INTRODUCTION

Course webpage

http://stat.columbia.edu/~porbanz/UN3106S18.html

THIS CLASS

What to expect

- This class is an introduction to machine learning.
- Topics: Classification; “learning”; basic neural networks; etc

Homework

- Programming + “theoretical” questions.
- All programming will be done in R.

What this class is not

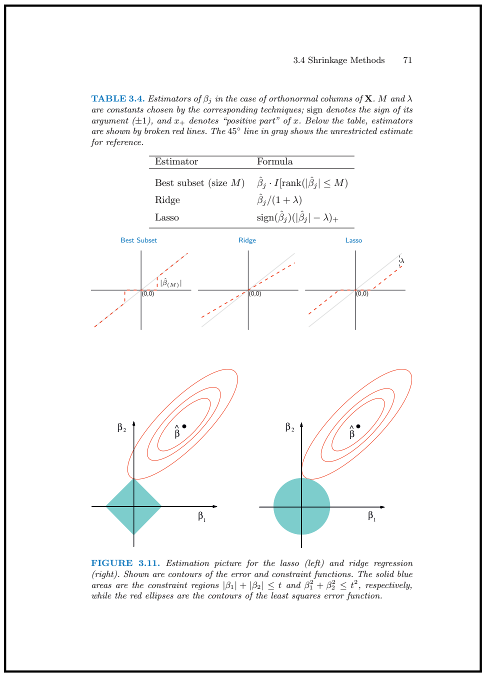
- Applied.

The purpose of this class is to understand how some of the most important machine learning methods work.

Books

T. Hastie, J. Friedman and R. Tibshirani:
"The Elements of Statistical Learning".
2nd Edition, Springer, 2009.

Available online.



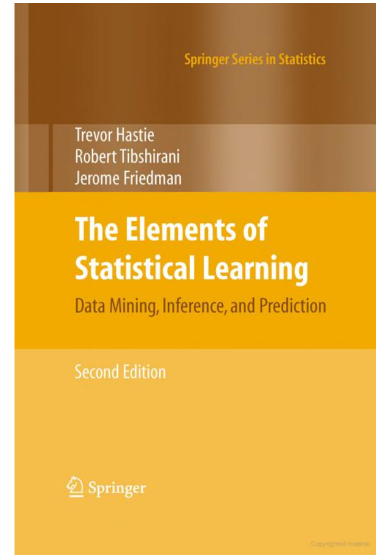
3.4 Shrinkage Methods 71

TABLE 3.4. *Estimators of β_j in the case of orthonormal columns of $\mathbf{X}$. M and λ are constants chosen by the corresponding techniques;* sign *denotes the sign of its argument (± 1), and x_+ denotes "positive part" of x. Below the table, estimators are shown by broken red lines. The $45°$ line in gray shows the unrestricted estimate for reference.*

Estimator	Formula
Best subset (size M)	$\hat\beta_j \cdot I[\text{rank}(\lvert\hat\beta_j\rvert \le M)$
Ridge	$\hat\beta_j/(1+\lambda)$
Lasso	$\text{sign}(\hat\beta_j)(\lvert\hat\beta_j\rvert - \lambda)_+$

FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions $|\beta_1| + |\beta_2| \le t$ and $\beta_1^2 + \beta_2^2 \le t^2$, respectively, while the red ellipses are the contours of the least squares error function.*

$\longleftarrow$ It's much prettier inside.



Links to this book and other potentially useful references will be added to the class homepage as they become relevant. All of these are optional; the relevant material are the course slides.

Background Knowledge

- Euclidean space; vectors
- Scalar products
- Derivatives and gradients of functions
- Probability distributions and densities. Example:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \qquad \text{or} \qquad p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

- Gaussian distribution on $\mathbb{R}$ and $\mathbb{R}^d$
- (Eigenvalues and eigenvectors)
- Regression

Problem setting

Classification methods subdivide data into several, distinct classes. More formally:

- Data $x_1, x_2, \ldots$.
- Each observations falls into one of K categories (the *classes*).
- Learning task: Find a classification function

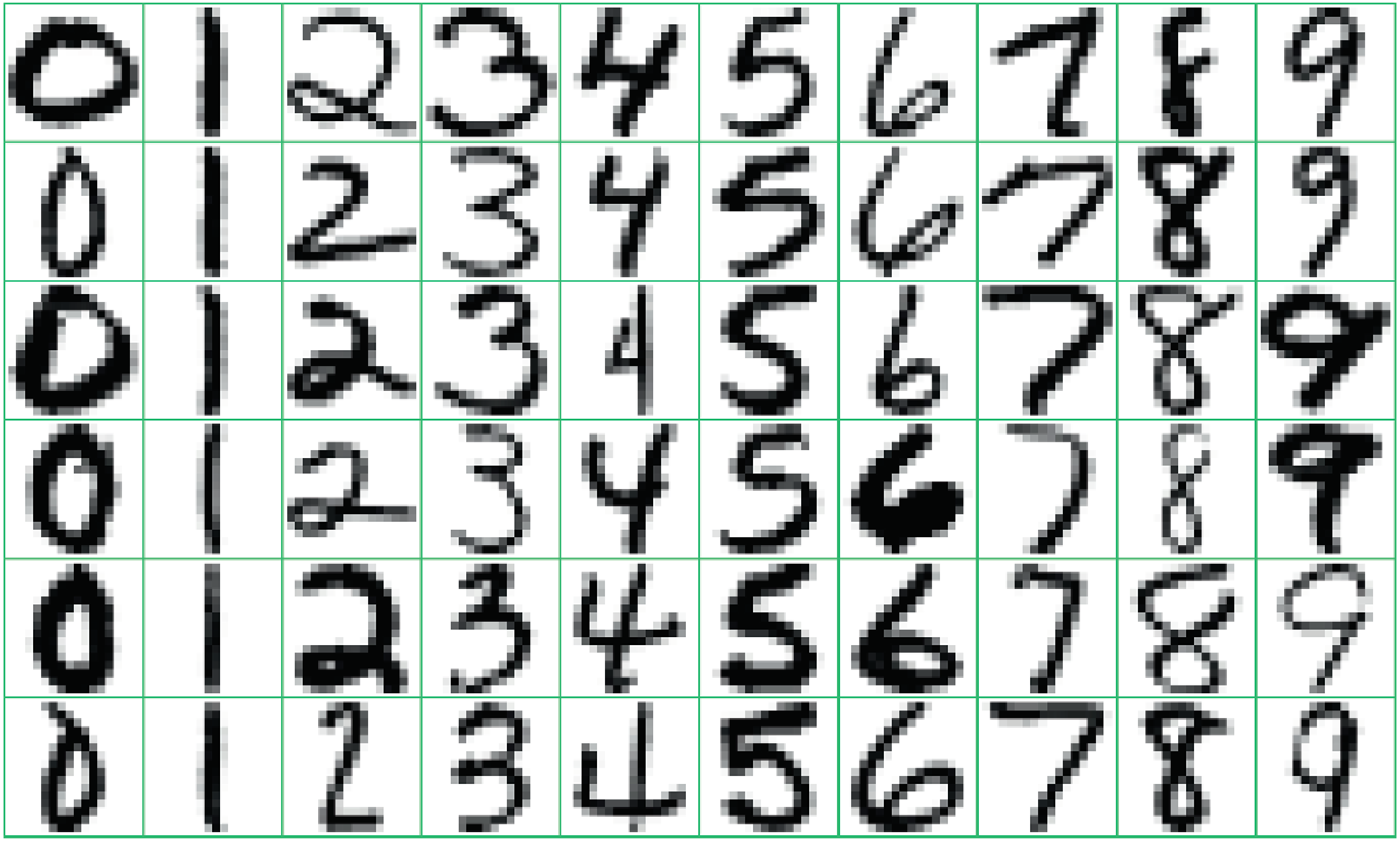
$$f : \mathbf{X} \to \{1, \ldots, K\} \ .$$

- Input of the learning problem: Correctly categorized examples $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n$.

Approach

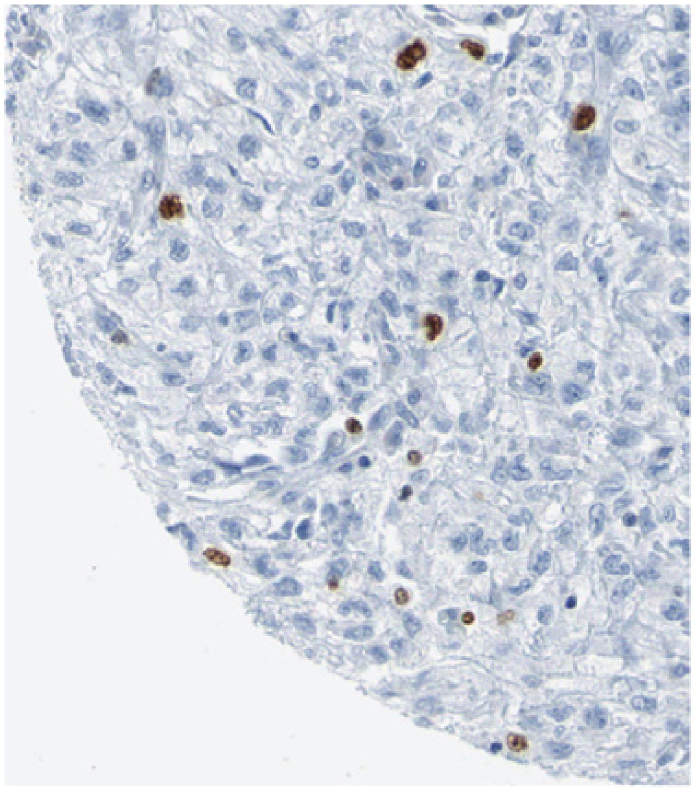
- Define:
  1. A set of possible classification functions f (the *hypothesis set*).
  2. A *cost function* which assumes a large value when mistakes are made.
- To find a good classifier, search the hypothesis class for the f which keeps costs as small as possible.
- Different types of errors can be more or less expensive.

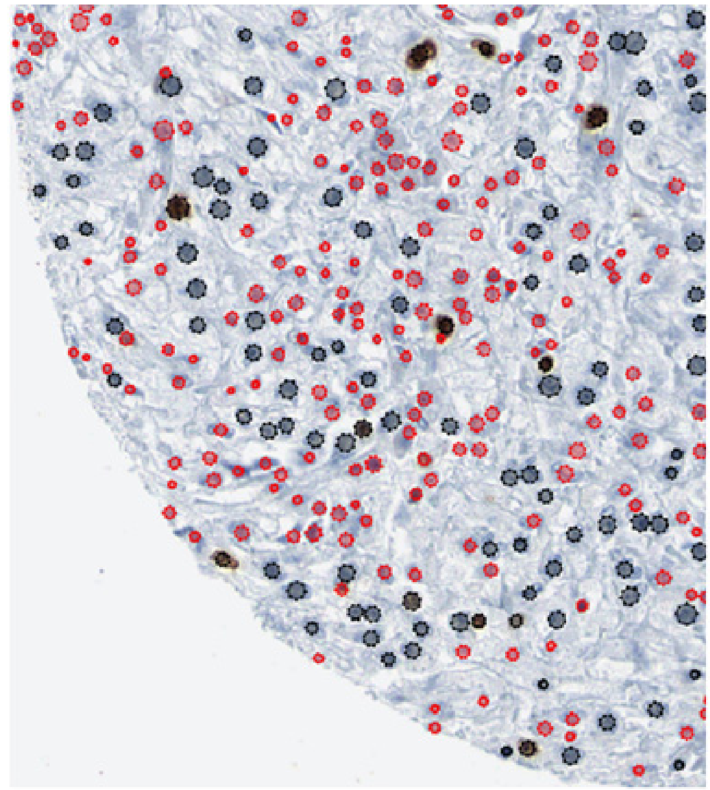
USPS Data



Each digit: 16×16 pixels, i.e. $x \in \mathbb{R}^{256}$

CANCER DIAGNOSIS





Previous examples as classification problems

- USPS data: 10 classes (+ one “outlier class”)
- Cancer diagnosis: 4 classes

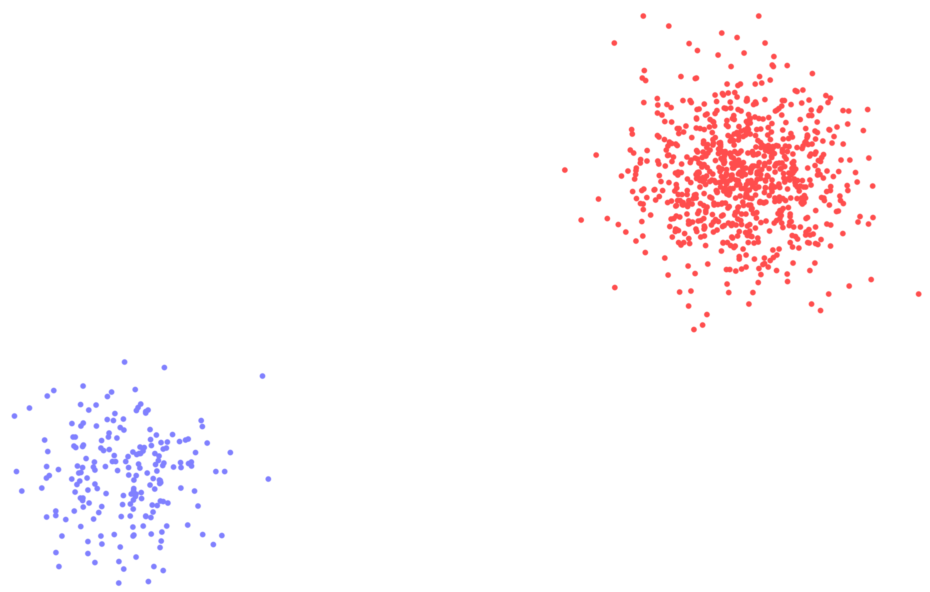
Face recognition

- Hard problem, but much recent progress.
- Deployable systems can now have around 90+% accuracy on people in their database.
- 1 class per person in data base + 1 class for “none of those”.

Fingerprint recognition

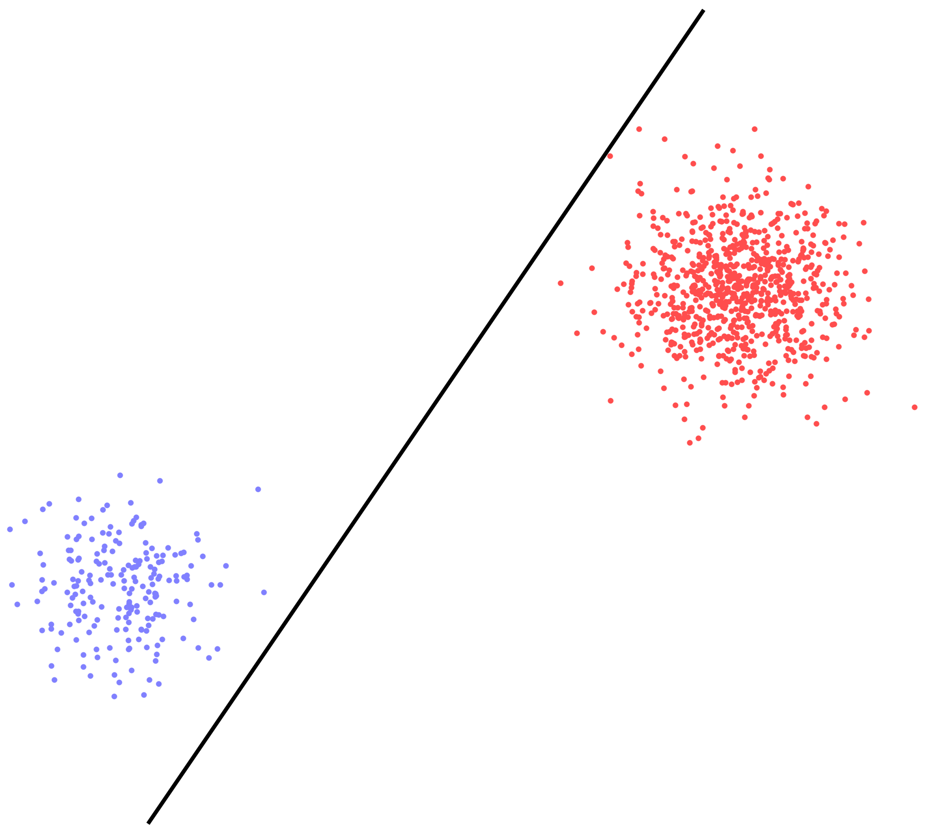
- Again: 1 class per person in data base + 1 class for “none of those”.
- Deployable systems have been available for ca. 15 years.
- Development of computer systems lead to reassessment of human error rates.

Two-class classification: Basic ideas



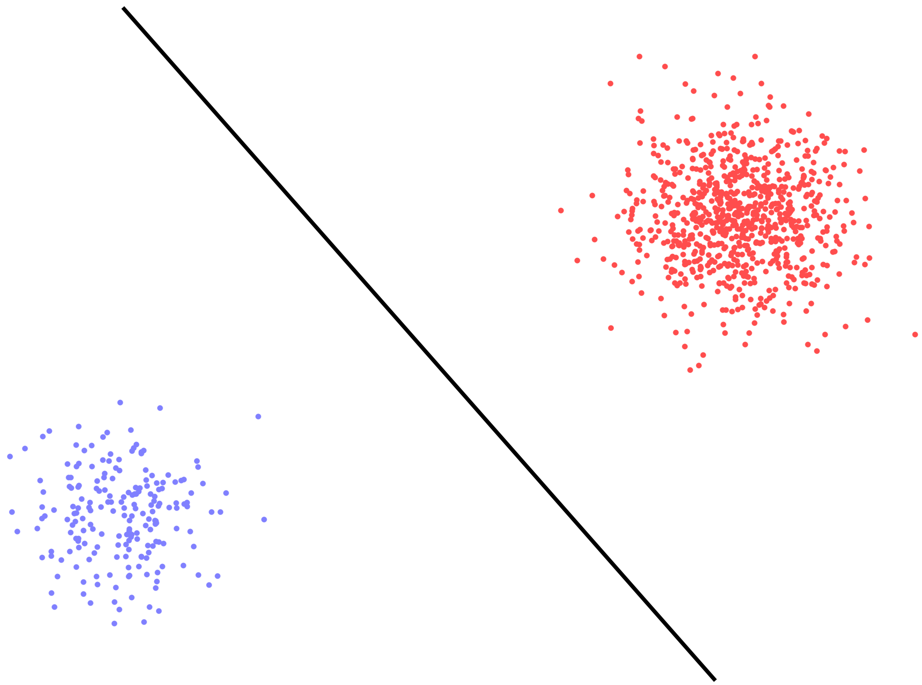
1. Represent each object as a point; axes = measurements ($\rightarrow$ *vector spaces*)
2. Separate classes by hyperplane ($\rightarrow$ *scalar products*)
3. Define a function that measures how well the plane separates classes; small values indicate a good fit.
4. Find "good" hyperplane by minimizing function ($\rightarrow$ *derivatives, gradients, Hessians, etc*)

TWO-CLASS CLASSIFICATION: BASIC IDEAS



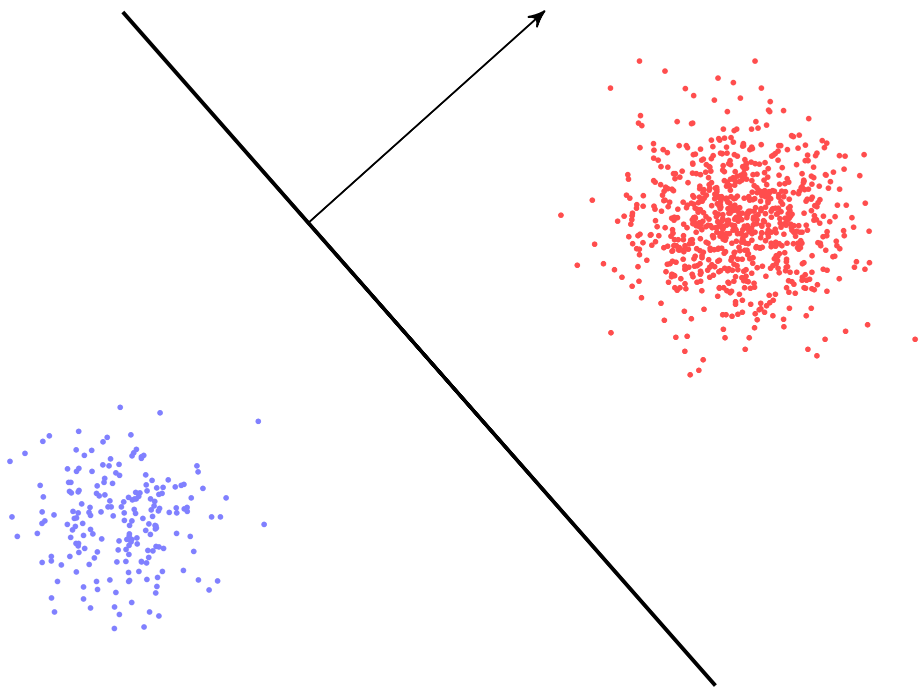
1. Represent each object as a point; axes = measurements ($\rightarrow$ *vector spaces*)
2. Separate classes by hyperplane ($\rightarrow$ *scalar products*)
3. Define a function that measures how well the plane separates classes; small values indicate a good fit.
4. Find "good" hyperplane by minimizing function ($\rightarrow$ *derivatives, gradients, Hessians, etc*)

Two-class classification: Basic ideas



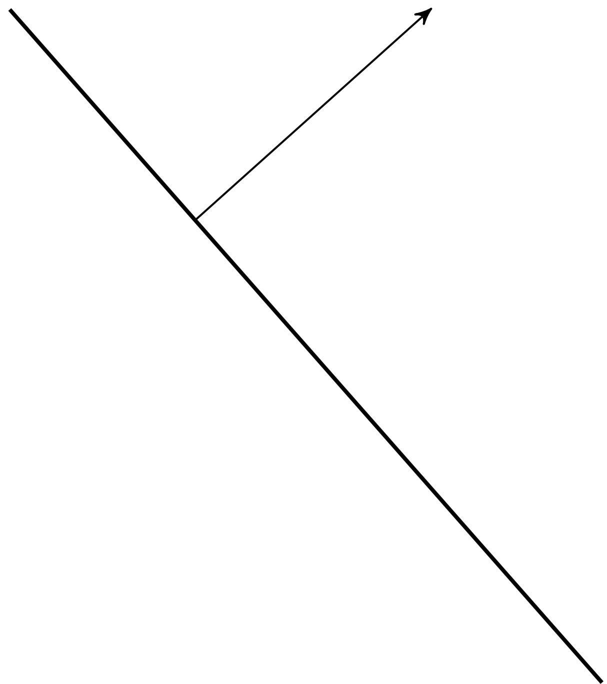
1. Represent each object as a point; axes = measurements ($\rightarrow$ *vector spaces*)
2. Separate classes by hyperplane ($\rightarrow$ *scalar products*)
3. Define a function that measures how well the plane separates classes; small values indicate a good fit.
4. Find "good" hyperplane by minimizing function ($\rightarrow$ *derivatives, gradients, Hessians, etc*)

Two-class classification: Basic ideas



1. Represent each object as a point; axes = measurements (→ *vector spaces*)
2. Separate classes by hyperplane (→ *scalar products*)
3. Define a function that measures how well the plane separates classes; small values indicate a good fit.
4. Find "good" hyperplane by minimizing function (→ *derivatives, gradients, Hessians, etc*)

TWO-CLASS CLASSIFICATION: BASIC IDEAS



1. Represent each object as a point; axes = measurements (→ *vector spaces*)
2. Separate classes by hyperplane (→ *scalar products*)
3. Define a function that measures how well the plane separates classes; small values indicate a good fit.
4. Find "good" hyperplane by minimizing function (→ *derivatives, gradients, Hessians, etc*)

PENDULUM
(WORK OF MARC DEISENROTH AND CARL EDWARD RASMUSSEN)

Task
Balance the pendulumn upright by moving the sled left and right.

- The computer can control *only* the motion of the sled.
- Available data: Current state of system (measured 25 times/second).

Formalization

State = 4 variables (sled location, sled velocity, angle, angular velocity)
Actions = sled movements

The system can be described by a function

$$\begin{aligned} f: \quad \mathcal{S} \times \mathcal{A} &\to \mathcal{S} \\ (\text{state}, \text{action}) &\mapsto \text{state} \end{aligned}$$

Historical origins: Artificial intelligence and engineering

Machines need to...

- recognize patterns (e.g. vision, language)
- make decisions based on experience (= data)
- predict
- cope with uncertainty

Today

- There is no clear dividing line between machine learning and statistics anymore.
- Engineering aspects (such as software development and specialized hardware) have become much more important as machine learning systems get deployed.

Modern applications: (A few) Examples

- medical diagnosis
- face detection/recognition
- speech and handwriting recognition
- web search
- recommender systems
- bioinformatics
- natural language processing
- computer vision