coming re: speed of course, etc.
Today

• NN for machine learning problems
• Density estimation to classify cats versus dogs
• Continuous versus discrete random variables
NN for machine learning

• Nearest neighbors is a fundamental ML technique
  – Used for classification, regression, etc.
• We will study (and implement!) NN algorithms
  – Exact algorithms on Tuesday next week
  – Approximate algorithms starting Thursday
• Today we will focus on understanding the use of NN
Classification and NN algorithms

• Suppose the height/weight of your query animal is very similar to the cats you have seen, and unlike the dogs you have seen, then it’s probably a cat

• There’s actually a lot going on in the sentence above:
  – “very similar”
  – “the cats you have seen”
  – “probably”

• You can classify directly from NN
NN and k-NN classification

• Find the animal you’ve seen that’s most similar to your query
  – NN classification

• What can go wrong?

• More robustly, look at the k most similar animals and take the mode (most common label)
  – k-NN classification
  – Choice of k?
k-NN classifier small example

Slide credit: https://www.cs.rit.edu/~rlaz/PatternRecognition/, Richard Zanibbi
(k-)NN classifier example
Classification and NN algorithms

• For many applications it’s way more useful to have the density
  – Tells you a lot more about your data
• To classify we need to estimate the density
• Good way to do this is from nearest neighbors
  – Lots of other algorithms also but NN is very popular
• Basic intuition: most of the cats are where the cat density is high, and vice-versa
  – “When you hear hoofbeats, think of horses not zebras”
Cats versus dogs (simple version)

• Suppose we want to classify based on a single number
  – Such as weight
• Simple case: cats and dogs have their weights described by a Gaussian (normal) distribution
• Cats occur about as frequently as dogs in our data
• We just need to estimate the density for cats vs dogs
• This allows us to build our classifier
Cat vs dog classification from weight

“Simple” case

A not so simple case
NN Density estimation

• Lots of practical questions boil down to density estimation
  – Even if you don’t explicitly say you’re doing it!
    • “How much do typical cats weigh?”
      – Google says: 7.9 – 9.9 lbs

• Your classes generally have some density in feature space
  – Hopefully they are compact and well-separated

• Given a new data point, which class does it belong to?
  – We just maximize $P(data|class)$, called the likelihood
    • Formalizes what we did on the previous slide
What is a density?

- Consider an arbitrary function \( p \) where

\[
p(x) \geq 0
\]

- Can view it as a *probability density function*
  - The PDF for a real-valued random variable
  - If we weighed \( \infty \) cats, what frequency of weights would we get?

\[
p(x)
\]
A pitfall of interpreting densities

• The value of PDF at \( x \) is **not** the probability we would observe the weight \( x \)
  – Which is always zero (think about it!)
  – Instead, it gives the probability of getting a weight in a given interval

\[
\int_{\alpha}^{\beta} p(x) \, dx = P[\alpha \leq \text{weight} \leq \beta]
\]
Discrete case is easier

- Sometimes the values of the random variable are discrete
  - Instead of a PDF you have a probability mass function (PMF)
  - I.e., a histogram whose entries sum to 1
    - No bucket has a value greater than 1
- This is the true relative frequencies
  - I.e., what we would get in the limit as we weigh more and more cats
Sampling from a PDF

• Suppose we weigh a bunch of cats
  – This generates our sample (data set)
  – How does this relate to the true PDF?

• It simplifies life considerably to assume:
  – All cats have their weights from the same PDF (identical distributions)
  – No effect between weighing one cat and another (independence)
Welford’s online mean algorithm

• Suppose we know that cats come from a Gaussian distribution but we have too many cats to store all their weights
• Can we estimate the mean (and variance) online?
• Online algorithms are very important for modern applications
• For the mean we have

\[ \mu_n = \frac{1}{n} \sum_{i=1}^{n} x_i = \mu_{n-1} + \frac{1}{n}(x_n - \mu_{n-1}) \]
Non-parametric approach

• We knew the underlying distribution
  – All we needed was to estimate the parameters
  – Obviously, this gives bad results when the true distribution isn’t what we think it is
  – Non-Gaussian distributions are in general rare, and hard to handle
    • But they occur a lot in some areas
• Box’s law: All models are wrong but some are useful
Is there a free lunch?

- Suppose we simply plot the data points
  - Assume 1-D for the moment (cat weights)

- How to compute density from data?
Histogram representation

Histogram 10 bins

Histogram 20 bins

Histogram 90 bins
Bin-size tradeoffs

• Fewer, larger bins give a smoother but less accurate answer
  – Tend to avoid “gaps”
    • i.e., places where the density is declared to be 0
• More, smaller bins have opposite property
• If you don’t know anything in advance, there’s no way to predict bin size
  – Also, note that this method isn’t a great idea in high dimensions
Histogram-based estimates

• You can use a variety of fitting techniques to produce a curve from a histogram
  – Lines, polynomials, splines, etc.
  – Also called regression/function approximation
  – Normalize to make this a density

• If you know quite a bit about the underlying density you can compute a good bin size
  – But that’s rarely realistic
  – And defeats the whole purpose of the non-parametric approach!
Nearest-neighbor estimate

- To estimate the density, count number of nearby data points
  - Like histogramming with sliding bins
  - Avoid bin-placement artifacts

\[ \hat{p}(x) = \frac{\# \{ x_i \mid \| x_i - x \| \leq \epsilon \} }{n} \]

- Can fix epsilon and compute this quantity, or we can fix the quantity and compute epsilon
NN density estimation

Slide credit: https://www.cs.rit.edu/~rlaz/PatternRecognition/, Richard Zanibbi
Suppose we want to “smooth” a histogram, i.e. replace the values by the average over a window.

- How can we do this efficiently?