

# 15-494/694: Cognitive Robotics

Dave Touretzky

Lecture 15:

Machine learning with  
scikit-learn

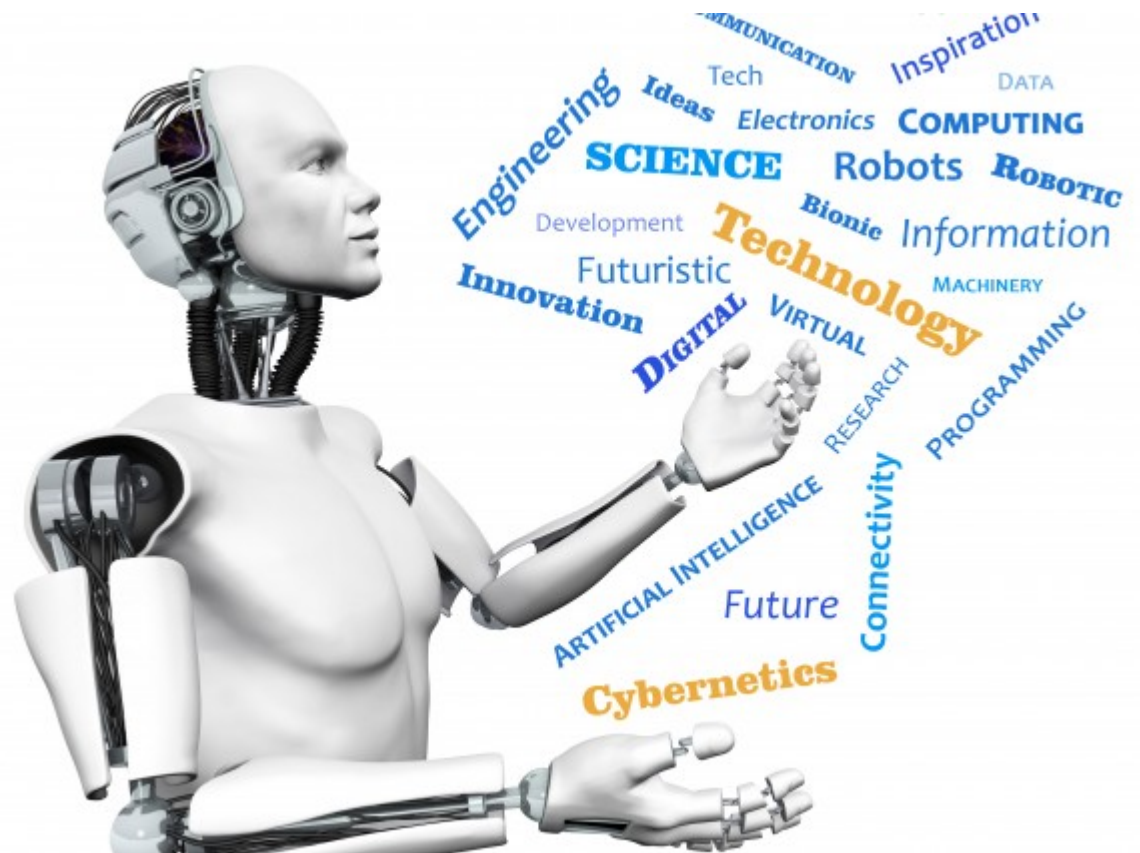


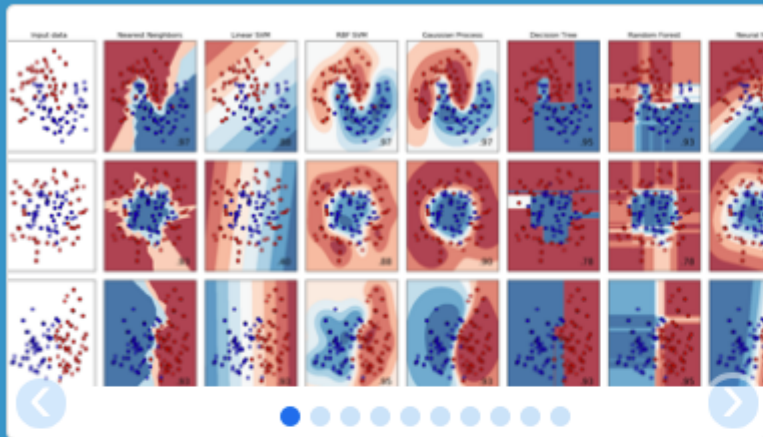
Image from <http://www.futuristgerd.com/2015/09/10>

# Machine Learning

- ML is a branch of Artificial Intelligence.
- “Learning” does not mean human-like learning.
- It means extracting information from data. This has many uses in robotics.
- Types of learning algorithm:
  - Supervised (labeled data)
  - Unsupervised (unlabeled data)
  - Reinforcement

# scikit-learn

- Open source collection of machine learning algorithms implemented in Python.
- Documentation at [scikit-learn.org](http://scikit-learn.org)
- Install: **`pip3 install scikit-learn`**
- To use the module:  
**`import sklearn`**



# scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

## Classification

Identifying to which category an object belongs to.

**Applications:** Spam detection, Image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, ... — Examples

## Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.  
**Algorithms:** SVR, ridge regression, Lasso, ... — Examples

## Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes  
**Algorithms:** k-Means, spectral clustering, mean-shift, ... — Examples

## Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency

**Algorithms:** PCA, feature selection, non-negative matrix factorization. — Examples

## Model selection

Comparing, validating and choosing parameters and models.

**Goal:** Improved accuracy via parameter tuning  
**Modules:** grid search, cross validation, metrics. — Examples

## Preprocessing

Feature extraction and normalization.

**Application:** Transforming input data such as text for use with machine learning algorithms.  
**Modules:** preprocessing, feature extraction. — Examples

# Supervised Learning

- For each training point, there is a desired output value.
- Error measure: difference between actual output and desired output.
  - Example: sum-squared error

$$E = \frac{1}{2} \sum (d_i - y_i)^2$$

- Learning adjusts the model parameters to reduce the error.

# Unsupervised Learning

- Data points are unlabeled: there is no “correct” answer.
- Learning discovers structure in the data.
- Examples:
  - Clustering: finding categories.
  - Dimensionality reduction: finding key features and relationships between features. Useful for data compression.

# Reinforcement Learning

- Used for *sequential decision problems*.
- Model is trained via a reinforcement signal that tells it how well it is doing.
- We don't tell it the right answer, just reward it when it does well.
- Example:
  - Learning to play a game by reinforcing wins. Program can learn by playing against itself.

# Supervised Learning: Classification

- Desired outputs may be binary, or probabilities of class membership.
- Examples:
  - Tell “spam” from “not spam”.
  - Distinguish images containing cats from those without cats.
  - Recognize handwritten digits 0-9.



# Supervised Learning: Regression

- Desired outputs are continuous, possibly vectors.
- Examples:
  - Interpolate values of a nonlinear function.
  - Predict stock prices.
  - Calculate inverse kinematics solutions for a non-linear robot.

# Parametric vs. Non-Parametric Models

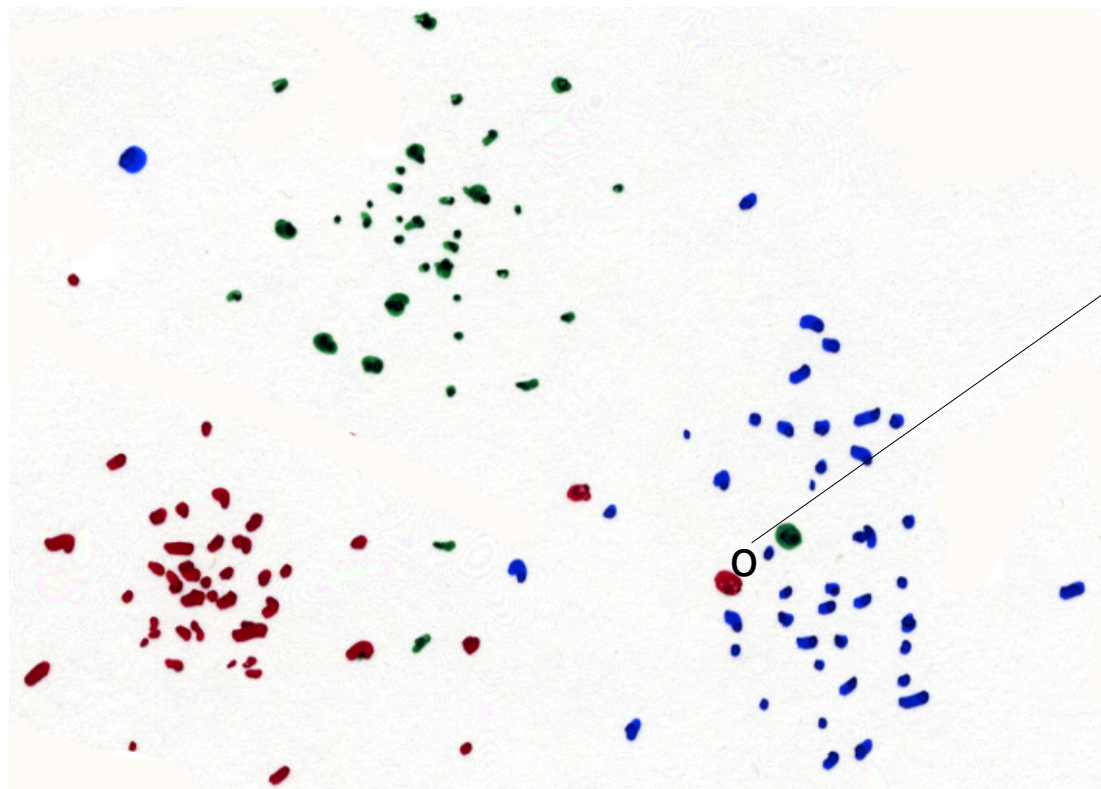
- Parametric models describe data using equations with a small number of parameters.
  - Example: Gaussian distribution.
  - Parameters are mean  $\mu$  and variance  $\sigma$
- **Pros:** compact representation; easy to test new data points.
- **Cons:** what if your data doesn't fit the equation?

# Parametric vs. Non-Parametric Models

- Non-parametric models don't make any assumptions about the distribution of the data. The data represents itself.
  - Particle filters are non-parametric models.
- **Pros:** “training” is instantaneous. Can represent arbitrary distributions.
- **Cons:** can take a lot of memory to store all the data, and classifying new points can be slow.

# Nearest-Neighbor Classifier

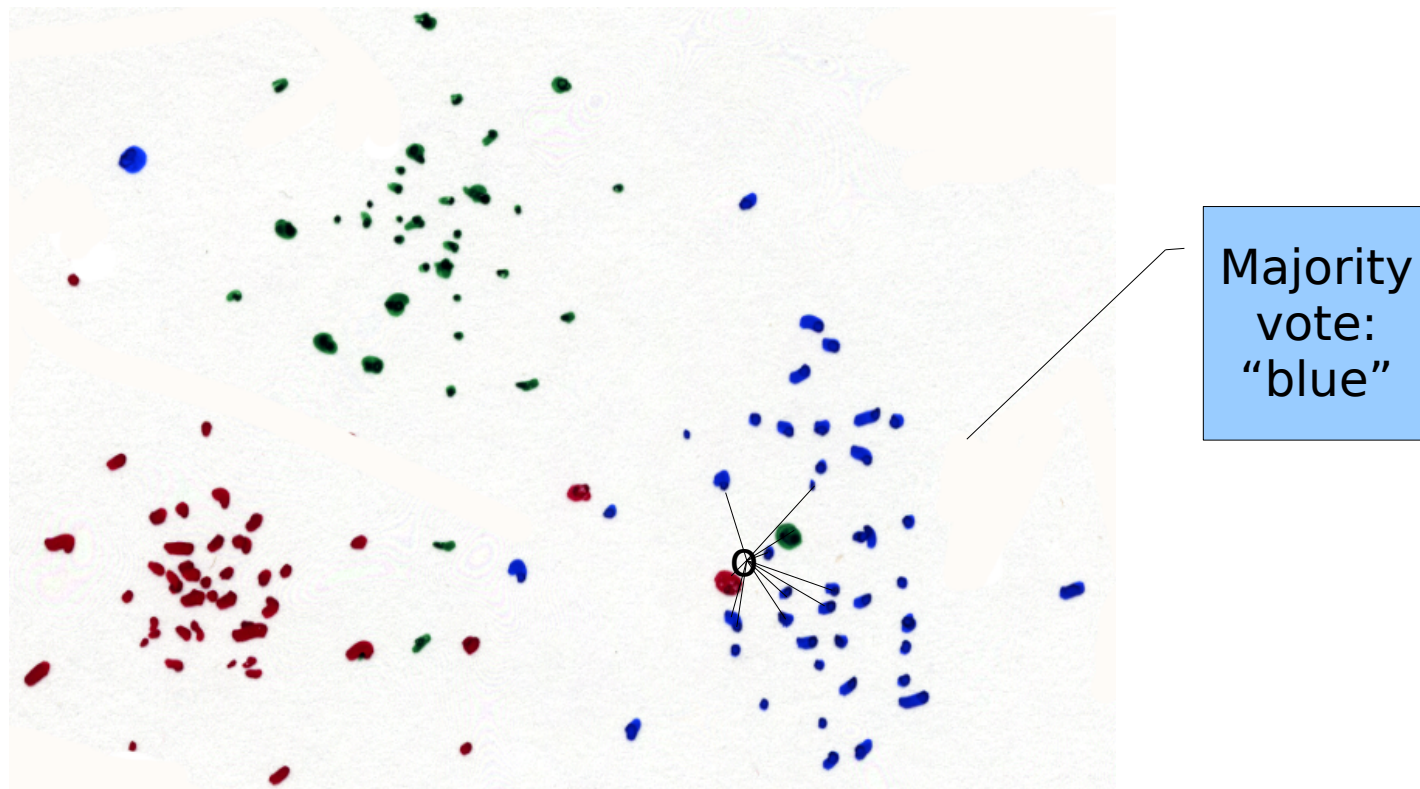
- Simplest non-parametric classifier.
- Noisy data can be a problem.



Will get  
this one  
wrong.

# k-Nearest-Neighbor

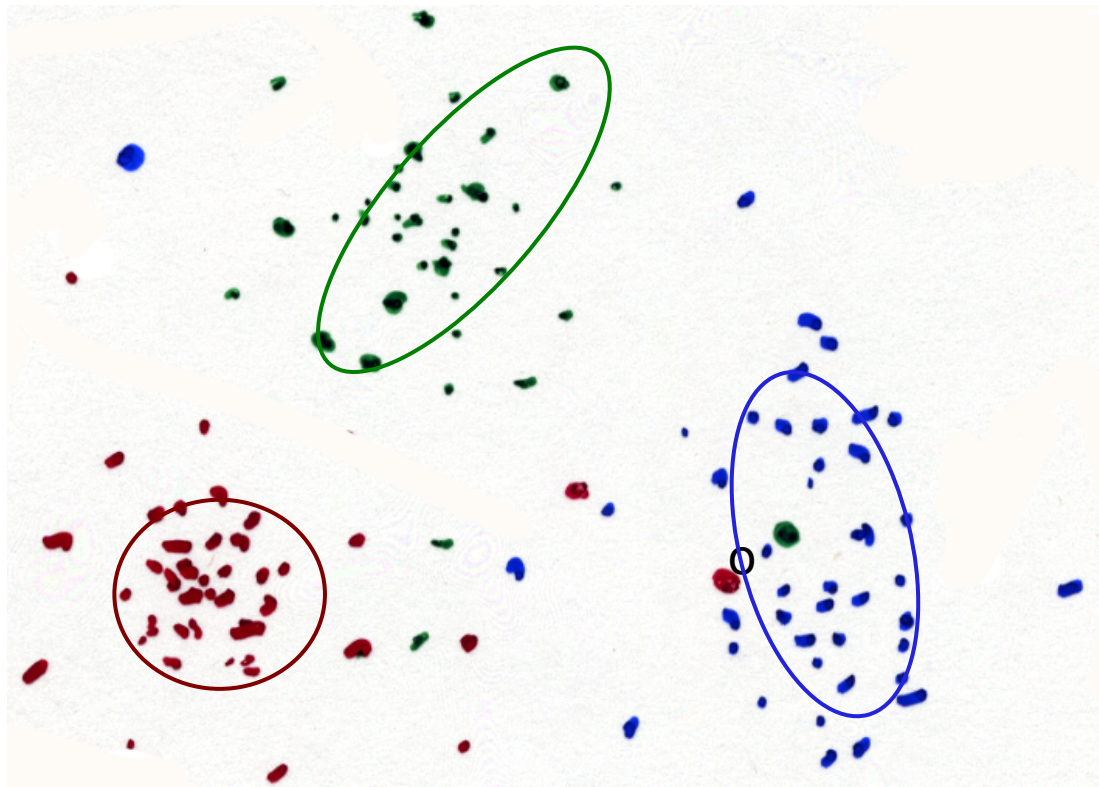
- Still a non-parametric classifier.
- Majority vote to find the correct class.



# Gaussian Mixture Model

- Parametric model: Gaussian distributions.

$$p(x \mid \mu, \sigma) = e^{\frac{-(x-\mu)^2}{\sigma^2}}$$



How do we find the correct values of the parameters?

Learning algorithm!

# Sample Learning Problem: Color Classes

- Assume objects come in a small number of colors.
- We want to know what the colors are.
  - This is a clustering problem.
- Given a new object, we want to determine its color class.
  - This is a classification problem.



# Training Data: RGB Values





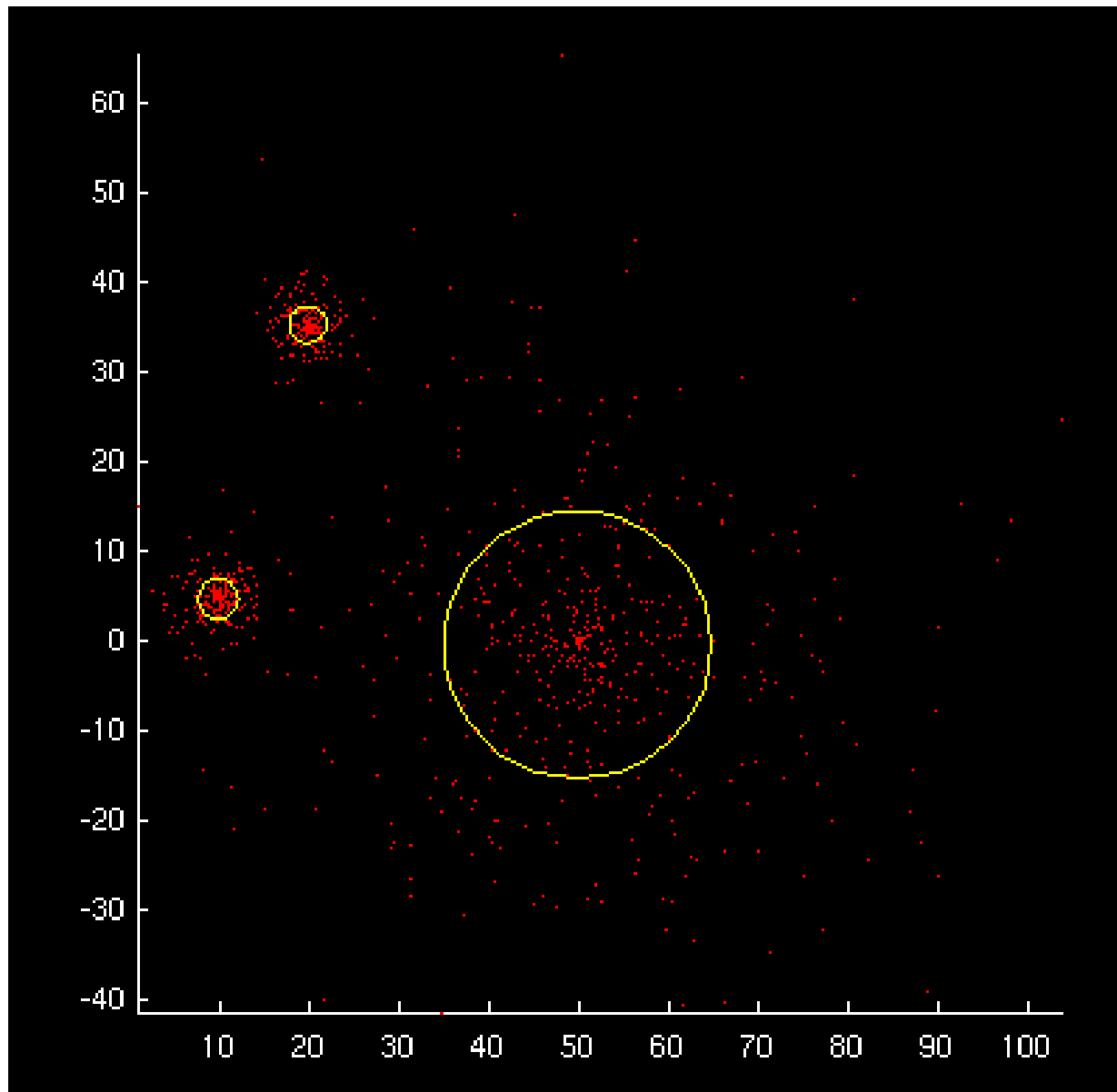
# Feature Space

- Three-dimensional feature space: RGB.
- $320 \times 240 = 76,800$  data points per image.
- Let's assume that each color class can be modeled as a gaussian distribution:
  - Mean color  $\mu$
  - Covariance matrix  $\Sigma$

# Expectation-Maximization Algorithm

- Unsupervised learning algorithm for finding clusters in data.
- Learns the  $\mu$  and  $\Sigma$  parameters for a set of gaussians.
- You must guess the number of classes.
- Runs quickly but can get stuck in local minima.

# E-M Clustering



# E-M Algorithm

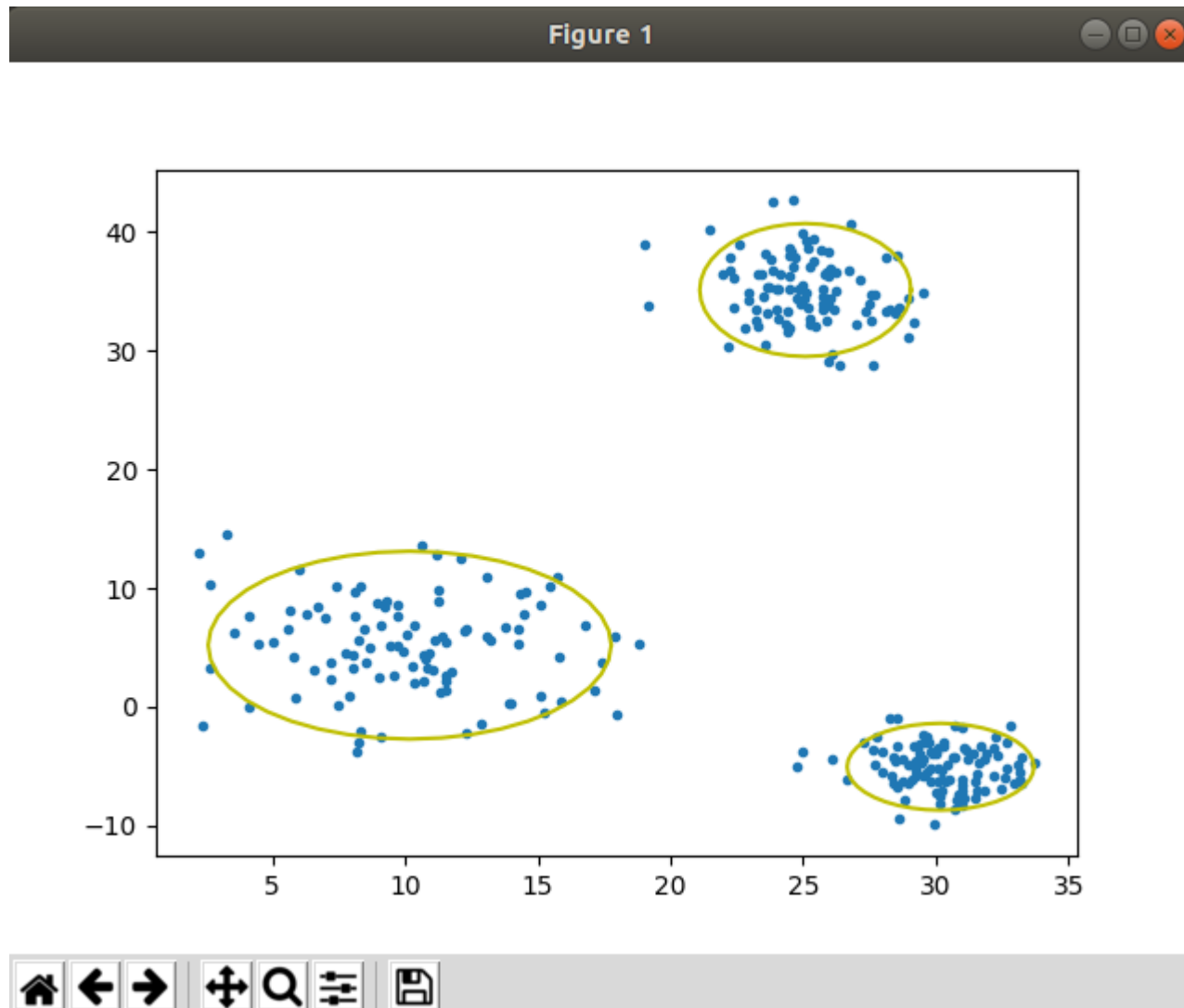
- Expectation step:
  - For each point  $\mathbf{x}$ , for each gaussian  $(\mu_i, \Sigma_i)$ , calculate the likelihood of  $\mathbf{x}$  having been generated by the  $i$ -th gaussian:  
 $P(\mathbf{x} \mid \mu_i, \Sigma_i)$ .
- Maximization step:
  - For each gaussian, recalculate its mean and covariance  $\mu_i, \Sigma_i$  based on the likelihood-weighted data points.
- Repeat for several iterations.

# E-M in scikit-learn

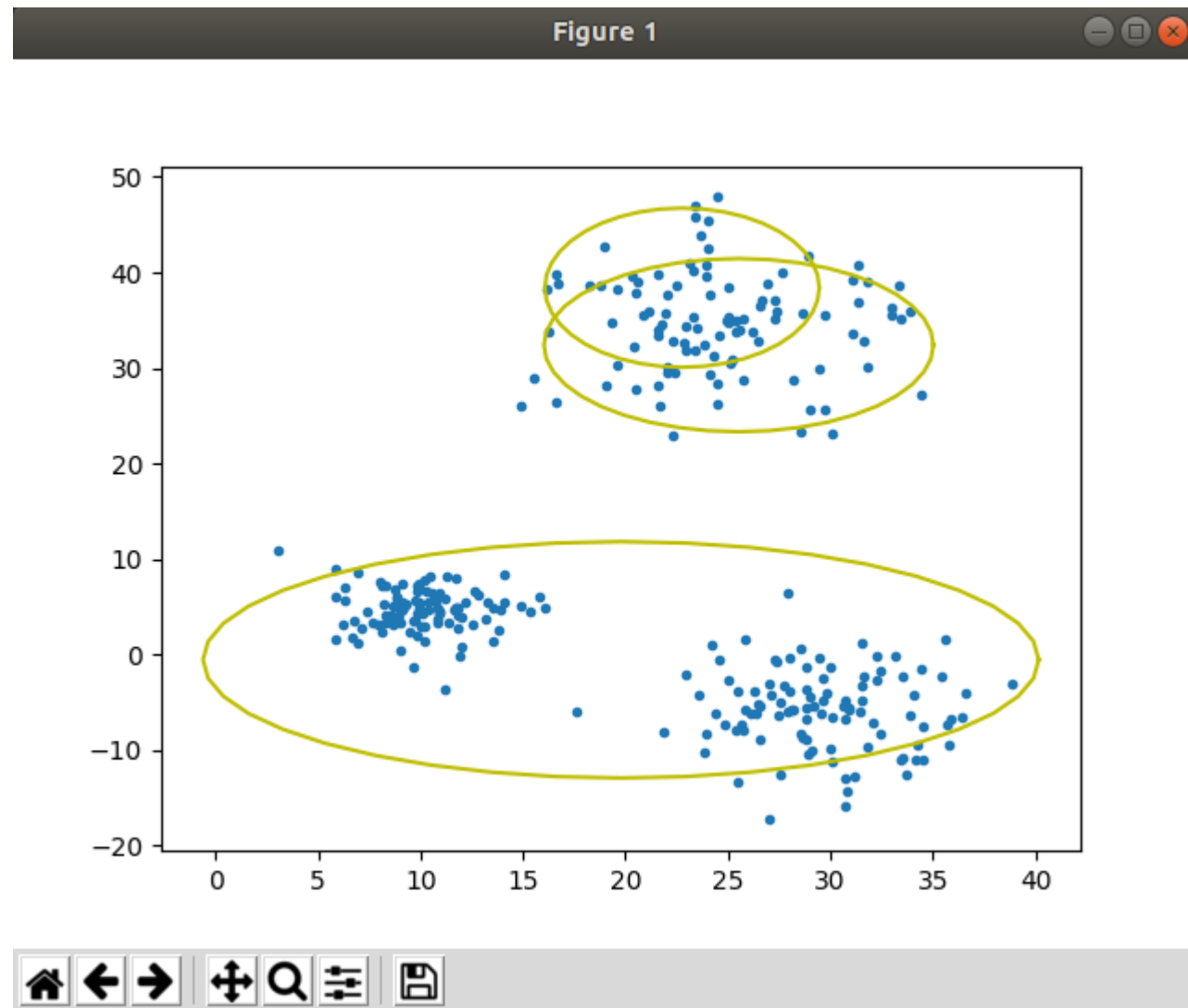
```
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=7)
gmm.fit(data)
means = gmm.means_
covariances = gmm.covariances_

classes = gmm.predict(new_data)
```

# emdemo.py



# EM doesn't always succeed

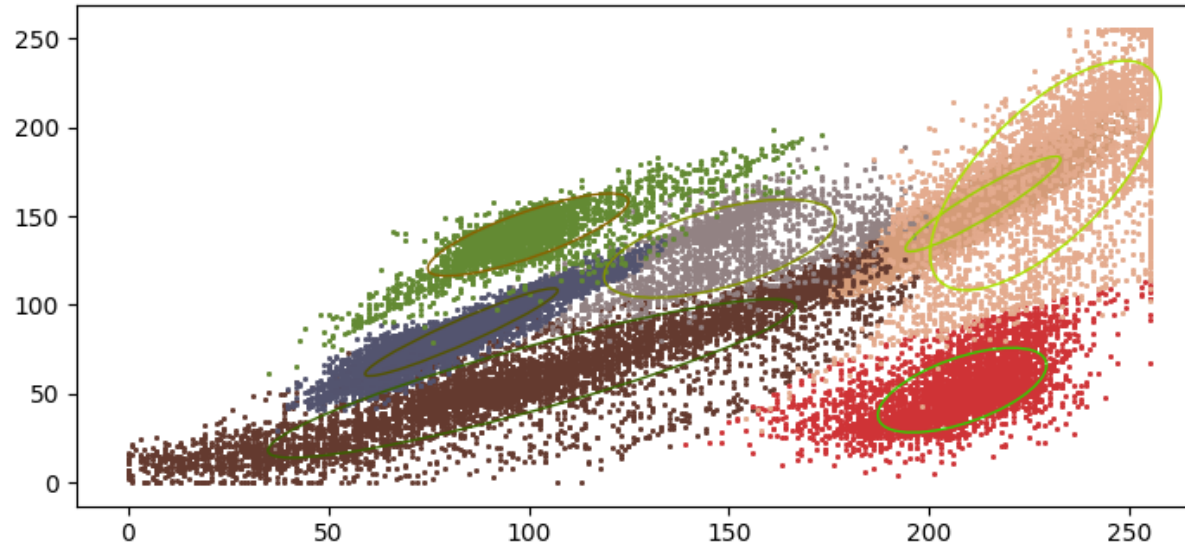


# Color Classification: Visualizing the Result

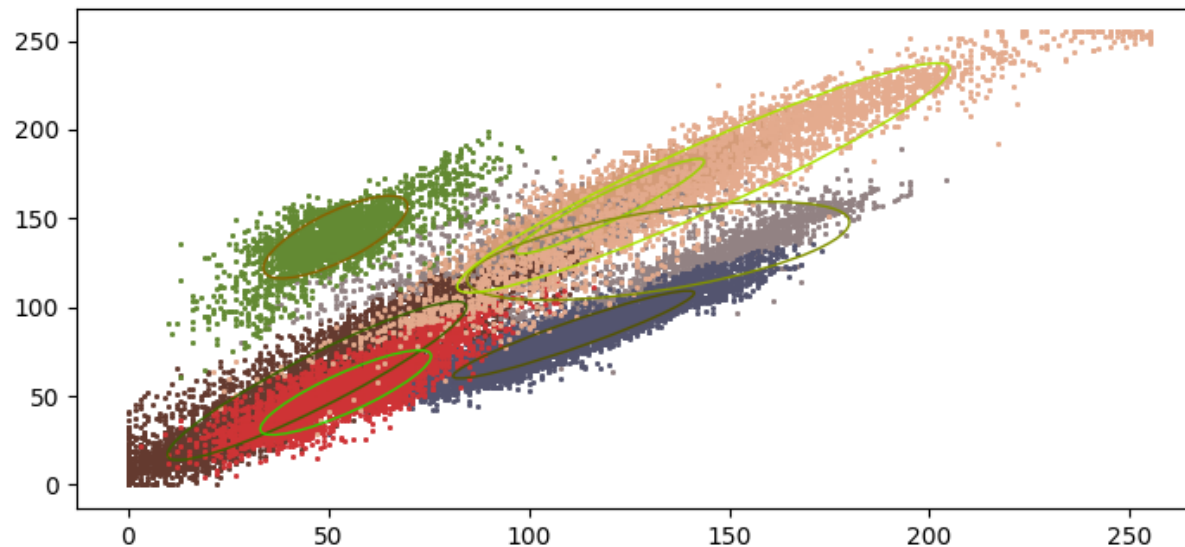
- Each color class is modeled as an ellipsoid in 3D space (RGB space).
- Too hard to plot. So instead:
  - Generate R-vs-G and B-vs-G plots.
  - Draw the ellipses in feature space determined by the covariance values.



# Scatter Plots With Gaussian Ellipses



Red  
vs.  
Green

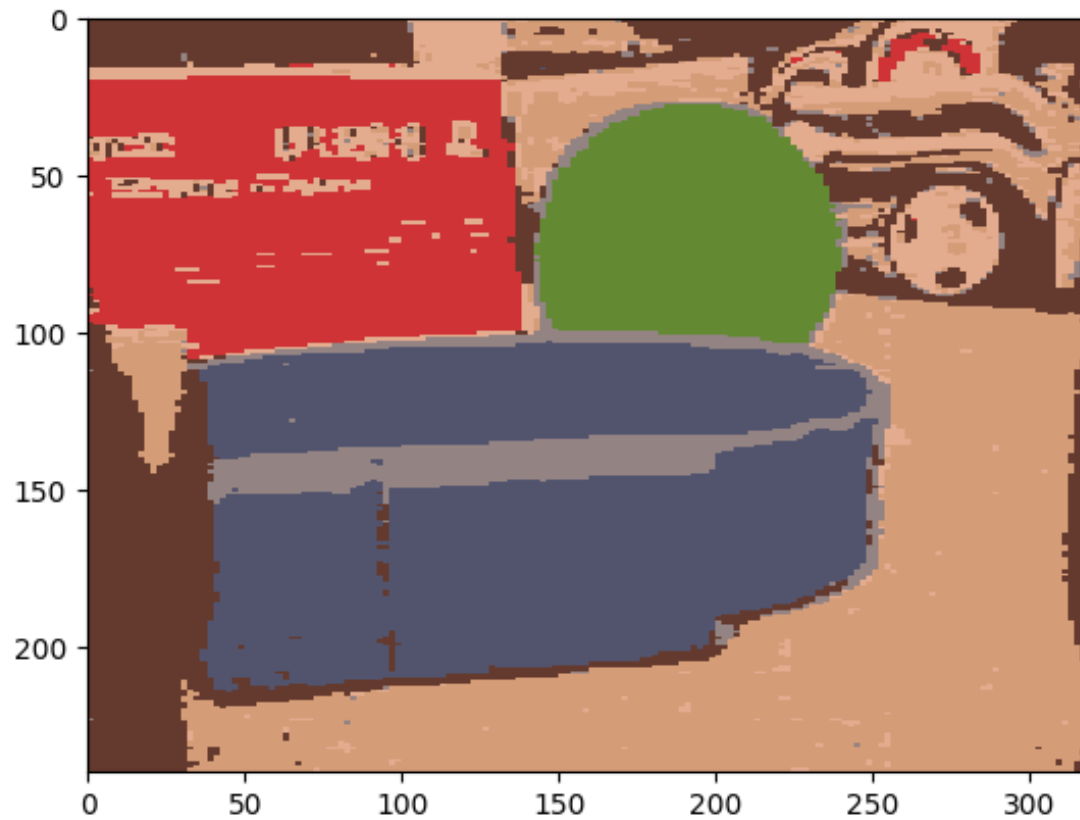


Blue  
vs.  
Green

# Classification

- Once we've learned the color classes, we can assign a class to each pixel.
  - This gives us a color-quantized image.
- Can then use these color classes to classify new images the same way.

# Classified Image



Original Image



# Refinements

- Can detect local minima by checking the density of points near the mean of the gaussian.
- Split/merge EM can reallocate gaussians if some are being wasted and others are spread between two clusters.
- BayesianGaussianMixture class in scikit-learn can infer the number of effective components.