

perClass 3.2 cheat-sheet

<http://perclass.com/doc>



Labels

create labels by specifying individual sample labels

```
lab=sdslab({'apple','banana','apple','pear'})
```

create labels by giving class list and vector of class sizes

```
lab=sdslab({'apple','banana'},[10 5 20])
```

create labels by giving class name and number of samples

```
lab=sdslab('apple',10,'banana',5)
```

create labels by per-sample indices and class list

```
lab=sdslab([1 1 2 3],{'apple','banana','lemon'})
```

find samples of a class

```
ind=find(lab=='apple')
```

find given class index

```
ind=find(lab==2)
```

get class index by name

```
lab.list('banana')
```

get class name by index

```
lab.list(2)
```

number of classes

```
length(lab.list)
```

number of entries (samples)

```
length(lab)
```

class sizes

```
lab.sizes
```

class priors (read only)

```
lab.priors
```

subset of samples by index

```
lab(1:10)
```

test if class is present

```
isempty(lab.list('A'))
```

▶ `lab.list` contains only classes present in `lab`

display label details (class names, sizes and fractions):

```
lab'
```

Data sets

create data set

```
a=sddata(matrix,lab)
```

raw data matrix

```
+a; double(a)
```

access class labels

```
a.lab
```

display data set details

```
a'
```

subset of samples

```
a(1:10)
```

remove samples

```
a(1:10)=[]
```

subset of features by indices

```
a(:,[1 2 5])
```

subset of features by names

```
a(:,{'Skew','Entropy'})
```

select classes by names

```
a(:,:,{'apple','pear'})
```

select classes by indices

```
a(:,:,1:2)
```

create new labels

```
a.patient=sdslab('Pat1',10,'Pat2',30)
```

create numerical property

```
a.pixel=1:1024
```

select samples by property

```
a(a.pixel<50)
```

test if property exists

```
isprop(a,'patient')
```

remove sample property

```
rmprop(a,'patient')
```

make patient labels current

```
a=setlab(a,'patient')
```

find current label set

```
name=setlab(a)
```

▶ `a.lab` now points to `a.patient`

label all samples in one class

```
a.lab='pear'
```

change label for a subset

```
a(1:10).lab='banana'
```

Subsets

sample subset by values

```
b=subset(a,'lab','cancer','patient',1:3)
```

▶ get cancer samples from the first three patients

return also the **rest of samples**

```
[b,rest]=subset(a,'patient',5)
```

select classes **matching regular expression**

```
b=subset(a,'/substring')
```

iterate over classes

```
for i=1:length(a.lab.list), b=a(:,:,i); end
```

Random subsets

100 samples **per class**

```
sub=randsubset(a,100)
```

select 30% **per class**

```
b=randsubset(a,0.3)
```

select 100 samples **per patient**

```
sub=randsubset(a,'patient',100)
```

select 100 samples **from complete data set**

```
[sub,rest]=randsubset(a,100,'all')
```

Classifiers

train a classifier

```
p=sdlinear(a)
```

```
p=a*sdlinear
```

apply classifier model to new data (returns soft outputs)

```
out=b*p
```

add default decision function

```
pd=sddecide(p)
```

all classifier decisions

```
pd.list
```

one-step training until decisions

```
pd=a*(sdlinear*sddecide)
```

get decisions

```
dec=b*pd
```

get data with decision labels

```
c=b.*pd
```

▶ `dec.list` or `c.lab.llist` contains **all classes present** in

`dec` (may be less than in `pd.list`)

▶ `p.lab` gives labels of classifier output (empty if returns `dec`)

Detectors

train a detector (data, target class, model)

```
pd=sddetector(a,'banana',sdknn)
```

▶ change decision names with 'target' and 'non-target' opts.

train a one-class detector rejecting 10% of bananas

```
pd=sddetector(a,'banana',sdparzen,'reject',0.1)
```

train a detector **on all samples**

```
pd=sddetector(a,'all',sdknn,'reject',0.1)
```

▶ use target name that is not present in `a.lab.list`

▶ must specify the fraction to be rejected to set threshold

provide a set to estimate ROC (does not split `tr`)

```
pd=sddetector(tr,'banana',sdgauss,'test',val)
```

Relabeling

relabel data set (what is not *stone* becomes *fruit*)

```
a2=sdrelabel(a,{'~stone','fruit'})
```

relabel first two classes into *fruit*, *stone* class in *other*

```
a=sdrelabel(a,{1:2,'fruit','stone','other'})
```

relabel by **regular expression** (*apple* or *pear* becomes *fruit*)

```
a=sdrelabel(a,{'/apple|pear','fruit'})
```

add prefix to class names

```
a2=sdrelabel(a,'add','new-')
```

add prefix to all class names **in all sets of labels**

```
a2=sdrelabel(a,'add to all','new data-')
```

relabel classifier decisions by providing a list with new names

```
pd=sdrelabel(pd,sdlist('target','others'))
```

Rejection

Add reject option to a classifier (reject 5% of samples in `a`)

```
p=sdparzen(a); pr=sdreject(p,a,'reject',0.05)
```

Find samples that are not rejected

```
dec=a*pr; a(dec~='reject')
```

Evaluation

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

```
err=sdtest(pd,b)
```

compute **specific performance measures**

```
perf=sdtest(pd,ts,'measures',{'TPr','apple','mean-error','precision','banana'})
```

confusion matrix (use `'norm'` option to normalize)

```
sdconfmat(a.lab,a*pd)
```

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

```
sdconfmat(a.lab,a*pd,'classes',{'lemon','apple','decisions',{'lemon','apple'}})
```

get sample indices **in a specific entry** of confusion matrix

```
ind=sdconfmatind(a.lab,p*pd,'apple','banana')
```

▶ `a(ind)` are *apple* samples, labeled as *banana*

cross-validate a classifier (10-fold rotation)

```
p=sdlinear*sddecide;
```

```
[s,res]=sdcrossval(p,a)
```

▶ `s` is a string summary, `res` struct with results

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

```
s=sdcrossval(p,a,'method','random',0.8,...'folds',30,'seed',42)
```

retrieve training, test subsets and the classifier in fold 5

```
[s,res,e]=sdcrossval(p,a);
```

```
tr=gettrdata(e,a,5); ts=gettsdata(e,a,5);
```

```
pd=e(5); dec=ts*pd
```

leave-one-out over patients

```
res=sdcrossval(p,a,'method','loo','over','patient')
```

simple leave-one-out **loop over patients**

```
for i=1:length(a.patient.list)
    [ts,tr]=subset(a,'patient',i);
    pd=tr*(sdlinear*sddcide);
    err(i)=sdtest(ts,pd);
end
```

Cascades and hierarchical classifiers

Create **hierarchical classifier**

```
▶ pd is fruit/non-fruit detector, pd2 apple/banana classifier
pc=sdcascade(pd,'fruit',pd2)
```

▶ for further stages continue adding decision, classifier pairs

ROC analysis

estimate ROC characteristic (two- or multi-class)

```
[tr,ts]=randsubset(a,0.5); p=sdlinear(tr);
out=ts*p; r=sdroc(out)
```

draw interactive ROC plot

```
sddrawroc(r)
```

▶ select op.point by clicking; press 's' to save it back to workspace

create classifier **access ROC** stored in the classifier

```
pd=p*r pd.roc
```

one-line training and ROC estimation (split as above)

```
pd=a*(sdlinear*sdroc([], 'split', 0.5))
```

store confusion matrices in ROC object

```
r=sdroc(out, 'confmat')
```

▶ press 'c' in sddrawroc figure to show interactive confusion matrix

create ROC **with specific measures**

```
r=sdroc(out, 'measures', {'FPr', 'apple', 'TPr', 'apple'})
```

get performances at op.point 10

```
r(10)
```

get class weights

```
r.ops(10).data
```

estimate ROC for user-defined class weights

```
ops=sdops('w', rand(10000,3), a.lab.list)
r=sdroc(out, 'ops', ops)
```

constrain ROC to a subset with error on apple<0.3

```
r2=constrain(r, 'err(apple)', 0.3)
```

set curent operating point by index

```
r=setcurop(r, 100); ind=getcurop(r);
```

set curent operating point by minimizing error (max.performance)

```
r=setcurop(r, 'min', 'err(apple)')
r=setcurop(r, 'max', 'TPr(apple)')
```

set operating point **minimizing the cost** (confmat is stored)

```
M=ones(3); M(1,2)=10; r=sdroc(out, 'confmat');
r=setcurop(r, 'cost', M)
```

General commands

display perClass version

```
sdversion
```

provide on-line feedback, submit bug reports

```
sdfeedback
```

Interactive visualization

show interactive scatter plot

```
sdscatter(a)
```

sdscatter keyboard commands

- change feature →/←/↑/↓
- change z-order of classes +/-
- cycle through classes one at a time </>
- legend l
- hide current class h
- show only this class o
- class to top t
- switch to label set 1:9
- switch between full data set and subset axes a
- show feature distributions d
- return to previous sample filter f
- show all samples (remove filter) F

classifier decisions (also in multi-dimensional space)

```
pd=a*(sdlinear*sddcide); sdscatter(a,pd)
```

▶ change color of decision backdrop under cursor c

open also connected **ROC plot for scatter/image view**

```
sdscatter(a,p*r, 'roc'); sdimage(im,p*r, 'roc')
```

show feature distributions

```
sdfeatplot(a)
```

▶ right-click to control interactive threshold selection

open image view **open image view from file**

```
sdimage(im) sdimage('image.png')
```

return **image data set**

```
a=sdimage(im, 'sddata')
```

return **image matrix**

```
im=sdimage(a, 'matrix')
```

▶ get image info (original image size) with **getiminfo(a)**

▶ view data subset as image **sdimage(a(1:1000))**

change image band ↑/↓

toggle label layer space

cluster with k-means c

apply pipeline to image and show decisions d

compute local image features (histograms, textures.)

```
b=sdextract(a, 'block', 8, 'step', 4, 'feat', 'hist')
```

Cluster analysis

cluster data with k-means algorithm into 10 clusters

```
b=sdcluster(a, sdkmeans, 10)
```

train k-means model returning cluster labels

```
pd=sdkmeans(a, 10, 'cluster') *sddcide
```

get labels

```
lab=a*pd
```

protect clustering from outliers

```
pr=sdreject(pd, a)
```

Dimensionality reduction

Principal Component Analysis

```
sdpca
```

Fisher projection

```
sdlda
```

Proximity representation

```
sdprox
```

Scaling data

```
sdscale
```

Feature selection

```
sdfeatsel
```

select fixed feature subset using regular expression

```
pf=sdfeatsel(a, [1 5 10]) sdfeatsel(a, '/substr')
```

▶ see **pf.lab** for feature names

Select features with non-zero variance **sdfeatsel(a, 'var>0')**

Available classifiers

d ... can be used for detection

Nearest mean

```
d sdnmean
```

Fisher linear discriminant

```
sdfisher
```

Gaussian model (for detection)

```
d sdgauss
```

Linear discriminant

```
sdlinear
```

Quadratic discriminant

```
sdquadratic
```

Parzen

```
d sdparzen
```

Mixture of Gaussians

```
d sdmixture
```

k-NN

```
d sdknn
```

k-means prototype extraction

```
d sdkmeans
```

Support Vector Machine

```
sdsvc
```

Neural network

```
sdneural
```

Naive Bayes

```
d sdnbayes
```

Decision tree

```
sdtree
```

Random forest

```
sdrandforest
```

Minimum distance classifier

```
d sdmindist
```

Look-up table (2D classifier approximation)

```
sdlut
```

Standard performance measures

		decisions		sum
		target	non-target	
true labels	target	TP	FN	Nt
	non-target	FP	TN	Nn

Accuracy

```
FN/Nt+FP/Nn
```

True positive ratio (recall,sensitivity)

```
TPr=TP/Nt
```

True negative rate (specificity)

```
TNr=TN/Nn
```

Precision (purity)

```
TP/(TP+FP)
```

Detection rate

```
(TP+FP)/Nt
```

ROC measures: mean-error, class-errors, TP,TN,FP, FN, TPr,FPPr,TNr,FPPr, sensitivity, specificity, precision, posfrac, detrate