NDD October 7th, 2021

1. Data experiment / OpenML
2. CNN – KDN → CNN?
3. New other data
4. High dimensional problem → sparse parity
5. Robust to contaminating distributions.
6. Run the benchmarks.

* The accuray is better than the chance accuray.
* FTE → no theoretically guarentees.
  can't say for sure since study's are empirical.
  Theory!! adversarial task! → hard to measure /task similarity
- recruiting gaussian distributions relevant to task.
- pruning the polytope
- KDF
- CNN??

- double counting → increase sample size.

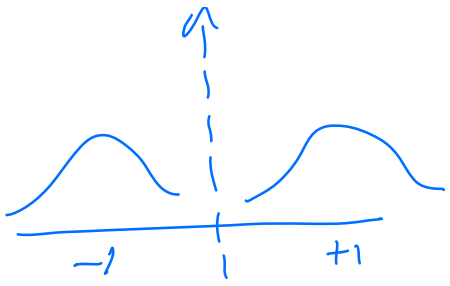
KDG / Jayanta Oct 11th 2024

2^5 possible polytopes. → 5 possibilities.

* penultimate layer →
* filtered same path → same partition.
* CIFAR-10 – simple data
* covariance matrix ~ dimensionality re
  → high dimensions
* KDN → feature selection.
* Deep Boltzmann Machines.
* High Dimensionality → KDN
* MNIST → • test the KD-CNN on a lower dim dataset
  CIFAR-10 • penultimate layer — ①
  • all the fully-connected layers — ②
  • Deep Boltzmann machines
  • Dealing with high-dimensionality
  • Histogram of polytopes

Weighted Maximum Likelihood

→ weighting using no # paths.



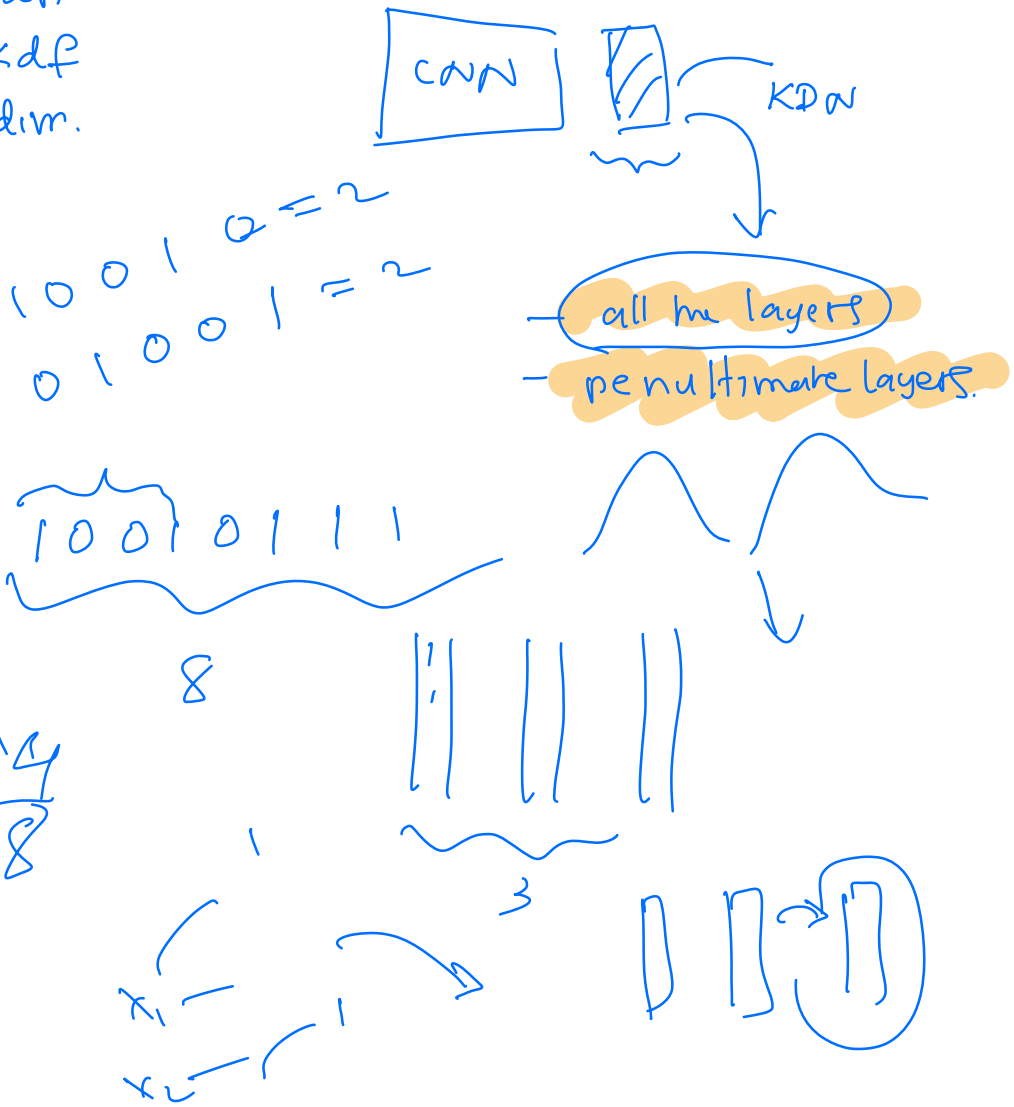
weighting!

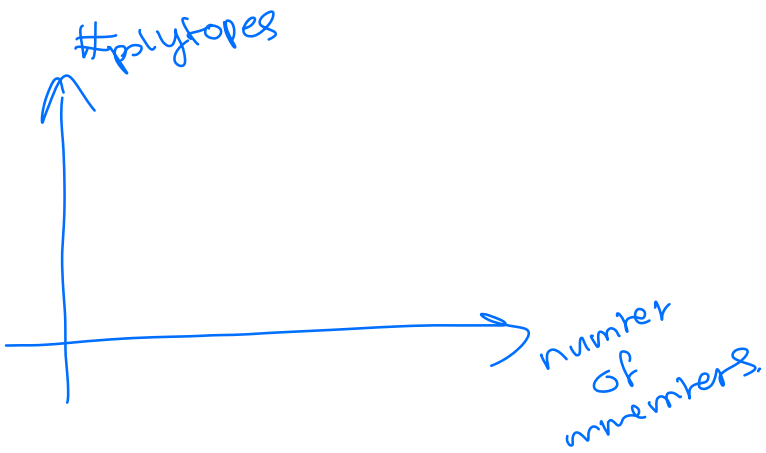
Dim-reduction!

sim_kdn

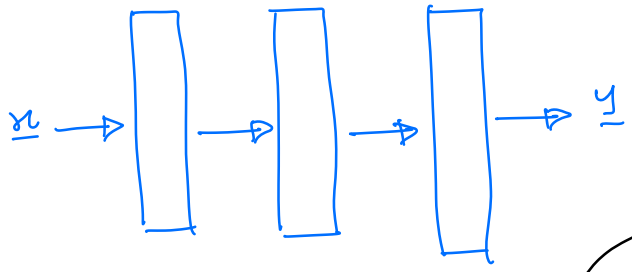
sim_kdf

high-dim.



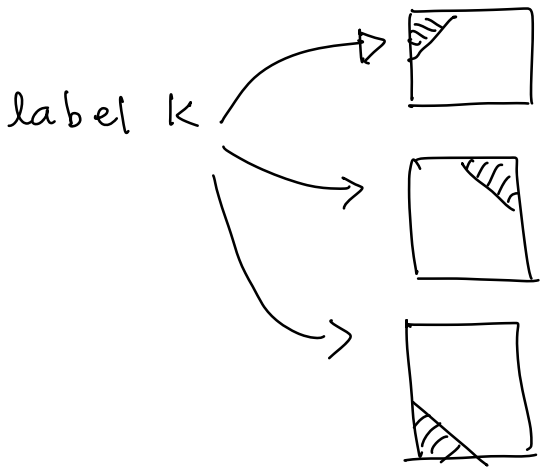


Kernel Density Network



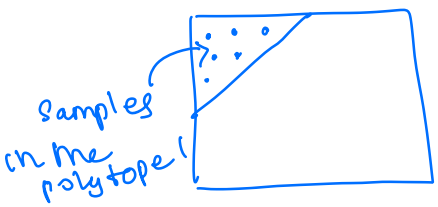
some polytope memters has weighting of 1

other memters would have a weighting based on their degree of match



polytopes 2 and 3 would have less weighting/ contribution

consider polytope 1 ⇒



samples ⇒ $(x_1, x_2, x_3, \ldots, x_m)$

where $x_i \in \mathbb{R}^d$ (d - feature dimension)

* we are interested in fitting a Gaussian over the samples (aka polytope members)

We need to estimate parameters μ, Σ ← $\phi(x \mid \mu, \Sigma)$

d-variate Gaussian

The joint distribution of the observed m polytope members. ⇒

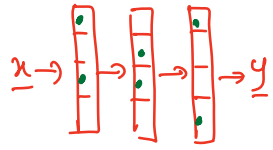
$$f(x_1, x_2, \ldots, x_m \mid \mu', \Sigma')$$

If x_i $(i = 1, \ldots, m)$ are iid,

$$f(x_1, x_2, \ldots, x_m \mid \mu', \Sigma') = \prod_{i=1}^{m} \phi(x_i \mid \mu, \Sigma)$$

$$l(\mu, \Sigma) = \prod_{i=1}^{m} \phi(x_i \mid \mu, \Sigma)$$

$$\hat{\mu}, \hat{\Sigma} = \underset{\mu, \Sigma}{\operatorname{argmax}} \prod_{i=1}^{m} \phi(x_i \mid \mu, \Sigma)$$



* In the KDN, all the sample points in a given polytope have the same activation pattern

eg:-

$x_1 \rightarrow$ [1010 0110 1001]

$x_2 \rightarrow$ [1010 0110 1001]

$x_3 \rightarrow$ [1010 0110 1001]

⋮

$x_m \rightarrow$ [1010 0110 1001]

* when $d >> m$, it's not feasible to estimate Σ ($d \times d$ matrix) from just the m observations

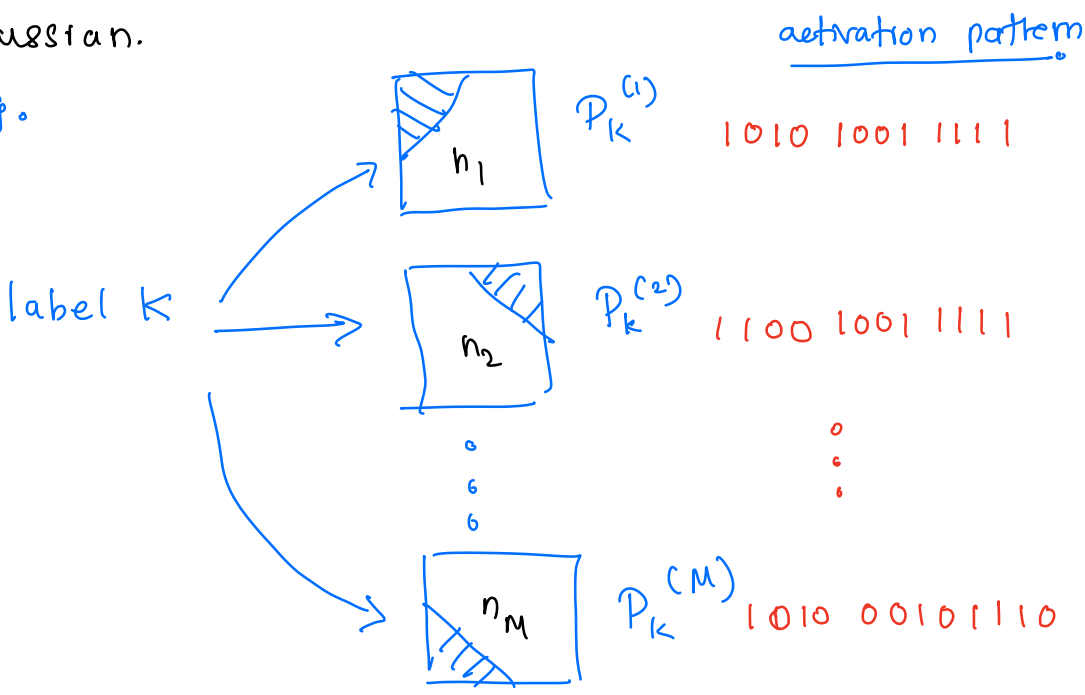
* However, there might be other samples in the same class (but in different polytopes) that are approximately-matched with the activation patterns of the samples in the given polytope.

eg:- $x_1 \rightarrow [1\,0\,1\,0\;\;0\,1\,1\,0\;\;1\,0\,0\,1]$
$x^* \rightarrow [1\,0\,1\,0\;\;1\,0\,1\,0\;\;1\,0\,0\,1]$
$\Big\}\; x_1 \tilde{=} x^*$

---

* How does Jayanta's weighting help with the incorporation of approximately-matched samples?

* we incorporate the samples from other polytopes of the same class to enrich the estimate we get for covariance & mean of underlying Gaussian.



Consider fitting a Gaussian to $P_k^{(1)}$.

$P_k^{(1)}$ has n_1 samples but $n_1 << d$. Therefore we need more samples. $P_k^{(\ell)}$ where $\ell \neq 1$ are neighboring polytopes of $P_k^{(1)}$ that belong to the same class. We can leverage these neighbors to increase the sample size.

However the contributions from other polytopes would not be same as that in $P_k^{(1)}$. To correct for this contribution mismatch we can weight the samples from other polytopes.

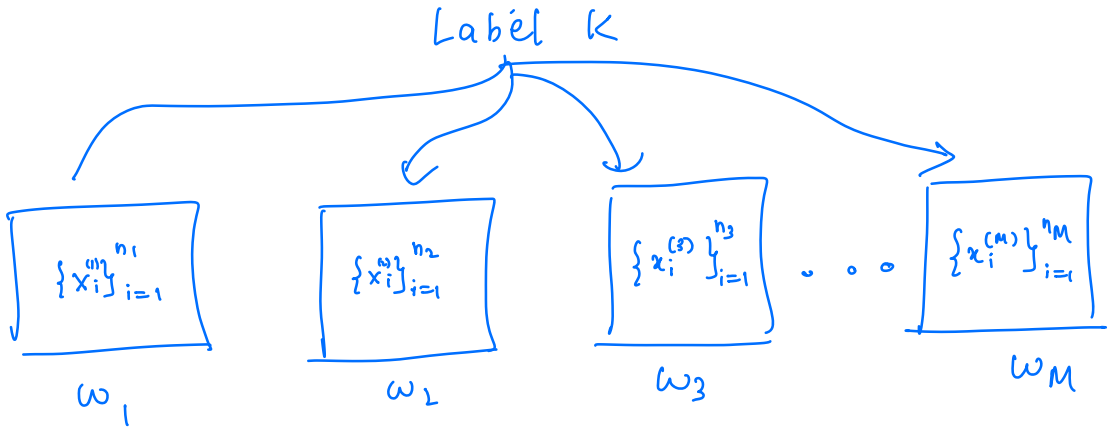
We can leverage the degree of activation pattern mismatch (match) to compute the weights for each polytope.

eg:- 1 0 1 0 1 0 0 1 1 1 1 1 ← $P_k^{(1)}$'s activation pattern

1 0 1 1 0 0 0 1 0 1 1 1 ← $P_k^{(\ell)}$ " "

∴ w_ℓ (weight assigned to $P_k^{(\ell)}$ samples) should reflect this degree of similarity / disagreement / match.

contd...



$x_1 \quad x_2 \quad x_3 \quad \cdots \quad x_m$

$\omega_1 \quad \omega_2 \quad \omega_2 \quad \cdots \quad \omega_m$

iid

$$f(x_1, x_2, \ldots, x_m \mid \mu, \Sigma) = \prod_{i=1}^{m} \phi(x_i \mid \mu, \Sigma)$$

$$l(\mu, \Sigma \mid x_1, \ldots, x_m) = \prod_{i=1}^{m} \phi(x_i \mid \mu, \Sigma)$$

since the contributions of each sample point to the likelihood function is different, we apply the weighting to each $\phi(x_i \mid \mu, \Sigma)$

$$l_{\text{weighted}}(\mu, \Sigma \mid x_1, \ldots, x_m) = \prod_{i=1}^{m} \phi(x_i \mid \mu, \Sigma)^{\omega_i}$$

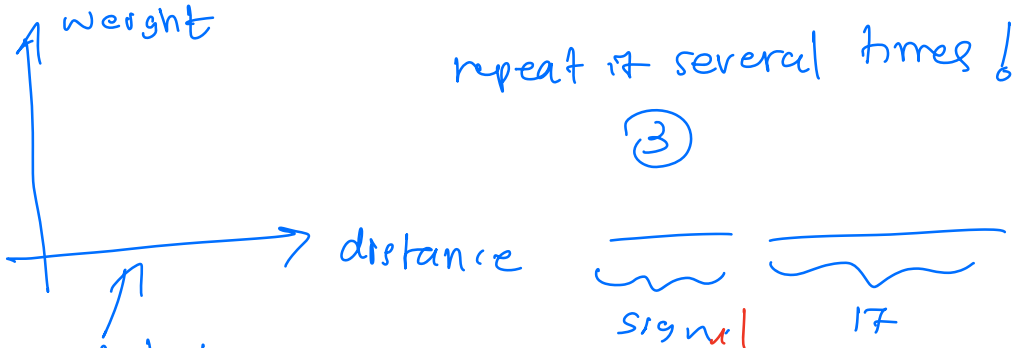
using MLE

$$\hat{\mu} = \frac{\sum_{i=1}^{m} \omega_i x_i}{\sum_{i=1}^{m} \omega_i} \quad \& \quad \hat{\Sigma} = \frac{\sum_{i=1}^{m} \omega_i (x_i - \hat{\mu})(x_i - \hat{\mu})^T}{\sum_{i=1}^{m} \omega_i}$$

* Through this weighting scheme we can incorporate the data from the neighboring polytope to estimate the gaussian kernel parameters. Since more data is being used for the estimation, the $\hat{\mu}$, $\hat{\Sigma}$ would be better estimates.

---

* weight based on the ⇒
  weight based on each layer activations!

* 

repeat it several times!

(3)

euclidean distance ✓

Mahalanobis ✗

KDE → GMM → KDF

Ledoit-Wolf →

→ experiment with weighting scheme.

DO PR!! after this

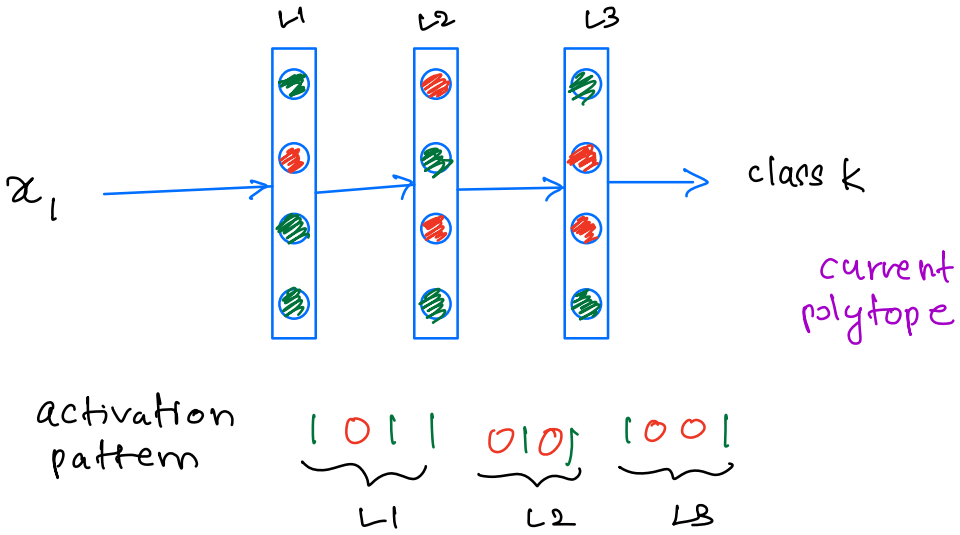
* Deep Boltzmann Machine → read!
* check Jayantiass new proof -

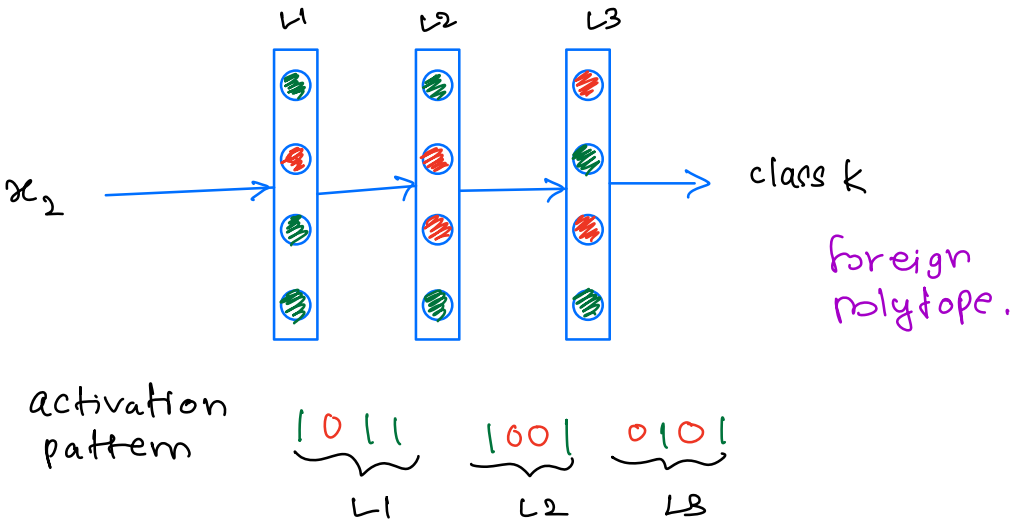
Total error^2 = bias^2 + varrance^2
error error

* All the covarunce matrees ave the same across all the polytopes.
* common covarreunce matrx. → inducing bias (because we make assumptions)
simple model → Jayanti's Proof.
* AIC/BIC criterion → evaluate estimator performance.
a lot of data but less params
better estimator
* making assumptions → bias → if it's in the right direction, we might win!
high dimensional problem may be solved.

Task enaware Learning

Weighting Schemes for KDN & KDCNN





Total Matches-based Weighting

$$w = \frac{\text{Total number of matches in the two activation patterns}}{\#\text{ FC neurons under consideration}}$$

eg:-

x_2's weight $w_2 = \frac{4+2+2}{12} = \frac{8}{12} = \frac{2}{3}$

First Mismatch-based weighting

$$w = \frac{\text{number of matched nodes up to the first mismatch}}{\text{\# FC neurons under consideration}}$$

eg :-

$$w_2 = \frac{4}{12} = \frac{1}{3}$$

Layer by Layer weighting

$$w = \frac{\text{\#L1 matches} + \text{\# L2 matches} + \dots + \text{\# Ln matches}}{\text{\# total layers}}$$

eg:- $w_2 = \frac{\frac{4}{4} + \frac{2}{4} + \frac{1}{4}}{3} = \frac{\frac{8}{4}}{3} = \frac{2}{3}$ → same as TM.

* give a larger weight to the matches towards the penultimate layer.

Other metrics ⇒ Jaccard index
Dice score
Cosine similarity
Rajski's Distance.