SVMs

gabe heafitz

okay, so, wonderful, but what can we do with it?

— handwriting recognition
— identifying the most promising sales leads
— rescue blob spikes
— e-commerce fraud detection?
— fraudulent transactions at the register?

plan of attack

1. demo of SVM running on some example data
2. interpreting the results
3. review of fundamental concepts of machine learning; how is machine learning helpful?
4. how do SVMs work?
5. how do you train an SVM?
6. some classic usage examples
7. some publically available implementations

demo

train an SVM against example data

example: can we predict heart disease?

what does our data look like?

heart dataset from UCI

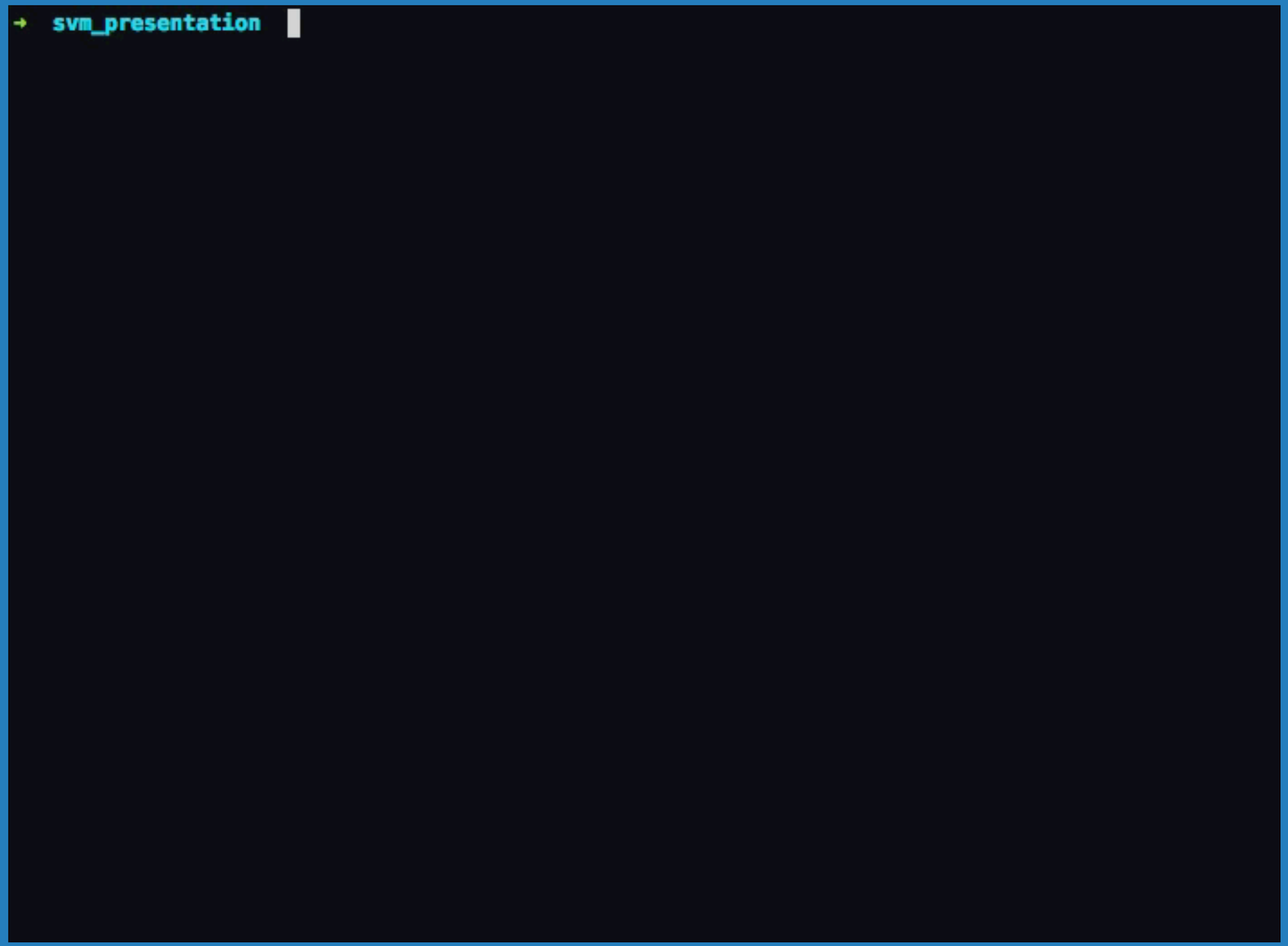
```
age  sex      bp                          hr                                  ?
63.0 1.0 1.0 145.0 233.0 1.0 2.0 150.0 0.0 2.3 3.0 0.0 6.0 1
67.0 1.0 4.0 160.0 286.0 0.0 2.0 108.0 1.0 1.5 2.0 3.0 3.0 0
67.0 1.0 4.0 120.0 229.0 0.0 2.0 129.0 1.0 2.6 2.0 2.0 7.0 1
37.0 1.0 3.0 130.0 250.0 0.0 0.0 187.0 0.0 3.5 3.0 0.0 3.0 0
41.0 0.0 2.0 130.0 204.0 0.0 2.0 172.0 0.0 1.4 1.0 0.0 3.0 0
56.0 1.0 2.0 120.0 236.0 0.0 0.0 178.0 0.0 0.8 1.0 0.0 3.0 0
```

1. age
2. sex
3. chest pain type (4 values)
4. resting blood pressure
5. serum cholesterol in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2)
8. maximum heart rate achieved
9. exercise induced angina

...

okay, finally ready to SVM

let's download, compile, and run an open-source SVM library


svm_presentation

demo

evaluate the results

what ML is for

finding patterns in the data

machine learning uses statistics. in engineering, we're used to solving a deterministic problem where our solution solves the problem all the time. if we're asked to write software to control a vending machine, it had better work all the time, regardless of the money entered or the buttons pressed. there are many problems where the solution isn't deterministic.

that is, we don't know enough about the problem or don't have enough computing power to properly model the problem. for these problems we need statistics.
-- from Machine Learning in Action, by Peter Harrington

terminology

attribute

one of the columns in your data

examples from the heart data set:

- age
- gender
- chest pain type (this is a set of 4 nominal values)
- resting blood pressure
- serum cholesterol in mg/dL
- fasting blood sugar > 120 mg/dL

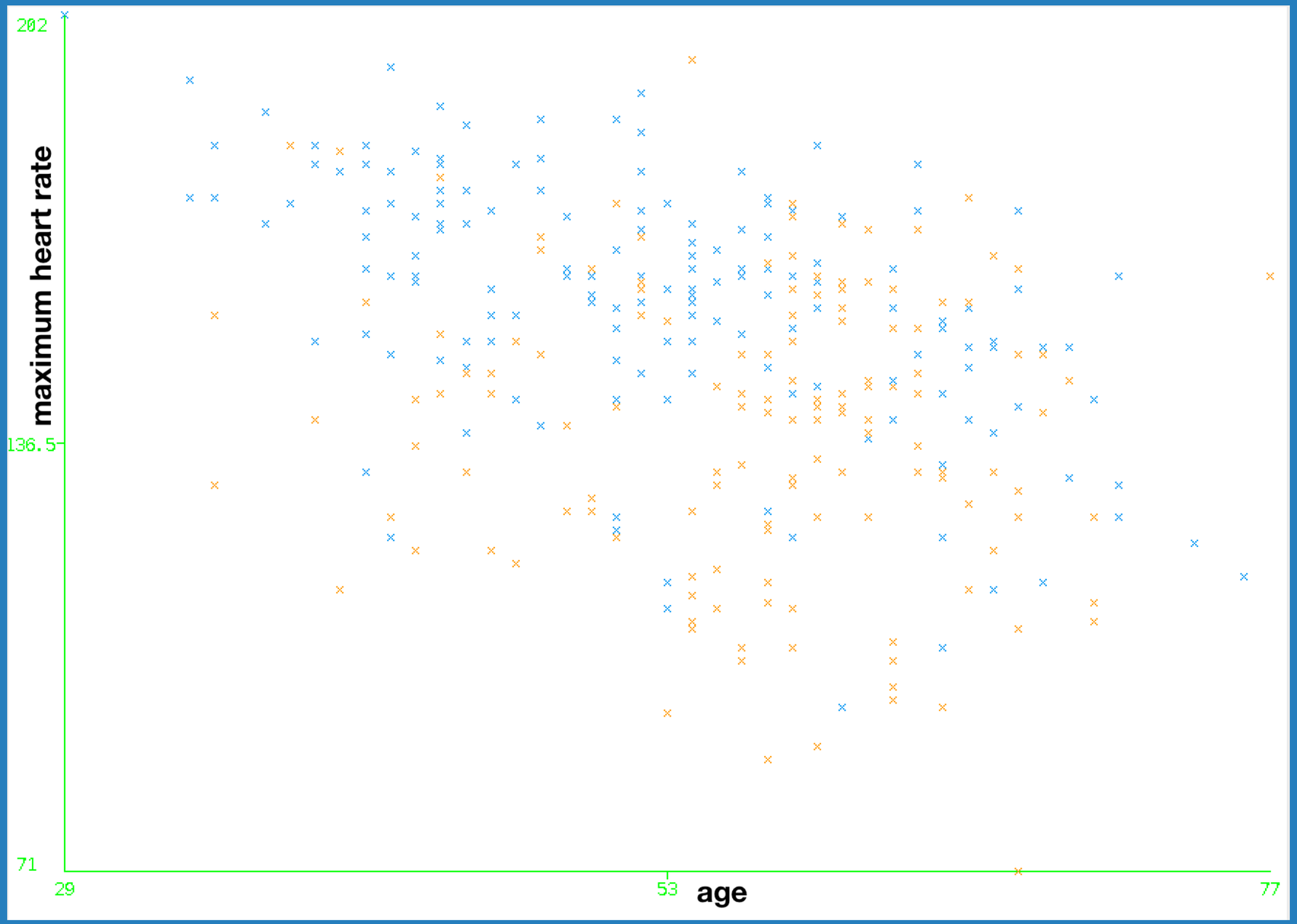
classification

tell me which of 2 classes the row of data fits in

example from the heart data set:
whether heart disease is present or absent

2 attributes of the heart data set

- age
- maximum heart rate


202
maximum heart rate
136.5
71
29
53
age
77

supervised vs. unsupervised

whether your data is already labeled with the "answers"

you divide your data into subsets

1. training
2. validation
3. production data

training data

validation data

knowledge representation

can a human interpret the patterns that the machine has found, or is the model an opaque black box?

how to train and use a machine learning model

step 1

collect a data set

step 2

encode the data set in such a way that the machine learning algorithm will be able to ingest

step 3

filter out any known noise or other useless attributes or rows in the data, so that the machine learning algorithm doesn't get distracted by something you know to be irrelevant

step 4

train the algorithm: "you feed the algorithm good clean data from the first two steps and extract knowledge or information." the "knowledge" or "information" *is* your model

step 5

exercise your model on the validation set -- this will show how well your model performs with unknown data

step 6

use your model in production. you may want to periodically re-train your model, as you collect more data in your system

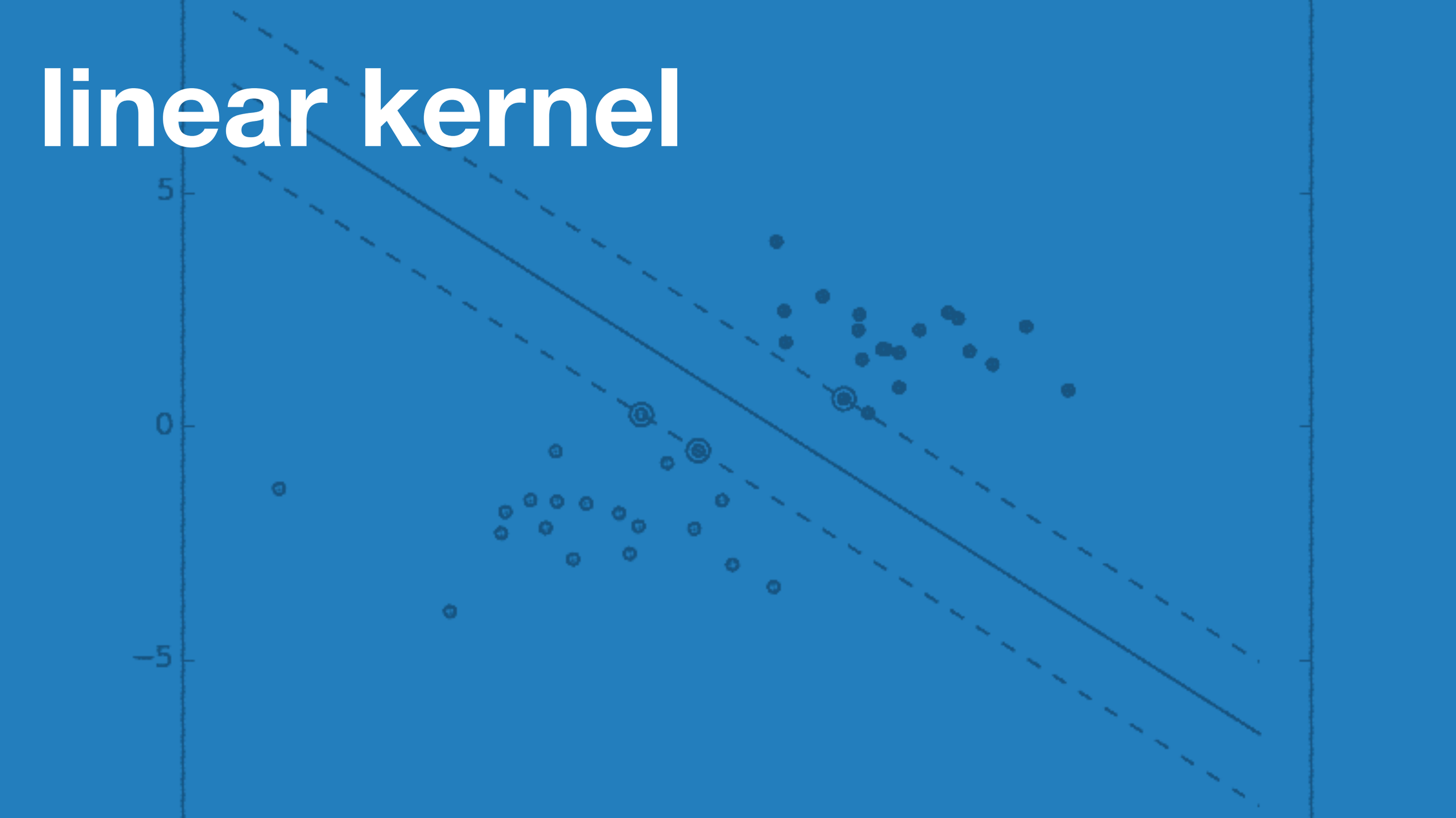
step 7[1]

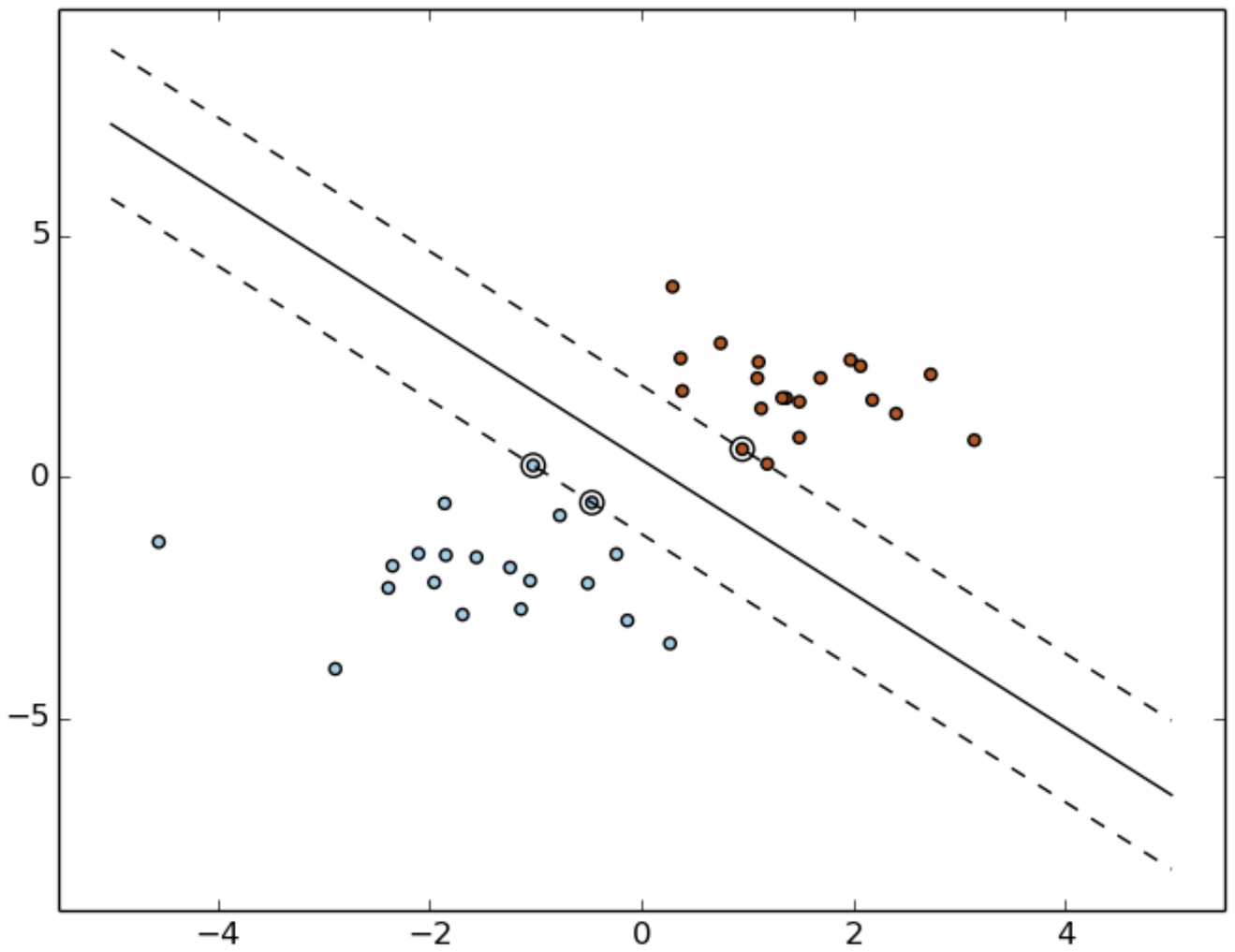
profit?

[1] This one's not from the Harrington book. I added this one myself.

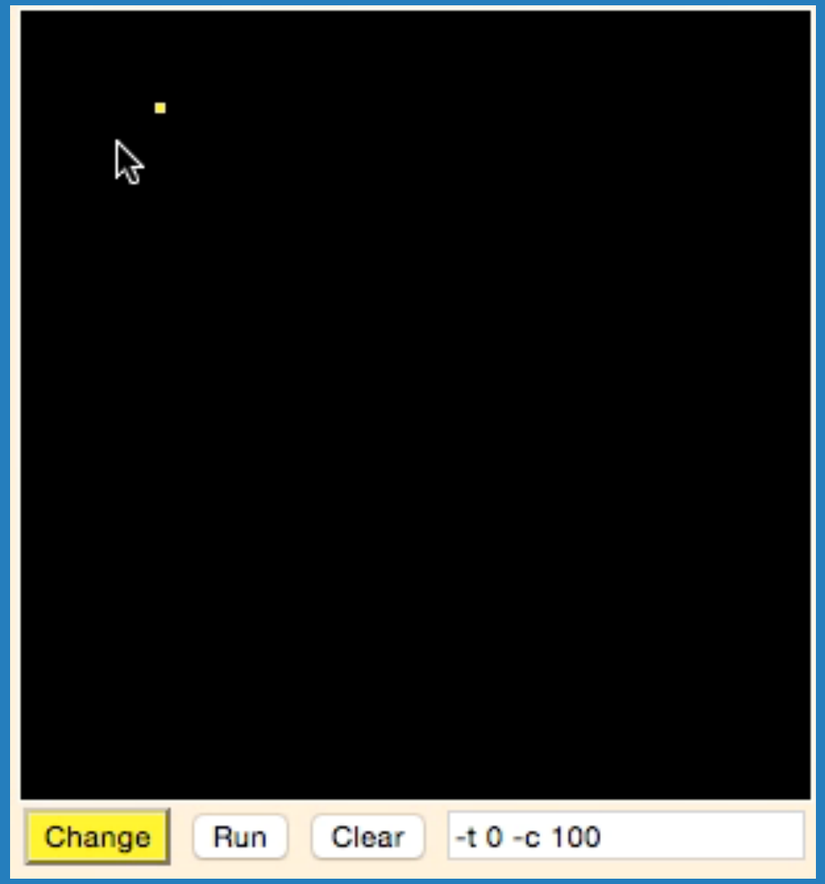
how SVMs work

linear kernel




5
0
−5
−4
−2
0
2
4

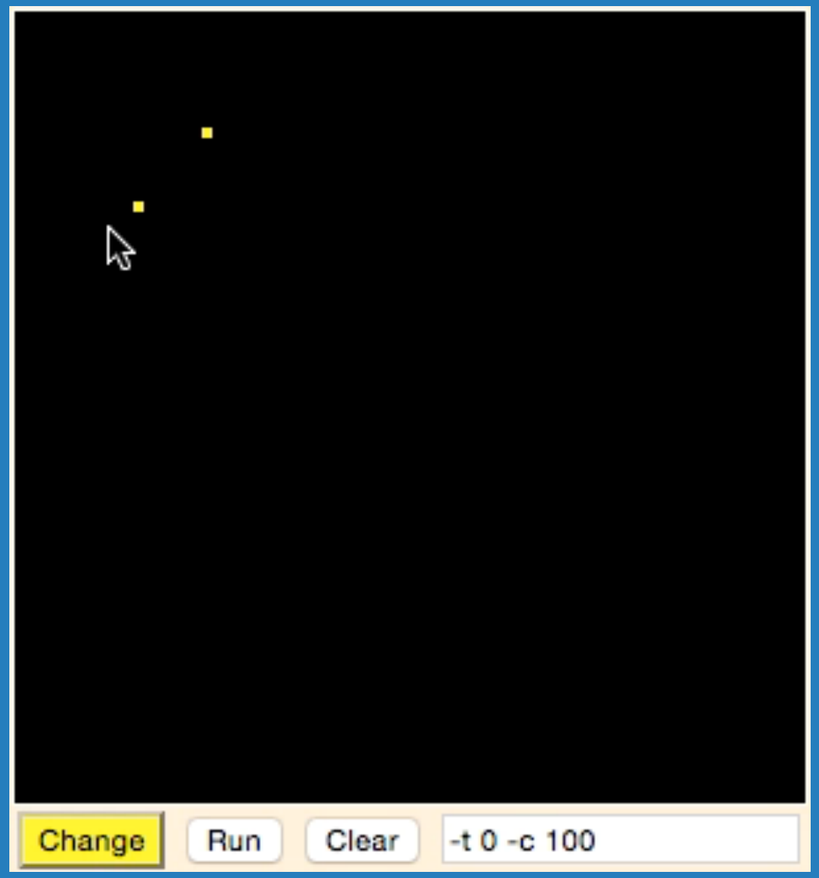
demo of linear kernel SVM



but what if the data is not linearly separable?

slack variables allow us some room for error and to find a line that doesn't necessarily separate the data perfectly, but still to do the best job we can

slack allows for some error



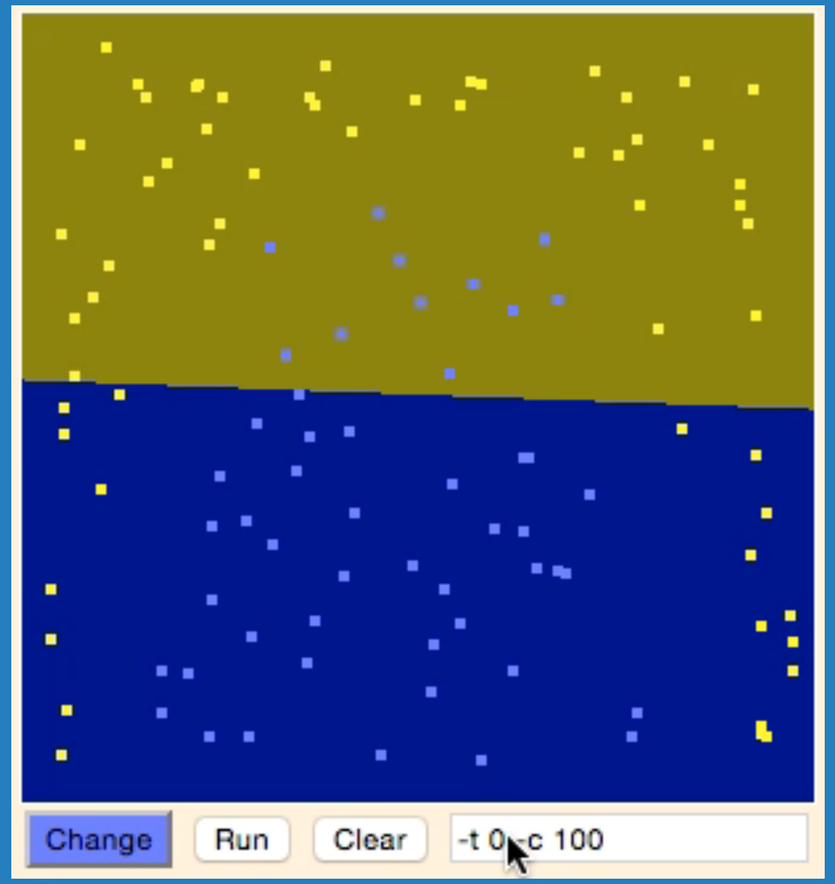
what if the data are *really* not linearly separable?

what if the data are *really* not linearly separable?

higher order SVMs

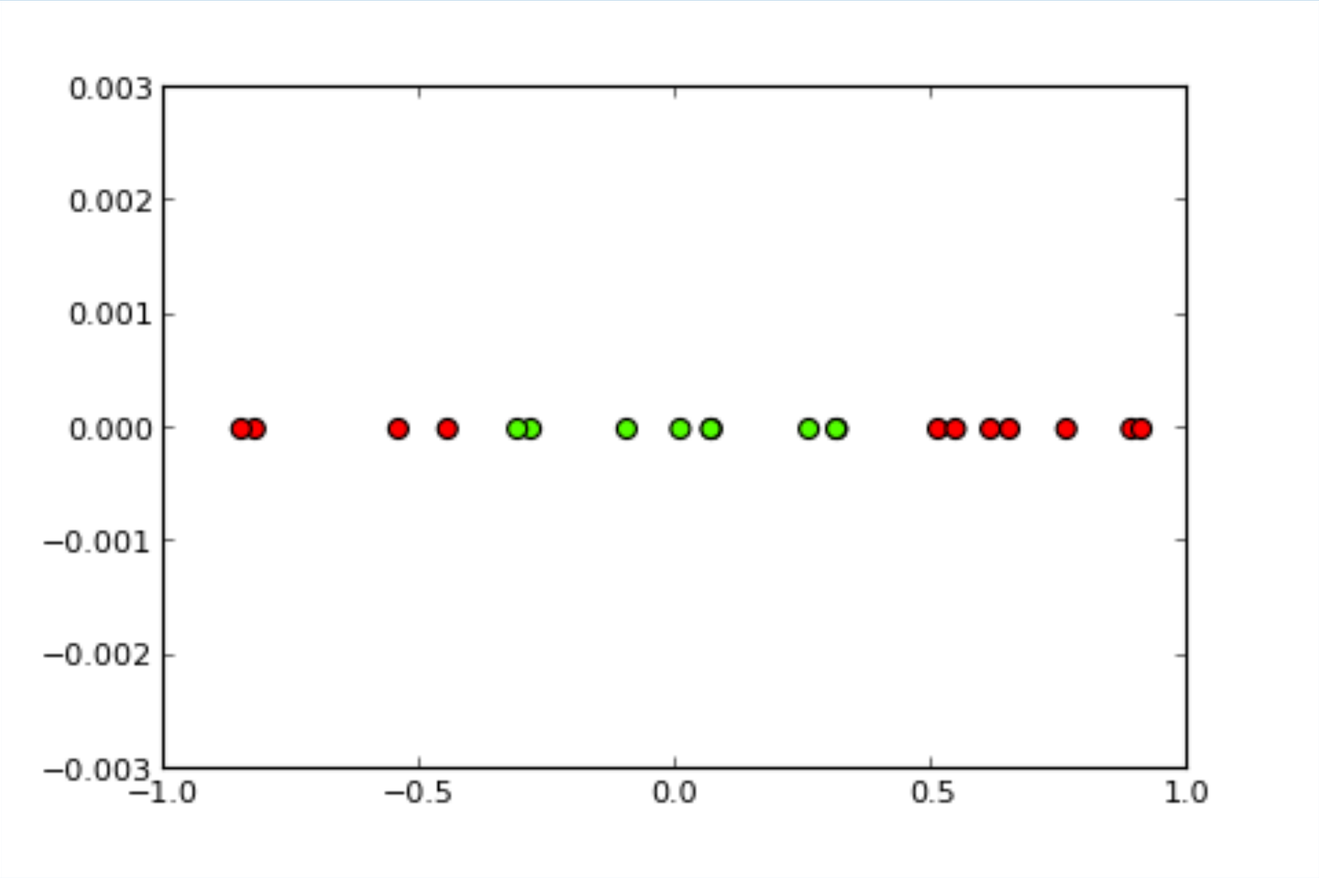
- polynomial function
- Gaussian radial basis function

higher order SVMs

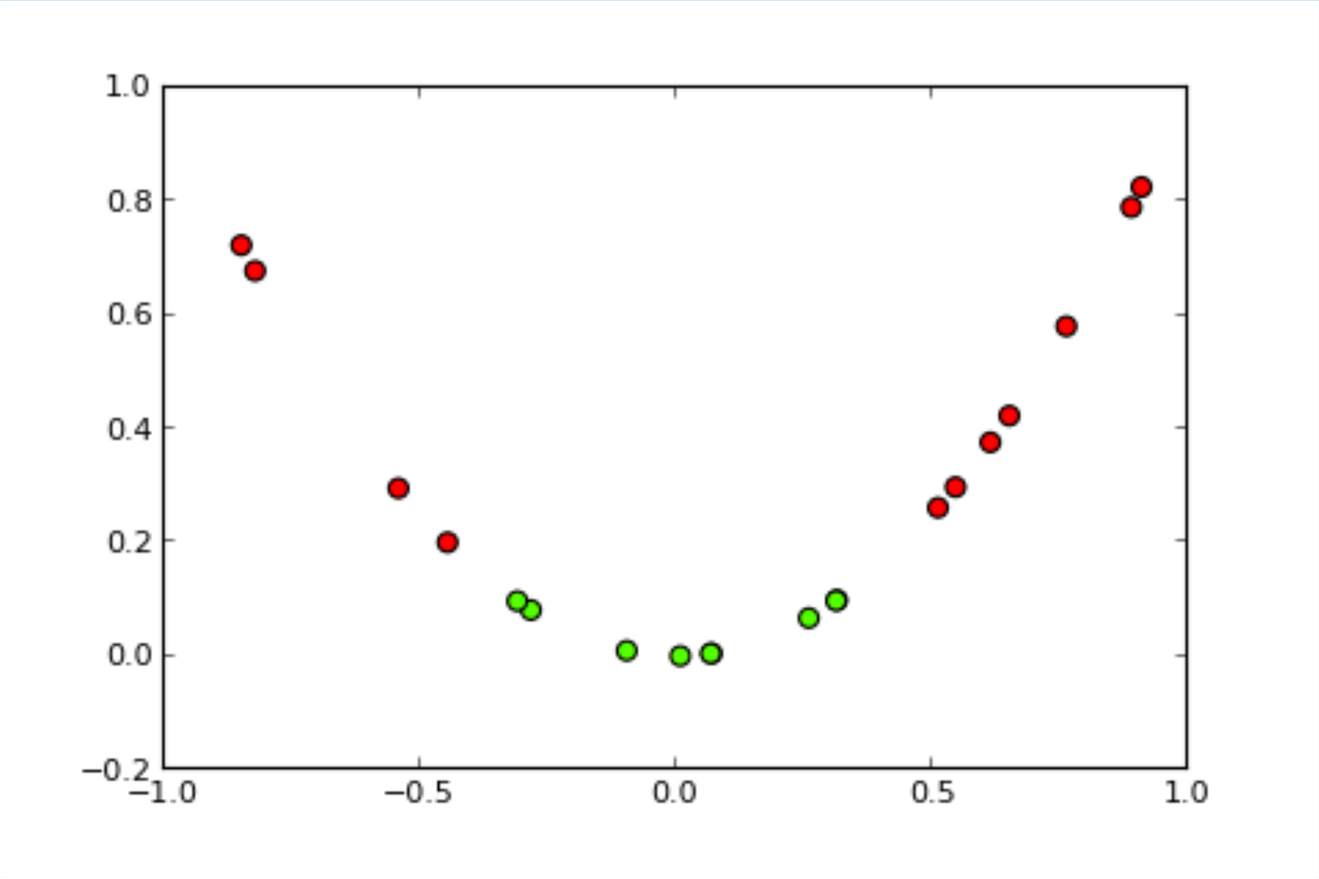


what is going on here?

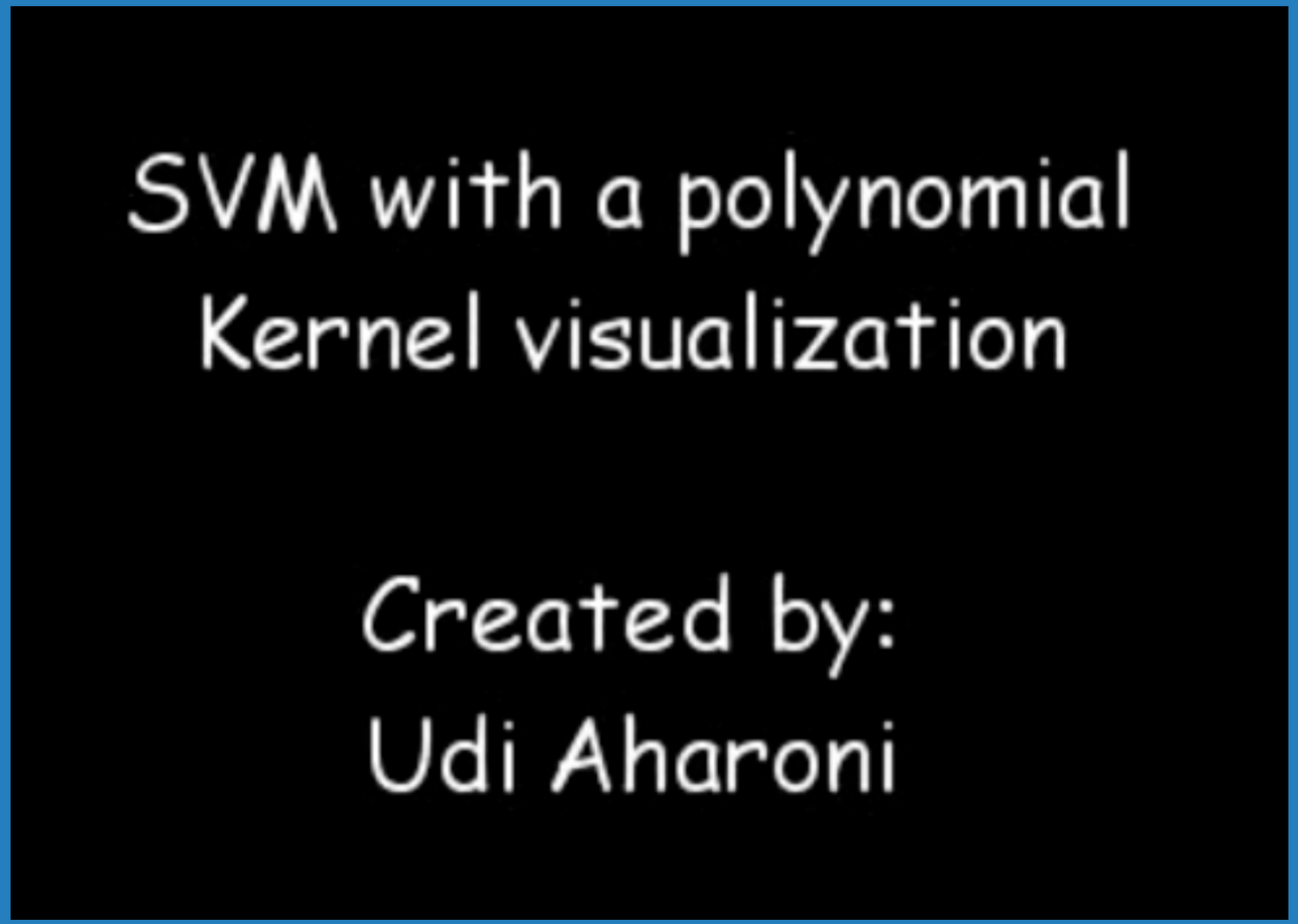
data that can't possibly be separated by a straight line

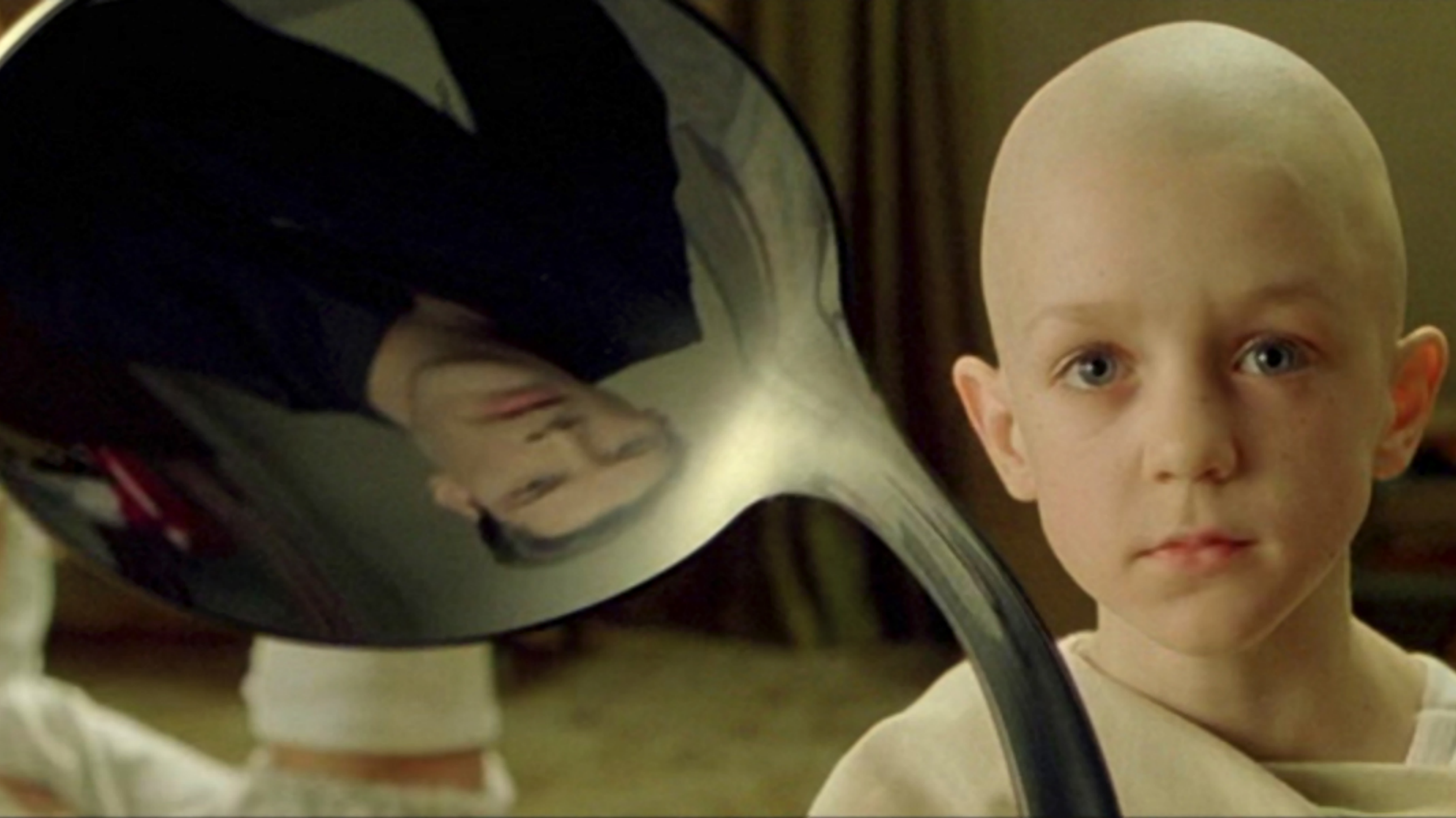


data that can't possibly be separated by a straight line



another example, this time in 3-D

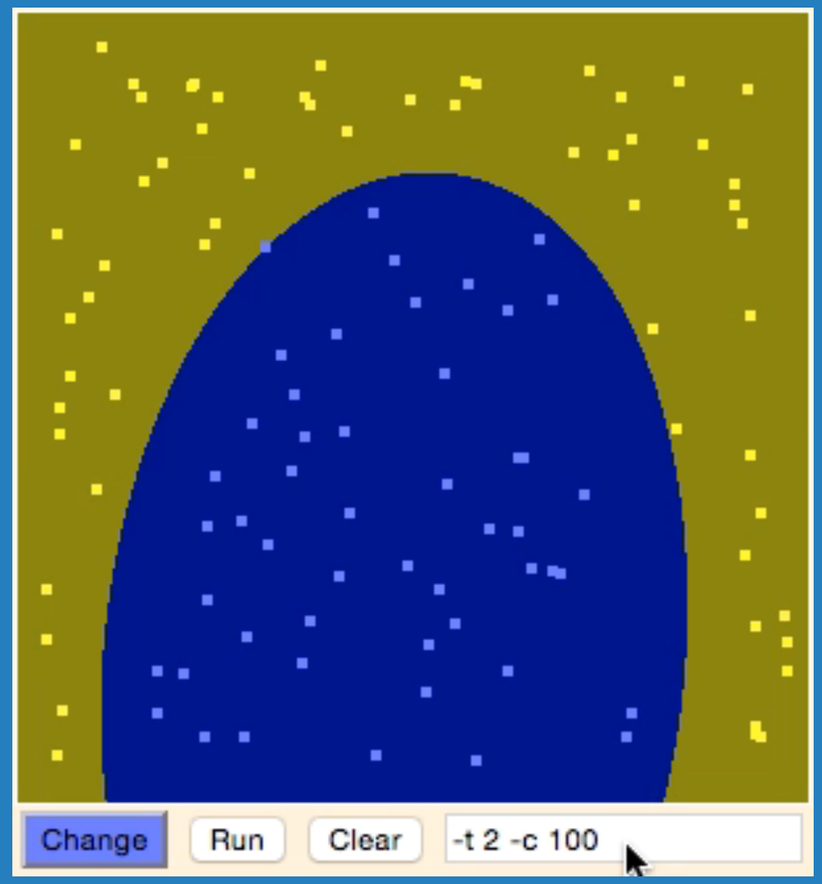




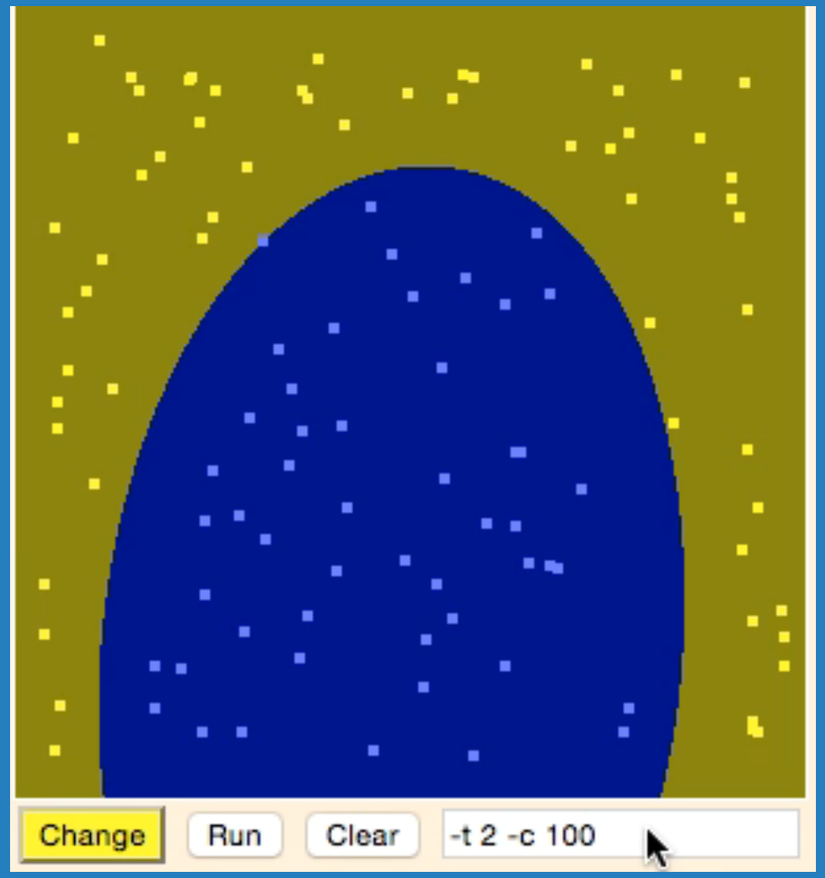
gaussian radial basis function parameters

- C
- gamma

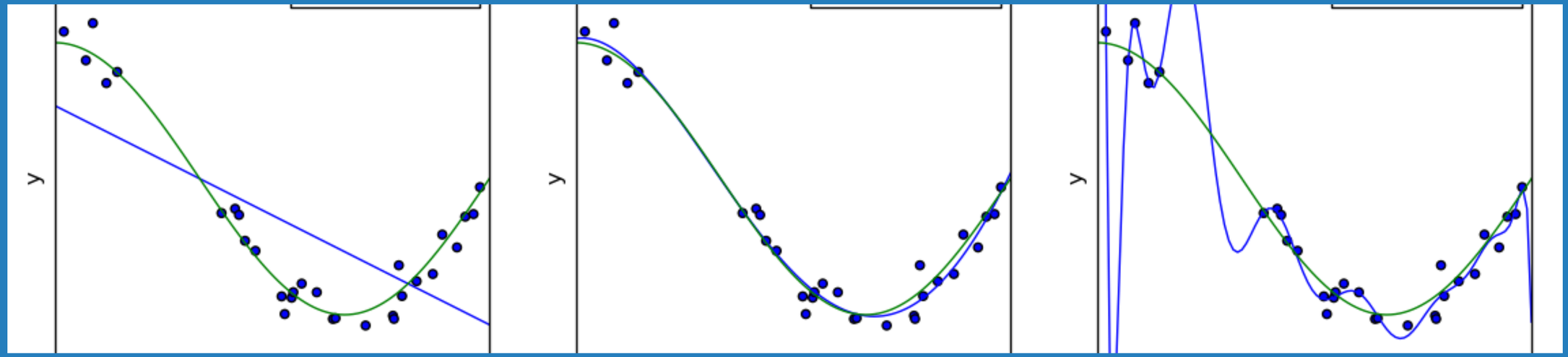
C, the cost parameter

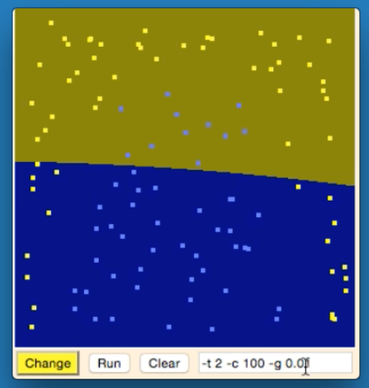


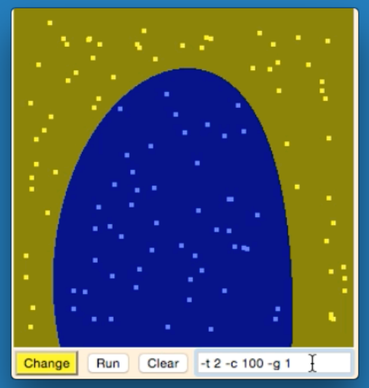
gamma, the spikiness parameter

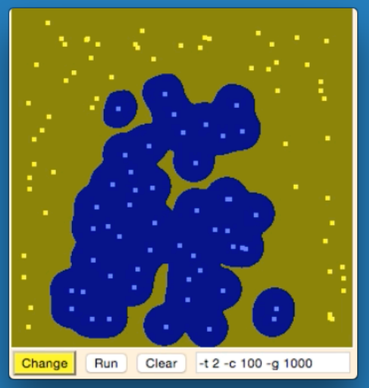


underfitting / overfitting









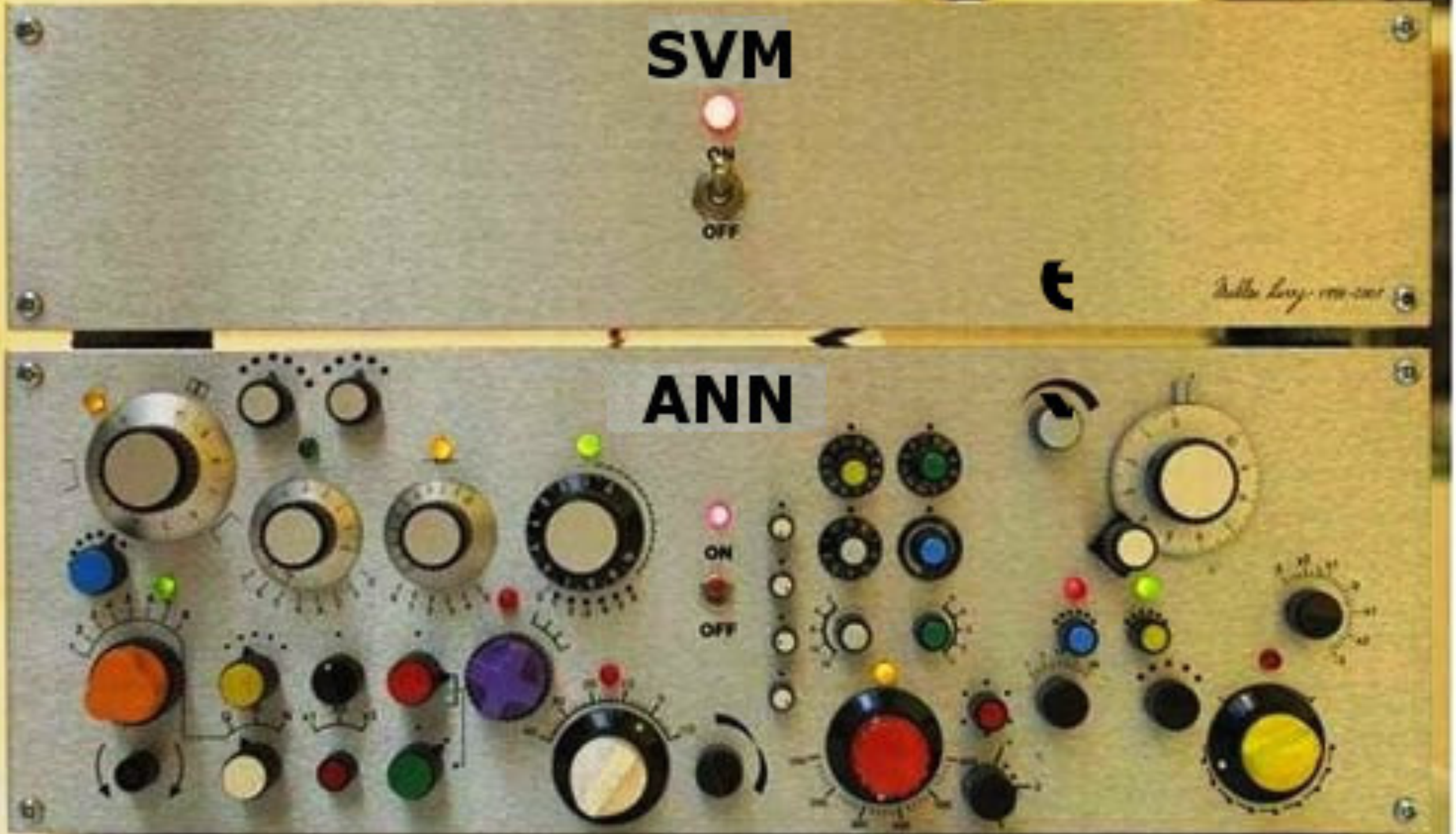
gaussian radial basis function parameters

- C: the cost parameter
- gamma: controls sharpness of bumps

SVMs: the advantages and disadvantages

advantages

- widely considered to be the best "stock" learning algorithm
- relatively very simple to configure and require very little tuning of parameters


SVM
ON
OFF
ANN
ON
OFF

disadvantages

- black box: tough to extract knowledge representation

disadvantages

— can be slow

disadvantages

— but not too bad if you're using a linear kernel

popular real-world applications of SVMs

- text categorization
- image classification
- protein classification in medical research
- handwritten character recognition

popular SVM implementations

and they're easy to use, too!

- LibSVM
- SVMLight
- scikit-learn

easy to use in your own code

— nice, clear APIs
— python example, using scikit-learn

```
from sklearn.svm import SVC

X = ... # my training data
y = ... # my training labels (the class or classification of each training data point)

clf = SVC(C=1.0)
clf.fit(X, y)

X_test = ... # my production data

predictions = clf.predict(X_test)

# done!
```

popular machine learning libraries

import your data once and run it on any number of both popular and obscure ML algorithms

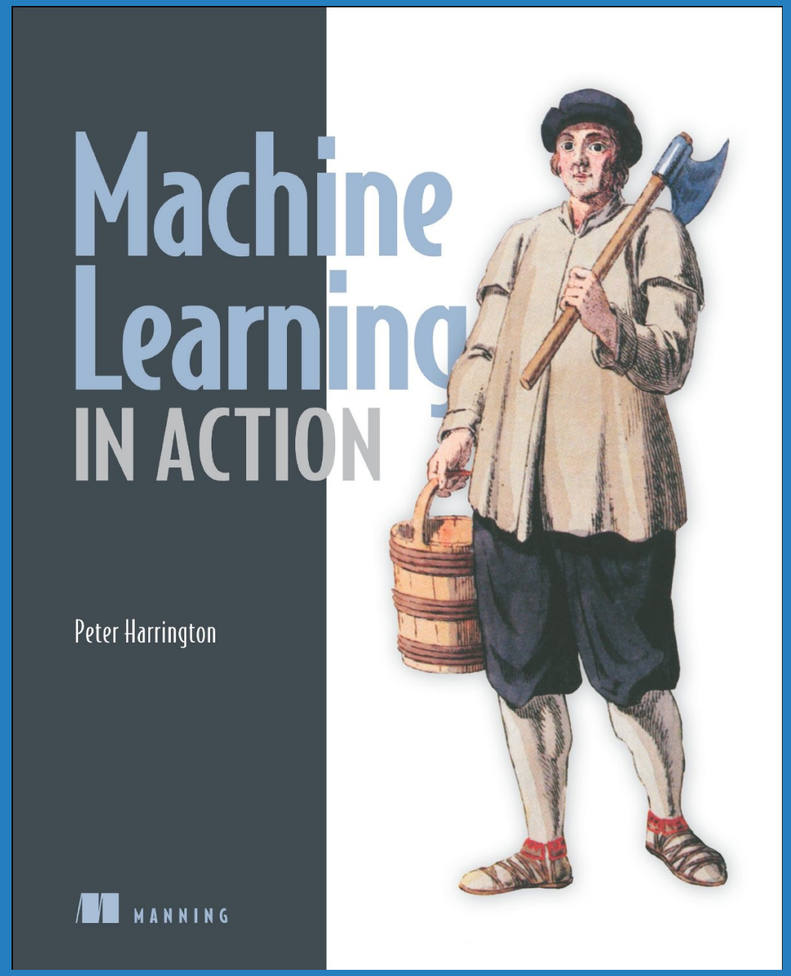
adapters for calling from many popular languages, including Ruby

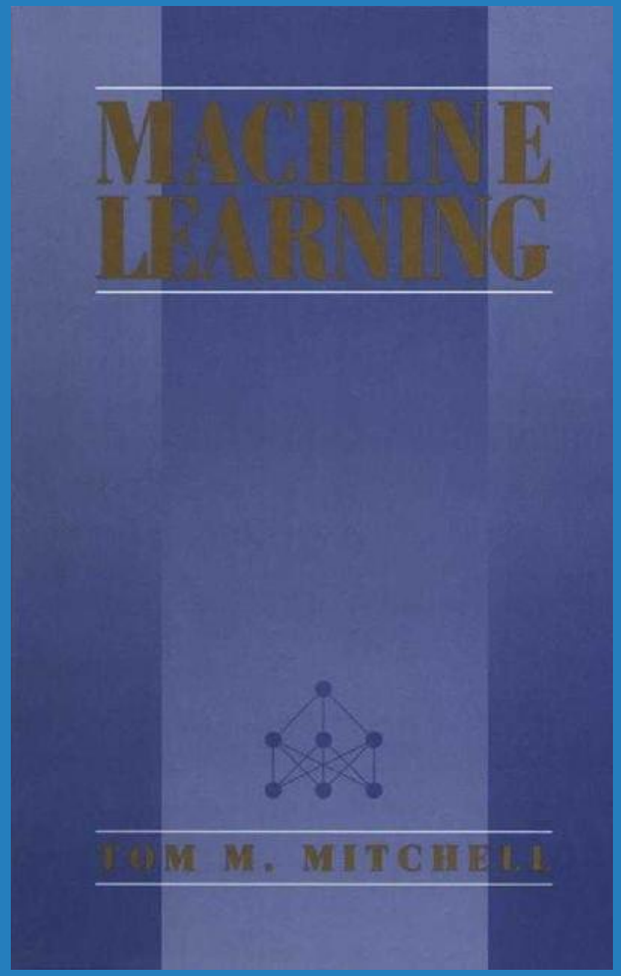
- Weka -- written by kiwis
- Shogun -- on github but looks like it's a giant pain to use, with confusing documentation
- scikit-learn
- Matlab -- but not open source

summary cum laude

1. demo of SVM running on some example data
2. interpreting the results
3. review of fundamental concepts of machine learning
4. how do SVMs work?
5. how do you train an SVM?
6. some classic usage examples
7. some publically available implementations

i have books available to lend/give you





many thanks go to data will, gessner, and josh

does anyone have any questions?

examples:

- **are SVMs a good source of riboflavin?**
- **can i wash my cat with an SVM?**


fin