SGN-13006 Introduction to Pattern Recognition and Machine Learning
TAU Computing Sciences
Exercise 3 *Visual classification (CIFAR-10 dataset)*

Be prepared for the exercise sessions (watch the demo lecture). You may ask TAs to help if you cannot make your program to work, but don't expect them to show you how to start from the scratch.

1. **CIFAR-10 – Bayesian classifier (super simple)** (40 points)

   To start with something simple, we assume that the only important feature of CIFAR-10 images is their *average color*. That means that you calculate the mean color of 32×32 images and each of the $i = 1, \ldots, 50000$ CIFAR-10 images is then represented by only three values $\boldsymbol{x}^i = (r^i, g^i, b^i)$. Write a function *def cifar10_color(X)* that converts the original images in X ($50000 \times 32 \times 32 \times 3$) to Xf ($50000 \times 3$).

   Moreover, we make the assumption that all channels are independent (no correlation) and therefore their probability distribution is a single Gaussian, i.e.

$$\begin{aligned} P(class_1|\boldsymbol{x}) &= \frac{P(\boldsymbol{x}|class_1)P(class_1)}{\sum_j P(\boldsymbol{x}|class_j)P(class_j)} \\ &= \frac{\mathcal{N}(r;\mu_{r,c_1},\sigma_{r,c_1})\mathcal{N}(g;\mu_{g,c_1},\sigma_{g,c_1})\mathcal{N}(b;\mu_{b,c_1},\sigma_{b,c_1})P(c_1)}{\sum_j \mathcal{N}(r;\mu_{r,c_j},\sigma_{r,c_j})\mathcal{N}(g;\mu_{g,c_j},\sigma_{g,c_j})\mathcal{N}(b;\mu_{b,c_j},\sigma_{b,c_j})P(c_j)} \end{aligned}$$

   To use the above rule, you need to compute the mean μ and variance σ of the three color channels for each class. Write a function *def cifar_10_naivebayes_learn(Xf,Y)* that computes the normal distribution parameters *(mu, sigma, p)* for all ten classes (mu and sigma are 10×3 and priors p is 10×1).

   Finally write function *def cifar10_classifier_naivebayes(x,mu,sigma,p)* that returns the Bayesian optimal class c for the sample x.

   Run your classifier for all CIFAR-10 test samples and report the accuracy (write another script for this). For evaluation you can use the functions implemented for the previous experiments.

2. **CIFAR-10 – Bayesian classifier (better)** (20 points)

   In this experiment we continue the previous Bayesian classifier, but we relax the naive assumption that the red, green and blue channels are independent. Instead of three 1-dimensional Gaussians we assume a single 3-dimensional Gaussian, i.e. *multivariate normal distribution* (Python: numpy.random.multivariate_normal()). Classification becomes

$$\begin{aligned} P(class_1|\boldsymbol{x}) &= \frac{P(\boldsymbol{x}|class_1)P(class_1)}{\sum_j P(\boldsymbol{x}|class_j)P(class_j)} \\ &= \frac{\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{c_1},\Sigma_{c_1})P(c_1)}{\sum_j \mathcal{N}(\boldsymbol{\mu}_{c_j},\Sigma_{c_j})P(c_j)} \end{aligned}$$

   You need to write new functions to replace the naive versions. *def cifar_10_bayes_learn(Xf,Y)* computes the multivariate normal distribution parameters *(mu, sigma, p)* for all ten classes (mu is 10×3, sigma $10 \times 3 \times 3$ and priors p is 10×1). *def cifar10_classifier_bayes(x,mu,sigma,p)* computes probabilities.

   Compute the classification accuracy for the whole test set and compare it to the naive version - which one is better and why?

3. **CIFAR-10 – Bayesian classifier (super powerful)** (20 points)

   Extend *def cifar10_color(X)* to *def cifar10_2x2_color(X)* that resizes the 32×32 image to 2×2 and computes 4 color features for each sub-window. Now the feature vector $\boldsymbol{x}$ length is $2 \times 2 \times 3 = 12$.

   Your previous bayesien learning and classification functions should still work, now only *mu* is 12×1 and *sigma* is 12×12.

   Compute the performance for 1×1, 2×2, 4×4, $\ldots$, 32×32 images (if only covariance can be computed) and report how the performance improves as the function of the image size (plot a graph).