

ML in computer vision

Júlia Križanová, Dominik Farhan

1 Section A - a very detailed answer is required

1.1 Describe SVM and its training procedure. What is a kernel and what is it used for in SVM?

SVMs – Support Vector Machines – are binary linear classifiers.

We would like to solve

$$\arg \max_w \frac{2}{\|w\|}$$

with the conditions:

$$(w^T x_i + b)k_i \geq 1$$

where

$$k_i = 1 \text{ or } -1 \text{ based on the class}$$

which is equivalent to solving the following minimization

$$\arg \min_w \frac{1}{2} \|w\|^2$$

with the same constraints. We can use Lagrange method. Start with the Lagrangian:

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^N \alpha_i (w^T x_i + b)k_i + \sum_{i=1}^N \alpha_i$$
$$\alpha_i \geq 0$$

Now we observe that weights can be expressed as a some linear combination of x_i multiplied by k_i and α_i – provided that α_i is non-negative and the $\sum_i \alpha_i k_i = 0$. We substitute this to the Lagrangian and obtain

$$L(w, b, \alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j k_i k_j x_i^T x_j$$

with conditions

$$\alpha_i \geq 0,$$
$$\sum_{i=1}^N \alpha_i k_i = 0.$$

We can solve this with quadratic programming.

Now we note, that $\alpha_i = 0$ for most i . This x_i don't give any information to the weights so we can ignore them. For classification only important examples are those where $\alpha_i > 0$. We call these examples support vectors because they determine (support) the boundary.

We can find the bias as:

$$b = \frac{1}{|\text{support vectors}|} \sum_{x_j \in \text{support vectors}} (k_j - \sum_{x_i \in \text{support vectors}} \alpha_i k_i x_i^T x_j)$$

There are two types of non-linearly-separable data. Nearly separable and not separable. Nearly separable data can be fitted with soft margin SVM which relaxes the condition of linear separability by introducing hinge loss. This loss is expressed as:

$$\max(0, 1 - y_i(w^T x - b))$$

which is zero for correctly classified and for incorrectly it gives value proportional to the distance to the margin. SVMs have C regularization parameter which states how much should hinge loss attribute to the total loss. Small value of C will missclassify more examples but the hyperplane margin will be greater. For high C the converse will happen and less examples will be missclassified.

To solve non-separable data, we introduce non-linear kernels. Often in higher dimensions data are linearly separable even if in the original space they are not. The idea is to observe that everywhere in the SVM formulation features maps exists only as pairs

$$\varphi(x_i)^T \varphi(x_j),$$

so we don't need to compute φ but only their dot products. A symmetric function $K(x, y)$ can be expressed as dot product if for some φ , matrix with entries $K(x, y)$ is positive semidefinite. Most popular kernel choices are:

- polynomial $(ax^T y)^p$
- Gaussian $e^{-\frac{\|x-y\|^2}{2\sigma^2}}$

1.2 Describe forward neural networks with error propagation and the procedure of their training.

Inspired by brain.

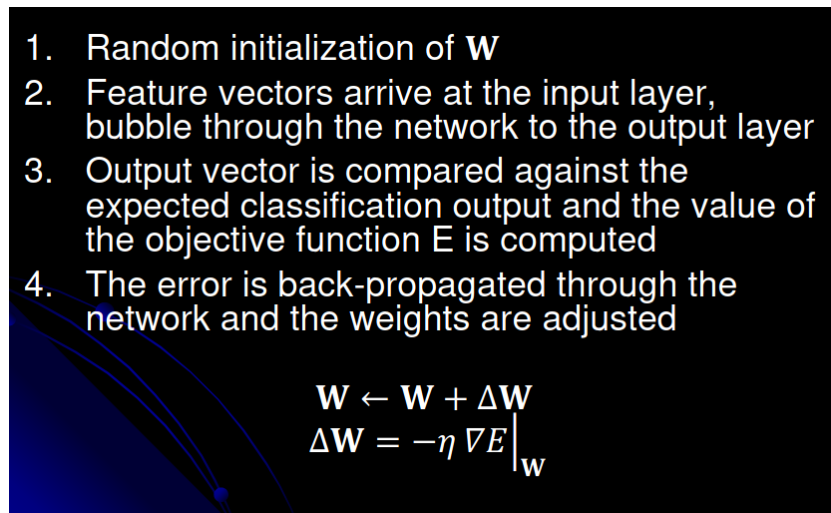
Directed graphs consisting of layers. Outputs from the previous layer are weighted, summed and increased by bias, then non-linear function is applied. Non-linearity is important, otherwise adding layers wouldn't result in any increase of capacity of the model because combination of linear maps is still linear and therefore expressible with a single linear transformation.

Universal Approximation Theorem: informally, big enough neural network with one hidden layer and non-linear activation can approximate any continuous function.

1.2.1 Activation functions

- Sigmoid $\frac{1}{1+e^{-x}}$
- Logistic $\frac{1}{1+e^{-\alpha x}}$
- Hyperbolic tangent $2 \cdot \sigma(x) - 1$
- Hyperbolic tangent with β : $\frac{e^{\beta x} - e^{-\beta x}}{e^{\beta x} + e^{-\beta x}}$
- ReLU
- Leaky ReLU
- ELU
- GELU

1.2.2 Learning



1. Random initialization of W
2. Feature vectors arrive at the input layer, bubble through the network to the output layer
3. Output vector is compared against the expected classification output and the value of the objective function E is computed
4. The error is back-propagated through the network and the weights are adjusted

$$W \leftarrow W + \Delta W$$
$$\Delta W = -\eta \nabla E \Big|_W$$

1.2.3 Backpropagation

Efficient algorithm for computing partial derivatives with respect to parameters of our network.

1.2.4 Learning rate

Backprop needs learning rate – how great should the update in each step be. Small lr can lead to very slow learning, high lr can overshoot the optimum and never find a feasible solution. Heuristics for the learning rate

- multiple learning rate by a constant in each step/every k steps
- when the error significantly increases, discard the update and decrease learning rate
- allow small increase in error to leave local minimum
- if errors are getting lower, decrease learning rate because we are heading towards minimum

1.2.5 Other learning algorithms

- Newton's method. Uses Hessian
- Quasi-Newton method. Approximation of Hessian
- Conjugate gradient, new gradient and the previous search direction
- Levenberg-Marquardt. Hessian approximated by Jacobian

1.3 Describe the self-organizing maps and the procedure of their training.

SOMs are networks where the neurons form a lattice designed to visualize similarity in more dimensional data. Usually lattice is 1D or 2D. In the network the input data is connected to all possible neurons in the lattice.

1.3.1 2 spaces of SOM:

SOM lattice topological structure:

Neurons of the network have their own neighbors.

Weight space:

Each neuron has a weight and the dimensionality of the weight is the same as the input space.

Here, the neurons can be far away from each other when we initialize the weights randomly. During the training weights use the topological structure to get closer together and fit the data of the input training set.

1.3.2 SOM learning algorithm:

It is a winner takes all algorithm, because we identify closest node and update this node.

1. Randomly assign the weights to the neurons
2. Select randomly one input vector
3. We compare the weights of each neuron with the input vector and we identify the closest node (using Euclidian distance):

$$i^* = \underset{i}{\operatorname{argmin}} \|x - w_i\|$$

4. The weight of the node is updated (it is moved to be closer to the input vector)
5. The weights of the adjacent nodes are also updated, but not to the same degree. The neighborhood is identified by the neighborhood function:

$$w_i(t+1) = w_i(t) + \eta(t)O(i, i^*, t)\|x - w_i(t)\|$$

where $\eta(t)$ = learning rate
 $O(i, i^*, t)$ = neighborhood specification

6. Reduce the intensity of the update progressively. In the beginning the network is in the cooperation phase where the neurons cooperate in the movement, so not only the winner is moved, but also his neighbors and after reaching some number of the iterations, the network moves to the adaptation phase, where only the individual neuron moves towards the input vector.
7. The training is done for the set number of iterations
8. At the end the nodes of the network fit the input data

1.3.3 Neighborhood specification:

The neighborhood specification can be done by for example a Gaussian function, given like:

$$O(i, i^*, t) = e^{-\frac{\|r_{i^*}(t) - r_i(t)\|^2}{2\sigma^2(t)}}$$

where $\sigma(t) = \sigma_0 e^{\frac{-t}{T_{max}}}$

Or it can be given by the distance in a different metrics.

We can use the SOM to identify the cluster in the data

1.4 Describe the linear classifier and its training procedure.

In order to work with a linear classifier we set two assumptions:

1. there are only two classes of data
2. Classes are linearly separable (\exists hyperplane that separates data from the 2 classes so that they lie in different subspaces)

Hyperplane is a decision boundary of the classifier with the following equation:

$$w^T x + b = 0$$

where w is the normal vector of the hyperplane and b is the offset

The points that lie on the hyperplane have the value $w^T x + b = 0$ and points lying outside the hyperplane are either positive or negative and this will help us to classify the points:

$$f(x) \geq 0 \text{ for } x \in \omega_1$$

$$f(x) < 0 \text{ for } x \in \omega_2$$

In order to simplify the conditions we will do some changes in the notation:

$$u = [b, w^T]^T \text{ and } y = [1, x^T]^T$$

Then the conditions can be written as $f(x) = u^T \cdot y$.

Another trick: We set a new variable z_i lying between $-y$ and y . \Rightarrow All points from the classes lie on the positive side of the hyperplane. The solution doesn't change but the conditions become just one condition that $u^T x \cdot z_i > 0$.

1.4.1 Gradient method:

Method that optimizes the chosen objective function $O(u)$. We choose the objective function which is minimal in the solution of the problem.

Idea of the gradient method:

If a function $F(x)$ is defined and differentiable in a neighborhood of a point y , then $F(x)$ decreases the most and the fastest when we go in the direction of the negative gradient in the point y .

It is an iterative method which starts with a random vector u_1 and computes the gradient of the objective function in this point and then updates the value of u by choosing a vector from the neighborhood of u_1 in the direction of the negative gradient:

$$u_{i+1} = u_i - \eta(i) \nabla O|_{u_i}$$

where $\eta(i)$ is the learning rate in step i

We can find the optimal value of η by doing the Taylor expansion of the objective function:

$$O(a) \approx O(u_i) + \nabla O^T(a - u_i) + 1/2(a - u_i)^T \cdot H \cdot (a - u_i)$$

where H is the Hessian matrix of second partial derivative of the function O .

minimized for $\eta(i) = \frac{\|\nabla O\|^2}{\nabla O^T \cdot H \cdot \nabla O}$

We can choose to minimize the number of misclassified examples meaning that if we apply the vector u on all points z , the ones which are negative are misclassified, because we constructed the point z such that they all lie in the positive part of the space. If we compute the sum of the results of the misclassified examples, we can set this as our obj function so we would like to minimize this sum. This function is called piece-wise linear perceptron function:

$$O_p = \sum_{z \in Z} -u^T z$$

Z is a set of misclassified examples

1.4.2 Batch processing vs sequential learning:

Batch processing:

We can compute the value of the function by doing it in batch processing meaning that we classify all objects and compute the whole sum:

$$\nabla O_p = \sum_{z \in Z} -z$$
$$u_{i+1} = u_i + \eta(i) \sum_{z \in Z} z$$

Sequential learning:

We classify each example one by one and then the algorithm looks like this:

We start with a vector set of all ones, and we apply this vector on each sample. When we hit a sample that is misclassified, we update the solution by adding this sample to the vector u , and we do it until all samples are classified correctly, meaning $u^T z_j$ is greater or equal to zero for all examples.

```
set  $u_0 = 1, k = 0, i = 0$ 
repeat
  if  $z_k$  is misclassified
     $u_{i+1} = u_i + z_k$ 
     $i = i + 1$ 
  endif
   $k = (k + 1) \bmod N$ 
until convergence ( $u^T z_j \geq 0$  for all  $z_j$ )
```

1.4.3 Objective function 2

we can work with quadratic function $O_Q = \sum_{z \in Z} (u^T z)^2$

We set a small epsilon that will assure that the convergence will not be reached on the boundary. We can work with the quadratic function where the gradient is continuous so we don't have the sum of the values but sum of the quadratic values, we have then the continuous gradient.

1.5 Describe the K-means algorithm, the choice of K and possible alternatives.

K-means clustering is a non hierarchical method, that creates k clusters. Optimality of the clusters is given by the minimal intra-cluster scatter:

$$W = \sum_{k=1}^K \sum_{x_i \in C_k} \sum_{x_j \in C_k} \|x_i - x_j\|^2 = \sum_{k=1}^K 2N_k \sum_{x_i \in C_k} \|x_i - m_k\|^2 = \sum_{k=1}^K WSS_k$$

where m_k is centroid of cluster k N_k is number of points in cluster k

1.5.1 K-Means Algorithm:

Initialization:

Randomly place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.

Algorithm follows:

Assign objects to the group that has the closest centroid:

$$C(x) = \underset{k}{\operatorname{argmin}} \|x - m_k\|^2$$

When all objects have been assigned, recalculate the positions of the K centroids:

$$m_k = \frac{\sum_{x:C(x)=k} x}{N_k}, k = 1, \dots, K$$

Repeat until the stopping criteria is met. Can be that the MSE is less than some threshold or that the centroids do not change in the consecutive steps.

1.5.2 K-Means Clustering:

K-Means clustering is guaranteed to converge. It won't always achieve the global optimum, but it always achieves a local optimum.

It is quite sensitive to noise and outliers in the data, and also it is sensitive to the initial assignment of the centroids. We said that the initial position of the centroid is random, so K-means is not a deterministic algorithm and so the clusters computed by each run of the algorithms do not have to be consistent.

1.5.3 Choice of K:

We have to be careful about the number of clusters we want to compute. There are many methods that show us which K is optimal:

1. elbow point

We compute the total WSS(= within-cluster sum of squares)

2. Gap value method

Gap value is given by: $Gap_N(K) = E_N^* W_K - \log W_K$

$$W_K = \sum_{k=1}^K \frac{1}{2N_k} WSS_k$$

3. The silhouette value

tells us how similar is a point to the points of its own cluster compared to points of other clusters.

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Where $a(i)$ is the average distance of point i from the points of its own cluster
 $b(i)$ average distance of the closest cluster

If the average value from the own neighbors is less than the average distance from other cluster, then the point i is happy within his cluster. If this value is negative, then another cluster is closer to this point i , than its own cluster.

1.5.4 Possible alternatives:

There are several alternatives for K-means:

1. K-medoids

Instead of means we take the medoids of each cluster

Medoid is a point which belong to the original training points. Mean is a point which doesn't have to belong to the training set

2. K-medians

Instead of means we use medians. Object might not be in the set.

3. Using fuzzy clustering: Fuzzy C-means:

We assign a probability or membership function to each point, so each point can be assigned to any cluster with some weight.

Soft membership function:

$$\sum_{k=1}^K \sum_{x_i \in C_k} w_{ik}^f \|x_i - m_k\|^2$$

4. Buckshot

We combine an agglomerative clustering method with the k-means clustering, because as we said k-means is quite sensitive to the initial position of the centroids, so we can run an agglomerative clustering method on a sample of the data, to get the initial position.

5. Bisecting K-means method

We are doing the divisive clustering, and we use 2 means in each step.

```
for i=1 to K-1 do
{
  pick a leaf cluster C to split
  For J=1 to ITER do
  {
    Use K-means to split C into two sub-clusters, C1 and C2
  }
  Choose the best of the above splits and make it permanent
}
```

2 Section B – a moderately detailed answer is required

2.1 Describe the FLDA method in detail.

Fisher's linear discriminant analysis, FLDA, is a special case of a more general method, LDA, when there are only 2 classes computations.

LDA is the direct extension of Fisher's idea on situation of any number of classes and uses matrix algebra devices (such as eigendecomposition) to compute it. It is a supervised method that takes class labels into account when reducing the number of dimensions. Its goal is to find a feature subspace that best optimizes class separability by investigating intraclass and interclass relations.

We have D-dimensional feature vectors $\{x_1, \dots, x_n\}$ and we have C classes where the number of objects in each class $|\omega_j| = N_j$

If we look at the mean of the class, it's the mean of all objects in that class:

$$\bar{x}_j = \frac{1}{|\omega_j|} \sum_{x \in \omega_j} x = \frac{1}{N_j} \sum_{x \in \omega_j} x$$

Total mean of all objects and all classes is:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

where N is the number of all objects, and then we investigate the sum of the objects.

Total scatter in the data

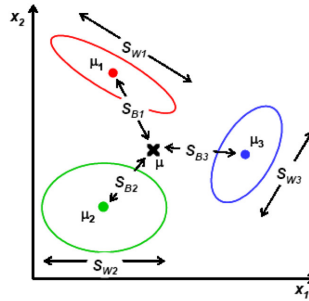
$$S = \sum_{i=1}^N (x_i - \bar{x})(x_i - \bar{x})^T.$$

$$S = S_M + S_V$$

Interclass scatter

$$S_M = \sum_{j=1}^C N_j (\bar{x}_j - \bar{x})(\bar{x}_j - \bar{x})^T$$

$$S_V = \sum_{j=1}^C \sum_{x \in \omega_j} (x - \bar{x}_j)(x - \bar{x}_j)^T = \sum_{j=1}^C S_j$$



Intraclass scatter

$$S_j = \sum_{x \in \omega_j} (x - \bar{x}_j)(x - \bar{x}_j)^T.$$

In the illustration we have 3 classes, where color dots are the class means and the black cross is the overall mean of the data. We want to separate the classes, so we want the distances from the black cross to the color dots to be as big as possible.

The second condition is that we want in each class, the variance(= the scatter) to be as small as possible, meaning we want the classes to be as tight together as possible.

S_M interclass scatter \rightarrow scatter of the means of the classes

S_V intraclass scatter \rightarrow sum of all scatters in all classes

If we project to the direction w , what will happen to the scatter? Instead of original data we will use the projected data and we will show that the total projected scatter is the sum of projected interclass(Q_M) and intraclass(Q_V). After calculations, total scatter Q equals to the direction w transposed * the original scatter S * the direction w .

The Fisher's criterion which we want to maximize is then $J(w)$. Why is it so? Because we want the interclass scatter to be as big as possible so it's in the numerator, and the intraclass we want it to be as small as possible so it is as small as possible.

Projection to w

$$x'_i = w^T x_i \quad \bar{x}'_j = w^T \bar{x}_j$$

$$Q = Q_M + Q_V$$

$$Q = \sum_{i=1}^N (w^T x_i - w^T \bar{x})(w^T x_i - w^T \bar{x})^T = w^T S w$$

$$Q_M = \sum_{j=1}^C N_j (w^T \bar{x}_j - w^T \bar{x})(w^T \bar{x}_j - w^T \bar{x})^T = w^T S_M w$$

$$Q_V = \sum_{j=1}^C \sum_{x \in \omega_j} (w^T x - w^T \bar{x}_j)(w^T x - w^T \bar{x}_j)^T = w^T S_V w.$$

$$J(w) = \frac{w^T S_M w}{w^T S_V w}$$

$$S = S_M + S_V$$

$$S_M = \sum_{j=1}^C N_j (\bar{x}_j - \bar{x})(\bar{x}_j - \bar{x})^T$$

$$S_V = \sum_{j=1}^C \sum_{x \in \omega_j} (x - \bar{x}_j)(x - \bar{x}_j)^T = \sum_{j=1}^C S_j$$

In case of 2 classes, the FLDA, is looking for one vector w which maximizes this fishers criterion. With 2 classes we are looking for 1 direction to project. To find the maximum, we take the derivative of the object function. In case of C classes we are looking for $C - 1$ directions and so w are no longer vectors but matrices W .

2.2 Describe the ICA method in detail.

Method similar to PCA, however here the components are not orthogonal. By using these components we are able to separate the sources of the composed data set. That means that by using ICA we try to transform set of vectors into maximally independent set. The example of this is a so called Cocktail Party where there are multiple microphones collecting different combinations of sounds and our goal is to separate the original voice of each person on the party.

In ICA the vector is represented as linear combination of non-Gaussian random variables (=independent components)

Key assumptions of ICA are that independent components are:

- statistically independent (probability of observing all signals at once, equals to the probabilities of observing them separately) $p(s_1, s_2, \dots, s_n) = p(s_1)p(s_2) \dots p(s_n)$
 - non-Gaussian
 - data is uncorrelated $E(s_i) = 0$
 - $Var(s_i) = 1$
- \Rightarrow the covariants matrix is equal to the unit matrix $E\{SS^T\}=I$

We would like our data to follow these assumptions, so we have to preprocess them:

- (1) We center the data, meaning that we subtract the mean $X' = X - \bar{X}$
- (2) We whiten the data, we put covariant matrix sigma = I $\tilde{X} = BX'$ such that $\Sigma_{\tilde{X}} = I$
- (3) We find the transformation matrix B by using the eigenvalues approach:

$$X'X'^T = \Sigma = VSV^T$$

$$d_{ij} = s_{ij}^{-1/2}$$

$$\tilde{X} = VDV^T X'$$

In the separation step we try to find the vectors that maximize the non-Gaussianity, because we suppose that the sources are non-Gaussian. The W should maximize the non-Gaussianity Y: $Y = W\tilde{X}$
We can measure the non-Gaussianity by using skewness and kurtosis, or we can use the negentropy:

$$J(y) = H(y_G) - H(y)$$

ICA algorithm:

- w - matrix, that maximizes non-gaussianity
- $J(w^T x)(E(G(w^T x))) - E(G(y_G))^2$

- constraint $\|w\|^2 = 1$
- We use Lagrangian method: $L = (G(w^T x)) - \lambda(w^T w - 1)$
- Derivation: $L' = E(x \cdot g(w^T x)) - \lambda w \equiv 0$
- If we note this expression as a function of w , we can solve this problem using the Newton root finding method (method where we iteratively update the values w by looking at the function and the derivative of the function f until we find the root.)

There are some properties of the data that cannot be computed using ICA algorithm. For example:

- Amplitudes of separated signals cannot be determined.
- There is a sign ambiguity associated with separated signals.
- Most problematic in the case of dimensionality reduction is that the order of separated signals cannot be determined.

2.3 Describe in detail the K nearest neighbors' method.

Use targets of the most similar training data for prediction. For this we need to store the whole training set so it's rather heavy for memory. Requires three things

- The training set
- Some distance metric
- K

Some popular metrics are

- Euclidean
- Manhattan
- Chebychev $\max |x_i - y_i|$
- Mahalanobis $d(x_i, x) = \sqrt{(x_i - x)^T A (x_i - x)}$, A is matrix. If identity then we get Euclidean, other choices are covariance matrix, or weights of features on the diagonal.

KNN can use weights. Some simple weights are softmax – proportional to the softmax of negative distances, inverse distance, relevance of features, mutual information, chi square.

Choosing optimal K is not easy. We need to experiment with our validation set. Small K gives finer structure and is prone to noise, large K decreases sensitivity to noise but can include too distant training data in the prediction.

For the KNN to work it needs numerical data, so we need to convert categorical to numerical.

For fast classification (so we don't need to compare against all training data points) kd-trees or ball trees are used. We also use condensing or editing. Condensing removes data that are not needed to form the decision boundary, or they change it only slightly. Editing removes points that do not agree with the majority of their NNs – removes outliers.

If we need to do regression task we can use (weighted) average of neighbors.

2.3.1 Discriminant adaptive NN

Using discriminant adaptive analysis we can determine better metric for each neighborhood and modify it. This increases performance especially for tasks in space with a high number of dimensions.

2.4 What is the difference between the two basic approaches to dimension reduction (wrapper and filter), what evaluation measures are used for them?

Filter does not depend on the classifier and uses some fitness function to evaluate the subset of features.

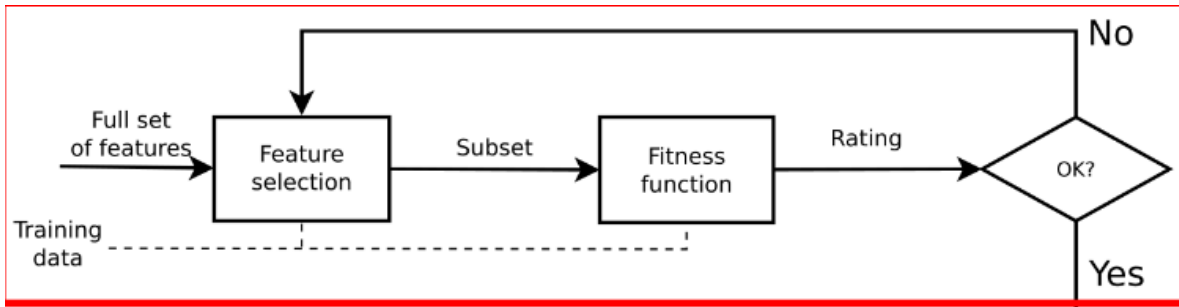


Figure 1: Filter pipeline

Wrapper depends on the classifier and uses the loss of the classifier to choose the best subset. This should lead to better results because it directly optimizes the classifier however it is more computationally expensive.

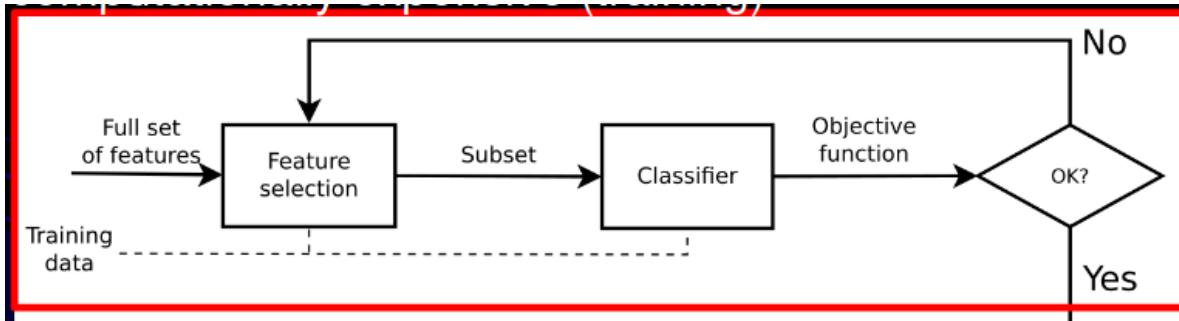


Figure 2: Wrapper pipeline

Measures:

- Consistency: Feature subset must classify consistently within the set. When some feature is present in multiple classes it is inconsistent.
- Independence: Features that are highly correlated provide the same information, so it is not important to have all of them. $J(\tilde{X}) = \frac{kr_{cf}^2}{\sqrt{k+k(k-1)r_{ff}^2}}$, where r_{cf} is a mean correlation coefficient of feature-class and r_{ff} of feature-feature.

- $NLCC = \sqrt{1 - e^{-2I(X;Y)}}$ (Non-Linear Correlation Coeff)

- $pearson = \frac{COV(X,Y)}{\sqrt{VAR(X)VAR(Y)}}$.

- Information theoretical measures

- Hartley's Information Measure

– Shannon’s Information Measure. Entropy is given as the expected value of information, for discrete case $H(A) = -\sum_{a \in \Omega} P(A = a) \log_2(P(A = a))$. We use entropy to compute mutual information $I(X; Y)$ which tells us how much information on variable gives about another. We can use it to evaluate subset of features as $J(\tilde{X}) = I(\tilde{X}; y)$.

- Interclass distance We want classes that put features far away. $J(\tilde{X}) = \sum_{i=1}^C P(\omega_i) \sum_{j=i+1}^C P(\omega_j) D_{\tilde{X}}(\omega_i, \omega_j)$, where D denotes the average distance computed from all pairs of objects with class i and j .

2.5 What is the difference between the two basic approaches to dimension reduction (wrapper and filter), what procedures are used to construct the subsets?

Filter does not depend on the classifier and uses some fitness function to evaluate the subset of features.

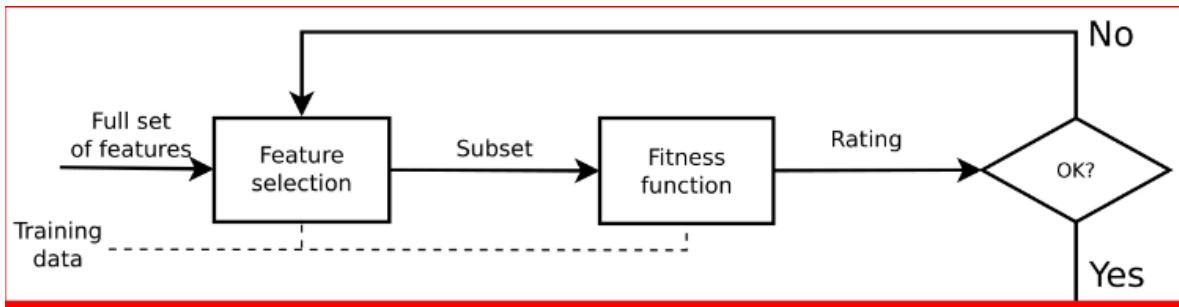


Figure 3: Filter pipeline

Wrapper depends on the classifier and uses the loss of the classifier to choose the best subset. This should lead to better results because it directly optimized the classifier however it is more computationally expensive.

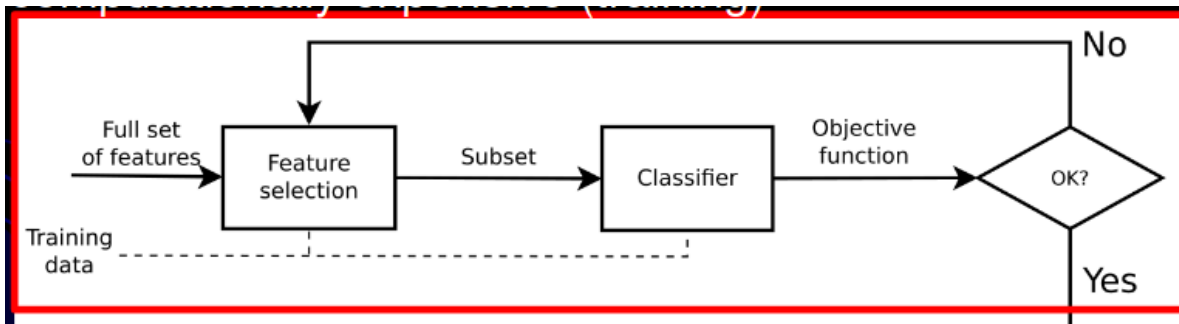


Figure 4: Wrapper pipeline

2.5.1 One-step feature selection

We evaluate each feature separately and then add K features with the highest score

2.5.2 Sequential forward selection

We add features sequentially. We evaluate set of already chosen features augmented with the considered feature. In each iteration we add the best performing feature to the set of already chosen features.

2.5.3 One step backward elimination

We start with full set. For each feature we compute its score and then remove the K lowest-score features from the set.

2.5.4 Sequential backward elimination

Similar to sequential forward selection. In each iteration we remove the feature x that decreases the fitness of $X - x$ least.

2.5.5 Combined selection and elimination

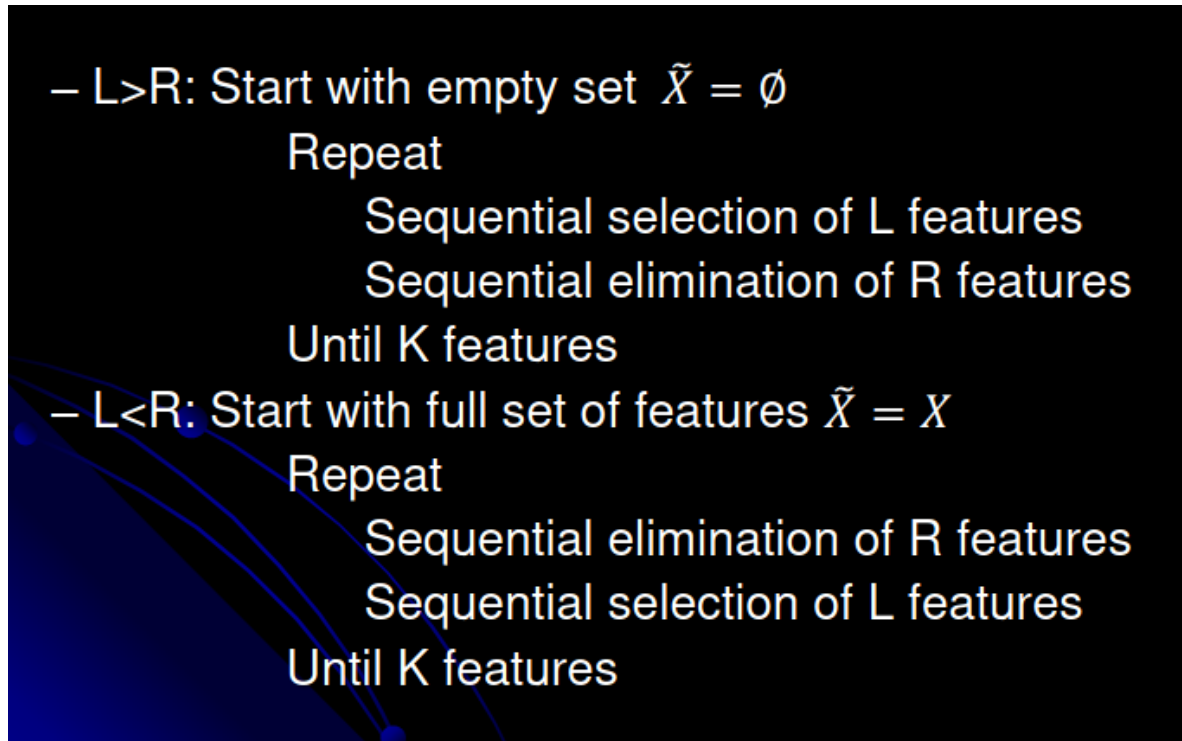


Figure 5: Combined selection and elimination pseudocode

2.5.6 Other methods

There are also other methods like genetic algorithms or simulated annealing.

2.6 Describe the PCA method in detail. What is the difference between PCA and ICA?

Describe the PCA method in detail.

PCA or principal component analysis is a method that enables us to transform the set of features into less dimensional space. The PCA method is an unsupervised method, where we don't take into account the information about the class of the data, but we want to minimize the information loss by diminishing the dimensionality of the space. As such, PCA is not a dimensionality reduction method. It takes the axis of the space, translates and rotates the axis such that the direction of the maximal variance in the data is found.

What is the difference between PCA and ICA?

PCA – vector is represented as linear combination of orthogonal vectors

ICA – vector represented as linear combination of non-Gaussian random variables (=independent components)

PCA – typically compresses information

ICA – separates information (separates the voices of several speakers into independent components)

PCA – finds uncorrelated components ICA – finds independent components.

PCA – optimizes the covariance matrix of the data which represents second-order statistics.

ICA – optimizes higher-order statistics such as kurtosis.

2.7 Describe the PCA method in detail. How do we use it to reduce the dimension?

Describe the PCA method in detail.

PCA or principal component analysis is a method that enables us to transform the set of features into less dimensional space. The PCA method is an unsupervised method, where we don't take into account the information about the class of the data, but we want to minimize the information loss by diminishing the dimension of the space. As such, PCA is not a dimensionality reduction method. It takes the axis of the space, translates and rotates the axis such that the direction of the maximal variance in the data is found.

How do we use it to reduce the dimension?

By using only some of the computed components, we can reduce the data dimensionality. If we want to lower the dimension we will use only the first axis. Generally speaking, we have d -dimensional feature vectors, and we want to find a new orthonormal basis. We have to map the new coordinates to the original space. We do it by this equation:

$$y_i = p + \sum_{j=1}^D y_{ij} b_j$$

Error we get by the backward mapping to the original space can be computed by comparing the original values and the back projected values. We want to minimize this error between original and projected vectors:

$$\begin{aligned} E &= \sum_{i=1}^N \|x_i - y_i\|^2 = \sum_{i=1}^N \|x_i - (p + \sum_{j=1}^D y_{ij} b_j)\|^2 = \\ &= \sum_{i=1}^N \|x_i - p\|^2 - 2 \sum_{i=1}^N \sum_{j=1}^D y_{ij} b_j^T (x_i - p) + \sum_{i=1}^N \sum_{j=1}^D y_{ij}^2 \\ & \quad y_{ij} = b_j^T (x_i - \bar{x}) \end{aligned}$$

⇒ new origin is the mean of the original data: $p = \bar{x}$

2.8 Describe the PCA method in detail. How do we determine the number of significant eigenvectors?

Describe the PCA method in detail.

PCA or principal component analysis is a method that enables us to transform the set of features into less dimensional space. The PCA method is an unsupervised method, where we don't take into account the information about the class of the data, but we want to minimize the information loss by diminishing the dimension of the space. As such, PCA is not a dimensionality reduction method. It takes the axis of the space, translates and rotates the axis such that the direction of the maximal variance in the data is found.

How do we determine the number of significant eigenvectors?

As we know, PCA only rotates and translates the coordinates but we want to reduce the dimensionality, so we will only keep first k components after PCA. There are several methods of how to find the correct k (=number of principal components).

(1) Scree plot is a graph of the explained variance where we have the eigenvectors ordered in decreasing order. There is the inflection point after which the eigenvalues stop becoming interesting and relevant for us (the boundary after which unimportant eigenvalues start). So the optimal number is the number of eigenvalues before the inflection point.

(2) cumulative explained variance - if the cumulative proportion reaches 0.9-0.95 then we cut off the eigenvalues.

2.9 Describe in detail the training of the Bayesian classifier. What do we know about the error of this classifier?

Bayes rule is expressed as follows:

$$P(A_i|B) = \frac{P(B|A_i) \cdot P(A_i)}{P(B)}$$

In this classifier the object is assigned to class ω_i , which is the most probable given the feature vector x :

$$P(\omega_i|x) = \frac{P(x|\omega_i) \cdot P(\omega_i)}{P(x)}$$

So we have to know how probable is the object in the class ω_i , how probable is the class itself and how probable is the object at all.

We can express the probability of the object x as: $P(x) = \sum_{j=1}^M P(x|\omega_j) \cdot P(\omega_j)$

The decision function of Bayes classifier is that the object is assigned to class ω_i , where the following inequality holds for any other class:

$$\frac{P(x|\omega_i) \cdot P(\omega_i)}{P(x)} \geq \frac{P(x|\omega_j) \cdot P(\omega_j)}{P(x)}$$

Since in the inequality is the same, it's enough to evaluate nominators, so the decision function $f(x)$ will be the class ω_i , where:

$$i = \operatorname{argmax}_j P(x|\omega_j) \cdot P(\omega_j)$$

$P(\omega_i|x)$ = posterior

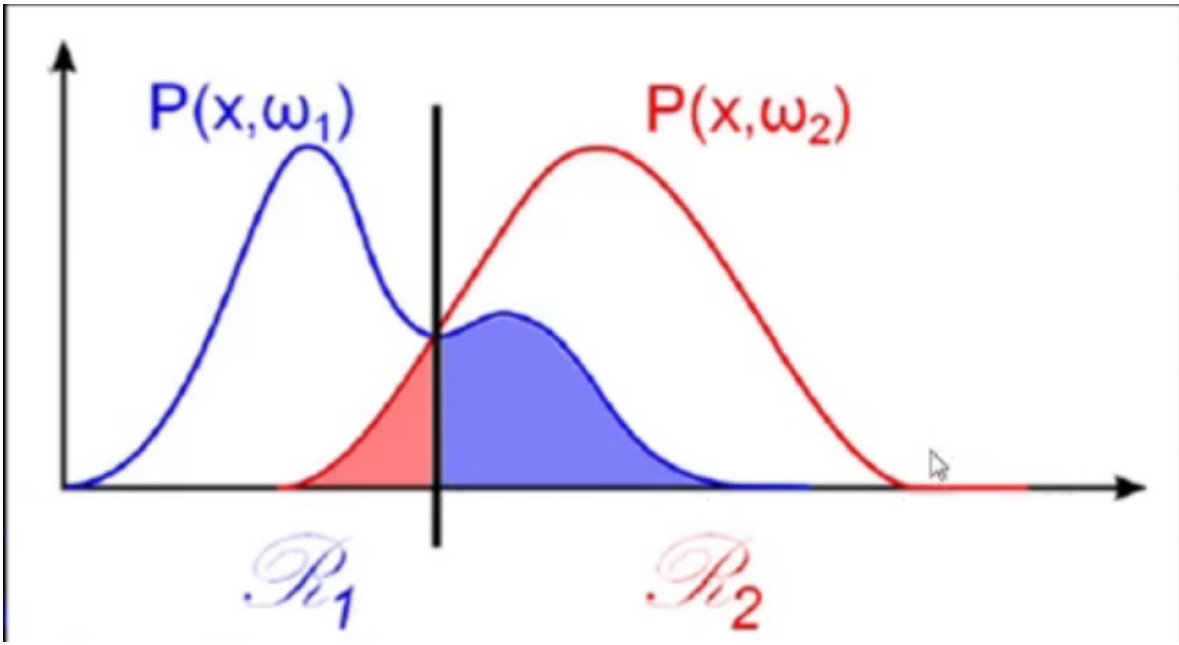
$P(x|\omega_i)$ = likelihood

$P(\omega_i)$ = prior

$P(x)$ = evidence

Depending on what we compute, we do either the MLE (maximum likelihood estimation)=we evaluate only the nominator, or MAP(maximum a posteriori estimation)=we evaluate the whole expression.

Since we choose the class to hold this inequality, we can illustrate it on the two classes:



What is the error we made by classifying the object in this manner?
 We misclassified all the objects under the red and the blue curve, so the probability of the total error is the sum of these two probabilities of misclassifying objects from class 1 and class 2:

$$P(\text{error}) = \int_{R_1} P(x, \omega_2) dx + \int_{R_2} P(x, \omega_1) dx$$

If we move the decision boundary, the error can only become higher, so the Bayesian decision boundary is optimal from the point of the error.

Naive Bayes classifier:
 Naivety is in the assumption that the features are independent.

2.9.1 Training:

We need to estimate the probability of each feature given each class.
 For categorical features:

$$P(x_k | \omega_i) = \frac{N^{i,k}}{N^i}$$

$$P(\omega_i) = \frac{N^i}{N}$$

where:
 $N^{i,k}$ = number of objects from class ω_i where feature k takes the value x_k
 N^i = number of objects from class ω_i
 N = number of all objects

Don't forget to mention: Laplace smoothing, how different distributions can be used to model the probability of x given some class, calculating logarithms rather than raw probabilities, some use cases.

2.10 Describe how decision trees are created.

Building an optimal tree is hard. So we use greedy algorithm to construct them.
 To evaluate the split we use either gini impurity or information gain (basically entropy).

2.10.1 Information gain

$$IG(S, F) = H(S) - \sum_{f \in \text{eval}(F)} P(F = f)H(S_f)$$

2.10.2 Gini

$$p_k = \frac{|S_k|}{|S|}$$
$$G(S) = \sum_{k=1}^c p_k(1 - p_k)$$

Information gain likes features that have many values because they fragment the dataset a lot. One possibility is to penalize features with many values in the splitting process. Another is to allow only binary splits. In binary splits we usually check all possible thresholds. If we needed an increase in speed we can check only some thresholds.

2.10.3 Pruning

Prepruning – stop growing the tree before it reaches the points where it clasifies everything correctly.

Postpruning – allow the tree to overfit, then delete branches. We use validation set to dicide what branches to delete. We iteratively try to remove nodes and see how the validation loss changes. In each step we remove the node that decreases the validation loss the most. C4.5 pruning methods.

2.10.4 Some stopping criteria

- Number of nodes in the tree
- Number of examples in the node
- depth

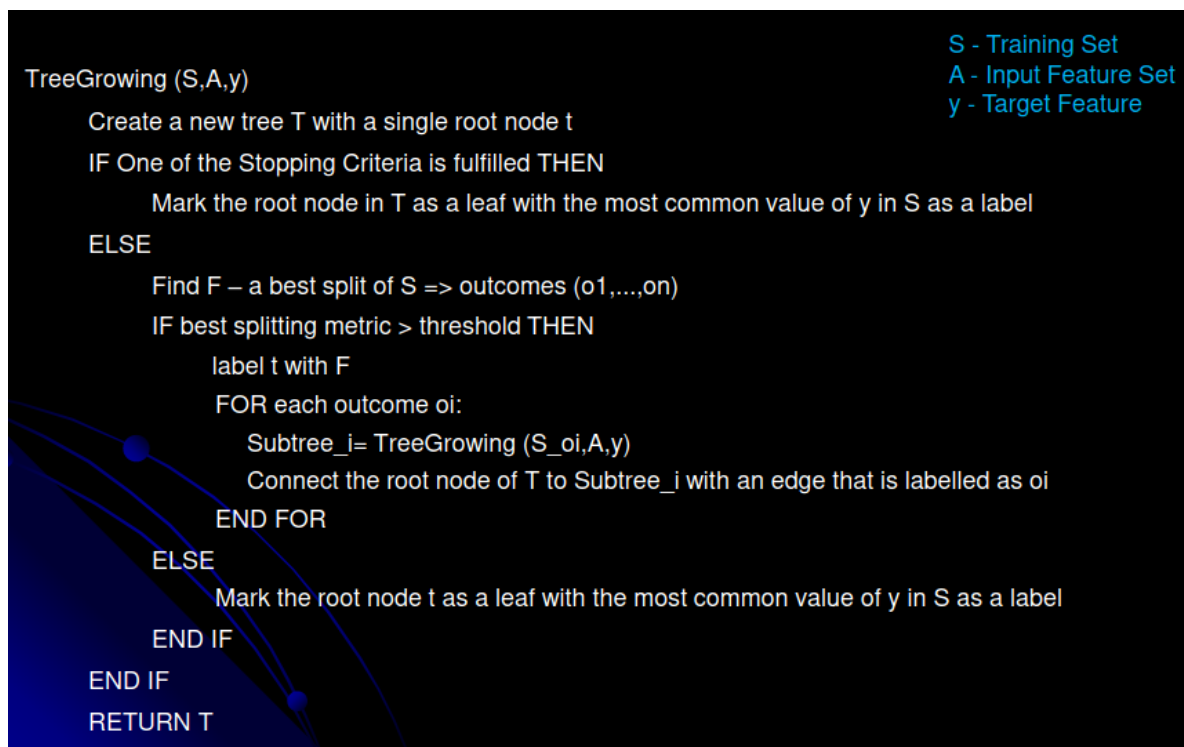


Figure 6: Decision tree learning

2.11 Describe the cross-validation methods and the bootstrap method. What is the main difference between them? What are they used for?

They are used for error estimation.

In the cross validation we divide the data into K subsets and use each subset as a test set for error estimation. At the end we compute the average error $E = \frac{1}{K} \sum_{k=1}^K E_k$

2.11.1 Cross validation methods:

There are several possibilities of how to divide data into K subsets.

1. Random subsampling:

We take random samples from the data without returning.

In each draw, each object can be in the test set only once, but the same object can be in the test set in several draws.

2. K-fold cross validation:

It takes each object to the test set exactly once. We divide whole data into K equally sized subsets.

3. Stratified K-fold validation:

Create K subsets that follow the class distribution of the whole set.

K-fold cross, validation can be done if we have enough data. In case of small sample size, we use Leave one out approach.

4. Leave one out:

We test each time on one sample only.

$K=N$ (size of the set)

2.11.2 Bootstrap method:

In bootstrapping we pick the data for the training set, and we do it with replacement. The training set size in bootstrap is the same as the data size and some objects will not make it to the training set and those will then later form a test set. The test set size can vary across the draws. At least one object must end up in the testing set.

The probability of ending up in the test data is: $(1 - \frac{1}{N})^N \approx e^{-1} = 0.368$

The training data will contain approximately 63.2% of the instances.

2.11.3 Cross validation methods vs bootstrap:

Cross validation:

- Cross-validation gives the cross-validation estimate for assessing how well a model can perform on new data.
- Sampling without replacement
- Does not rely on random sampling
- Forms subsets for testing

Bootstrap:

- The bootstrap tells how accurate a sample statistic is compared to the true population statistic.
- Sampling with replacement
- Relies on random sampling
- Used to form the training set

2.12 Describe convolutional networks in detail.

Inspired by biological eyes. They are significantly easier to compute than feed forward nets (much fewer parameters) and they work well. They are shift invariant so they work well with image/video data.

The typical CNN consists of many blocks of kernels and pooling layers. They were pioneered by LeCun and Hinton.

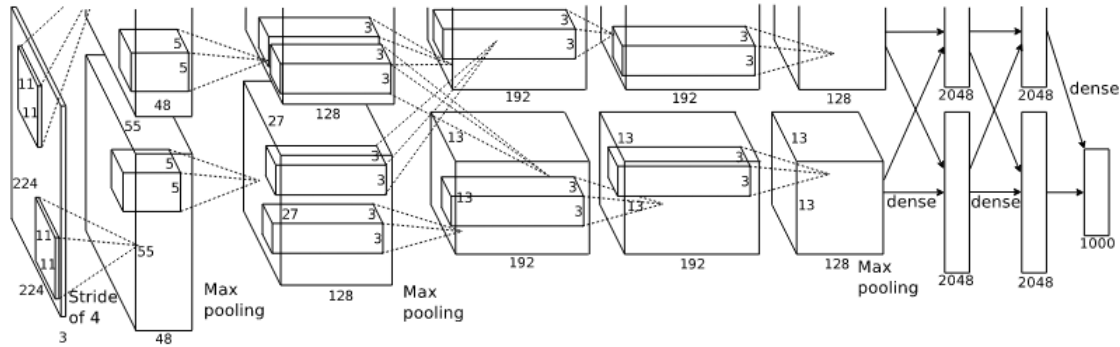


Figure 7: CNN (from Krizhevsky et al. 2017)

2.12.1 Discrete 2D convolution

$$h(k, l) = (I * w)(k, l) = \sum_{m=-M}^M \sum_{n=-N}^N I(m, n) \cdot w(k - m, l - n)$$

Popular activation functions

- ReLU
- Leaky ReLU
- ELU
- GELU

Most common types of pooling: max and average.

2.12.2 Stride

The stride is the number of pixels that the analysis window moves on each iteration. A stride of 2 means that each kernel is offset by 2 pixels from its predecessor.

2.12.3 Padding

Padding is the addition of pixels on the borders of an image. Padding prevents some information loss. Without it the bordering pixels would participate in only some kernel computations.

It also prevents image from shrinking too much after convolutions.

2.12.4 Feature extractors

Sometimes CNNs are pretrained on a different problem and then inserted into a new model with frozen weights. They can extract features well and on their output another model not necessarily CNN is trained.

2.13 Describe the methods of hierarchical clustering in detail.

Seeks to build a hierarchy of clusters. Two approaches: agglomerative (bottom-up), divisive (top-down). Result of hierarchical clustering can be usually described with a dendrogram – binary tree showing how clusters are merged/split hierarchically. In a dendrogram leaves are singleton clusters.

2.13.1 Divisive approach

Start with all objects in one cluster. Repeat until individual objects: Pick cluster to split. Split the cluster.

Divisive ANALYSIS Basic algorithm for divisive clustering. When splitting cluster choose the object most dissimilar to other objects and create new cluster from it. Now for each object in the old cluster, reassign it to the new if the new one is closer. Repeat this until all objects are in their closer clusters.

Principal Directions Divisive Partitioning

1. Find the principal axis
2. Split point at mean projection
3. Find new centroids

Median cut Used for color quantization.

2.13.2 Agglomerative approach

Individual objects are clusters. Until only one cluster, merge two closest clusters (determined by some metric).

Typical cluster distances:

- single-link – closest
- complete-link – furthest
- average-link – average between all pairs
- centroid-link – centroids
- Ward's methods $C_{A,B} = \frac{N_A N_B}{N_A + N_B} \|c_a - c_b\|^2$, where N_k is the number of objects in cluster K and c_k is its centroid.

Iterative shrinking In each step select cluster to remove and repartition objects to nearby clusters.

3 Section C - the answer consists of 2-3 sentences

3.1 What is a confusion matrix, how is it created? What is it used for?

truth \ prediction	Positive ✓	Negative	# of examples
Positive ✓	TP (True Positive)	FN (False Negative)	P
Negative	FP (False Positive)	TN (True Negative)	N

Figure 8: A binary confusion matrix.

Actual	Elephant	25	3	0	2
	Monkey	3	53	2	3
	Fish	2	1	24	2
	Lion	1	0	2	71
		Elephant	Monkey	Fish	Lion
		Predicted			

Figure 9: Multi class confusion matrix.

One vs. all

		Prediction					Total
		c_1	c_2	c_3	...	c_n	
Actual	c_1	TP	FN				N_1
	c_2	FP	TN				N_2
	c_3						N_3

	c_n						N_n
Total	\hat{N}_1	\hat{N}_2	\hat{N}_3	...	\hat{N}_n	N	

Figure 10: One vs All confusion matrix and types of results.

3.2 What is the ROC curve, how is it generated? What is it used for? What values can we derive from it?

Receiver Operating Characteristics. TPR (True Positive Rate) againsts FPR.

$$TPR = \frac{TP}{TP + FN} = \frac{TP}{P}$$

$$FPR = \frac{FP}{FP + TN} = \frac{FP}{N}$$

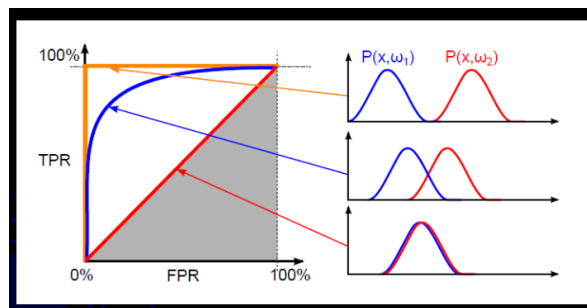


Figure 11: ROC and distribution types

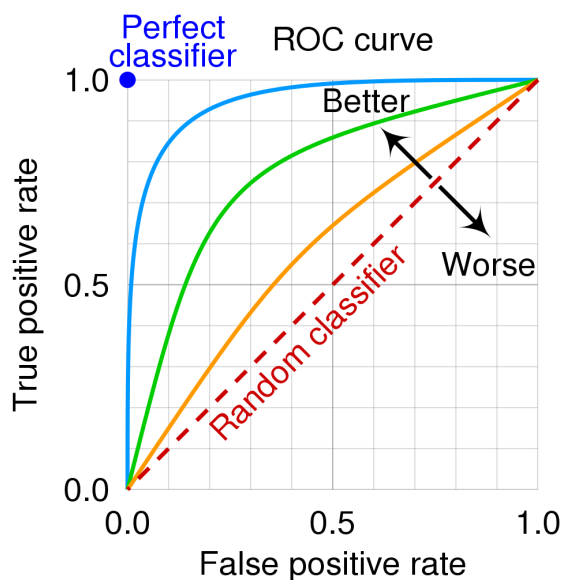


Figure 12: ROC for different classifiers

3.3 What methods do we use to select a training and test set?

- cross validation – Divides data into K subsets, use each subset for error estimation
- random subsampling with no return
- k-fold cross validation – divide data into k equally sized subset, train on $k - 1$ validate on the remaining. Do this for each of the k subsets.
- stratified k-fold – try to obtain subsets so that each subset has the same distribution of classes
- leave one out – test on one sample only
- bootstrap – sample the original dataset (of size N) N times with replacement. Use data not occurring in the train set for testing.

3.4 What is the difference between PCA and ICA?

PCA – vector is represented as linear combination of orthogonal vectors

ICA – vector represented as linear combination of non-Gaussian random variables (=independent components)

PCA – typically compresses information

ICA – separates information (separates the voices of several speakers into independent components)

PCA – finds uncorrelated components ICA – finds independent components.

PCA – optimizes the covariance matrix of the data which represents second-order statistics.

ICA – optimizes higher-order statistics such as kurtosis.

3.5 What is the difference between PCA and LDA?

- two classes are easily separable by PCA but not LDA
- for small number of training data, PCA gives better results than LDA, however if we have enough data, LDA is better

- In the case where the difference between the classes lies in the variance of the class not in the mean, so the classes are almost in the same position but differ only in the variance, then LDA fails, because it is trying to separate the means of the classes to be as far as possible.
- Sometimes the dimensionality is too high and we only have a few classes so going from a dimensionality eg. 1000 to 2 using the LDA is not good because we lose too much information. What can be done is to combine PCA and LDA. We first use PCA to lower the dimension, so we will find the mapping which keeps as much variance as possible but lowers the dimensionality and then we find the discriminative directions using the LDA.
- As both PCA and LDA assume Gaussian data, they fail in the non-Gaussian case.

3.6 In what ways do we normalize features?

3 σ scaling:

$$\tilde{x}_i = \frac{\frac{x_i - \mu}{3\sigma} + 1}{2}$$

Algorithms that require scaled input:

- SVM
- NN
- Perceptron
- PCA, ICA

3.7 What is the error function of the classifier? How is the classifier error estimated?

Indicates penalization for wrong answer.

$$L : Y \times Y \rightarrow \mathbb{R}^+$$

Cross validation. Random subsampling K-fold CV. Stratified K-fold CV. Leave one out Bootstrap – pesimistic.

3.8 What are the filter and wrapper for? What is the difference between them?

Filter Chooses features without any consideration of the classifier. Needs some fitness function.

Wrapper based on classifier.

3.9 Describe the classification using the Bayesian classifier.

Choose class with the highest prob conditioned on x :

$$\arg \max_k P(C_k | x) = \arg \max_k \frac{P(x | C_k)P(C_k)}{P(x)} = \arg \max_k P(x | C_k)P(C_k)$$

Models $P(x | C_k)$ and $P(C_k)$.

3.10 Describe the classification using a linear classifier.

Model the decision boundary as a hyperplane. The hyperplane can be modeled by the following equation where w is the normal vector and b is the offset from the origin

$$w^T x + b = 0$$

$$f(x) = w^T x + b$$

positive for one class, negative or equal for the other. We can also hide bias and model f as

$$f(x) = u^T y,$$

where $u = [b, w^T]^T$ and $y = [1, x^T]^T$.

3.11 Describe the classification using decision trees.

In each inner node there is a feature according to which we split. When we get to a leaf we need to classify the example. For that, we do the majority voting. If we were doing a regression task we could average values in the leaf.

3.12 Describe the SVD method. What is it used for?

Singular value decomposition of a matrix(SVD) is one of the methods used to find PCA. We use this kind of decomposition namely when $N < D$ (dimensionality). It is a factorization of the matrix into three matrices: $A = USV^T$, where:

- U is orthogonal matrix containing normalized eigenvectors of $S_{left} = AA^T$ arranged in descending order
- V^T is orthogonal matrix containing normalized eigenvectors of $S_{right} = A^T A$ also in descending order that are transposed
- Rectangular diagonal matrix of sigmas(sigmas are called singular values. The eigenvalue is the square of the singular value) having the same dimension as matrix A

$$[A]_{N \times D} = [U]_{N \times N} [S]_{N \times D} [V]_{D \times D}^T$$
$$U = [u_1, \dots, u_N], V = [v_1, \dots, v_D], S = \text{diag}(\sigma_1, \dots, \sigma_{\text{rank}(A)})$$

We are looking for d basis vectors (so we are interested in matrix V). Instead of computing the covariance matrix X , we can compute the singular value decomposition, which gives us directly eigenvectors of the covariance matrix without having to evaluate the covariance matrix.

The algorithms for computing the SVD are numerically stable.

3.13 Describe the measures used to evaluate features.

- Consistency Feature subset must classify consistently within the set. When some feature is present in multiple classes it is inconsistent.
- Independence Features that are highly correlated provide the same information so it is not important to have all of them. $J(\tilde{X}) = \frac{k r_{cf}^-}{\sqrt{k+k(k-1)r_{ff}^-}}$, where r_{cf}^- is a mean correlation coefficient of feature-class and r_{ff}^- of feature-feature.

$$- NLCC = \sqrt{1 - e^{-2I(X;Y)}}$$
$$- pearson = \frac{COV(X,Y)}{\sqrt{VAR(X)VAR(Y)}}$$

- Information theoretical measures
 - Hartley's Information Measure
 - Shannon's Information Measure. Entropy is given as the expected value of information, for discrete case $H(A) = -\sum_{a \in \Omega} P(A = a) \log_2(P(A = a))$. We use entropy to compute mutual information $I(X; Y)$ which tells us how much information on variable gives about another. We can use it to evaluate subset of features as $J(\tilde{X}) = I(\tilde{X}; y)$.
- Interclass distance We want classes that put features far away. $J(\tilde{X}) = \sum_{i=1}^C P(\omega_i) \sum_{j=i+1}^C P(\omega_j) D_{\tilde{X}}(\omega_i, \omega_j)$, where D denotes the average distance computed from all pairs of objects with class i and j .

3.14 Define the terms entropy and mutual information. What are they used for when choosing features?

Introduced by Shannon. Entropy is given as the expected value of information, for discrete case $H(A) = -\sum_{a \in \Omega} P(A = a) \log_2(P(A = a))$. We use entropy to compute mutual information $I(X; Y)$ which tells us how much information on variable gives about another. We can use it to evaluate subset of features as $J(\tilde{X}) = I(\tilde{X}; y)$.

$$I(Y; X) = \sum_{x \in \Omega_X} \sum_{y \in \Omega_Y} P(X = x, Y = y) \log \frac{P(X = x, Y = y)}{P(X = x)P(Y = y)} = H(Y) - H(Y|X)$$

3.15 Define the terms statistical independence and consistency. What are they used for when choosing features?

Events are independent if $P(A \cap B) = P(A)P(B)$. If events are independent they are uncorrelated (converse is however not always true).

Feature subset must classify consistently with the whole set. Features that classify consistently are consistent. We calculate inconsistency $IC(x)$ as

$$IC(x) = M - \max_i m_i,$$

where $M = \sum_{i \in \text{classes}} m_i$ and m_i is the number of instances in ω_i .

Fitness is then

$$1 - \frac{\sum_{x \in \text{Unique}(\tilde{X})} IC(x)}{N}$$

3.16 What clustering methods do we know? Give basic division and examples of methods.

- K-means clustering
- K-medoids
- K-medians
- Fuzzy C-means
- Buckshot
- Bisecting K-means
- Self-organizing maps
- DIANA
- Median cut
- Agglomerative clustering
- Shrinking

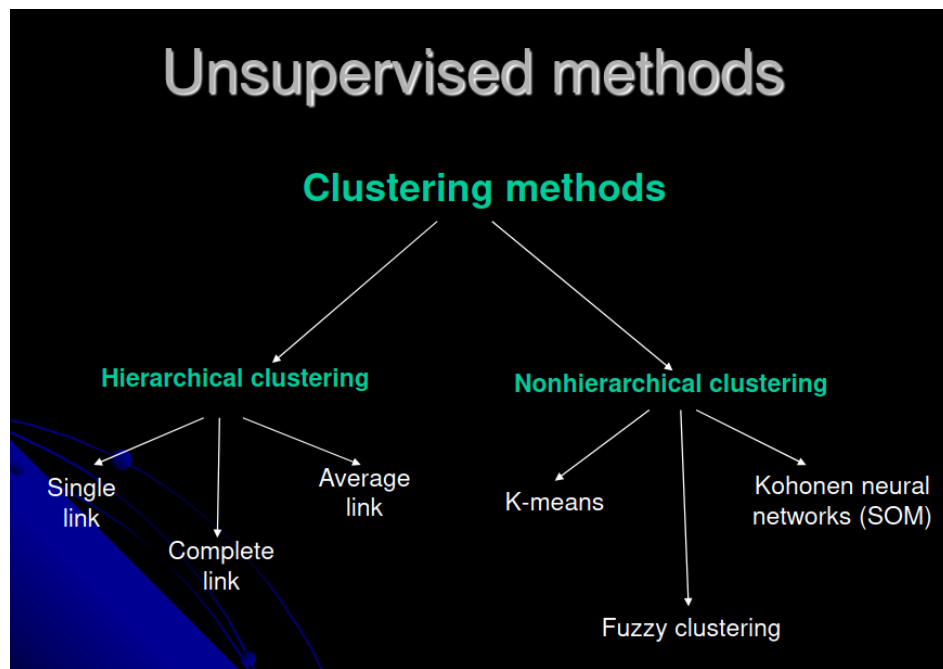


Figure 13: Clustering division

4 Section D - practical questions

- 4.1 Convert data from 2D to 1D space using PCA.
- 4.2 Use the naive Bayesian classifier to classify the object.
- 4.3 Create a decision tree.
- 4.4 Calculate the number of CNN parameters and the size of the individual outputs in the layers