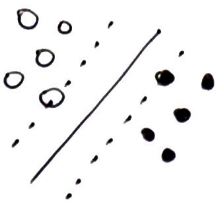
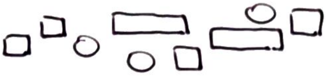
Neuroscience Using Machine Learning



SVM: the separating hyperplane that best separates the data is the maximal distance between the closest data points in the two classes.

KM: $\sum_{i=1}^{K} \sum_{x_j \in S} \| x_j - \mu_i \|^2$ lowering centroid-to-data distances

data features: 

with K=2, □ and ▭ are more similar and so clustered together

Magnetoencephalography (MEG) → 306 sensors → 306 columns

MEG_data 125x306 → 125 trials: 25 images each trial

stim_ID 1x125 → image ID (1→25) seen by the subject

cat_ID 1x125 → scene category (1→5) ~ stimulus for each image
→ 5 clusters excepted

MEG_data(i, j) contains response in the j^{th} sensor to the i^{th} stimulus.

$\boxed{IDX = Kmeans(X, 5)}$ % with $\boxed{X = MEGdata(:, [\overbrace{20, 273}^{\text{2 channels/sensors}}])}$

% with X an 125x2 matrix

% Kmeans() returns an n x 1 vector containing the cluster indices of each point

we want to plot entries of vector B that have a X at their corresponding location in vector A against entries of vector that have a Y at their corresponding location in vector A.

→ conditional coloring

```
figure
load('Kmeans_results.mat');
x1 = X(IDX==1,1); y1 = X(IDX==1,2);
plot(x1, y1, 'b.', 'MarkerSize', 16);
hold on
x2 = X(IDX==2,1); y2 = X(IDX==2,2);
% ...
% random centroid points generated each time Kmeans() is called.
```

SVM binary classifier → Separates beach from forest.

MetaData → train_cat_labels and test_cat_labels data

cross-validate

svmStruct = fitcsvm(train_data, train_cat_labels, 'Standardize', 'on');

↳ parameters that describe the hyperplane

% test the classifier

pred = predict(svmStruct, test_data)

actual = test_cat_labels' % transposition

A = sum(pred == actual);

accuracy = A / length(pred)
