Visual feature learning

Winter 2013

Roland Memisevic

Lecture 9, February 19, 2013

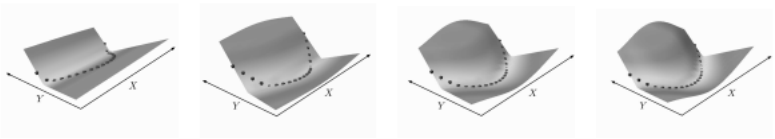
Last time

- Limitations of ICA
- Overcomplete codes
- Energy based models

Energy based models

- We may eliminate the partition function altogether and define the model as an “energy landscape” that we form through learning.
- This gives us even more freedom in devising schemes that push or pull on the energy landscape.
- (LeCun, et al. 2006)

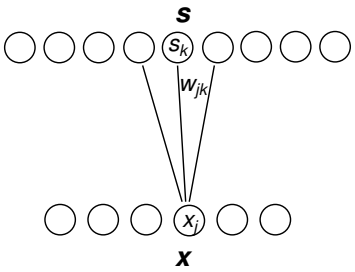
Energy based models



- It is common to define energies as $-q(\boldsymbol{x}; W)$, in which case we want to *minimize* energy near the data.
- Energy based models can be used in a variety of tasks, but for feature learning, they practically always involve hidden variables which can be visualized conveniently as a bi-partite graph.
- Pushing up the energy away from the data typically translates into a *capacity constraint* on the hiddens.

Feature learning and bi-partite networks

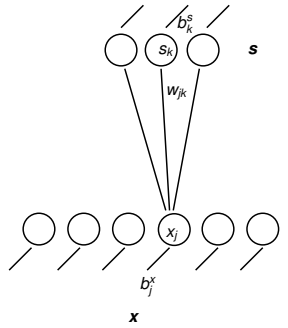
$$\boldsymbol{s} = W^{\mathrm{T}}\boldsymbol{x}$$



- Most feature learning models are based on a variation of the encoder/decoder equations.
- PCA is a special case with linear dependencies and low-dimensional $\boldsymbol{s}$

Feature learning

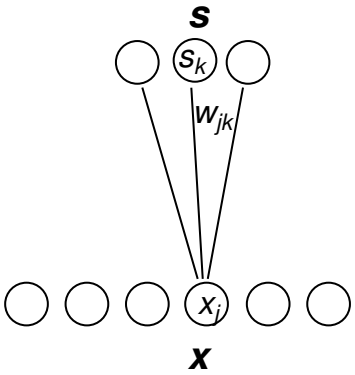
$$\boldsymbol{s} = W^{\mathrm{T}}\boldsymbol{x} + b^s$$



- In practice, one typically adds *bias terms* to obtain affine not linear dependencies.
- But one often drops these in derivations.

Feature learning models

$$\boldsymbol{s} = W^{\mathrm{T}}\boldsymbol{x}$$
$$\boldsymbol{x} = W\boldsymbol{s}$$

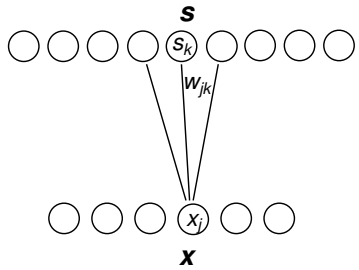


PCA

- Training: minimize $\|\boldsymbol{x} - W\boldsymbol{s}\|^2 = \|\boldsymbol{x} - W^{\mathrm{T}}W\boldsymbol{x}\|$ s.t. orthogonality constraint.

Feature learning models

$$\boldsymbol{x} = W\boldsymbol{s}$$

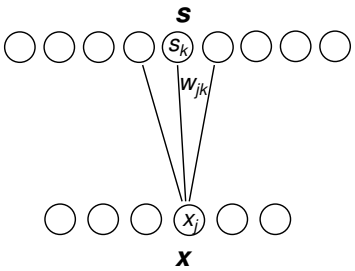


Sparse coding (Olshausen/Fields)

- Energy: $\sum_i \|\boldsymbol{x}_i - W\boldsymbol{s}_i\|^2 + \sum_i |\boldsymbol{s}_i|$ (optimize wrt. $\boldsymbol{s}_i$ and W)
- Inference: *Search* for $\boldsymbol{s}$ that minimizes the same cost.

Feature learning models

$$\boldsymbol{s} = \text{sigmoid}(W^{\mathrm{T}}\boldsymbol{x})$$
$$\boldsymbol{x} = \text{sigmoid}(W\boldsymbol{s})$$

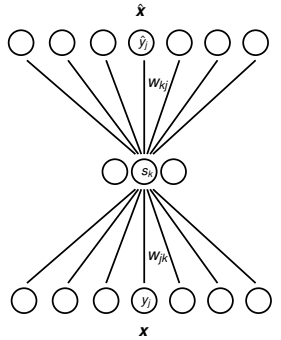


(Binary) Restricted Boltzmann Machine (RBM)

- Density: $p(\boldsymbol{x}, \boldsymbol{s}) = \frac{1}{Z} \exp\left(\sum_{jk} w_{jk} x_j s_k\right)$
- Contrastive divergence learning (Hebbian/anti-hebbian learning rule)
- Probability model with $G_i(\boldsymbol{x}) = \left(1 + \exp(\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x})\right)$

Feature learning models

$$\boldsymbol{s} = \text{sigmoid}(W^{\mathrm{T}}\boldsymbol{x})$$
$$\boldsymbol{x} = W\boldsymbol{s}$$

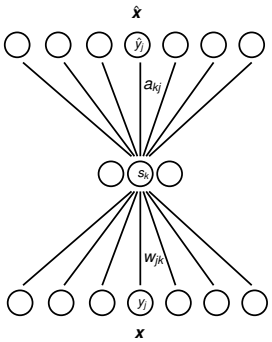


Autoencoder

- Energy: $\sum_i \|\boldsymbol{x}_i - W\text{sigmoid}\left(W^{\mathrm{T}}\boldsymbol{x}_i\right)\|^2$
- Can add a sparsity penalty for $\boldsymbol{s}$, or *corrupt inputs during training* (Vincent et al., 2008)

Feature learning models

$$\boldsymbol{s} = \text{sigmoid}(A^{\mathrm{T}}\boldsymbol{x})$$
$$\boldsymbol{x} = W\boldsymbol{s}$$

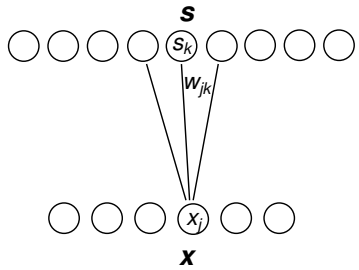


Autoencoder with "untied" weights

- Energy: $\sum_i \|\boldsymbol{x}_i - A\text{sigmoid}\left(W^{\mathrm{T}}\boldsymbol{x}_i\right)\|^2$
- Can add a sparsity penalty for $\boldsymbol{s}$, or *corrupt inputs during training* (Vincent et al., 2008)

Feature learning models

$$\boldsymbol{s} = \text{wta}(W^{\mathrm{T}}\boldsymbol{x})$$
$$\boldsymbol{x} = W\boldsymbol{s}$$

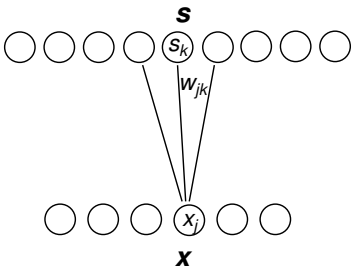


k-means clustering

- Energy: $\sum_i \|\boldsymbol{x}_i - W\boldsymbol{s}\|^2$ with $\boldsymbol{s}$ infered as follows:
- Inference: Set $s_i = 1$ for the i with maximal $\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}$, 0 otherwise ("winner-takes-all")

Feature learning models

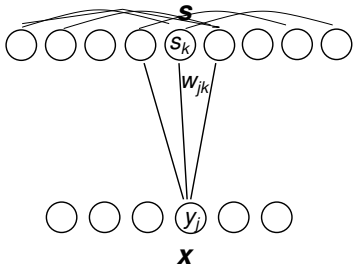
$$\boldsymbol{s} = \mathrm{wta}(W^{\mathrm{T}}\boldsymbol{x})$$
$$\boldsymbol{x} = W\boldsymbol{s}$$



Mixtures of Gaussians

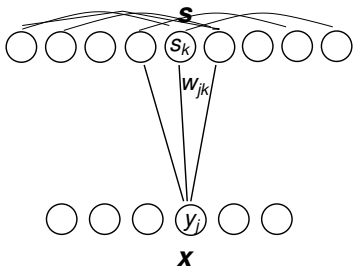
- Probalistic version of kmeans.
- Density:
  $p(\boldsymbol{x}) = \sum_k p(s_k)p(\boldsymbol{x}|s_k) = \sum_k p(s_k)\mathcal{N}(\mathrm{x}|W_{\cdot k}, \Sigma_k)$
- Inference: Bayes' rule.

Lateral interactions



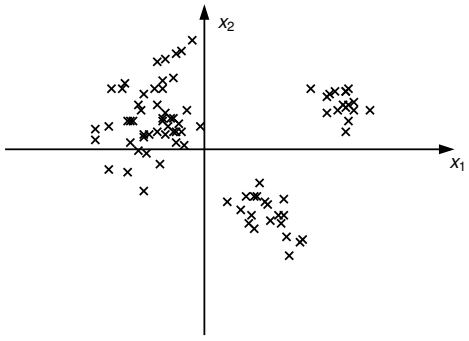
- In some of these models, inference requires **lateral interactions** or **feedback**.
- This means that hidden units need to talk to each other to figure out their joint activity pattern.
- In most cases, the interactions are *inhibitory* and lead to *competition*.
- Examples include sparse coding, k-means, MoG.

Lateral interactions



- *End-stopping* is one known effect that can be explained with competition among simple cells.
- A process related to lateral interaction is top-down feedback from higher processing layers.

K-means clustering



- K-means is traditionally a clustering algorithm.
- Learning: Fit K prototypes μ_k (these will be the rows of W) to training data-points $\boldsymbol{x}_n$.
- Inference: Given a point, find the nearest prototype.

K-means clustering

- Think of $\boldsymbol{s}$ as a one-hot encoding of the discrete variable representing the index of the cluster center.
- There are K *prototypes* $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ that represent the K clusters. The dimensionality of the prototypes is the same as that of the data $\boldsymbol{x}$.
- Assume we *knew* the cluster assignments $\boldsymbol{s}_n$ for each point $\boldsymbol{x}_n$.
- The K-means objective function measures the *average distance between points* $\boldsymbol{x}$ *and their representatives*:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} s_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$$

K-means clustering

- *Learning* amounts to finding *both* the prototypes $\boldsymbol{\mu}_k$ *and* the assignments $\boldsymbol{s}_n$ for each point, so as to minimize J.
- This seems like a tricky optimization problem, because the $\boldsymbol{s}_n$ are discrete and the $\boldsymbol{\mu}_k$ are continuous.
- But learning gets easy if we decouple learning the $\boldsymbol{s}_n$ from learning the $\boldsymbol{\mu}_k$.
- This gives rise to a *block coordinate-descent method*, which is a special case of the *EM-algorithm* for training mixtures of Gaussians.

K-means clustering

Finding the optimal $\boldsymbol{s}_n$

- Note that given the $\boldsymbol{\mu}_k$, we can optimize all the $\boldsymbol{s}_n$ independently, because the objective is just the sum over n.
- But the squared error will be smallest if we set $s_{nk} = 1$ for whichever $\boldsymbol{\mu}_k$ is *closest*.
- Formally, to optimize all $\boldsymbol{s}_n$, given the set of $\boldsymbol{\mu}_k$, set:

$$s_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

K-means clustering

Finding the optimal $\boldsymbol{\mu}_k$

- Given S, J is a quadratic function of $\boldsymbol{\mu}_k$ which we can minimize by setting the derivative to zero:

$$2 \sum_{n=1}^{N} s_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) = 0$$

- Solving for $\boldsymbol{\mu}_k$ yields:

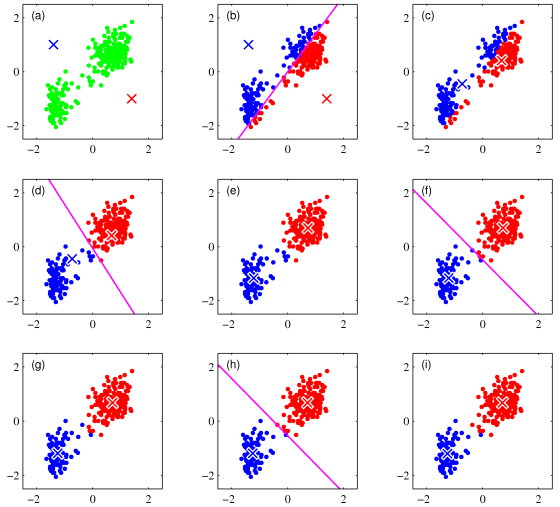
$$\boldsymbol{\mu}_k = \frac{\sum_n s_{nk} \boldsymbol{x}_n}{\sum_n s_{nk}}$$

- This solution has a simple interpretation: Set each $\boldsymbol{\mu}_k$ to the mean of all points currently assigned to cluster k !

K-means clustering

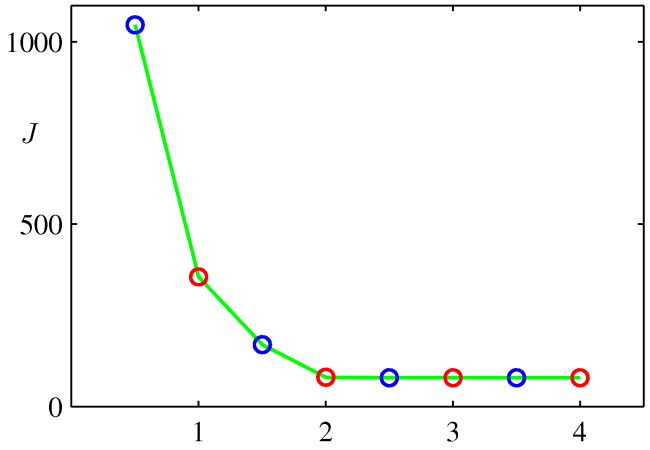
- Learning amounts to iterating inference of the $\boldsymbol{s}_n$, and adapting the parameters $\boldsymbol{\mu}_k$ until there are no more changes.
- This training procedure always converges: J is positive, and every step either decreases it or leaves it unchanged.
- But there can be local minima.
- One way to deal with this is to try multiple runs with different initializations for the parameters $\boldsymbol{\mu_k}$ and to pick the solution with the lowest final cost.
- (Bishop, 2006):

K-means example



- K-means with $K = 2$.

The value of J as learning progresses



K-means inference

- Given the trained model, we can infer the cluster-centers for new test-data points $\boldsymbol{x}$ not seen during training: Pick the nearest $\boldsymbol{\mu}_k$ like we did during training:

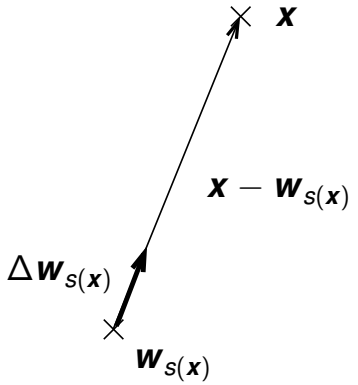
$$s_k = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

- So inference involves some *non-linear* interactions between the hiddens.
- The set of all K prototypes $\boldsymbol{\mu}_k$ is sometimes called *codebook*.
- Clustering and K-means are also known as *vector quantization*.

K-means via online learning

- The reconstruction error for training point $\boldsymbol{x}$ may be written
$$E(W) = \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{w}_{s(\boldsymbol{x})}\right)^2$$
- Its gradient is
$$\frac{\partial E(W)}{\partial \boldsymbol{w}_i} = -\left(\boldsymbol{x} - \boldsymbol{w}_{s(\boldsymbol{x})}\right)\delta_{s(\boldsymbol{x}),i}$$
- So we can use the online learning rule:
$$\boldsymbol{w}_{s(\boldsymbol{x})} \leftarrow \boldsymbol{w}_{s(\boldsymbol{x})} + \eta\left(\boldsymbol{x} - \boldsymbol{w}_{s(\boldsymbol{x})}\right)$$

Geometry of online K-means



- $\Delta \boldsymbol{w}_{s(\boldsymbol{x})} = \eta(\boldsymbol{x} - w_{s(\boldsymbol{x})})$ moves the winning weight vector towards the observation.

Online K-means and Hebbian learning

- We can interpret the online k-means updates as:

  Hebb-rule + competition + unlearning
- To this end write the update as
$$\Delta \boldsymbol{w}_k = \eta \delta_{ks(\boldsymbol{x})}\left(\boldsymbol{x} - \boldsymbol{w}_k\right)$$
  where
$$\delta_{ks(\boldsymbol{x})} = \begin{cases} 1 & \text{if } s(\boldsymbol{x}) = k \\ 0 & \text{else} \end{cases}$$
  is the "post-synaptic activity" determined by competition ("winner takes all" rule)
- There are two learning terms:

K-means and Hebbian learning

1. A Hebbian term:
$$\delta_{ks(\boldsymbol{x})}\boldsymbol{x}$$
2. An "unlearning" term:
$$-\delta_{ks(\boldsymbol{x})}\boldsymbol{w}_k$$

- The positive term decreases the energy near the data.
- The unlearning term increases the energy everywhere.
- "Hebb-rule + competition + unlearning" are present (not surprisingly) in a wide variety of learning algorithms, including contrastive divergence learning for RBMs.

K-means features from natural image patches