**Labels**

- create labels by specifying individual sample labels
  \[
  \text{lab} = \text{sdlab}(\text{\{'apple', 'banana', 'apple', 'pear'\}})\]
  - create labels by giving class list and vector of class sizes
  \[
  \text{lab} = \text{sdlab}(\text{\{'apple', 'banana', 'size', 10, 5\}})\]
  - create labels per-sample names and indices
  \[
  \text{lab} = \text{sdlab}(\text{\{'apple', 'banana', 'lemon', \{1, 2, 3\}\}})\]
  - create labels by giving class name and number of samples
  \[
  \text{lab} = \text{sdlab}(\text{\{'apple', '10', 'banana', '5\}})\]

**Data sets**

- find samples of a class
  \[
  \text{ind} = \text{findLab}(\text{\{'apple\}})\]
- get class index by name
  \[
  \text{lab.index(\{'banana\})} = \text{lab.index(\{'banana\})}\]
- number of classes
  \[
  \text{lab.numClasses} = \text{lab.numClasses}\]
- number of entries (samples)
  \[
  \text{lab.length(lab.index)} = \text{lab.length(lab.index)}\]
- class/category sizes
  \[
  \text{lab.length(lab.index)} = \text{lab.length(lab.index)}\]
- class/category fractions/priors
  \[
  \text{lab.fractons} = \text{lab.fractons}\]
- subset of samples by index
  \[
  \text{lab(index) = lab(index)}\]
- test if class is present
  \[
  \text{lab(index) \neq empty(lab.index)}\]
- display label details (classes, sizes and fractions)
  \[
  \text{lab' or info(lab)}\]

**Subsets**

- create labels by giving class list and vector of class sizes
  \[
  \text{lab} = \text{sdlab}(\text{\{'apple', 'banana', 'size', 10, 5\}})\]
  - create labels by giving class name and number of samples
  \[
  \text{lab} = \text{sdlab}(\text{\{'apple', '10', 'banana', '5\}})\]

**Random subsets**

- select 100 samples per class
  \[
  \text{sub = randsubset(a,100)}\]
- select 100 samples per patient
  \[
  \text{sub = randsubset(a,patient,100)}\]
- select 100 samples from complete data set
  \[
  \text{sub = randsubset(a,100,\{'apple', 'all'\})}\]

**Bootstrap sampling**

- use pd = sddetect(tr,\{'banana', 'gauss', 'test', 'val\})
- use pd.roc to access ROC curve (empty for one-class detectors)
- specify ROC performance measures with measures option

**Classifiers and combinings**

- Combine classifiers:
  \[
  \text{pd1 = \text{sdlinear}(a, \{'fruit', 'stone', 'another\})}\]
- \text{pd2 = \text{sdgauss}(a, \{'fruit', 'stone', 'other\})}\]
- \text{pdcombine([\{'1' - '2\}])}\]

**Available classifiers**

- \text{d ... can be used for detection}
- Nearest mean
  \[
  \text{sdmmean}\]
- Fisher linear discriminant
  \[
  \text{sdfisher}\]
- Gaussian model (for detection)
  \[
  \text{sdgauss}\]
- Linear discriminant
  \[
  \text{sdlinear}\]
- Quadratic discriminant
  \[
  \text{sdquadratic}\]
- Least mean square classifier (also regression)
  \[
  \text{sdlogistic}\]
- Logistic classifier
  \[
  \text{sdparzen}\]
- Mixture of Gaussians
  \[
  \text{sdmixture}\]
- k-NN
  \[
  \text{sdkn}\]
- k-means prototype extraction
  \[
  \text{sdmeans}\]
- Support Vector Machine
  \[
  \text{sdsvm}\]
- Neural network (feed-forward or RBF)
  \[
  \text{sdsnna}\]
- Deep convolutional network
  \[
  \text{sddeepnet}\]
- Support Vector Machine
  \[
  \text{sdsvr}\]
- Decision tree
  \[
  \text{sdsvm}\]
- Random forest
  \[
  \text{sdtree}\]
- Minimum distance classifier (on sdprox output)
  \[
  \text{sdmindist}\]
- Look-up table (2D classifier approximation)
  \[
  \text{sdlut}\]

**Detection**

- train on one class
  \[
  \text{pr = sdmmodelName(a, \{'fruit', 'stone', 'another\})}\]
- train on one-class classifier
  \[
  \text{pr = sdmmodelName(a, \{'fruit', 'stone', 'reject', 'all\}, 0.01)}\]
- reject low confidence: make normalized outputs with sdmnorm
- Rejection
  \[
  \text{p = sdmmodelName(a, \{'fruit', 'stone', 'reject', 'all\}, 0.05)}\]
- to reject specific number of samples
  \[
  \text{p = sdmmodelName(a, \{'fruit', 'stone', 'reject', 'all\}, 0.05)}\]
- to reject specific number of samples
  \[
  \text{p = sdmmodelName(a, \{'fruit', 'stone', 'reject', 'all\}, 0.05)}\]

**Sample subset**

- by indices
  \[
  \text{b = subset(a, \{'apple', 'banana', 'fruit\})}\]
- by indices
  \[
  \text{b = subset(a, \{'patient', '5\})}\]
- by indices
  \[
  \text{b = subset(a, \{'patient', '5\})}\]
- by indices
  \[
  \text{b = subset(a, \{'patient', '5\})}\]
- by indices
  \[
  \text{b = subset(a, \{'patient', '5\})}\]

**Relabeling**

- relabel classifier decisions
  \[
  \text{p = sdmmodelName(a, \{'fruit', 'stone', 'reject', 'all\}, 0.05)}\]
- relabel sample list
  \[
  \text{p = sdmmodelName(a, \{'fruit', 'stone', 'reject', 'all\}, 0.05)}\]
- relabel sample list
  \[
  \text{p = sdmmodelName(a, \{'fruit', 'stone', 'reject', 'all\}, 0.05)}\]
- relabel sample list
  \[
  \text{p = sdmmodelName(a, \{'fruit', 'stone', 'reject', 'all\}, 0.05)}\]
- relabel sample list
  \[
  \text{p = sdmmodelName(a, \{'fruit', 'stone', 'reject', 'all\}, 0.05)}\]
**Evaluation**

find error on a test set (default mean error over classes)

\[ \text{err} = \text{sdtest}(\text{p, b}) \]

compute specific performance measures

\[ \text{perf} = \text{sdtest}(\text{p, ts, 'measures', {'TPr', 'apple', 'mean-error', 'precision', 'banana'}}) \]

confusion matrix (use 'norm' option to normalize)

\[ \text{sdconfmat(a, lab, ap}) \]

specific order of classes (rows) or decisions (columns)

\[ \text{sdconfmat(a, lab, ap, 'classes', {'lemon', 'apple'}, 'decisions', {'lemon', 'apple'})} \]

confusion matrix in a figure

\[ \text{sdconfmat(a, lab, ap, 'figure')} \]

get sample indices in a specific entry of confusion matrix

\[ \text{ind} = \text{sdconfmatind(a, lab, ap, 'lemon', 'apple')} \]

\[ a(\text{ind}) \] are apple samples, labeled as banana

cross-validate a classifier (10-fold rotation)

\[ p = \text{sdlinear}([s, res]) \text{sdccrossval(p, a)} \]

\[ s \] is a string summary, \[ res \] struct with results

cross-validation by randomization (30-fold, 80% in training)

\[ s = \text{sdccrossval}(p, a, 'method', 'random', 0.8, 0.0, \ldots, 'folds', 30), 'seed', 42) \]

retrieve training, test subsets and the classifier in fold 5

\[ [s, res, e] = \text{sdccrossval}(p, a); \text{tr} = \text{gettsdata}(e, a), 5; \text{ts} = \text{gettsdata}(a, e, a); p(e(5)); \text{de} = \text{cts} \text{tp} \]

leave-one-out over patients

\[ res = \text{sdccrossval}(p, a, 'method', 'leave', 'over', 'patient') \]

simple leave-one-out loop over patients

for \( i = 1 : \text{length(a.patient.list)} \)

\[ [t, s, tr] = \text{subset(a.patient.list), i}; \text{p} = \text{sdlinear}(tr); \text{err(i)} = \text{sdtest}(ts, p); \]

end

**Performance measures**

<table>
<thead>
<tr>
<th></th>
<th>decisions</th>
<th>non-decisions</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>true labels</td>
<td>target</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td></td>
<td>non-target</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

**ROC analysis**

| estimate ROC characteristic (two- or multi-class) |
|---|---|---|
| \[ [ts, tr] = \text{randsubset(a, 0.5); p = \text{sdlinear(tr, tr)}; r = \text{sdroc(ts, p)} \] |
| estimate ROC from soft outputs (use -p to remove decision step) |
| \[ \text{out} = \text{ts} \text{p}; r = \text{sdroc(out)} \] |
| draw interactive ROC plot |
| \[ \text{sddrawroc(r)} \] |
| select op.point by clicking; press 's' to save it back to workspace |
| show interactive confusion matrix (also via ROC plot by pressing 'c') |
| \[ \text{sdconfmat(r)} \] |
| click on confusion entries to define constraints (also precision) |
| \[ \text{up/down or mouse scroll for direct error minimization} \] |
| add ROC to classifier access ROC stored in the classifier |
| \[ p = \text{p'} \text{r}; \text{p} \text{roc} \] |
| create ROC with specific measures |
| \[ r = \text{sdroc(out, 'measures', {'TPr', 'apple', 'FPPr', 'banana'})} \] |
| get performances at op.point 10 |
| \[ r(10) \] |
| estimate ROC for user-defined class weights |
| \[ p = \text{sddecide('w', rand(10000, 3), a.lab.list}; r = \text{sdroc(pd, out)} \] |
| constrain ROC to a subset with error on apple<0.3 |
| \[ r = \text{setcurop(r, 100)}, \text{ind} = \text{getcurop(r)} \] |
| set current operating point by index |
| \[ r = \text{setcurop(r, 'alpha'), p} \] |
| set curernt operating point by constraining TPr and minimizing FPr |
| \[ r = \text{setcurop(r, 'constrain'), 'TPr(apple)'}, 0.99, \ldots, 'min', 'FPPr(apple)'} \] |
| set operating point minimizing the cost (confmat is stored) |
| \[ s = \text{ones(3)}; M(1.2) = 10; r = \text{sdroc(out, 'cost', 'N')}; \]

**Dimensionality reduction**

Principal Component Analysis

\[ \text{sdPCA} \]

Fisher projection (class separation)

\[ \text{sdlda} \]

Proximity representation (distance to prototypes)

\[ \text{sdprox} \]

Scaling data

\[ \text{sdscale} \]

Polynomial feature space expansion

\[ \text{sdpolyn} \]

Preprocessing/spatial indices

\[ \text{sdprep} \]

Feature selection

\[ \text{sdfeatsel} \]

**Forward feature selection** with 1-NN error as criterion

\[ p = \text{sdfeatsel(data, 'forward') \] |

Backward selection using error of a specific model:

\[ p = \text{sdfeatsel(data, 'backward', 'model', sdfisher) \] |

select fixed feature subset

\[ p = \text{sdfeatsel(data, 'fixed', 'sub') \] |

see \[ p \text{.lab} \] for output features and \[ p \text{.inlab} \] for input features

Select features with non-zero variance:

\[ \text{sdfeatsel}(a, 'var>0') \]

**Interactive visualization**

**ROC analysis**

| interactive scatter plot: \[ \text{sdscatter(a)} \] keyboard commands: |
|---|---|---|
| \[ ? \] change feature |
| \[ +/- \text{ or } = \] change z-order of classes |
| \[ </> \] cycle through classes at one time |
| \[ h \] show/hide legend |
| \[ r \] rename class (also class merging) |
| \[ i \] hide current class |
| \[ o \] show only this class |
| \[ t \] invert filter on this label set |
| \[ L \] remove filter on this label set |
| \[ F \] tag sample under cursor (also via double-click) |
| \[ c \] label visible samples as... |
| \[ 1:9 \] switch to label set |
| \[ a \] switch between full data set and subset axes |
| \[ v \] switch axes limits between visible samples / space of total set |
| \[ d \] show/hide feature distributions |
| \[ c \] show/hide confusion matrix with visible/hidden samples |
| \[ f \] return to previous sample filter |
| \[ s \] show all samples (remove filter) |
| \[ F \] classifier decisions (also in multi-dimensional space) |
| \[ c \] change color of decision backdrop under cursor |
| \[ c \] open also connected ROC plot for scatter/image view |
| \[ s \] sdscatter(a, p, r, 'roc'); \text{sdimage(im, p, r, 'roc')} |
| \[ c \] show feature distributions |
| \[ s \] sdfeatplot(a) |

**Cluster analysis**

| cluster data with k-means algorithm into 10 clusters |
|---|---|---|
| \[ b = \text{sdcluster(a, 10, 'kmeans', 10)} \] |
| train k-means model returning cluster labels |
| \[ p = \text{sdcluster(a, 10, 'kmeans')} \] |
| get labels |
| \[ p = \text{sdreject(p, a)} \] |
| execute and set decisions as new labels |
| \[ b = a \text{p} \] |

**Execution out-of-Matlab**

**Export classifier** for execution: \[ \text{sdexport(p, filename)} \]

Classifier can be run from any application via perclass.dll runtime

For details see: \[ \text{http://perclass.com/doc/guide/deployment.html} \]

Export classifier as a C header file for static linking:

\[ \text{sdexport(p, filename, 'header')} \]
Feature extraction

- **general extraction syntax**
  - `out=sdxextract(data, domain, feat [,options] )`
  - **domains:**
    - **region:** Extract features from local image neighborhoods
    - **object:** Extract one feature vector per object (via object labels)
    - **bands:** Extract features for bands in spectra
    - **color:** Transform color space

Region features

- **raw** image content (in: any, out: block*block*inbands)
- **moments** mean and std per neighborhood (in: 1, out: 2)
- **hist** Local histogram (in: 1, out: bins)
- **histfeat** statistical features of local histogram (in: 1, out: 5)
- **cm** Co-occurrence matrix (in: 1, out: bins^2)
- **gauss** Convolution with Gaussian kernel or derivative (in: 1, out: 1)
- **sobel** Gradient magnitude and orientation (in: 1, out: 2)

Object segmentation (connected components)

- **gray shape** Hu moments and shape eigenvalues (in: 1, out: 9)
- **shape** Hu moments and shape eigenvalues (in: any, out: 9)
- **class fractions** Fraction of classes present in provided pixel/region labels

Object features

- **size** Object size (in: any, out: 1)
- **mean** Mean vector for each object (in: any, out: same as input)
- **sum** Sum vector for each object (in: any, out: same as input)
- **hist** Histogram of one input feature per-object (in: 1, out: bins)

Grid definition

- **Neighborhood size** with block (default: block=8)
- **Change grid** step (default: step=1)
- **Remove grid** from image (shrink to grid)

Image subsets

- **Subset of an image is still image**
  - `c=sdxcluster(im,sdkmeans,3);
    sub=c(:,1)` % choose cluster 1 (dice in our example)
  - `sdimage(im); sdimage(c); sdimage(sub)`

- **use rename class ("r" key) to assign meaningful names and merge**
  - `im=sdxsegment(im .* pobjdet) % separate objects/tray
    dec=im .* ppixel % get per-pixel defect decisions
    obj=sdxextract(im, 'object', 'class fractions', dec)`
  - `obj` contains one sample for each of length(im.object.list) objects
  - `Back project object decisions to orig.image`
Feature extraction

- **sdpca(a,frac)** Principal component analysis
  - frac <1 - fraction of preserved variance
  - frac >1 - number of output dimensions
- **sdlda** Linear discriminant analysis
- **sdscale(data,method)** Feature scaling
  - 'variance'

Non-parametric classifiers

- **sdknn** k-th nearest neighbor
  - **sdknn(data,10)** - by default returns distance to k-th neighbour in each class (can be used as a detector)
  - **sdknn(data,'classfrac','k',10)** - returns confidence (fraction of samples of each class in k neighbors). Discriminant only.
  - k may be any integer (also even)
  - trained pipelines allow change of k: p(1).k=7
- **sdkmeans** k-means as a classifier (detector or discriminant)
  - **sdkmeans(data,10)** runs k-means clustering for each class and returns 1-NN classifier to the extracted prototypes

- **sddeepnet** Convolutional neural network