ECE/CS 559 Lecture 17/18 Th 10/24 / T 10/29

Last time: CNNs (max pooling, resnets), PyTorch

① Unsupervised Learning

Before, data (x, y). Now, data is just x.

What is the task? Represent x "nicely".

- Clustering: Place x in one of a handful of "clusters"
- Dimensionality reduction: Represent $x \in \mathbb{R}^m$ by $x \in \mathbb{R}^k$ $k \ll m$.
- Density estimation: Find the distribution of x

Relationship to generative models & feature extraction.

(represent → create more) ("enough" to represent)

② k-means Clustering

Input $x \in \mathbb{R}^n$, parameters $\underbrace{c_1, c_2, \dots, c_k}_{C}$, $c_i \in \mathbb{R}^n$.

Output $f(x; C) = \underset{c \in C}{\operatorname{argmin}} \|x - c\|_2^2$

Loss: $\ell(x, f) = \|x - f\|_2^2$

Risk: $R(C) = \frac{1}{|\text{Data}|} \sum_{x \in \text{Data}} \ell(x, f(x; C))$

How do we minimize $R(C)$?

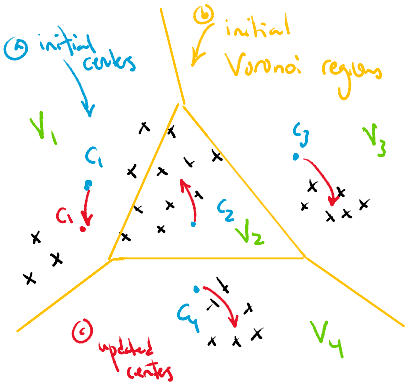
- It's possible to do gradient descent. It's non-convex!
- Or... use Lloyd's algorithm aka "the" k-means algorithm.

Lloyd's Algorithm

Input: Data, k

- Initialize $C = \{c_1, \dots, c_k\}$ (arbitrarily / randomly) ⓐ
- Repeat:
  - Cluster according to C (Voronoi regions) ⓑ
  - $c_i \leftarrow \text{Mean}(x : x \text{ closest to } c_i) = \text{Mean}(x : x \in V_i)$ ⓒ
- Stop when convergence slows.

Output: Cluster centers C

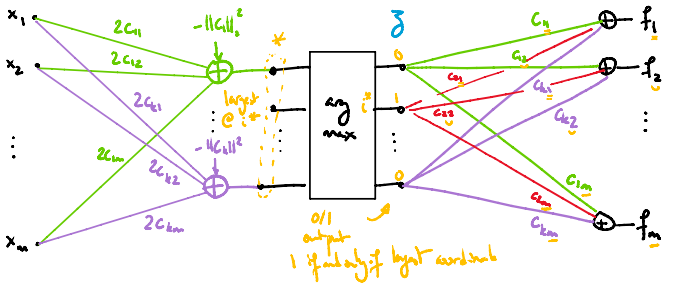


Voronoi regions partition the space according to proximity to specific centers

③ From k-Means to Hebbian updates

Since $\|x - c\|_2^2 = (x-c)^T(x-c) = \|x\|_2^2 + \|c\|_2^2 - 2x^Tc$ *

We can write the clustering as a neural network:



Note that f will be the closest center.

- Because argmax is not differentiable (constant, except at jumps), no gradient information backpropagates before it.
- We still have gradients after it, starting at: $\frac{\partial \ell}{\partial f_j} = -2(x_j - f_j)$

Then: $\frac{\partial \ell}{\partial c_{ij}} = \frac{\partial \ell}{\partial f_j} \cdot \frac{\partial f_j}{\partial c_{ij}} = -2 z_i (x_j - f_j)$ — for those $f_j = c_{ij}$

$\frac{\partial f_j}{\partial c_{ij}} = z_i$ ↑ only wiggles at $i = i^*$ (argmax index)

- Thus: $\nabla_{c_i} R = \sum_{x \in V_i} -2(x_j - f_j) = -2 \sum_{x \in V_i} x + 2|V_i| c_j$

Gradient descent becomes:

$$c_i \leftarrow c_i - \eta \nabla_{c_i} R = c_i + 2\eta \sum_{x \in V_i} x - 2\eta |V_i| c_j$$

if $\eta_i = \frac{1}{2|V_i|}$ then $= \frac{1}{|V_i|} \sum_{x \in V_i} x$ same as Lloyd's algorithm!

- Assume $\|x\| = \|w\|$ (or $\|c\|$) $= 1$ for all weights.
  - Reasonable by going one dimension higher
  - Distance → (minus) inner-products! cosine(angle) ↑ distance ↓
- That was full batch. With 1 point, we get instead

$\forall i$: $c_i \leftarrow c_i + \eta(x - c_i) \mathbb{1}\{\underset{i}{\operatorname{argmax}} (c_i^T x) = i\}$ — no bias needed

self-supervision $= z_i$

- z is the neurons' output (decision). In general, it doesn't have to be restricted to 0/1.
- Definition: Competitive (winner-takes-all) learning rule:

$w_i \leftarrow w_i + \eta z_i (x - w_i)$ only for $i = \underset{i}{\operatorname{argmax}}\, z_i$

- If $y_i \in (0,1)$ then

$$w_i \leftarrow (1-\eta y_i) w_i + \eta y_i x$$

↳ weights that look not like x are made to look more like x (they specialize at identifying things similar to x / clusters)

④ Oja's rule:

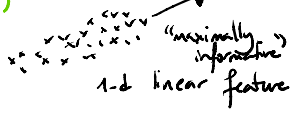
Definition: $w_i \leftarrow w_i + \eta y_i (x - w_i y_i)$ — difference; doesn't affect it if 0/1

- Claim, for small η, $w + \eta y (x - wy) \approx \frac{(w + \eta x y)}{\|w + \eta x y\|}$

vector scalar scalar vector (for each i)

- Extracts principle component:

$$\rightarrow \underset{w:\|w\|=1}{\operatorname{argmax}} \sum_{x \in \text{data}} (w^T x)^2$$



⑤ Hebbian rules

- Most of these rules are "Hebbian":
  "Neurons that fire together, wire together"
- If x is coming from another neuron and w is a neuron that mimics it, then it will strengthen.
- **Definition:** Original Hebbs rule: $w_i \leftarrow (w_i + \eta x y_i)$
  (compare to Perceptron with true supervision)
- It can also cluster! But it's unstable:
  Fixes: Oja's rule, $w_i \leftarrow \alpha w_i + \eta x y_i$
  ↳ dampening < 1.

Variants: $w_i \leftarrow w_i + \eta (x - \bar{x})(y_i - \bar{y}_i)$

Example: $y_i = \text{Sign}(w_i^T x)$

- Say $x = 1$ vector, fixed $w_i(0) = 1$ vector also initially. $\eta = 1$.
  $w_i(1) = 1 + 1 = 2$ $w_i(2) = 2 + 2 = 4$ $w_i(3) = 8 \ldots$
- If we stabilize with $\alpha < 1$:
  $w_i(1) = \alpha + 1$ $w_i(2) = \alpha(\alpha+1) + 1 = \alpha^2 + \alpha + 1$
  $w_i(3) = \alpha(\alpha^2 + \alpha + 1) + 1 \quad \cdots \rightarrow \frac{1}{1-\alpha}$
- In the notes: Examples of Hebbs rule and competitive