

Intro to Machine Learning

Geoff Gordon
ggordon@cs.cmu.edu

February 7, 2006

What is machine learning?

Machine learning = computing predictions about the future from data about the past

Examples

Information retrieval (Google)

Collaborative filtering (Netflix, Amazon)

Self-driving cars (CMU, Stanford)

- Lane departure warning systems (most auto makers)

Handwriting recognition (USPS)

Anomaly detection (SDSS)

Deciding who to hire (monster.com)

Self-organizing data structures (e.g., binary search trees where the item you want is always near the root)

Others...

Comp bio examples

Microarray analysis

Sequence alignment

High-throughput screening

Others. . .

Formally

Can formalize at various levels of generality

One thing is constant: the real world never quite fits the formalism you want it to

Classification and regression

Training data: $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

Test data: $(x_{n+1}, ?)$

$x_i \in \mathbb{R}^d$ (or discrete, doesn't really matter)

If $y_i \in \mathbb{R}$: regression

If $y_i \in \{1, \dots, k\}$: classification

If $y_i \in$ something else: structured regression/classification, other

Formally

Assume training data is an independent sample of size n from distribution $D = P_D(x, y)$

Test data sampled from x marginal of D , $P_D(x)$

Prediction problem is to guess $P(y | x_{n+1})$

Difficulties

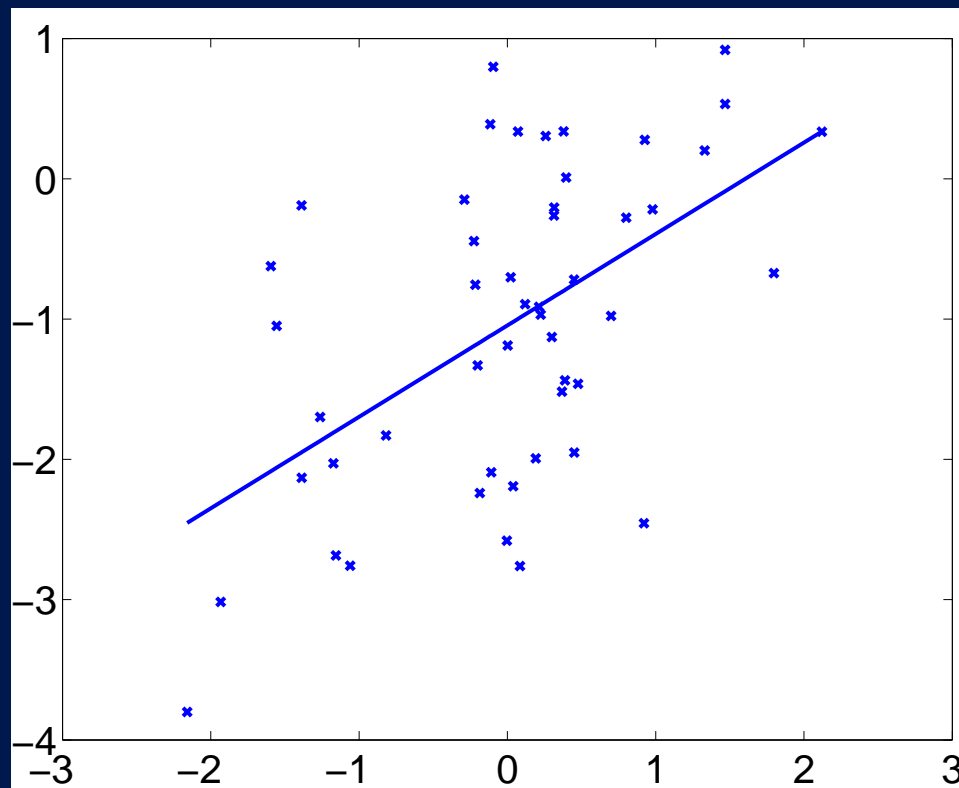
In real world, training data is never independent, training distribution never matches test distribution

Often hard to “featurize” our training examples

- web pages
- job applicants

But things seem to work out anyway

Example: linear regression



Linear regression

Out of all linear predictors, pick the one that does best on the training data

Best = least sum of squared errors

Minimize

$$\sum_i (y_i - x_i \cdot \theta)^2$$

Linear regression

In matrices, minimize

$$(y - X\theta) \cdot (y - X\theta) = \theta^T X^T X \theta - 2y^T X \theta + \dots$$

Set gradient to 0:

$$X^T X \theta = X^T y$$

“Normal equations”—Matlab, etc., can solve for θ

A bad example

Bad algorithm: *memorization learning*

If $x_{n+1} = x_i$ for some $i \in \{1 \dots n\}$ then predict y_i

If no training example matches exactly, predict 0

What happens when we try to use this algorithm?

What's the difference?

Q: Why does linear regression work better than memorization?

A: It doesn't always.

Why not?

Suppose # features k is \gg # training examples, and each x_i has one feature that's on in x_i but not x_j for $j \neq i$

Then regression \approx memorization

Hypothesis complexity and generalization

Big difference between regression (w/ small k) and memorization: hypothesis class complexity

Hypothesis class H = set of prediction functions we are willing to use

H for linear regression:

$$\{f(x) = x \cdot \theta \mid \theta \in \mathbb{R}^k\}$$

H for memorization: sums of delta-functions

$$f(x) = \sum_{i=1}^n y_i \delta(x - x_i)$$

Hypothesis complexity

If $\theta \approx \theta'$, then $x \cdot \theta \approx x \cdot \theta'$

Intuitively: not that many truly different linear predictors

If $x_i \approx x'_i$ and $y_i \approx y'_i$, memorized function can still make very different predictions

No way to make a small list of prediction functions that always contains one “kind of like” the one the algorithm comes up with

Regularization

What if we really want to use a complicated H with not much data?

Solution: regularization (also “smoothing,” “structural risk minimization,” etc.)

Divide H into

$$H_1 \subseteq H_2 \subseteq H_3 \subseteq \dots \subseteq H$$

Only move from H_i to H_{i+1} if there is “enough evidence”

Regularization

Maximize

$$\text{evidence}(h) - \text{complexity}(h)$$

evidence(h):

- number of data n , or
- performance of h

complexity(h):

- (some function of) smallest i with $h \in H_i$

Can have continuous grades of complexity as well as discrete

Regularization in linear regression

Define complexity($h = \theta$) for predictor $x \cdot \theta$ to be $\|\theta\|$

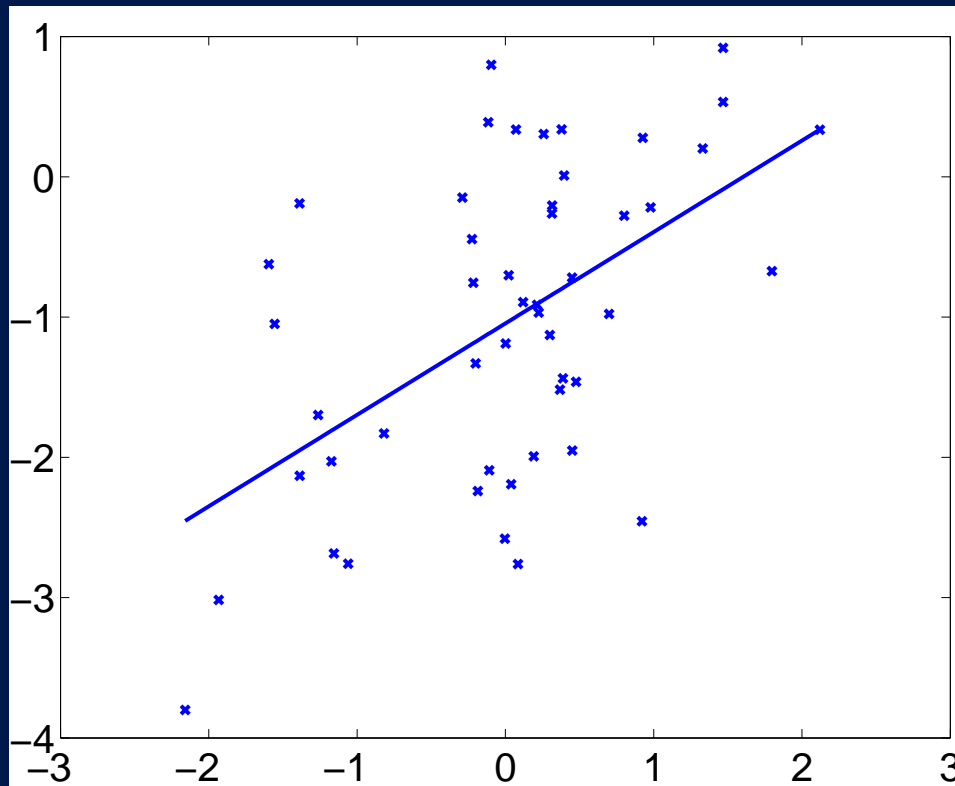
Minimize $\sum_i (y_i - x_i \cdot \theta)^2 + C\|\theta\|^p$

$\|\cdot\|_2^2$: ridge regression

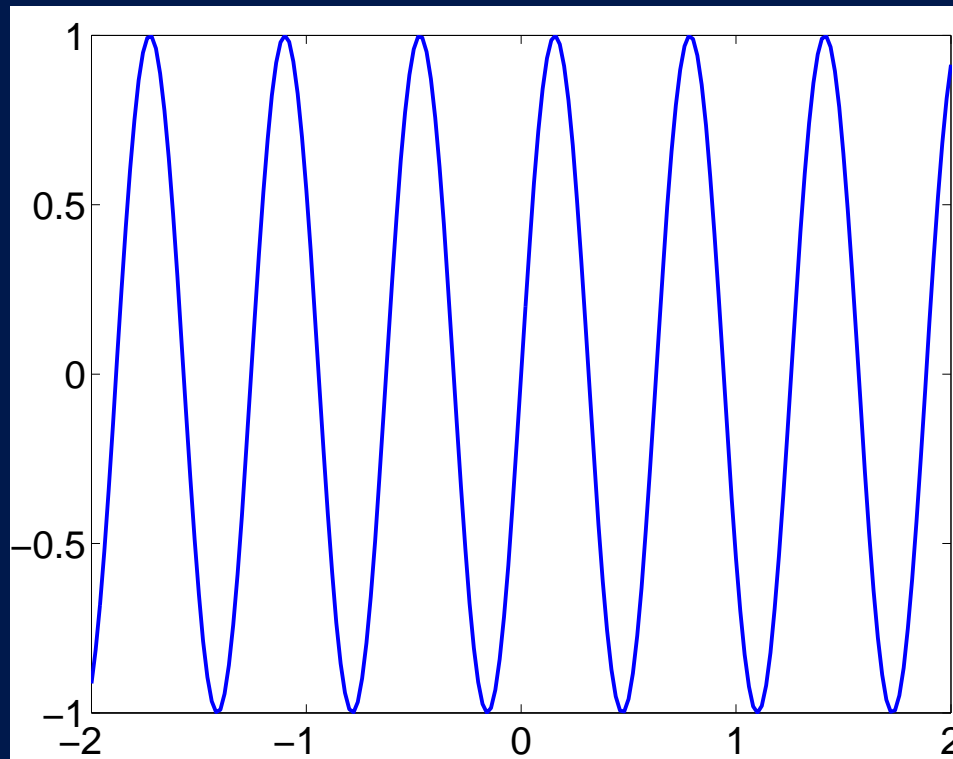
$\|\cdot\|_1$: LASSO

Others...

Linear regression example from before

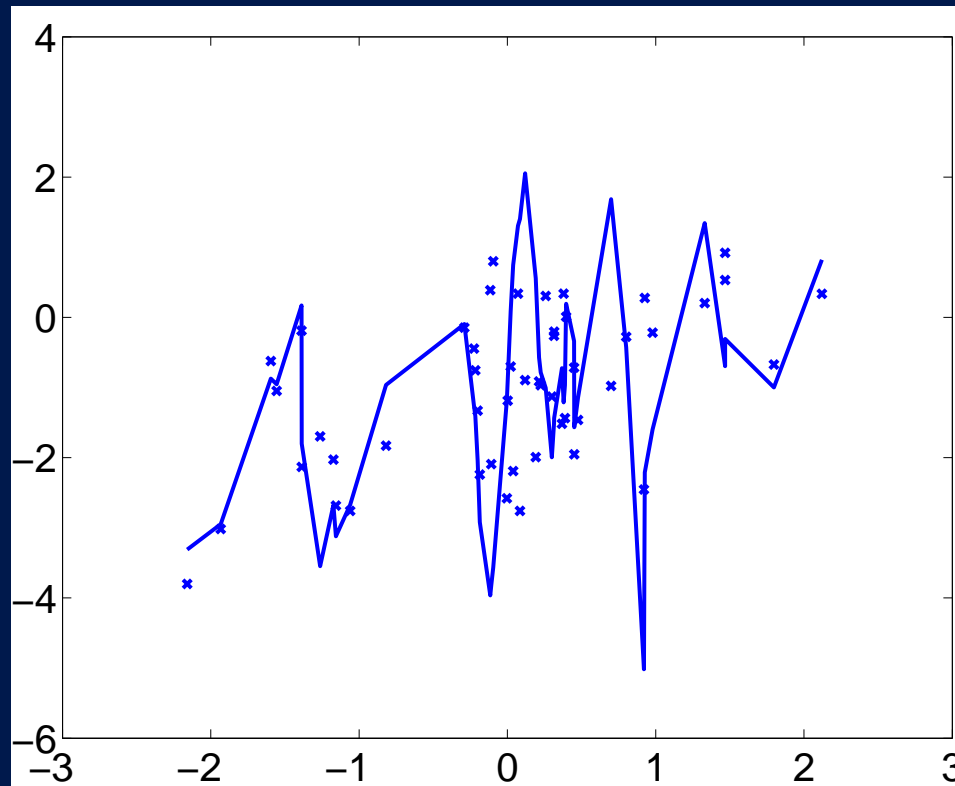


A distractor feature



Added 50 of these w/ different frequencies

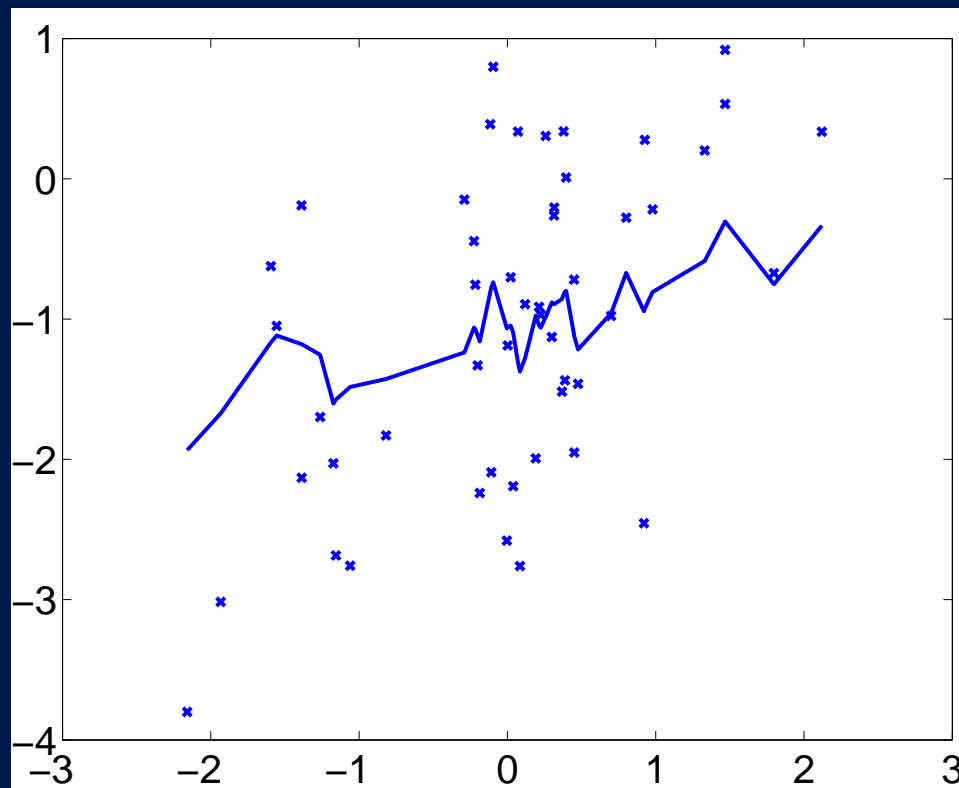
Plain linear regression



Numerical errors

Pays too much attention to distractors

Ridge regression



LASSO

Works particularly well for sparse θ

Draw...

Evaluating performance

Suppose we use linear regression or memorization to learn a predictor

$$y = f(x)$$

How good is this predictor?

Ways of evaluating

- training data (bad)
- penalized training data
- holdout set
- cross-validation

Nonlinearity

Have been learning functions of form $f(x) = f(x \cdot \theta)$

Limited expressive power

XOR: $(0, 0) \mapsto 0, (0, 1) \mapsto 1, (1, 0) \mapsto 1, (1, 1) \mapsto 0$

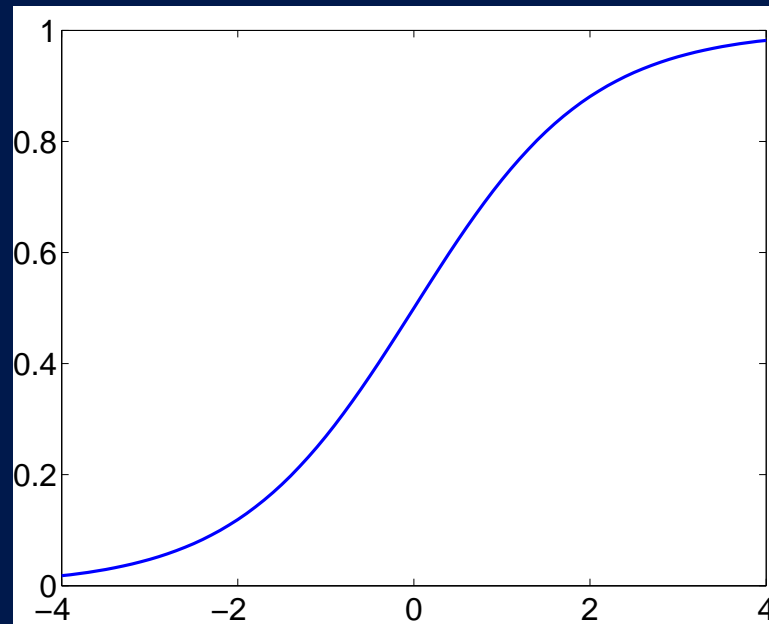
Solution: predict with a nonlinear function of x

Problem: there are a lot more nonlinear functions than linear ones

- computation
- generalization

But people do it anyway (probably more often than they should...)

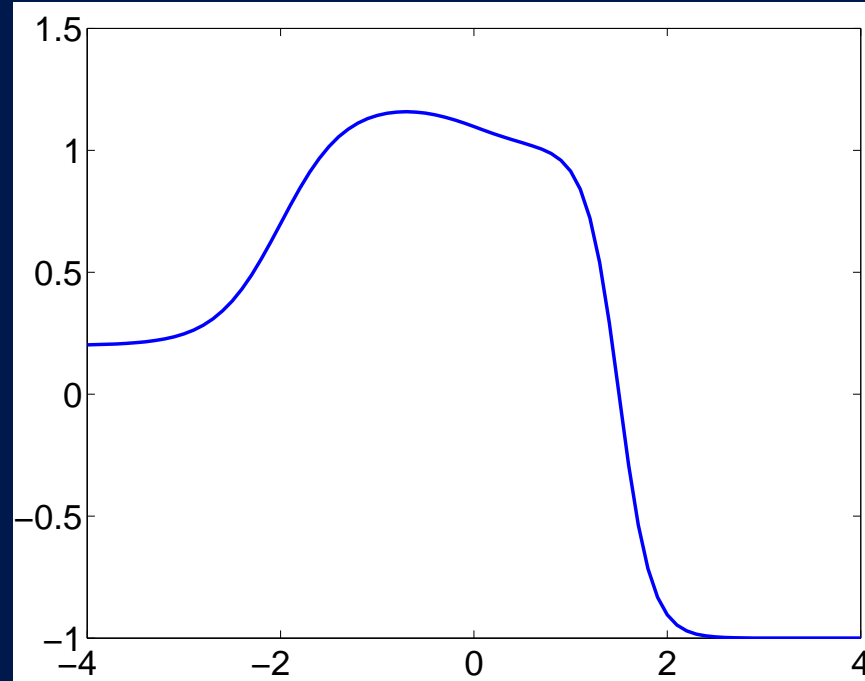
Example: neural nets



A specific form of nonlinear predictor: compositions and weighted sums of sigmoid functions

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Single-layer neural nets



$$f(x) = \sum_j w_j \sigma(\theta_j \cdot x)$$

Adjustable parameters $w_j \in \mathbb{R}$, $\theta_j \in \mathbb{R}^k$

Nonparametric methods

Just as with linear regression, minimize

$$\sum_i (y_i - x_i \cdot \theta)^2$$

plus optional regularization term

$$\sum_j (C_j \|\theta_j\|^p + D_j \|w_j\|^p)$$

This time, optimization is hard

Gradient descent (“backprop”), conjugate gradient, Newton’s method—
all in prepackaged toolboxes

Local minima

Nonparametric methods

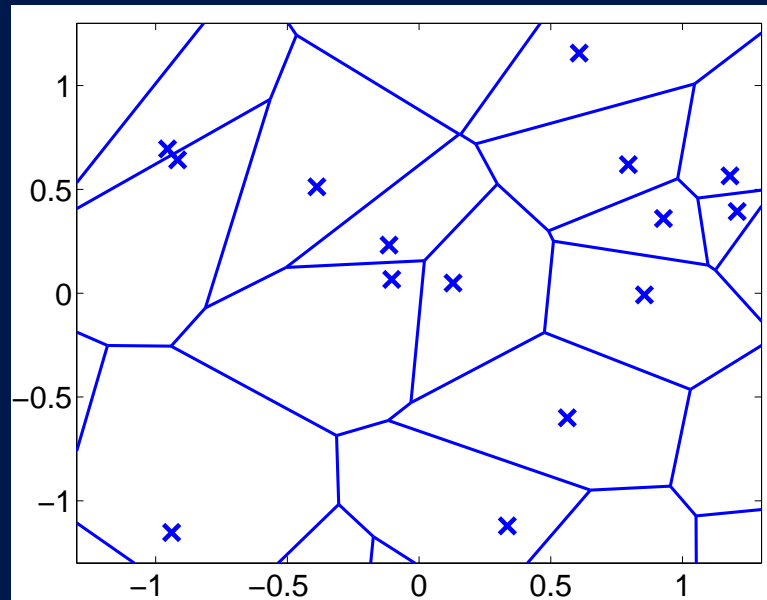
Hypothesis class H = all functions $f(x)$

Big! *Must* use regularization

$$H_1 \subseteq H_2 \subseteq H_3 \subseteq \dots \subseteq H$$

Often H_j depends on observed values of x_i —this is OK if we're careful

Example: 1-nearest-neighbor



Divide input space into bins based on x_i

H = functions which are constant on bins

Nearest neighbors cont'd

Best prediction in H : for $x_{n+1} \in \text{bin}_j$, predict average of y_i for $x_i \in \text{bin}_j$

k -nearest-neighbor: average of y_i where x_i is one of k closest (not just the single closest)

k larger: greater regularization (e.g., if $k \geq n$ we just predict average of all y_i)

Generalization in k -NN

Fixed k : complexity of H grows too quickly

May generalize OK early, but doesn't reach right answer in limit

If we let k grow with training set, can learn any function in a large class (e.g., functions with bounded derivatives)

Unsupervised learning

Training data x_1, x_2, \dots, x_n

Test data x_{n+1}

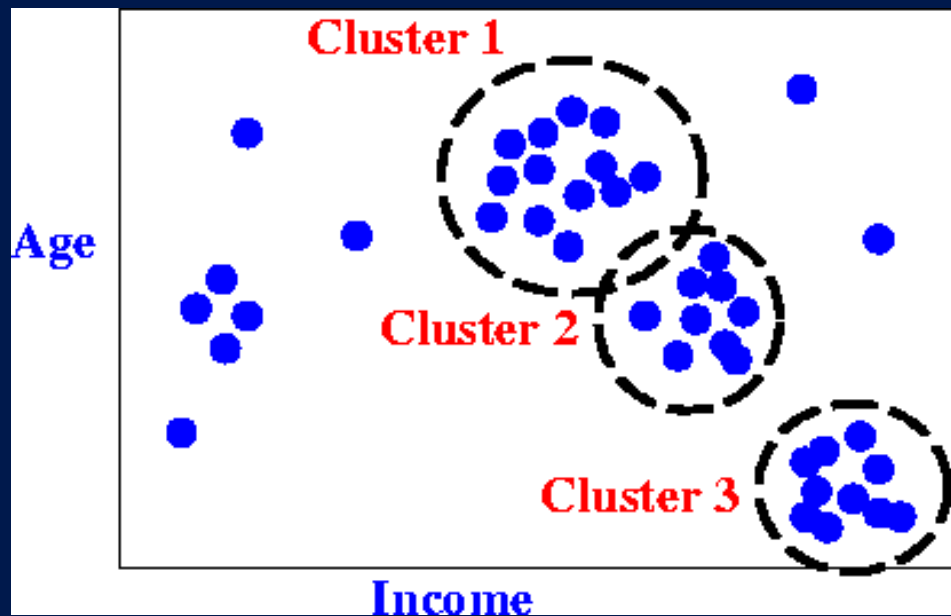
Task: what will x_{n+1} look like?

E.g., what will $\|x_{n+1}\|^2$ be, what will $t \cdot x_{n+1}$ be for some unknown vector t

Generally, can answer such questions if we know $P(x)$

Unlike regression and classification, no one distinguished feature that we know we want to predict—harder problem

Example: clustering



Partition x_i into “clusters”

Clustering

Cluster = subset b_k of the examples, with $x_i, x_j \in b_k$ similar to each other in some way

Predict that test example x_{n+1} will be

- selected from one of the clusters b_k at random
- have properties similar to the average properties of $x_i \in b_k$

Example: factor analysis

Approximate matrix X of training data as

$$X \approx UV$$

for

$$U \in \mathbb{R}^{n \times l} \quad V \in \mathbb{R}^{l \times k}$$

with small l

Used for, e.g., collaborative filtering

- x_{ij} is whether customer i likes item j
- U is matrix of “customer features”
- V is matrix of “item features”

Back to supervised learning: linear classification

[see supplemental slides]

Summary

Often easier to learn something from past data than to write it down from scratch

But have to worry about

- whether we can collect relevant data
- how best to make use of it (what to optimize)
- whether we can find the optimum
- whether we have enough data for our desired hypothesis class
- how to evaluate the chosen hypothesis
- ...