Il Diocane: Introduction to Data Mining Cheatsheet

Kotatsu, the Bringer of Jewishness

1 Similarity and dissimilarity

Entropy:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

Sample entropy:

$$H(X) = -\sum_{i=1}^{n} \frac{m_i}{m} \log_2 \frac{m_i}{m}$$

Mutual information:

$$I(X,Y) = H(X) + H(X) - H(X,Y)$$

where $H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} \log_2 p_{ij}$. For discrete variables the maximum mutual information is

$$\log_2(\min\{n_x, n_y\})$$

where n_x is th number of values that X can take.

We can combine similarities with

$$\text{similarity}(\mathbf{x},\mathbf{y}) = \frac{\sum_{k=1}^{n} w_k \delta_k s_k((\mathbf{x},\mathbf{y}))}{\sum_{k=1}^{n} w_k \delta_k}$$

with

$$\delta_k = \begin{cases} 0 & \text{if both attribute are asymmetric AND they are both zero or if one of them is missing} \\ 1 & \text{otherwise} \end{cases}$$

2 Clustering

Number of possible clusters:

$$B(n) = \sum_{k=0}^{\infty} \frac{k^n}{k!}$$

Sum of squared error (what we want to minimize):

$$\text{SSE} = \sum_{i-1}^{K} \sum_{x \in C_i} \mathsf{dist}(m_i, x)^2.$$

So we are trying to minimize the loss function, for the centroids of K clusters $\mathbf{c} = (c_1, \ldots, c_K)$:

$$\text{L}(\mathbf{c}) = \sum_{i=1}^{n} \min_{j=1,\ldots,K} \|x_i - c_j\|_2^2.$$

We alternate between

- updating $z_i = \arg\min_{j=1,\ldots,k} \|x_i - c_j\|_2^2$ (maps point x_i to a cluster j);
- updating $c_j = \frac{1}{|\{i|z_i = j\}|} \sum_{i|z_i=j} x_i$ (recomputes the cluster centroids)

Unsupervised measures of cluster validity

- **Cohesion**: within-cluster sum of squares (SSW)

$$\text{SSW} = \sum_{i=1}^{K} \sum_{x \in C_i} (x - m_i)^2.$$

- **Separation**: between-cluster sum of squares (SSB)

$$\text{SSB} = \sum_{i} |C_i|(m - m_i)^2$$

where $|C_i|$ is the size of cluster i and m is the global centroid.

- **Silhouette coefficient**: for a point P_i calculate the avg distance a to the points of the cluster and the minimum avg distance b to the points of another cluster. The silhouette coefficient is

$$s = \frac{b - a}{\max\{a, b\}}.$$

Supervised measures of cluster validity

- **Label probability per cluster**:

$$p_{ij} = \frac{m_{ij}}{m_j}$$

where m_j: size of cluster j and m_{ij}: number of elements of cluster j that are labelled i.

- **Entropy of cluster j**:

$$h_j = \sum_{i=1}^{L} p_{ij} \log_2 p_{ij}.$$

Total entropy:

$$h = \sum_{j=1}^{K} \frac{m_j}{m} h_j.$$

- **Purity**:

$$\text{purity}_j = \max\{p_{ij}\}.$$

Total purity:

$$\text{purity} = \sum_{j=1}^{K} \frac{m_j}{m} \text{purity}_j.$$

- **Precision**:

$$\frac{TP}{TP + FP} = \frac{m_{ij}}{m_j} = p_{ij}.$$

- **Recall**:

$$\frac{TP}{TP + FN} = \frac{m_{ij}}{m_i}.$$

- **F-measure**:

$$2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Remember that we have

		cluster	
		same	different
class	same	f_{11}	f_{10}
	different	f_{01}	f_{00}

More supervised measures

- **Rand statistic**:

$$R = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}.$$

- **Jaccard coefficient**:

$$J = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}.$$

- **Adjusted Rand Index**:

$$\text{ARI} = \frac{R(L,C) - \mathbb{E}\left[R(L,C)\right]}{\max\left\{R(L,C), \mathbb{E}\left[R(L,C)\right]\right\}}$$

with

$$\mathbb{E}\left[R(L,C)\right] = \frac{\pi(L)\pi(C)}{\frac{n(n-1)}{2}}$$

$$\max\left\{R(L,C)\right\} = \frac{1}{2}(\pi(L) - \pi(C))$$

with $\pi(C)$: number of objects pairs that belong to the same group C.

3 Fuzzy clustering

We generalize k-mean objective function

$$\text{SSE} = \sum_{j=1}^{k}\sum_{i=1}^{n} w_{ij}^{p}\text{dist}\left(\mathbf{x}_i, \mathbf{c}_j\right)^2.$$

So the procedure is

1. choose random weights w_{ij};
2. until centroids do not change:
   (a) $\mathbf{c}_j = \frac{\sum_{i=1}^{n} w_{ij}\mathbf{x}_i}{\sum_{i=1}^{n} w_{ij}}$ (updates centroids);
   (b) $w_{ij} = \frac{\left(\frac{1}{\text{dist}(\mathbf{x_i},\mathbf{c_j})^2}\right)^{\frac{1}{p-1}}}{\sum_{q=1}^{k}\left(\frac{1}{\text{dist}(\mathbf{x_i},\mathbf{c_q})^2}\right)^{\frac{1}{p-1}}}$ (updates weights).

4 Probabilistic clustering

Model the set of data points as arising from a mixture of distributions. Each cluster corresponds to one distribution from the mixture. Clusters are found by estimating the parameters of the statistical distributions using Expectation-maximization algorithm.

The probability that the j-th distribution with parameter ϑ_j is chosen to generate an object x_i belonging to one of K clusters is given by the weight w_j. So

$$\mathbb{P}(x|\Theta) = \sum_{j=1}^{K} w_j p_j(x|\vartheta_j).$$

If the m objects are generated independently then

$$\mathbb{P}(x|\Theta) = \prod_{i=1}^{m}\sum_{j=1}^{K} w_j p_j(x|\vartheta_j).$$

Probabilistic clustering

1. Estimation: for each point compute its probability under each distribution

$$\mathbb{P}(x_i|\vartheta_j).$$

2. Maximization: update the parameters.

Estimation is similar to k-means. Weights are probabilities, they are not raised to a power and are calculated using Bayes:

$$p(C_j|x_i) = \frac{p(x_i|C_j)p(C_j)}{\sum_{l=1}^{k} p(x_i|C_l)p(C_l)}.$$

We can estimate

$$p(C_j) = \frac{1}{m}\sum_{m}^{1} p(C_j|x_i).$$

Maximization step:

$$c_j = \frac{\sum_{i=1}^{m} x_i p(C_j|x_i)}{\sum_{i=1}^{m} p(C_j|x_i)}.$$

Complexity: $\mathcal{O}(d^2)$.

4.1 Density based algorithms

Grid based clustering

1. Define a set of grid cells.
2. Assign objects to the appropriate cells and compute the density of each cell.
3. Eliminate cells having a density below a specified threshold.
4. Form clusters from continuous groups of dense cells.

Complexity: $\mathcal{O}(n \log n)$.
The CLIQUE algorithm is based on the apriori principle:

$$\text{cluster in } k \implies \text{cluster in subsets of } k$$
$$\text{no cluster in } k \implies \text{no cluster in supersets of } k$$

CLIQUE

1. Find all dense areas in the one-dimensional spaces corresponding to each attribute. This is the set of dense one-dimensional cells.
2. $k \leftarrow 2$.
3. Until there are no candidate dense k-dimensional cells iterate:
   (a) Generate all candidate dense k-dimensional cells from dense $(k-1)$ dimensional cells.
   (b) Eliminate cells that have fewer than τ points.
   (c) $k \to k+1$.
4. Find clusters by taking the union of all adjacent high density cells.
5. Summarise each cluster using a small set of inequalities that describe the attribute ranges of the cells in the cluster.

Time complexity: exponential.

4.2 Graph-based clustering: chameleon

Three steps:

1. Sparsification of the proximity graph modifying the proximity matrix (can be recursive).
2. Partitioning into a large number of roughly equally sized groups (ise METIS).
3. Merging the large number of roughly equally sized groups (agglomerative hierarchical clustering based on dynamic modelling).

Example of merging schemes: MIN (minimum distance between clusters) or GROUP-AVERAGE (avg connectivity). Dynamic modeling: adapt to the characteristics of the data set.

Dynamic modeling measures

Relative interconnectivity:

$$\mathrm{RI}(C_i, C_j) = \frac{\mathrm{EC}(C_i, C_j)}{\frac{1}{2}\left(\mathrm{EC}(C_i) + \mathrm{EC}(C_j)\right)}.$$

Relative closeness:

$$\mathrm{RC}(C_i, C_j) = \frac{\overline{S}_{\mathrm{EC}}(C_i, C_j)}{\frac{m_i}{m_i+m_j}\overline{S}_{\mathrm{EC}}(C_i) + \frac{m_j}{m_i+m_j}\overline{S}_{\mathrm{EC}}(C_j)}$$

For example:

- combine pair of clusters which maximises $\mathrm{RI} \times \mathrm{RC}^{\alpha}$ and stop when there are K clusters left;
- combine clusters which have $\mathrm{RC} > t_c$ and $\mathrm{RC} > t_i$.

4.3 Spectral clustering

Given a proximity matrix $\mathbf{W}$ (weighted adjacency matrix) define a diagonal matrix $\mathbf{D}$ such that

$$D_{ij} = \begin{cases} \sum_l W_{il} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Compute the laplacian

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

We have that $\mathbf{L}$ is symmetric, positive semidefinite and all eigenvalues are positive for any vector $\mathbf{v}$:

$$\mathbf{v}^{\mathsf{T}}\mathbf{L}\mathbf{v} \geq 0.$$

Write $\mathbf{L}$ as eigenvalue decomposition:

$$\mathbf{L} = \mathbf{\Lambda}\mathbf{V}.$$

Spectral clustering

1. Create a sparsified similarity graph.
2. Compute the laplacian.
3. Create a matrix $\mathbf{V}$ from the first k eigenvalues of $\mathbf{L}$.
4. Apply k-means on $\mathbf{V}$.

4.4 SNN Clustering

SNN Algorithm

1. Compute similarity matrix.
2. Sparsify the similarity matrix (keep only k most similar neighbours).
3. Construct the SNN graph from the sparsified similarity matrix.
4. Find the SNN density of each point.
5. Find the core points using parameter *MinPts*.
6. For clusters from the core points.
7. Discard all noise points.
8. Assign all non noise, non core points to clusters.

Steps 4-8 are DBSCAN.

5 Anomaly detection

We can have:

- statistical approaches;
- distance-based approaches (from the object to its k-th nearest neighbor);
- density-based approaches: different density measures possible:

  $$\mathrm{density}(x, k) = \frac{1}{K_{\mathrm{dist}}}.$$

  Relative density:

  $$\begin{aligned} \mathrm{rel\_dens}(x, k) &= \frac{\sum_{i=1}^{k} \frac{\mathrm{density}(y_i, k)}{k}}{\mathrm{density}(x, k)} \\ &= \frac{K_{\mathrm{dist}}(x, k)}{\frac{\sum_{i=1}^{k} K_{\mathrm{dist}}(y_i, k)}{k}}. \end{aligned}$$

  You can use reachability distance

  $$R^K_{\mathrm{dist}}(x, y_1) = \max\left\{K_{\mathrm{dist}}(y_1), \mathrm{dist}(x, y_1)\right\}.$$

  LOF algorithm

  1. Compute the local reachability density of x as inverse of avg reachability distance:

     $$L^K_{\mathrm{density}}(x) = \frac{1}{\left(\frac{\sum_{y \in N_K(x)} R^K_{\mathrm{dist}}(X, y)}{|N_K(x)|}\right)}.$$

  2. Relate it to the one of the neighbours:

     $$\mathrm{LOF}_K(x) = \frac{\sum_{y \in N_K(x)} L^K_{\mathrm{density}}(y)}{|N_K(x)|} \frac{1}{L^K_{\mathrm{density}}(x)}$$

  Complexity is $\mathcal{O}(n^2)$;
- clustering-based approaches:
  - prototype based clusters;
  - density based clusters;
  - graph based clusters;
- reconstruction based approaches: use dimensionality reduction like PCA/autoencoders to project x to $\hat{x}$ and define reconstruction error as

  $$\|x - \hat{x}\|.$$

  Autoencoders minimize this. If this is high it is an anomaly;
- information theory approach: anomalies bring a decrease in information when they are deleted.

6 Classification

6.1 Decision trees

Many possible decision trees that can be built from the same dataset. The goal of the learning algorithm is to find the best tree among all possible trees.

Hunt's algorithm

1. Check if the tree is pure: if yes create a leaf.
2. Otherwise, split it according to the best condition.
3. Add a new node in the decision tree and associate it with that condition.
4. For each branch of the new node apply recursively the algorithm.

Continuous attributes need binning. We prefer nodes with purer class distributions. Measures of impurity of a node t ($p_c(t)$: frequency of class c at node t over total number of classes C):

- Gini index:
$$1 - \sum_{c=1}^{C} p_c(t)^2;$$
when a node is split the Gini index for the split is the weighted average of the Gini index evaluated on the resulting nodes:
$$\sum_{k=1}^{K} \frac{n_k}{n} \text{Gini}(t_k)$$
where n_k is the number of examples in the k-th split node;
- Entropy:
$$-\sum_{c=1}^{C} p_c(t) \log_2 p_c(t);$$
after splitting we have
$$\text{Entropy}(t_{parent}) - \sum_{k=1}^{K} \frac{n_k}{n} \text{Entropy}(t_k);$$
- Classification error:
$$1 - \max_c \{p_c(t)\}.$$

Choose the attribute test condition that yields the lowest impurity score. To avoid too many partition define the gain ratio

$$\text{GainRation} = \frac{\text{InformationGain}}{-\sum_{k=1}^{K} \frac{n_k}{n} \log_2 \frac{n_k}{n}}.$$

6.2 Rule-based classification

Uses a collection of if-then:

$$\text{Condition}_i = (A_1 \odot v_1) \wedge \ldots \wedge (A_k \odot v_k).$$

Coverage: fraction of instances that satisfy the rule's antecedent

$$\frac{|A|}{|D|}.$$

Accuracy: fraction of instances that satisfy the rule's antecedent and consequent

$$\frac{|A \cap y|}{|D|}.$$

Example: RIPPER.

6.3 Naive Bayes

We seek for the most probable hypothesis given the data and any initial knowledge about the prior probabilities. MAP hypothesis:

$$\begin{aligned} h_{\text{MAP}} &= \arg\max_{h \in \mathcal{H}} \mathbb{P}(h|D) \\ &= \arg\max_{h \in \mathcal{H}} \frac{\mathbb{P}(D|h)\mathbb{P}(h)}{\mathbb{P}(D)} \\ &= \arg\max_{h \in \mathcal{H}} \mathbb{P}(D|h)\mathbb{P}(h). \end{aligned}$$

If we assume that all hypotheses are equally likely a priori we can only consider the likelihood of the data (Maximum Likelihood hypothesis):

$$h_{\text{ML}} = \arg\max_{h \in \mathcal{H}} \mathbb{P}(D|h).$$

MDL principle:

$$\begin{aligned} h_{\text{MAP}} &= \arg\max_{h \in \mathcal{H}} \mathbb{P}(D|h)\mathbb{P}(h) \\ &= \arg\max_{h \in \mathcal{H}} \log_2 \mathbb{P}(D|h) + \log_2 \mathbb{P}(h) \\ &= \arg\min_{h \in \mathcal{H}} -\log_2 \mathbb{P}(D|h) - \log_2 \mathbb{P}(h). \end{aligned}$$

This is like minimizing the number of bits transmitted to encode the data and the hypothesis. Given a class $y_i \in \mathcal{Y}$ we get

$$\mathbb{P}(y_j|D) = \sum_{h \in \mathcal{H}} \mathbb{P}(y_j|h)\mathbb{P}(h|D)$$

so the Bayes Optimal Classifier is

$$y_{\text{Bayes}} = \arg\max_{y_j \in \mathcal{Y}} \sum_{h \in \mathcal{H}} \mathbb{P}(y_j|h)\mathbb{P}(h|D)$$

but we need to know the posterior of all hypotheses (often impossible).

Gibbs' algorithm

1. Choose an hypothesis $h \in \mathcal{H}$ at random according to the posterior probability over $\mathcal{H}$, which is $\mathbb{P}(h|D)$.
2. Use h to classify the new instance.

The error is at most twice the one of the OBC. The naive Bayes classifier makes the assumption that the attributes are conditionally independent given the class:

$$\mathbb{P}(a_1, \ldots, a_d|y_j) = \prod_{i=1}^{d} \mathbb{P}(a_i|y_j)$$

so the NBC is

$$y_{\text{NB}} = \arg\max_{y_j \in \mathcal{Y}} \mathbb{P}(y_j) \prod_{i=1}^{d} \mathbb{P}(a_i|y_i).$$

Probability are empyrical. Use laplace smoothing or m-estimate

$$\hat{\mathbb{P}}(a_i|y_i) = \frac{n_{a_i,y_i} + 1}{n_{y_j} + |A_i|}.$$

6.4 KNN

It is a lazy learner. Classifies based on the closest datapoint.

7 Model selection

Generalization error in the case of trees:

$$\text{err}_{\text{gen}}(T) = \text{err}(T) + \alpha \frac{k}{N_{\text{train}}}$$

MDL principle

Length of the shortest message that allows to reconstruct the labels.

$$\text{MDL}(D, M) = \text{Cost}(D|M) + \alpha \times \text{Cost}(M).$$

We can either do pre-pruning or post-pruning until generalization error cannot be improved.

Population Risk

Risk of a hypothesis $h \in \mathcal{H}$ as

$$R[h] = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[l(y, h(\boldsymbol{x}))\right]$$

Empirical risk

Given a dataset T whe have

$$\hat{R}_T[h] = \frac{1}{n}\sum_{(\boldsymbol{x},y)\in T} l(y, h(\boldsymbol{x})).$$

8 SRM

A given training set

$$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

is assumed to be drawn iid from a fixed and unknown distribution $p(x, y)$. We want to learn a function $h^\star$ such that $\forall(x_i, y_i) \in D,\ h^\star(x_i) = y_i$ (small population risk). We use some learning algorithm A which optimizes some performance measure l of the model (like a loss function). Expected risk minimization:

$$\begin{aligned} h^\star &= \arg\min_{h\in\mathcal{H}} R[h] \\ &= \arg\min_{h\in\mathcal{H}} \mathbb{E}_{(x,y)\sim p(x,y)}\left[l(h(x), y)\right] \end{aligned}$$

We use the empirical mean

$$h^\star = \arg\min_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} l(h(x_i), y_i).$$

So the generalization error is the sum of the error of the training set and the model complexity term

$$R[h] = \hat{R}_D[h] + \alpha \cdot \text{Complexity}(h).$$

VC dimension

The VC dimension of a hypothesis space $\mathcal{H}$ defined over an instance space X is the size of the largest finite subset of X shattered by $\mathcal{H}$.

$$\text{VC}(\mathcal{H}) = \max_{S\subset X} |S| : S \text{ is shattered by } \mathcal{H}.$$

The upper bound of the true risk is

$$R[h^\star] \le \hat{R}_D[h^\star] + \mathcal{O}\left(\sqrt{\frac{\text{VC}(\mathcal{H})}{n}\log\left(\frac{n}{VC(\mathcal{H})}\right) - \frac{\log(\delta)}{n}}\right)$$

9 SVM

Functional margin:

$$\mu(x_i) = y_i(w \cdot x_i - t) = y_i f(x)$$

Geometric margin:

$$\mu = \frac{1+t}{||\boldsymbol{w}||} - \frac{t-1}{||\boldsymbol{w}||} = \frac{2}{||\boldsymbol{w}||}$$

SVM optimization

$$\begin{aligned} &\min_{\boldsymbol{w},t} \frac{1}{2}||\boldsymbol{w}||^2 \\ &\text{s.t. } y_9(\boldsymbol{w}\cdot\boldsymbol{x}_i - t) \ge 1; \quad \forall i : 1 \le i \le n. \end{aligned}$$

Dual problem:

$$\begin{aligned} &\min_{\alpha} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i\boldsymbol{x}_j + \sum_{i=1}^{n}\alpha_i \\ &\text{s.t. } \alpha_i > 0 \\ &\sum_{i=1}^{n} y_i\alpha_i = 0 \end{aligned}$$

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Y}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{\alpha} \\ &\text{s.t. } \boldsymbol{\alpha} \succeq 0 \\ &\boldsymbol{y}^T\boldsymbol{\alpha} = 0. \end{aligned}$$

Written in matrix form we get

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Y}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{\alpha} \\ &\text{s.t. } \boldsymbol{\alpha} \succeq 0 \\ &\boldsymbol{y}^T\boldsymbol{\alpha} = 0. \end{aligned}$$

Introduce one slack variable for each training sample:

$$\begin{aligned} &\min_{\boldsymbol{w},t,\xi} \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{s.t.} y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i - t) \ge 1 - \xi_i \qquad \forall 1 \le i \le n \\ &\xi_i \ge 0 \qquad \forall 1 \le i \le n. \end{aligned}$$

Given a feature map ϕ we have

$$\boldsymbol{w} = \sum_{i=1}^{n}\alpha_i y_i \phi(\boldsymbol{x}_i)$$

and the decision function is

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{w}\cdot\phi(\boldsymbol{x}) - t \\ &= \sum_{i=1}^{n}\alpha_i y_i \phi(\boldsymbol{x}_i)\cdot\phi(\boldsymbol{x}) - t \\ &= \sum_{i=1}^{n}\alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) - t. \end{aligned}$$

SVM with kernels

The problem becomes

$$\begin{aligned} &\min_{\alpha} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) - \sum_{i=1}^{n}\alpha_i \\ &\text{s.t. } 0 \le \alpha_i \le C \\ &\sum_{i=1}^{n}\alpha_i y_i \ge 0. \end{aligned}$$

Polynomial kernel:

$$\kappa(\boldsymbol{x}, \boldsymbol{x'}) = (\boldsymbol{x}\cdot\boldsymbol{x'} + 1)^d$$

RBF (Gaussian) Kernel

$$\kappa(\boldsymbol{x}, \boldsymbol{x'}) = \exp\left\{-\frac{1}{2\sigma^2}||\boldsymbol{x} - \boldsymbol{x'}||^2\right\}$$

Mercer's conditions

Let us consider a function k and the gram matrix $\mathbf{M} = M_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. If there exists ϕ such that $\kappa(\boldsymbol{x}, \boldsymbol{x'}) = \langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x'})\rangle$ then $\mathbf{M}$ must be symmetric and positive semidefinite.

So designing a new kernel implies:

1. prove symmetry and positive definiteness of $\mathbf{M}$;
2. find an explicit mapping ϕ such that $\kappa(\boldsymbol{x}, \boldsymbol{x'}) = \langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x'})\rangle$;
3. derive the kernel from other valid ones using closure properties.

Representer Theorem

Let $\Omega : \mathbb{R} \to \mathbb{R}$ be any non-decreasing function of its arguments and consider the regularized risk

$$\hat{R}_\Omega[\boldsymbol{w}] = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \langle\boldsymbol{w}, \boldsymbol{x}_i\rangle\right) + \lambda\Omega\left(||\boldsymbol{w}||_2\right).$$

Then there exists some $\boldsymbol{\alpha} \in \mathbb{R}^n$ such that the optimal solution $\boldsymbol{w}^\star$ can be written as a linear combination of the input vector:

$$\boldsymbol{w}^\star = \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i$$

10 Neural networks

Rosenblatt's perceptron:

$$\text{output} = \begin{cases} 0 & \text{if } w \times x + b \leq 0 \\ 1 & \text{otherwise.} \end{cases}$$

The perceptron learning algorithm makes at most $\frac{R^2}{\gamma^2}$ updates.
Proof: for $w_1 = 0$ given a misclassified point x_j we have

$$\begin{aligned} w_{k+1} \cdot w^\star &= (w_k + y_j x_j) \cdot w^\star \\ &= w_k \cdot w^\star + y_j (x_j \cdot w^\star) \\ &> w_k \cdot w^\star + \gamma \\ &> k\gamma. \end{aligned}$$

Since

$$w_{k+1} \cdot w^\star \leq \|w_{k+1}\|\|w^\star\| = \|w_{k+1}\|$$

then

$$\|w_{k+1}\| < k\gamma.$$

Since coordinates are bounded

$$\begin{aligned} \|w_{k+1}\|^2 &= \|w_k + y_j x_j\|^2 \\ &= \|w_k\|^2 + \|y_j x_j\|^2 + 2(w_k \cdot x_j) y_j \\ &= \|w_k\|^2 + \|x_j\|^2 + 2(w_k \cdot x_j) y_j \\ &\leq \|w_k\|^2 + \|x_j\|^2 \\ &\leq \|w_k\|^2 + R^2 \\ &\leq kR^2 \end{aligned}$$

Perceptron representer theorem

During a run of the perceptron algorithm the weight vector $\boldsymbol{w}$ is always in the span of training data:

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i)$$

Cross-entropy:

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)]$$

We can show

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

and therefore

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z) - y)$$
$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_x (\sigma(z) - y).$$

Soft max activation:

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}.$$

Cost function with regularization term:

$$\mathcal{L}_{\text{reg}}(X, y; \vartheta) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f_\vartheta(x_i)) + \frac{\lambda}{n}\sum_{W \in \vartheta} \|W\|_F^2$$

10.1 4 fundamental BP equations

1. $\delta^L = \nabla_a C \odot \sigma'(z^L)$
2. $\delta^l = ((\mathbf{W}^{l+1})^T \delta^{l+1}) \circ \sigma'(z^l)$
3. $\frac{\partial C}{\partial b_j^l} = \delta_j^l$
4. $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l$

11 Privacy

11.1 ε-differential privacy

We have

- $\mathcal{Q} = q : \Omega \to \mathcal{R}$ the set of all queries
- $\mathcal{D}, \mathcal{D}' \in \Omega$ any pair of adjacent datasets (differ only by one record).

A mechanism $\mathcal{M} : \mathcal{Q} \times \Omega \to \mathcal{R}$ preserves ε-differential privacy if for any query and any response

$$e^{-\varepsilon} \leq \frac{\mathbb{P}(M(q, \mathcal{D}) = r)}{\mathbb{P}(\mathcal{M}(q, \mathcal{D}) = r)} \leq e^{\varepsilon}.$$

Global sensitivity: maximum variation of q on any two adjacent datasets

$$GS_n(q) = \max_{\mathcal{D} \sim \mathcal{D}'} \|q(\mathcal{D}) - q(\mathcal{D}')\|$$