INF5300, spring 2015 - Lab on feature transforms

"Curse of dimensionality":
Run and explore the script demonstrating the "curse of dimensionality" (see link on course page). Explain why we at some point start to get worse classification performance as we keep adding (in themselves) useful features.

Feature selection using the PRTools package
Run and explore the script demonstrating basic feature selection using the PRTools package (see link on course page). Uncomment appropriately to explore stepwise forward-selection and "forward-floating" selection, and also to determine whether it is anything to gain by using a wrapper-based criterion for this particular dataset. What can you say about computational requirements, based on your experience, for the filter vs. wrapper approach?

Principal component transform (PCA):

1. We will work on data from a six-band satellite image covering the Kjeller area. Load the file tm.mat from ~inf5300/www_docs/data.
2. Put all the image data into one long n*x*6 matrix, **X** (one row per pixel having 6 feature values).
3. Subtract the sample mean from each column of **X**. (Making the mean of each band zero.)
4. Compute the 6*x*6 sample covariance matrix (scatter matrix), **R**.
5. Use [V D] = eig(**R**) to find the eigenvectors and eigenvalues of the covariance matrix. Note: eig() sorts eigenvalues ascendingly.
6. Form the **A** matrix, in **y**=**A'x**, by letting each column be an eigenvector of the covariance matrix.
7. Get the 6 principal components from the just-designed linear transform (apply the **y**=**A'x** transform on each sample/pixel).
8. Compute the sample covariance matrix of the transformed variables. Verify that it is diagonal. Compare the matrix with your list of eigenvalues of **R**. Explain!
9. Reshape the data (the PCA-transformed data) into yet again six 2D images.
10. Display the principal component images one by one (the six new "bands" separately). Looking at them, how many have significant visual information? How many do you think are useful for per-pixel classification?
11. Plot the eigenvalues scaled by the sum of all eigenvalues, as well as the cumulative of that very function. How many bands is necessary to retain at least 90% of the variance of the original 6-band image?
12. Looking at the feature-weights that are used to form the (first) principle component; which of the six original images is the most influential? (Which has the highest weight?)
13. Is PCA scale invariant? Multiply one of the original images by, say, 1000, redo 4. and 5., then see if the eigenvectors change. Scale each input-feature by 1 divided by its standard deviation – does the answer in 12. (the most influential band) change if we do this?
14. Create a "random" vector of length 6 by e.g. **r** = randn(6,1);. Then repeat the following procedure, say N=100 times: a) **r** = **R*****r**; b) **r** = **r**/norm(**r**); Compare this vector with your eigenvectors. Explain! [Hint: Look at the decomposition of **R** on slide 9 in our PCA lecture. Hint II: Except for numerical issues, this is the same as doing a) $\mathbf{R}^{100}\mathbf{r}$**;** b) **r**=**r**/norm(**r**);]
15. Let $\mathbf{R}^*$ = **R** – c***r*****r'**, where c is the highest eigenvalue you found in 5. Repeat 14 using $\mathbf{R}^*$. What do you get? Explain!

Fisher's reduced-rank linear discriminant:

1. Load the training mask 'tm_train_mask.mat' and the feature vectors 'tm.mat'.
2. Compute the mean vectors and covariance matrices for each of the four classes.
3. Compute the between-class scatter matrix S_b.
4. Compute the within-class scatter matrix S_w.
5. Get the Fisher's reduced-rank linear transform by getting the eigenvectors and eigenvalues of $S_w^{-1}S_b$.
6. What is the rank of S_b? How many Fisher components can you get?
7. Display the Fisher component images one by one (the three "bands" separately). Compare with those from the PCA transform.

Classification

Compare the classification performance of a Gaussian-based classifier with equal class-covariance matrices using the original bands, all PCA bands and all "Fisher bands". What if we use a quadratic classifier (i.e., we do not assume equal class-covariance matrices) instead?

Which method yields the best classification accuracy (on the training set) using a single feature?

You can either implement the classifiers yourselves or use ldc() and qdc() in the PRTools toolbox. See the script under "curse of dimensionality" for a simple example of how to use the toolbox for a similar classification (it uses the qdc).