TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

9.2 Non-linear Clustering

Machine Learning 1: Foundations

Marius Kloft *(TUK)*

1 Linear Clustering

2 Non-linear Clustering

3 Hierarchical Clustering

Kernel k-means clustering

The k-means clustering algorithm can be "kernelized".

Kernel *k*-means clustering

The k-means clustering algorithm can be "kernelized".

function KMEANS(parameter k, inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$)
  initialize cluster centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$
  repeat
    for $i = 1 : n$ **do**
      label the input $\mathbf{x}_i$ as belonging to the nearest cluster,

$$y_i := \arg\min_{j=1,\ldots,k} \left\| \mathbf{x}_i - \mathbf{c}_j \right\|^2$$

    end for
    for $j = 1 : k$ **do**
      compute cluster center $\mathbf{c}_j$ as the mean of all inputs of the jth cluster,

$$\mathbf{c}_j := \mathrm{mean}(\{\mathbf{x}_i : y_i = j\})$$

    end for
  until convergence criterion is met
  return cluster centers $c_1, \ldots, c_k$
end function

Kernel *k*-means clustering

The k-means clustering algorithm can be “kernelized”.

function KMEANS(parameter k, inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$)
  initialize cluster centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$
  repeat
    for $i = 1 : n$ **do**
      label the input $\mathbf{x}_i$ as belonging to the nearest cluster,

$$y_i := \arg\min_{j=1,\ldots,k} \left\| \mathbf{x}_i - \mathbf{c}_j \right\|^2$$

    end for
    for $j = 1 : k$ **do**
      compute cluster center $\mathbf{c}_j$ as the mean of all inputs of the jth cluster,

$$\mathbf{c}_j := \mathrm{mean}(\{\mathbf{x}_i : y_i = j\})$$

    end for
  until convergence criterion is met
  return cluster centers $c_1, \ldots, c_k$
end function

- Never compute mean explicitly

Kernel *k*-means clustering

The k-means clustering algorithm can be "kernelized".

function KMEANS(parameter k, inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$)
    initialize cluster centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$
    repeat
        for $i = 1 : n$ **do**
            label the input $\mathbf{x}_i$ as belonging to the nearest cluster,

$$y_i := \arg\min_{j=1,\ldots,k} \left\|\mathbf{x}_i - \mathbf{c}_j\right\|^2$$

        end for
        for $j = 1 : k$ **do**
            compute cluster center $\mathbf{c}_j$ as the mean of all inputs of the jth cluster,

$$\mathbf{c}_j := \mathrm{mean}(\{\mathbf{x}_i : y_i = j\})$$

        end for
    until convergence criterion is met
    return cluster centers $c_1, \ldots, c_k$
end function

- Never compute mean explicitly
- Compute $\left\|\mathbf{x}_i - \mathbf{c}_j\right\|^2$ via kernel function

Kernel k-means clustering

- Let $\phi : \mathbb{R}^d \to \mathcal{H}$ be a kernel feature map, corresponding to a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Kernel k-means clustering

- Let $\phi : \mathbb{R}^d \to \mathcal{H}$ be a kernel feature map, corresponding to a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
- Let $I_j := \{i \in \{1, \ldots, n\} : y_i = j\}$ for all $j = 1, \ldots, k$

Kernel *k*-means clustering

- Let $\phi : \mathbb{R}^d \to \mathcal{H}$ be a kernel feature map, corresponding to a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
- Let $I_j := \{i \in \{1, \ldots, n\} : y_i = j\}$ for all $j = 1, \ldots, k$
- Then $\mathbf{c}_j := \frac{1}{|I_j|} \sum_{i \in I_j} \phi(\mathbf{x}_i)$

Kernel k-means clustering

- Let $\phi : \mathbb{R}^d \to \mathcal{H}$ be a kernel feature map, corresponding to a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
- Let $I_j := \{i \in \{1, \ldots, n\} : y_i = j\}$ for all $j = 1, \ldots, k$
- Then $\mathbf{c}_j := \frac{1}{|I_j|} \sum_{i \in I_j} \phi(\mathbf{x}_i) \in \mathcal{H}$ (high dimensional!)

Kernel *k*-means clustering

- Let $\phi : \mathbb{R}^d \to \mathcal{H}$ be a kernel feature map, corresponding to a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
- Let $I_j := \{i \in \{1, \ldots, n\} : y_i = j\}$ for all $j = 1, \ldots, k$
- Then $\mathbf{c}_j := \frac{1}{|I_j|} \sum_{i \in I_j} \phi(\mathbf{x}_i) \in \mathcal{H}$ (high dimensional!)
- However, $\left\|\phi(\mathbf{x}_i) - \mathbf{c}_j\right\|^2 = \underbrace{\|\phi(\mathbf{x}_i)\|^2}_{=k(\mathbf{x}_i,\mathbf{x}_i)} - 2\left\langle \phi(\mathbf{x}_i), \mathbf{c}_j \right\rangle + \left\|\mathbf{c}_j\right\|^2$

Kernel *k*-means clustering

- Let $\phi : \mathbb{R}^d \to \mathcal{H}$ be a kernel feature map, corresponding to a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
- Let $I_j := \{i \in \{1, \ldots, n\} : y_i = j\}$ for all $j = 1, \ldots, k$
- Then $\mathbf{c}_j := \frac{1}{|I_j|} \sum_{i \in I_j} \phi(\mathbf{x}_i) \in \mathcal{H}$ (high dimensional!)
- However, $\left\|\phi(\mathbf{x}_i) - \mathbf{c}_j\right\|^2 = \underbrace{\|\phi(\mathbf{x}_i)\|^2}_{=k(\mathbf{x}_i,\mathbf{x}_i)} - 2\left\langle \phi(\mathbf{x}_i), \mathbf{c}_j \right\rangle + \left\|\mathbf{c}_j\right\|^2$

  can be completely expressed in terms of kernel functions:
  - $\|\mathbf{c}_j\|^2 = \frac{1}{|I_j|^2} \sum_{i,i' \in I_j} k(\mathbf{x}_i, \mathbf{x}_{i'})$ and $\langle \phi(\mathbf{x}_i), \mathbf{c}_j \rangle = \frac{1}{|I_j|} \sum_{i' \in I_j} k(\mathbf{x}_i, \mathbf{x}_{i'})$ □

Kernel k-means clustering

- Let $\phi : \mathbb{R}^d \to \mathcal{H}$ be a kernel feature map, corresponding to a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
- Let $I_j := \{i \in \{1, \ldots, n\} : y_i = j\}$ for all $j = 1, \ldots, k$
- Then $\mathbf{c}_j := \frac{1}{|I_j|} \sum_{i \in I_j} \phi(\mathbf{x}_i) \in \mathcal{H}$ (high dimensional!)
- However, $\left\|\phi(\mathbf{x}_i) - \mathbf{c}_j\right\|^2 = \underbrace{\|\phi(\mathbf{x}_i)\|^2}_{=k(\mathbf{x}_i,\mathbf{x}_i)} - 2\left\langle \phi(\mathbf{x}_i), \mathbf{c}_j \right\rangle + \left\|\mathbf{c}_j\right\|^2$ can be completely expressed in terms of kernel functions:
  - $\|\mathbf{c}_j\|^2 = \frac{1}{|I_j|^2} \sum_{i,i' \in I_j} k(\mathbf{x}_i, \mathbf{x}_{i'})$ and $\langle \phi(\mathbf{x}_i), \mathbf{c}_j \rangle = \frac{1}{|I_j|} \sum_{i' \in I_j} k(\mathbf{x}_i, \mathbf{x}_{i'})$ $\square$

But what if we want to cluster images?

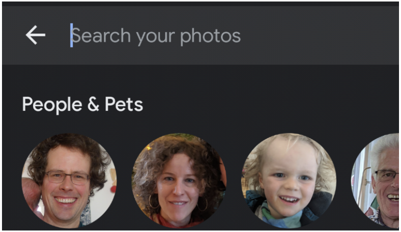
Example: Search Photo Collection For People

Given:

- Photo collection

Aim:

- Search for people



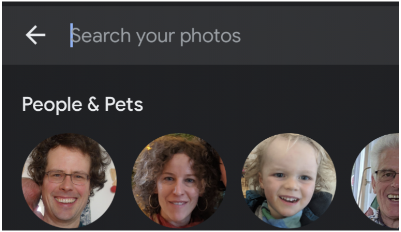
Example: Search Photo Collection For People

Given:

- Photo collection

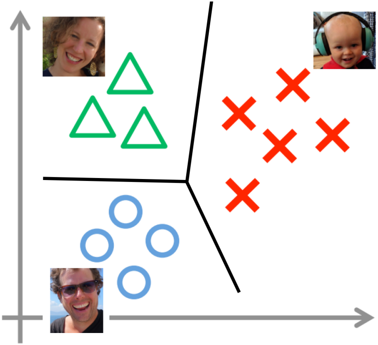
Aim:

- Search for people



Solution:

- Automatically crop out portraits
- **Cluster the portraits**



Deep Clustering

Can we make k-means deep?

Deep Clustering

Can we make k-means deep?

One can show:

- k-means minimizes the objective function

$$\min_{c=(\mathbf{c}_1,\ldots,\mathbf{c}_k)\in\mathbb{R}^{d\times k}} \sum_{i=1}^{n} \min_{j\in\{1,\ldots,k\}} \left\| \mathbf{x}_i - c_j \right\|^2$$

Deep Clustering

Can we make k-means deep?

One can show:

- k-means minimizes the objective function

$$\min_{c=(\mathbf{c}_1,\ldots,\mathbf{c}_k)\in\mathbb{R}^{d\times k}} \sum_{i=1}^{n} \min_{j\in\{1,\ldots,k\}} \left\|\mathbf{x}_i - c_j\right\|^2$$

Analog to regression and classification, we could "deepify" k-means as follows:

Deep k-means

$$\min_{W,\mathbf{c}} \; \frac{1}{2}\sum_{l=1}^{L} \left\|W^{(l)}\right\|_{\text{Fro}}^2 + \sum_{i=1}^{n} \min_{j\in\{1,\ldots,k\}} \left\|\phi_W(\mathbf{x}_i) - c_j\right\|^2$$

Deep Clustering

Can we make k-means deep?

One can show:

- k-means minimizes the objective function

$$\min_{c=(\mathbf{c}_1,\ldots,\mathbf{c}_k)\in\mathbb{R}^{d\times k}} \sum_{i=1}^{n} \min_{j\in\{1,\ldots,k} \left\|\mathbf{x}_i - c_j\right\|^2$$

Analog to regression and classification, we could “deepify” k-means as follows:

Deep k-means

$$\min_{W,\mathbf{c}} \; \frac{1}{2}\sum_{l=1}^{L} \left\|W^{(l)}\right\|_{\text{Fro}}^2 + \sum_{i=1}^{n} \min_{j\in\{1,\ldots,k} \left\|\phi_W(\mathbf{x}_i) - c_j\right\|^2$$

But this method does not always work so well

- ongoing research topic to get this working

Another Idea: Transfer Learning

We learned this trick already in the class on overfitting:

Another Idea: Transfer Learning

We learned this trick already in the class on overfitting:

1. Pre-training: Download an ANN with parameters W from the web that was pre-trained on huge amounts of images

Another Idea: Transfer Learning

We learned this trick already in the class on overfitting:

1. Pre-training: Download an ANN with parameters W from the web that was pre-trained on huge amounts of images
2. Deep representation of data: For each input $\mathbf{x}_i$, compute its vector of activations in the last hidden layer $\phi_W(\mathbf{x}_i)$

Another Idea: Transfer Learning

We learned this trick already in the class on overfitting:

1. Pre-training: Download an ANN with parameters W from the web that was pre-trained on huge amounts of images
2. Deep representation of data: For each input $\mathbf{x}_i$, compute its vector of activations in the last hidden layer $\phi_W(\mathbf{x}_i)$
3. Clustering: Run k-means on $\phi_W(\mathbf{x}_1), \ldots, \phi_W(\mathbf{x}_n)$