6.891 Machine learning and neural networks

Tommi Jaakkola
MIT AI LAB
tommi@ai.mit.edu

LECTURE 12 (October 19, 2000)

Mid-term is next Thursday

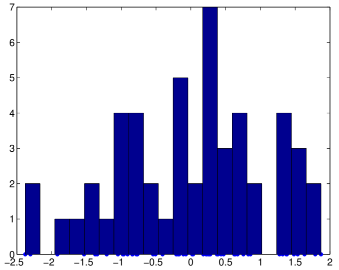
Topics

- Non-parametric density estimation
  - Histograms
  - Naive estimator
  - Parzen windows (1-dimensional case)
  - Nearest neighbor kernel width
  - Parzen windows (multi-dimensional case)
- Clustering
  - Mixture models, K-means
  - Hierarchical agglomerative clustering
  - Self-organizing maps
  - semi-supervised clustering (information bottleneck)

Histograms

- One dimensional example: n training points $x_1, \ldots, x_n$

  The real line divided into non-overlapping bins $[\mu_j - h, \mu_j + h)$ where $\mu_j = jh$ (with $j = \ldots -1, 0, 1, \ldots$) is the center of the j^{th} bin



The resulting density estimate is

$$\hat{P}_n(x) = \frac{1}{n} \times \frac{\#\text{ of } x_i \text{ in the same bin as } x}{\text{width of bin containing } x} \quad (1)$$

Naive estimator

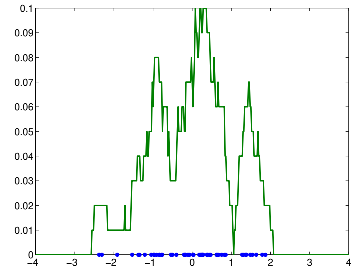
- Define a window function $w(x)$:

$$w(x) = \frac{1}{2}, \quad |x| < 1, \quad \text{and zero otherwise} \quad (2)$$

We can then construct the following (naive) density estimate

$$\hat{P}_n(x; h) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} w\left(\frac{x - x_i}{h}\right) \quad (3)$$

- Example: $n = 50$, $h = 0.02$



Naive estimator cont'd

- The naive estimator actually converges to the true density provided that both $n \to \infty$ and $h \to 0$ (appropriately).

  For a fixed x

$$\begin{aligned} \hat{P}_n(x;k) &= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h} w\left(\frac{x-x_i}{h}\right) \\ &\to \frac{1}{2h} P(x-h < X < x+h) \quad \text{as } n \to \infty \\ &\to P(x) \quad \text{as } h \to 0 \end{aligned} \tag{4}$$

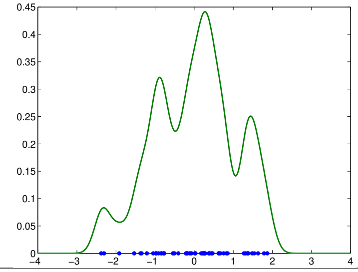
Parzen windows

- Instead of the naive window function, we can put a Gaussian (or other) bump on each training example

$$\hat{P}_n(x;h) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h} K\left(\frac{x-x_i}{h}\right) \tag{5}$$

  where $K(z) = \exp(-z^2/2)/\sqrt{2\pi}$ (this is also known as a kernel function; very different from SVM kernels).

- We get a much smoother estimate
- Example: $n = 50$, $h = 0.02$

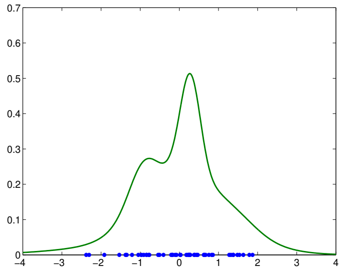


Parzen windows: variable kernel width

- We can set the kernel width locally
- k-nearest neighbor choice: let d_{ik} be the distance from x_i to its k^{th} nearest neighbor

$$\hat{P}_n(x;k) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{d_{ik}} K\left(\frac{x-x_i}{d_{ik}}\right) \tag{6}$$

- The estimate is smoother where there are only few data points
- Example: $n = 50$, $k = 10$



Parzen windows: optimal kernel width

- How do we choose the kernel width (smoothing parameter) h?

  A general solution: likelihood cross-validation

  Let $\hat{P}_{n-i}(x;h)$ be a parzen windows density estimate constructed on the basis of $n-1$ training examples leaving out x_i.

  We select h that maximizes the log-likelihood of the left-out examples

$$CV(h) = \sum_{i=1}^{n} \log \hat{P}_{n-i}(x;h) \tag{7}$$

- We can also use this to find the appropriate k in the variable (nearest neighbor) kernel width approach

Parzen windows: L_1 convergence

- We can measure the error in our density estimate relative to the L_1 norm:

$$err(x_1,\ldots,x_n) = \int_{-\infty}^{\infty} |P(x) - \hat{P}(x;h)|\, dx \tag{8}$$

(the error depends on the training examples)

- We would like to know how quickly this error converges to its expected value as a function of the number of training examples n for a fixed h

Theorem

$$P\left(\,|err(x_1,\ldots,x_n) - E\{\, err(x_1,\ldots,x_n)\,\}| > \epsilon\right) \leq 2\, e^{-n\epsilon^2/2} \tag{9}$$

- Note: this only gives us an idea how quickly we can get a good estimate of the error in the density estimate

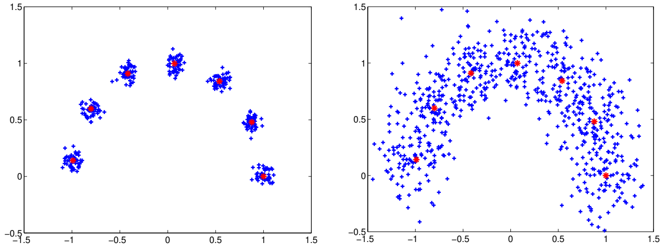
Parzen windows: multi-dimensional case

- Multi-dimensional Parzen windows estimate:

$$\hat{P}_{parzen}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} P(\mathbf{x}|\mathbf{x}_i, \sigma^2 I) \tag{10}$$

where n is the number of examples.

- The covariance matrices for the bumps $\Sigma_i = \sigma^2 \cdot I$ are all equal and spherical. The single parameter σ^2 now controls the smoothness of the density estimate



Clustering

- Clustering is *unsupervised* learning
- Unlike in classification, there's no "ground truth"...

- We have already seen (flat) clustering methods
  1. Mixture density models
  2. K-means clustering (cf. vector quantization)
- Other clustering methods: hierarchical agglomerative clustering, self-organizing maps, semi-supervised clustering

Hierarchical (bottom-up) clustering

- Given examples $\mathbf{x}_1,\ldots,\mathbf{x}_n$, we need
  1. A pairwise distance between the examples such as

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| \tag{11}$$

  2. A *linkage* method: we have to be able to measure distances between clusters of examples C_k and C_l

     a) Single linkage:

$$d_{kl} = \min_{i\in C_k, j\in C_l} d(\mathbf{x}_i, \mathbf{x}_j) \tag{12}$$

     b) Average linkage:

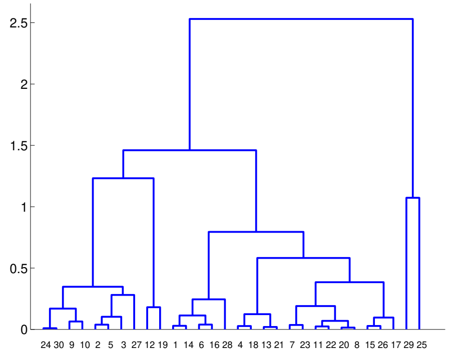
$$d_{kl} = \frac{1}{|C_l|\,|C_k|}\sum_{i\in C_k, j\in C_l} d(\mathbf{x}_i, \mathbf{x}_j) \tag{13}$$

     c) Centroid linkage:

$$d_{kl} = d(\bar{\mathbf{x}}_k, \bar{\mathbf{x}}_l),\quad \bar{\mathbf{x}}_l = \frac{1}{|C_l|}\sum_{i\in C_l}\mathbf{x}_i \tag{14}$$

Hierarchical (bottom-up) clustering

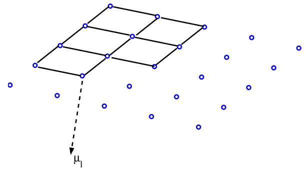
- A dendrogram representation of hierarchical clustering



(the linear ordering of examples is chosen for clarity of representation)

Self-organizing maps

- We want to cluster the data while preserving certain predefined topographic (neighborhood) relations among the clusters
- First we have to specify the desired cluster topology or grid



- Each grid point l has a cluster centroid μ_l associated with it (initially chosen at random)

  We have to update the cluster centroids somehow while preserving the topographic organization of the data

Self-organizing maps cont'd

- Competitive learning: the grid points are competing for examples

  For each training example $\mathbf{x}_i$

  1. find the cluster centroid μ_l closest to $\mathbf{x}_i$

  $$\mu_{l^*} = \arg\min_{l \in grid} d(\mathbf{x}_i, \mu_l) \tag{15}$$

  2. move the centroid *as well as nearby centroids in the grid* towards the training point

  $$\mu_l \leftarrow \mu_l + \epsilon\, \Lambda(l^*, l)\, (\mathbf{x}_i - \mu_l) \tag{16}$$

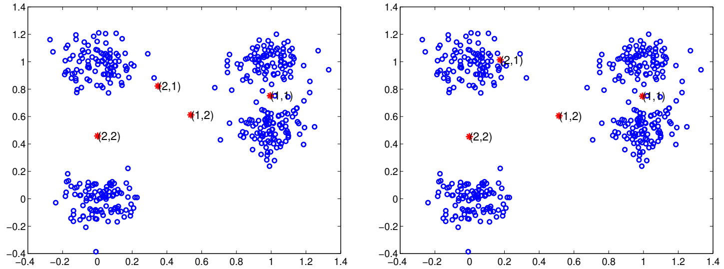
  where $\Lambda(l^*, l)$ is a neighborhood function (decreases with increasing distance in the original grid), e.g.,

  $$\Lambda(l^*, l) = \exp(-\|r_{l^*} - r_l\|^2/2) \tag{17}$$

  where r_{l^*} and r_l are the corresponding grid points.

Self-organizing maps: example

- 2x2 grid, centroids initialized to randomly selected points



Semi-supervised clustering

- Let's assume we have some additional *relevance* information about the examples

$\mathbf{x}_i$	Training example (e.g., a text document)
y	Relevance variable (e.g., a word)
$P(y\|\mathbf{x}_i)$	Relevance information per document (e.g., word distribution)

  where $i = 1, \ldots, n$.
- Suppose we wish to cluster the examples into larger groups without loosing information about word frequency
- Documents with similar word frequencies can be put into the same cluster

Semi-supervised clustering cont'd

We have to cluster $\{\mathbf{x}_i\}$ on the basis of $\{P(y|\mathbf{x}_i)\}$, the predictive distributions

- For any cluster C the predictive probability is based on randomly picking a document in the cluster

$$\hat{P}(y = j|C) = \frac{1}{|C|} \sum_{i \in C} P(y = j|\mathbf{x}_i) \quad (18)$$

$$\hat{P}(C) = \frac{|C|}{n} \quad (19)$$

- We can construct a new distance measure between the clusters

$$d(C_l, C_k) = \frac{|C_l| + |C_k|}{n} \cdot I(\, y\, ; \text{cluster identity}\,) \quad (20)$$

This mutual information distance measures how much we loose information about y if we don't know which cluster a random document comes from.

- We can make finer distinctions across larger clusters

Semi-supervised clustering cont'd

- The mutual information is given by

$$I(\, y\, ; \text{cluster identity}\,) = \hat{P}(C_l) \sum_{j=1}^{m} \hat{P}(y = j|C_l) \log \frac{\hat{P}(y = j|C_l)}{\hat{P}(y = j|C_l \cup C_k)} + \hat{P}(C_k) \sum_{j=1}^{m} \hat{P}(y = j|C_k) \log \frac{\hat{P}(y = j|C_k)}{\hat{P}(y = j|C_l \cup C_k)}$$

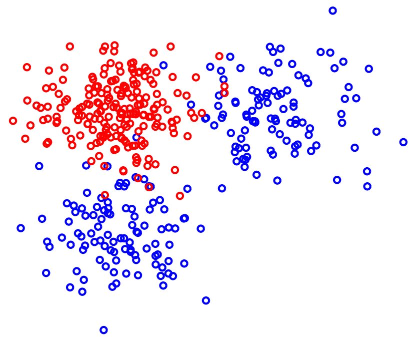
where

$$\hat{P}(y = j|C) = \frac{1}{|C|} \sum_{i \in C} P(y = j|\mathbf{x}_i)$$

$$\hat{P}(C) = \frac{|C|}{n}$$

Semi-supervised clustering: example

- Suppose we have a set of labeled examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$



- We can take the label as the relevance variable.

$$P(y|\mathbf{x}_i) = 1, \;\; \text{if } y = y_i \text{ and zero otherwise} \quad (21)$$

- What type of clustering do we get?