Introduction to Machine Learning

http://bootcamp.lif.univ-mrs.fr:8080/mainpage

PASCAL Bootcamp 2010

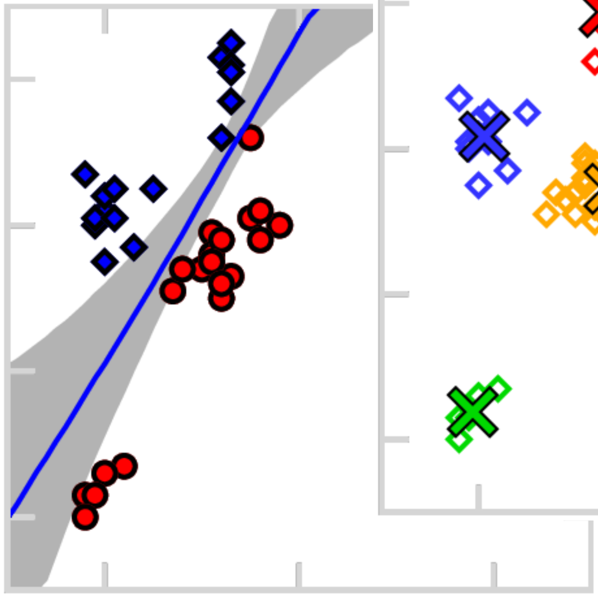
$$P(\theta|\mathcal{D},\mathcal{M}) = \frac{P(\mathcal{D}|\theta,\mathcal{M})\,P(\theta|\mathcal{M})}{P(\mathcal{D}|\mathcal{M})}$$

Sponsored links

Web-based Text Mining
Natural language processing API:
entity extraction, text categ, etc.
www.alchemyapi.com/

Machine Learning Jobs
Google is looking for **machine learning** experts
www.google.com/jobs

Seven Interesting Things
Machine learning + Trends + News
7 tasty **machine**-picked things a day
7interestingthings.blogspot.com



$$R(T_i) \le R_{\mathrm{emp}}(T_i) + \frac{\ln N - \ln\eta}{\ell}\left(1+\sqrt{1+\frac{2R_{\mathrm{emp}}(T_i)\,\ell}{\ln N - \ln\eta}}\right)$$

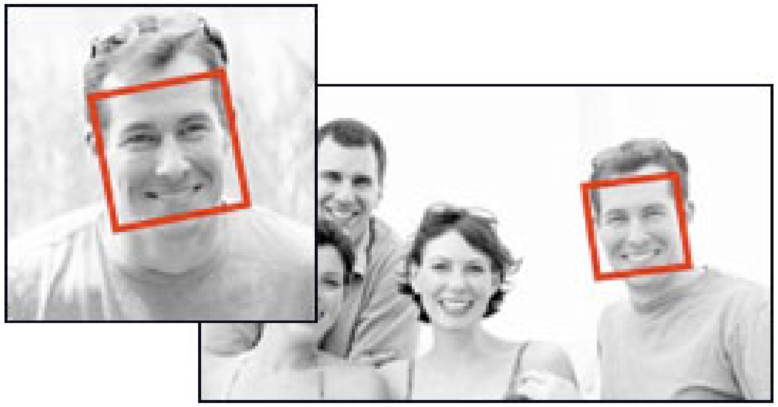
Iain Murray

http://homepages.inf.ed.ac.uk/imurray2/

Face detection

How would you detect a face?

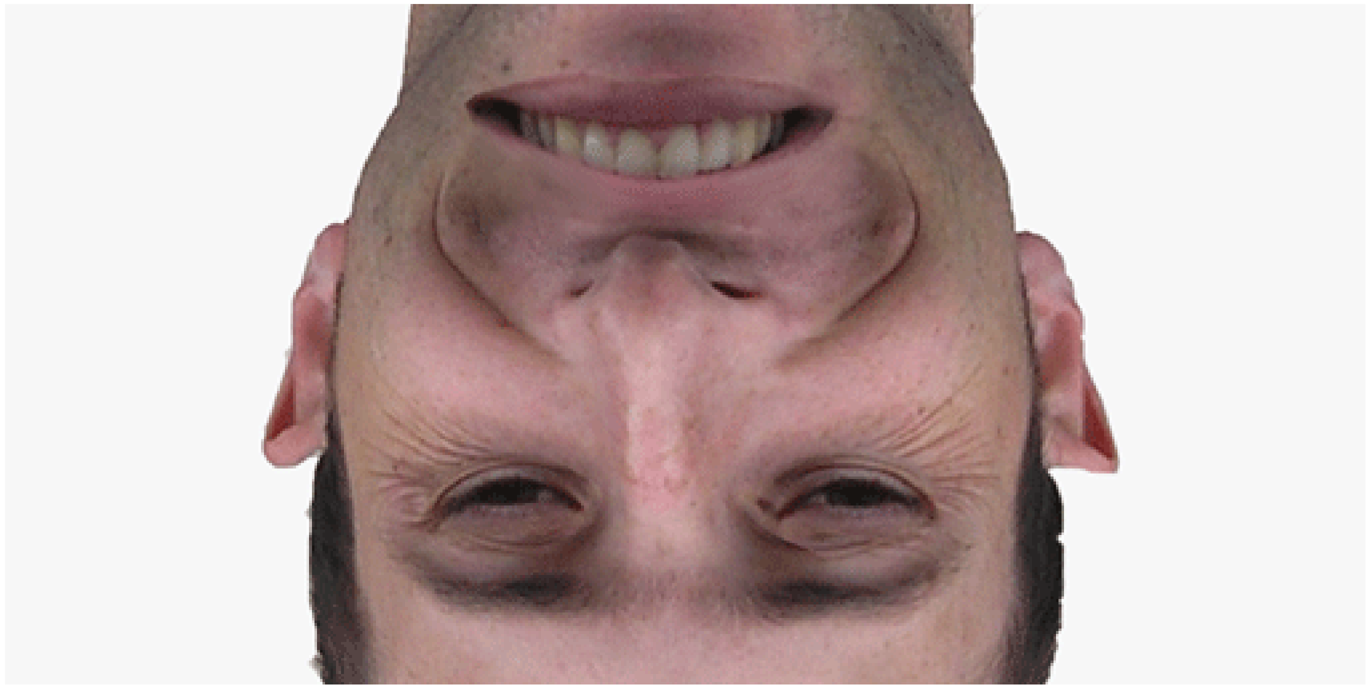
(R. Vaillant, C. Monrocq and Y. LeCun, 1994)



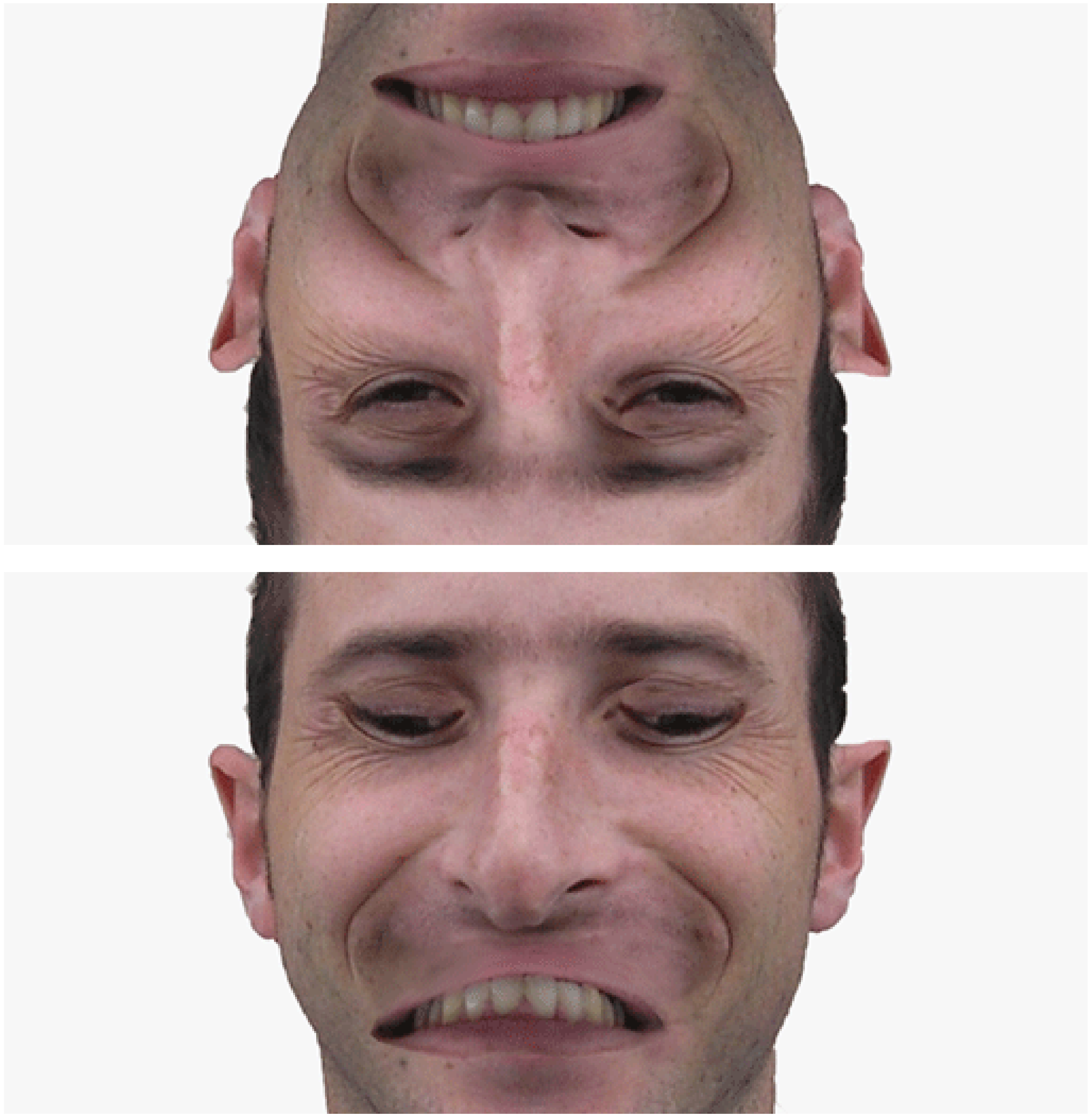
How does album software tag your friends?

http://demo.pittpatt.com/

What do we do?



http://brain.mada.org.il/upside-down-e.html



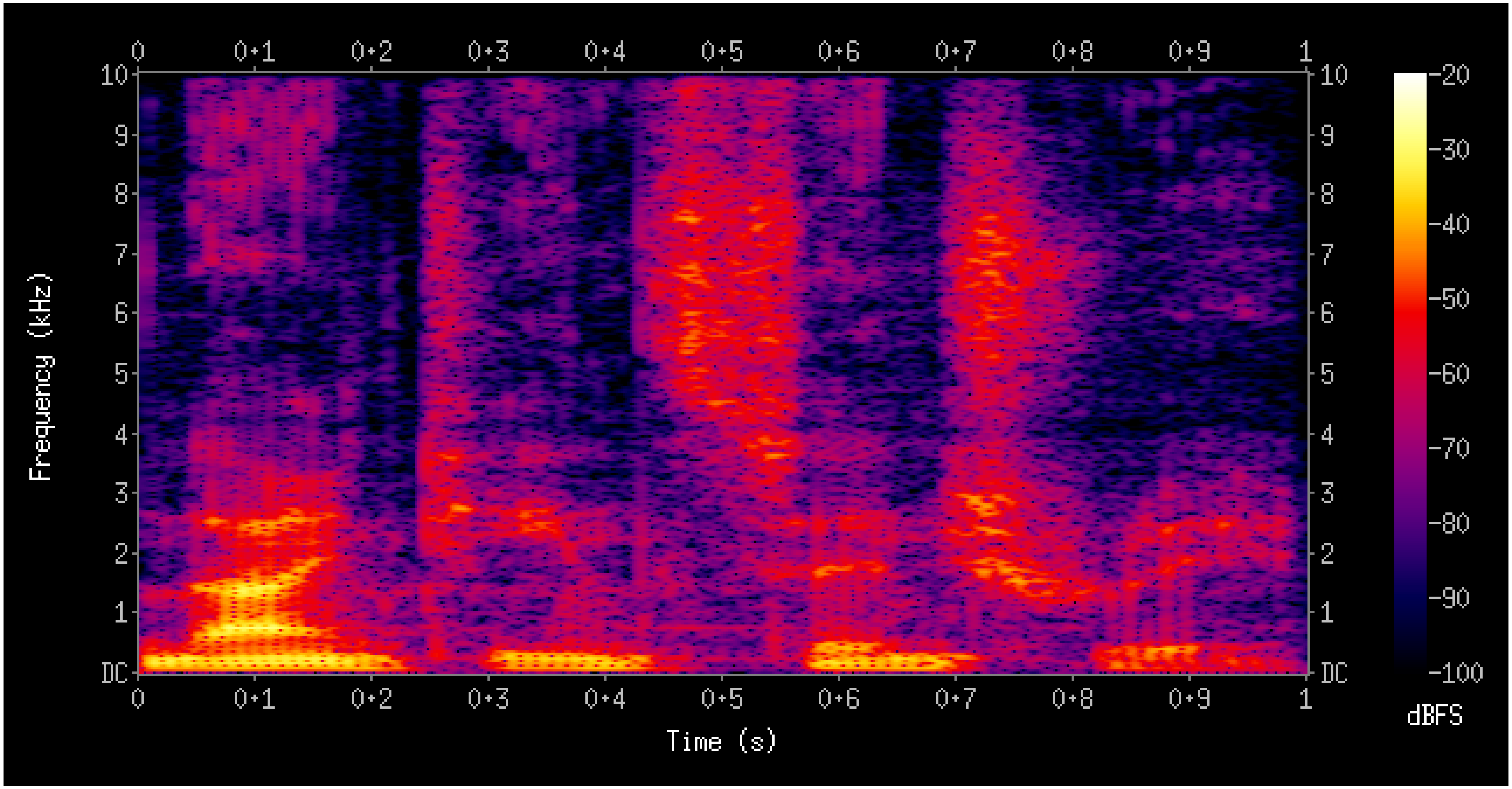
Response surface optimization

Consult on making a new:
concrete, weld, . . . widget?

(antmoose on Flickr, cc-by-2.0)

(Bhadeshia et al.)

Speech recognition

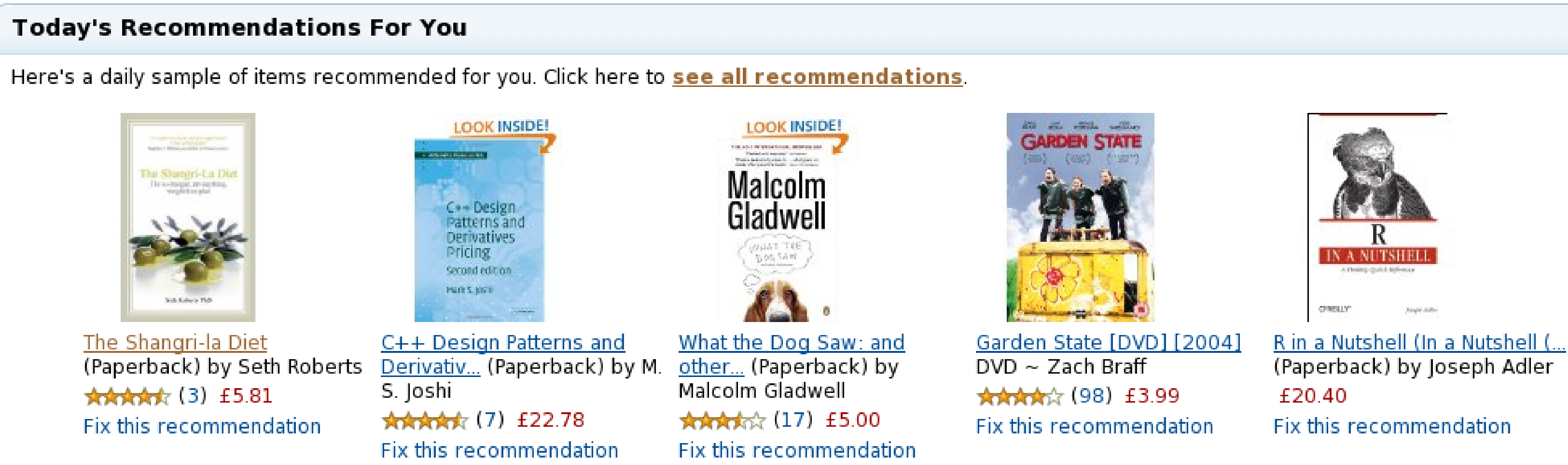


"nineteenth century"

Recommender systems

Amazon, Netflix, tell you what you might like

Collaborative filtering useful in many settings?


Today's Recommendations For You

Here's a daily sample of items recommended for you. Click here to **see all recommendations**.

The Shangri-la Diet (Paperback) by Seth Roberts (3) £5.81 Fix this recommendation

C++ Design Patterns and Derivativ... (Paperback) by M. S. Joshi (7) £22.78 Fix this recommendation

What the Dog Saw: and other... (Paperback) by Malcolm Gladwell (17) £5.00 Fix this recommendation

Garden State [DVD] [2004] DVD ~ Zach Braff (98) £3.99 Fix this recommendation

R in a Nutshell (In a Nutshell (... (Paperback) by Joseph Adler £20.40 Fix this recommendation

Roadmap

Binary classification

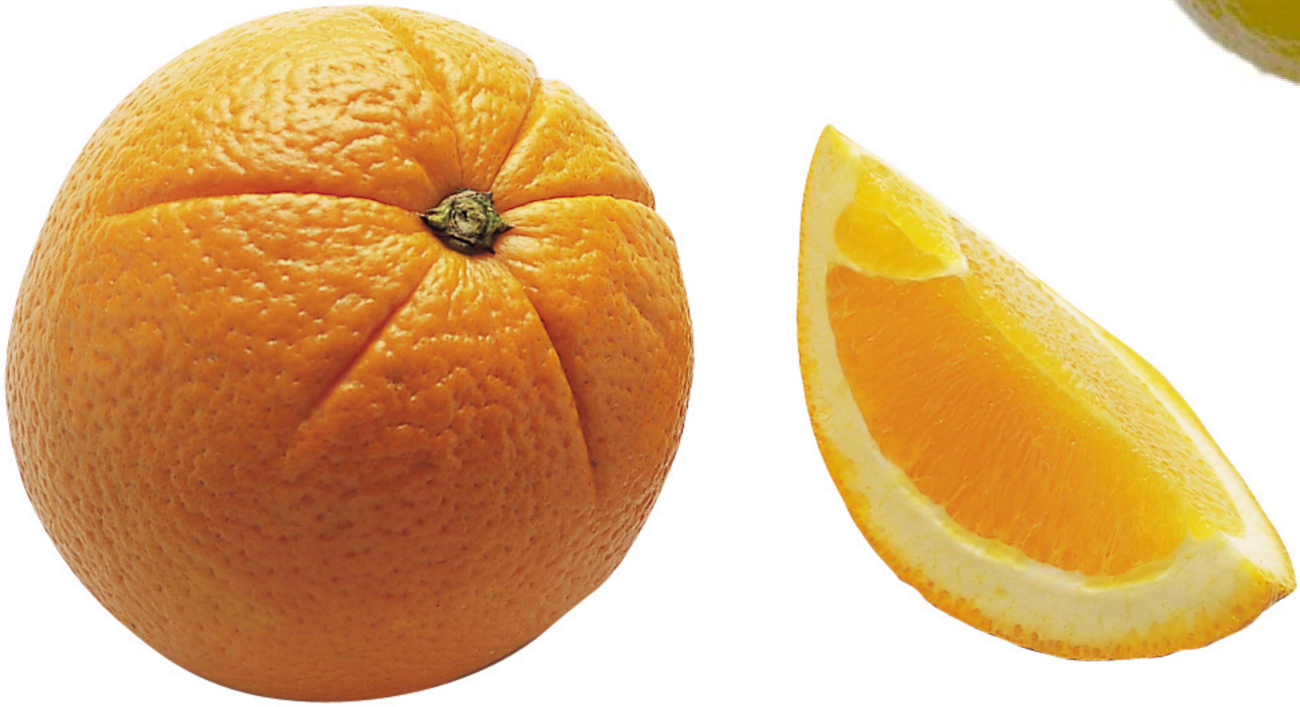
— Parametric and non-parametric prediction

Other ‘Supervised’ settings

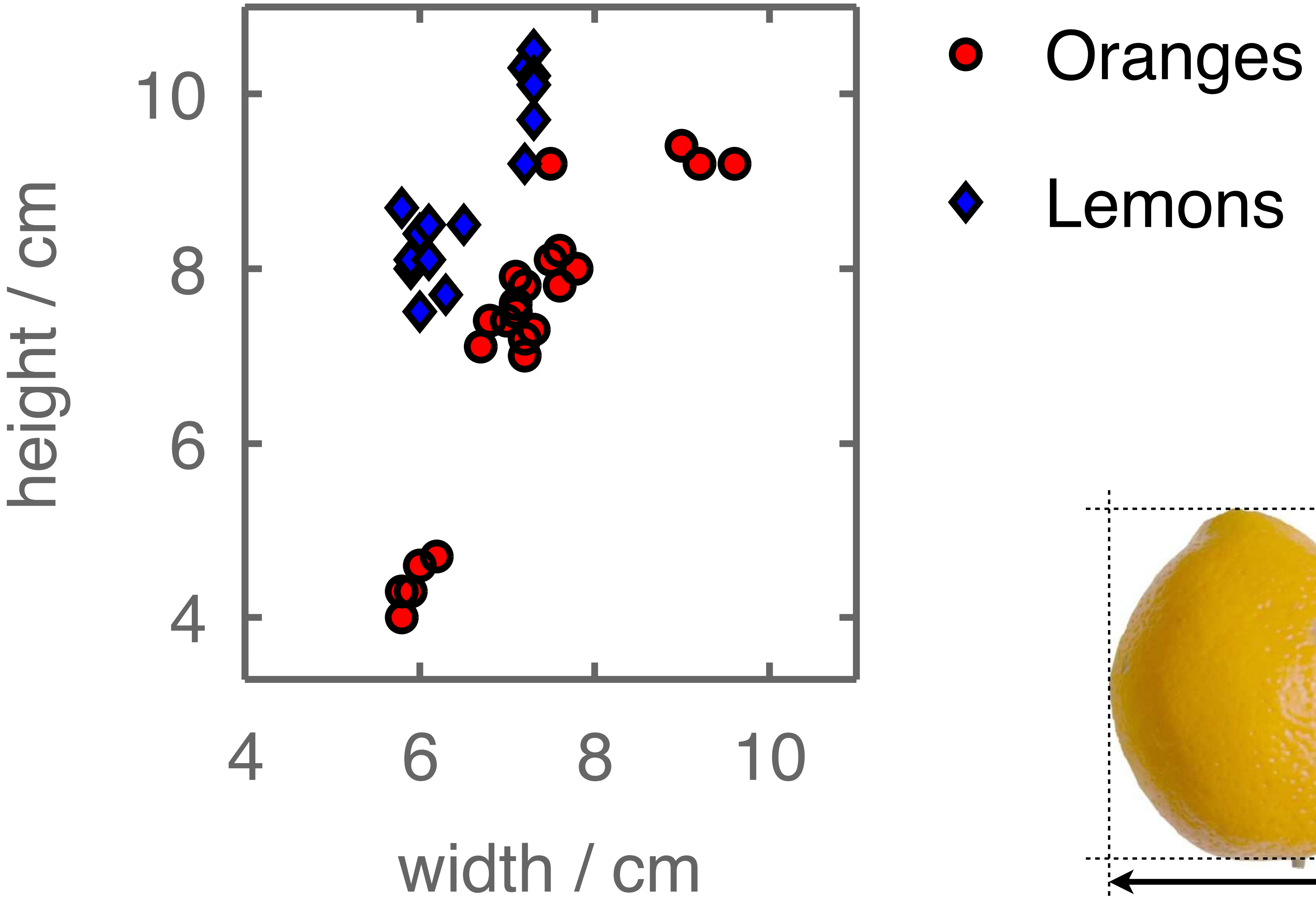
Principles for learning

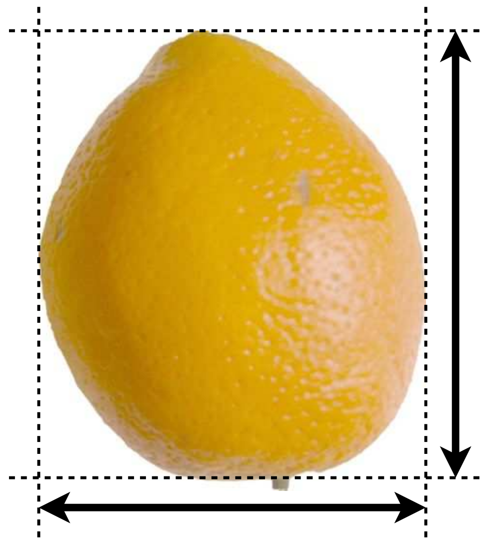
More machine learning paradigms

Oranges and Lemons

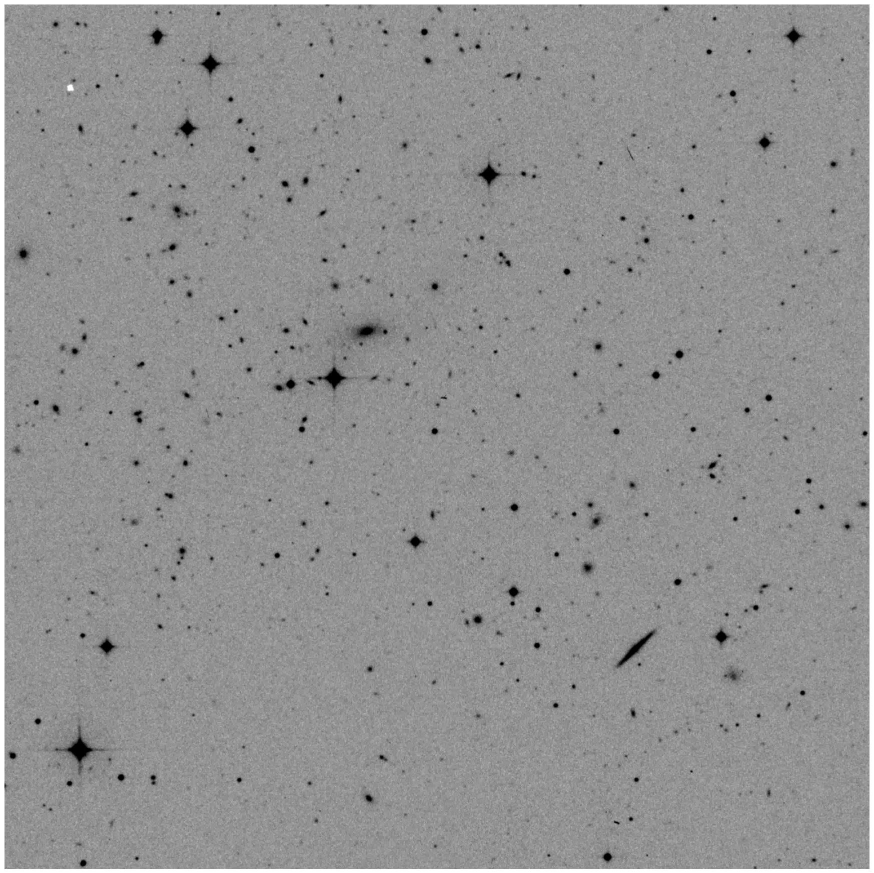


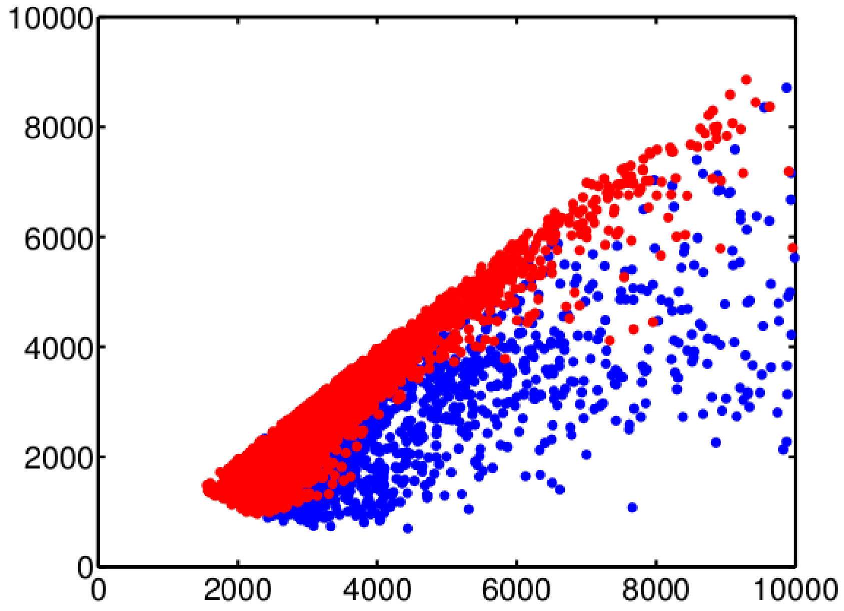
A two-dimensional space





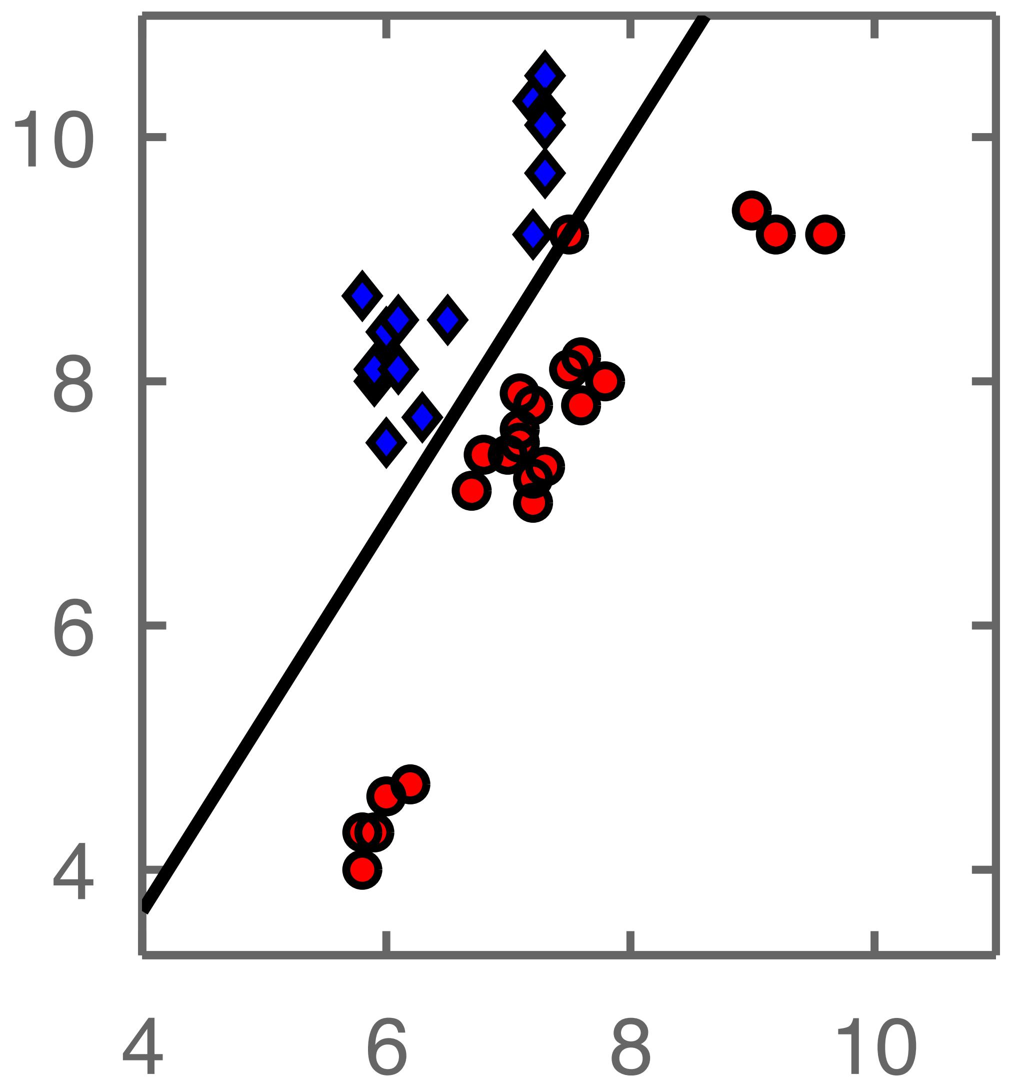
Stars and Galaxies


http://www-wfau.roe.ac.uk/sss/



Minor elliptical axis (y) against Major elliptical axis (x) for stars (red) and galaxies (blue). (Amos Storkey)

Linear classifier



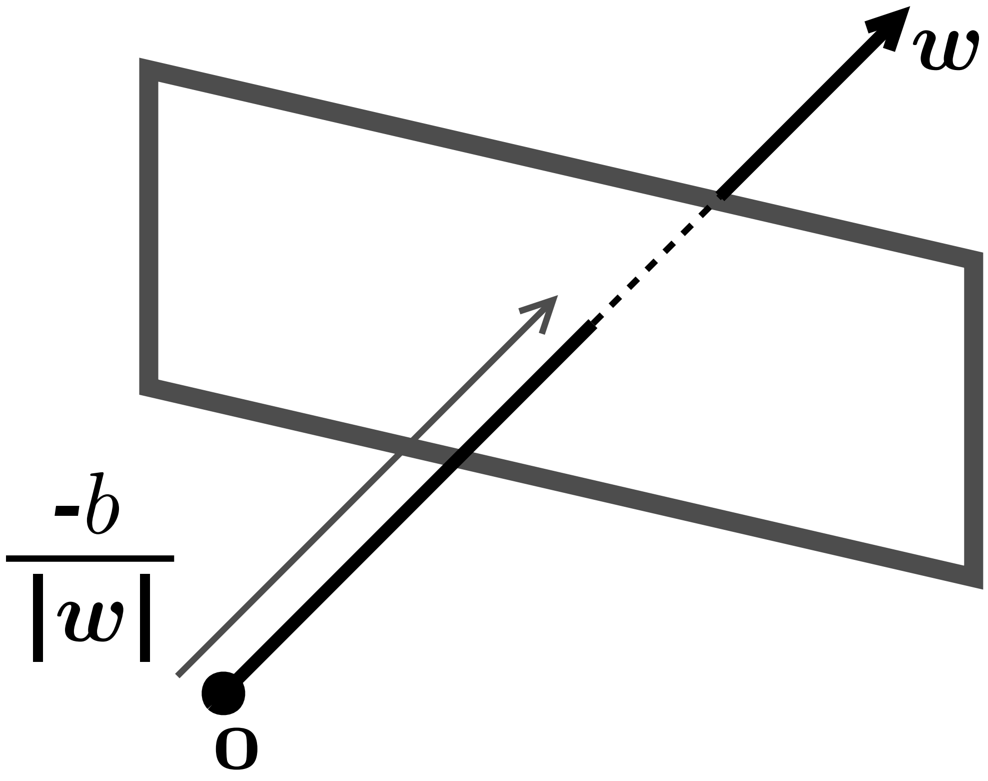
Straight cut dividing input space into two

The weight vector

Define positive class region: $\mathbf{w}^\top \mathbf{x} + b > 0$

$w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots b > 0$

$\sum_d w_d x_d + b > 0$



We will set $b = 0$:

$$\mathbf{w}^\top \mathbf{x} > 0$$

The weight vector

Positive class region $\mathbf{w}^\top \mathbf{x} > 0$

Matlab / Octave:

```
%    ww = Dx1 weights
% Xstar = MxD test cases
y_prediction = sign(Xstar*ww); % Mx1
```

Learn a numerical package: Python+NumPy, R, Matlab/Octave, Eigen+C++,. . .

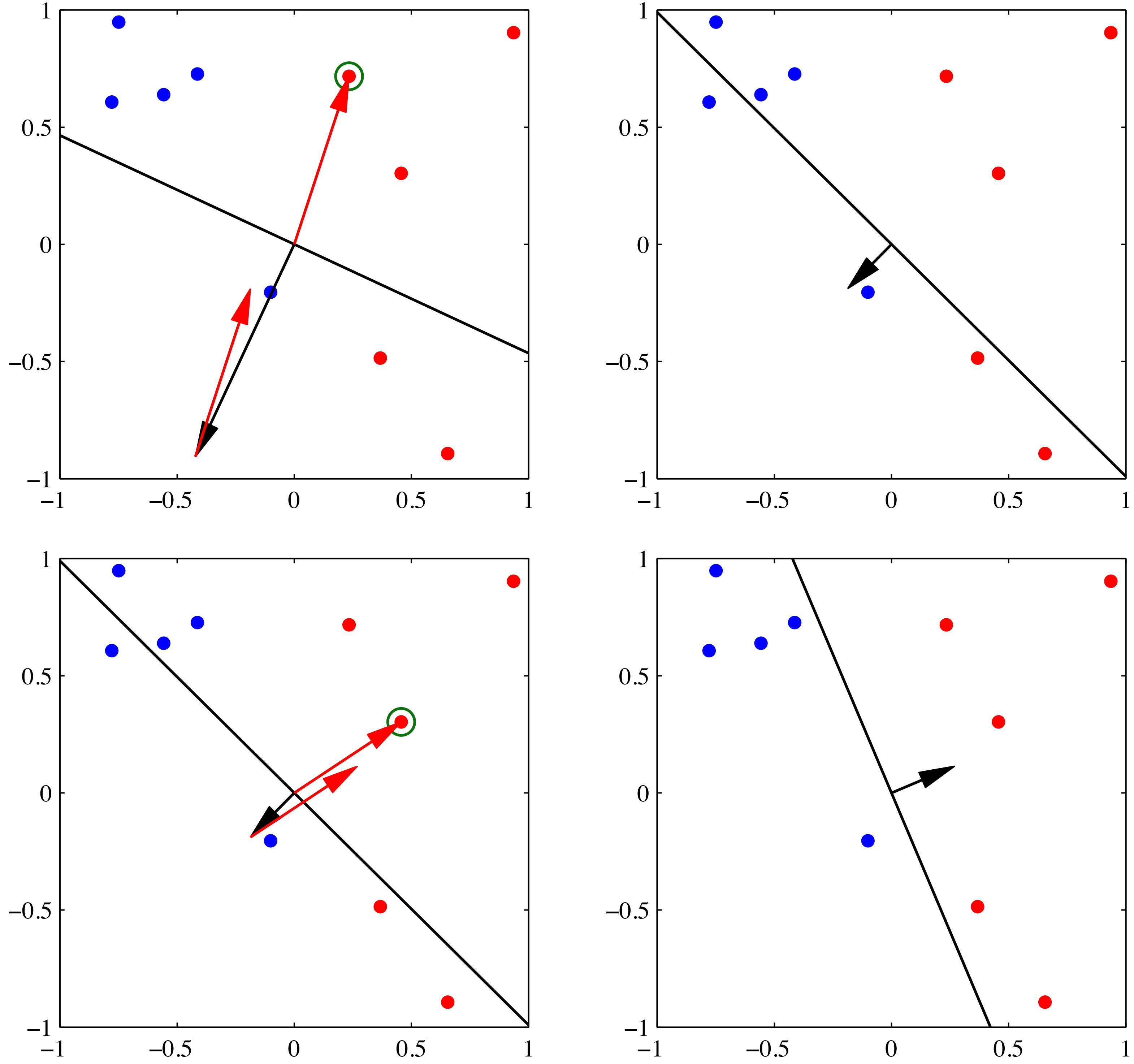
Learning the weights

A ‘perceptron’ learning rule:

$$\hat{y} \leftarrow \mathrm{sgn}(\mathbf{w}^\top \mathbf{x})$$

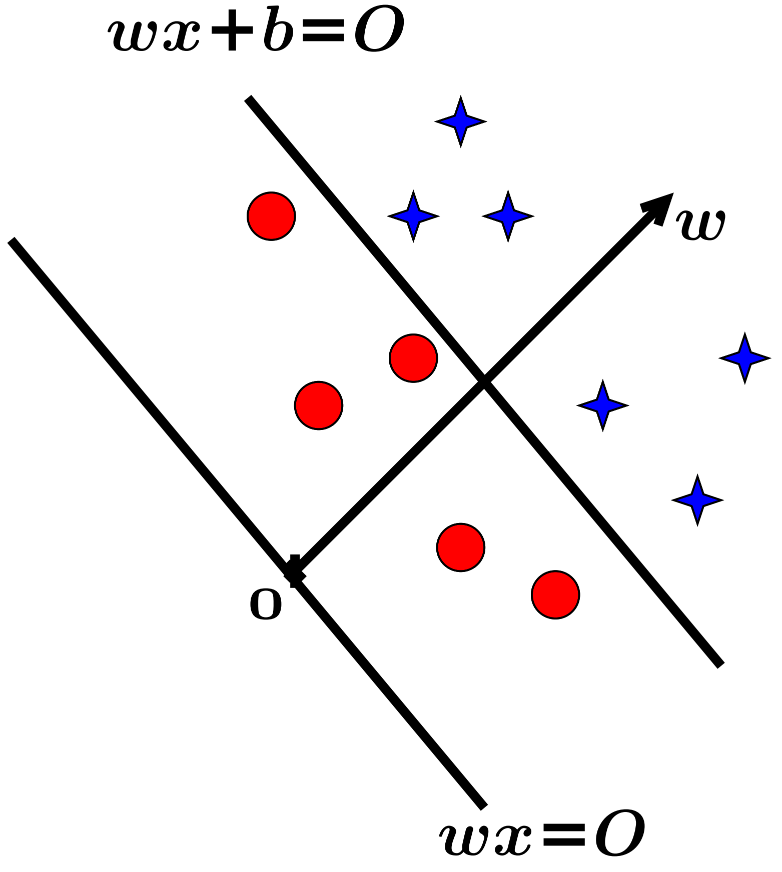
$$\mathbf{w} \leftarrow \mathbf{w} + (y - \hat{y})\mathbf{x}$$

```
% Matlab/Octave code
old_ww = []; ww = zeros(size(xx,2), 1);
while ~isequal(ww, old_ww)
    old_ww = ww;
    for ii = 1:N
       pred = sign(xx(ii, :)*ww);
       ww = ww + (yy(ii) - pred)*xx(ii, :)';
    end
end
```



Pattern Recognition and Machine Learning C. M. Bishop (2006)

Implementing the bias



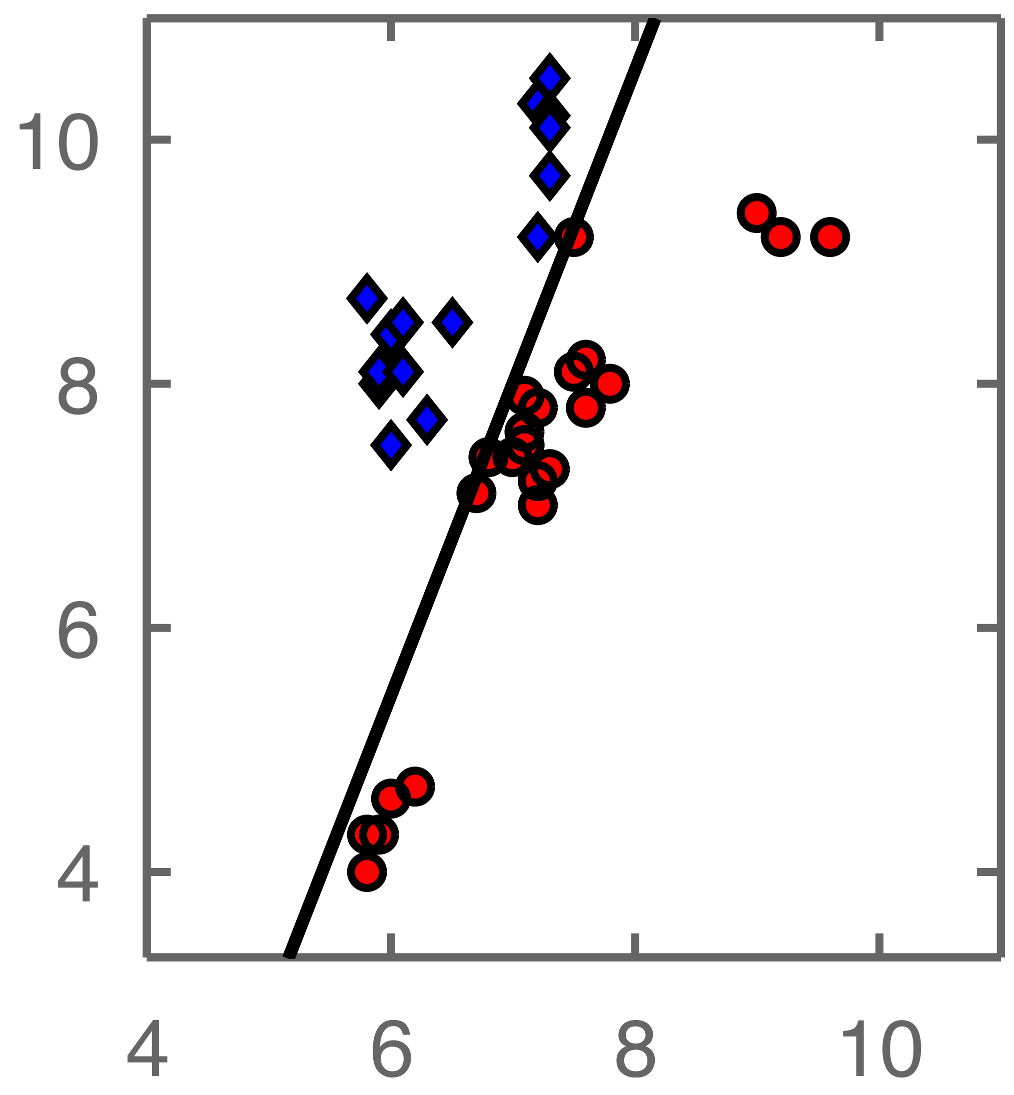
$\mathrm{sgn}(\mathbf{w}^\top \mathbf{x})$ model is inflexible

Need a bias parameter

Or add constant feature to $\mathbf{x}$:

$$\mathbf{x}' = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \quad \mathbf{w}' = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

Output of the perceptron



Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

If we are eventually to understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking, we must first have answers to three fundamental questions:

1. How is information about the physical world sensed, or detected, by the biological system?
2. In what form is information stored, or remembered?
3. How does information contained in storage, or in memory, influence recognition and behavior?

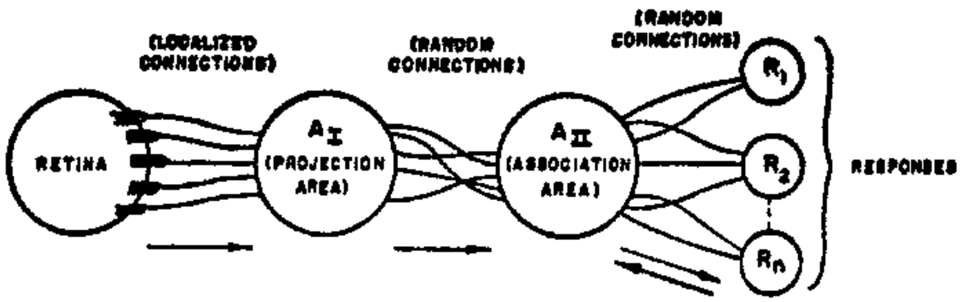
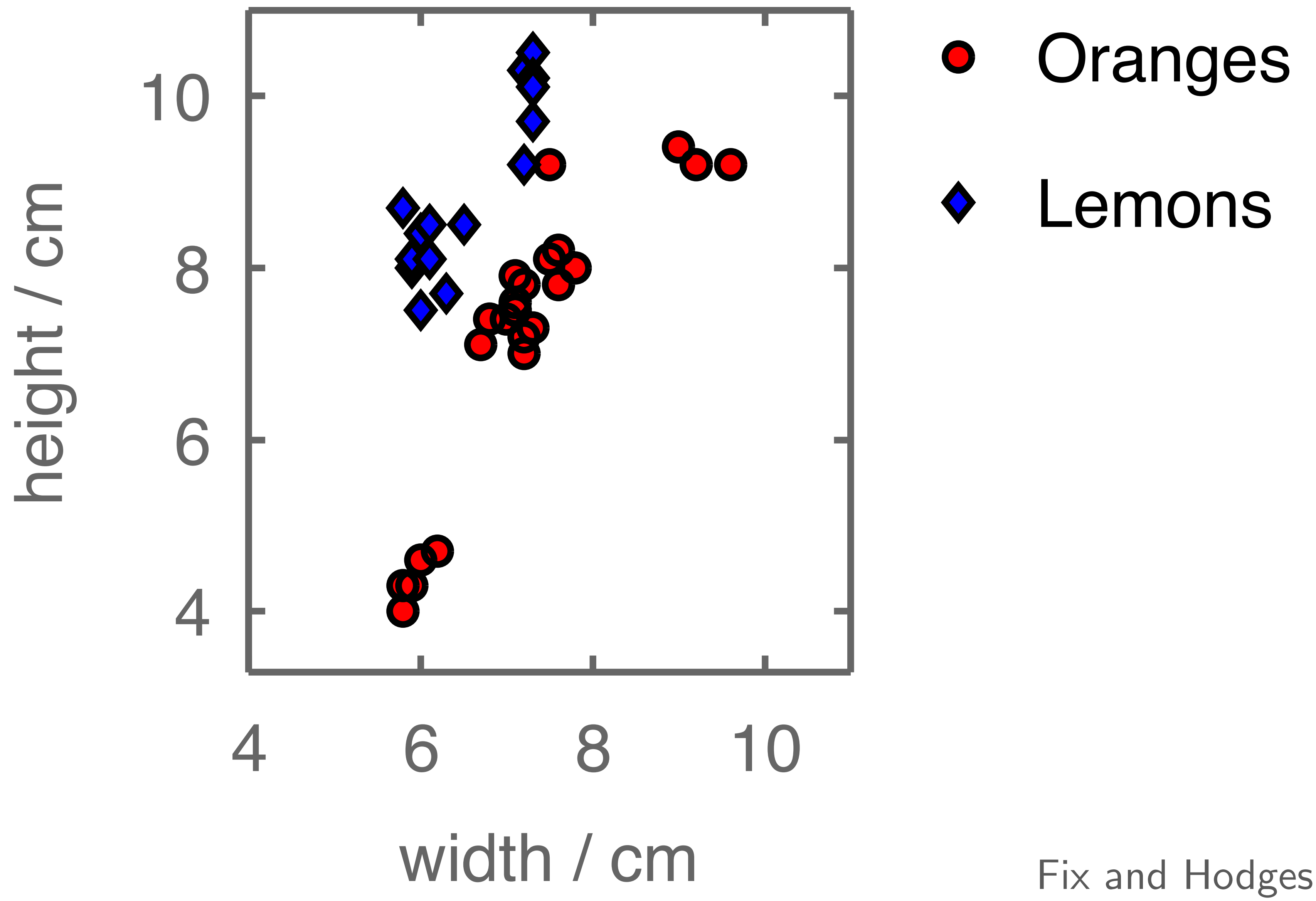


FIG. 1. Organization of a perceptron.

K-Nearest neighbours classifier

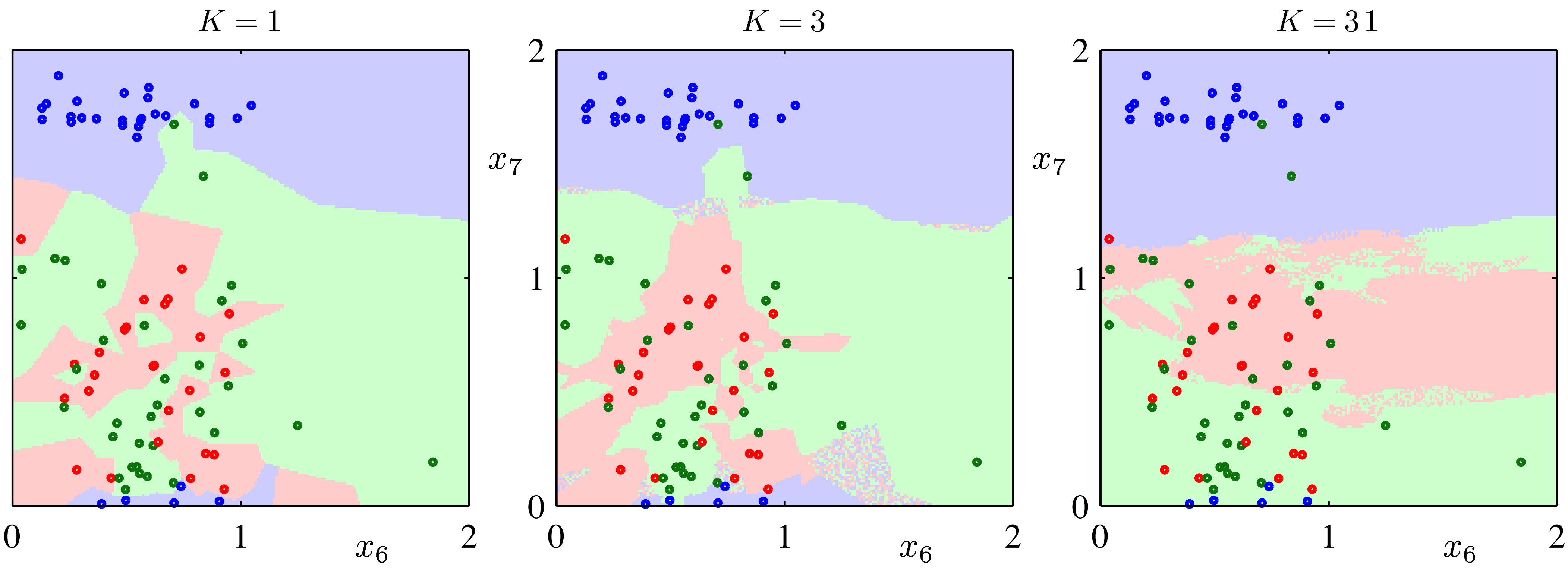


Fix and Hodges (1951)

The odd-ball orange

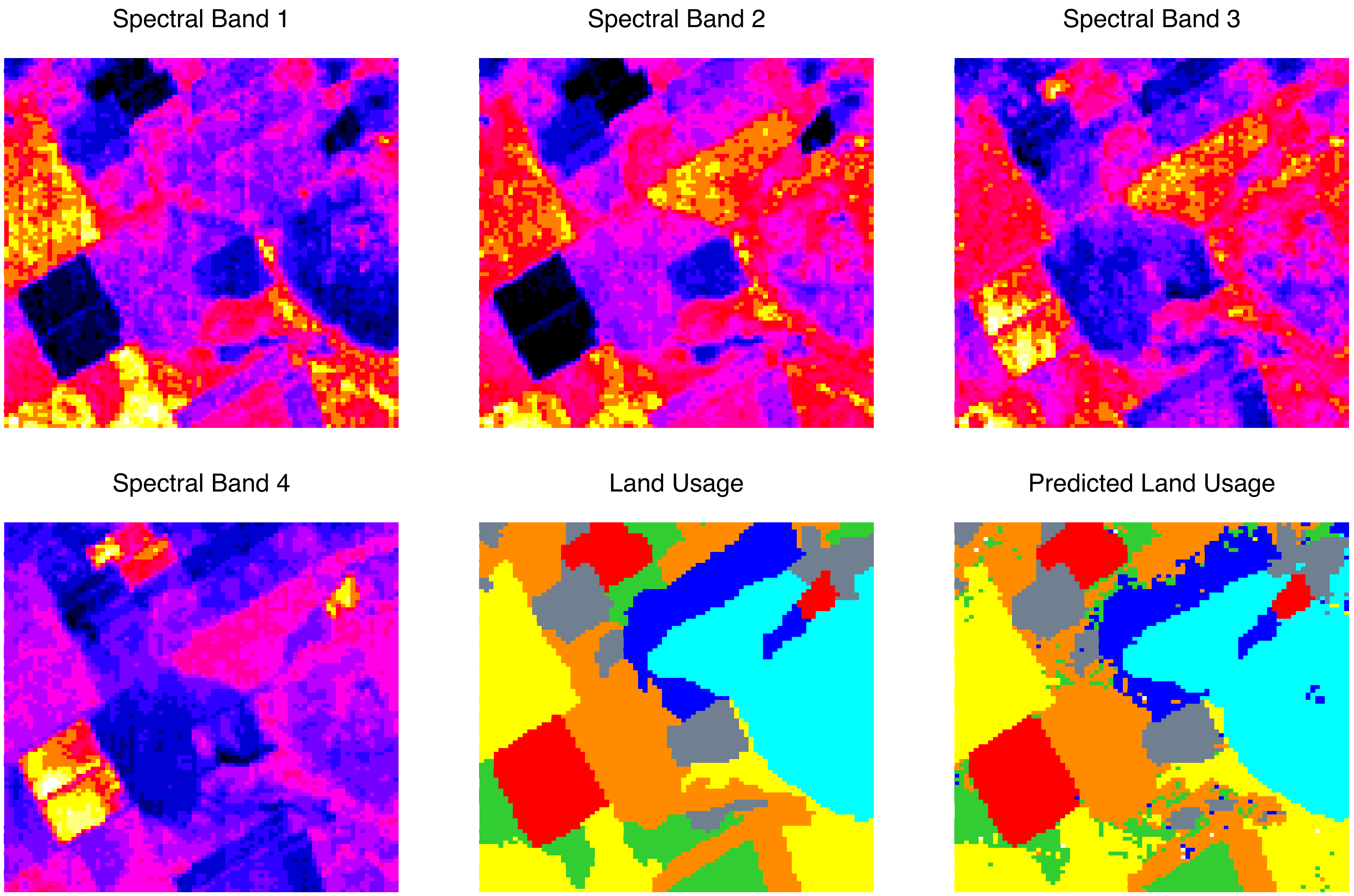
Decision boundaries

Classifying oil flow



Pattern Recognition and Machine Learning C. M. Bishop (2006)

LANDSAT application



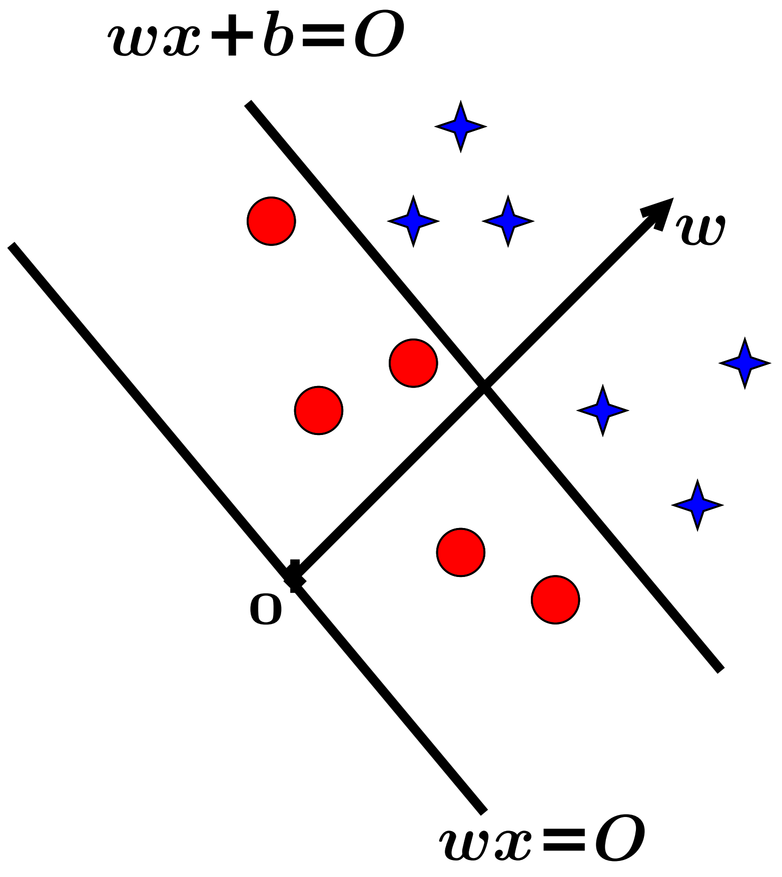
Parametric vs. non-parametric

Started assuming decision boundary is a plane

Non-parametric KNN has no fixed assumption: boundary gets more complicated with more data

Non-parametric methods may need more data and can be computationally intensive

Linear classifier revisited



$\mathrm{sgn}(\mathbf{w}^\top \mathbf{x})$ model is inflexible

Need a bias parameter

Or add constant feature to $\mathbf{x}$:

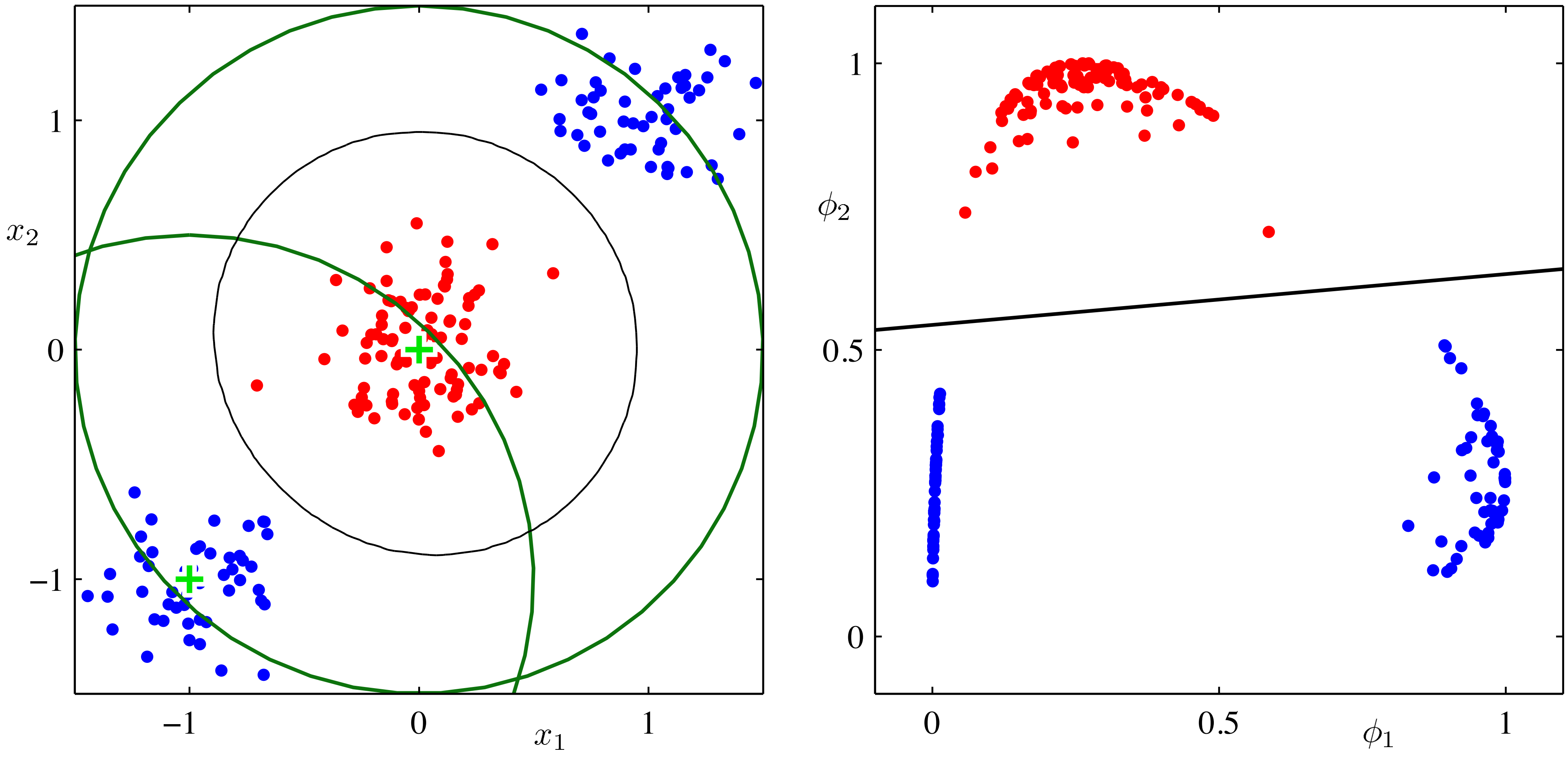
$$\mathbf{x}' = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \quad \mathbf{w}' = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

If not linear separable must extend model. . .

. . . or add features.

Nonlinear basis functions

Create new feature vector: $\mathbf{x}' = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ 1 \end{bmatrix}$



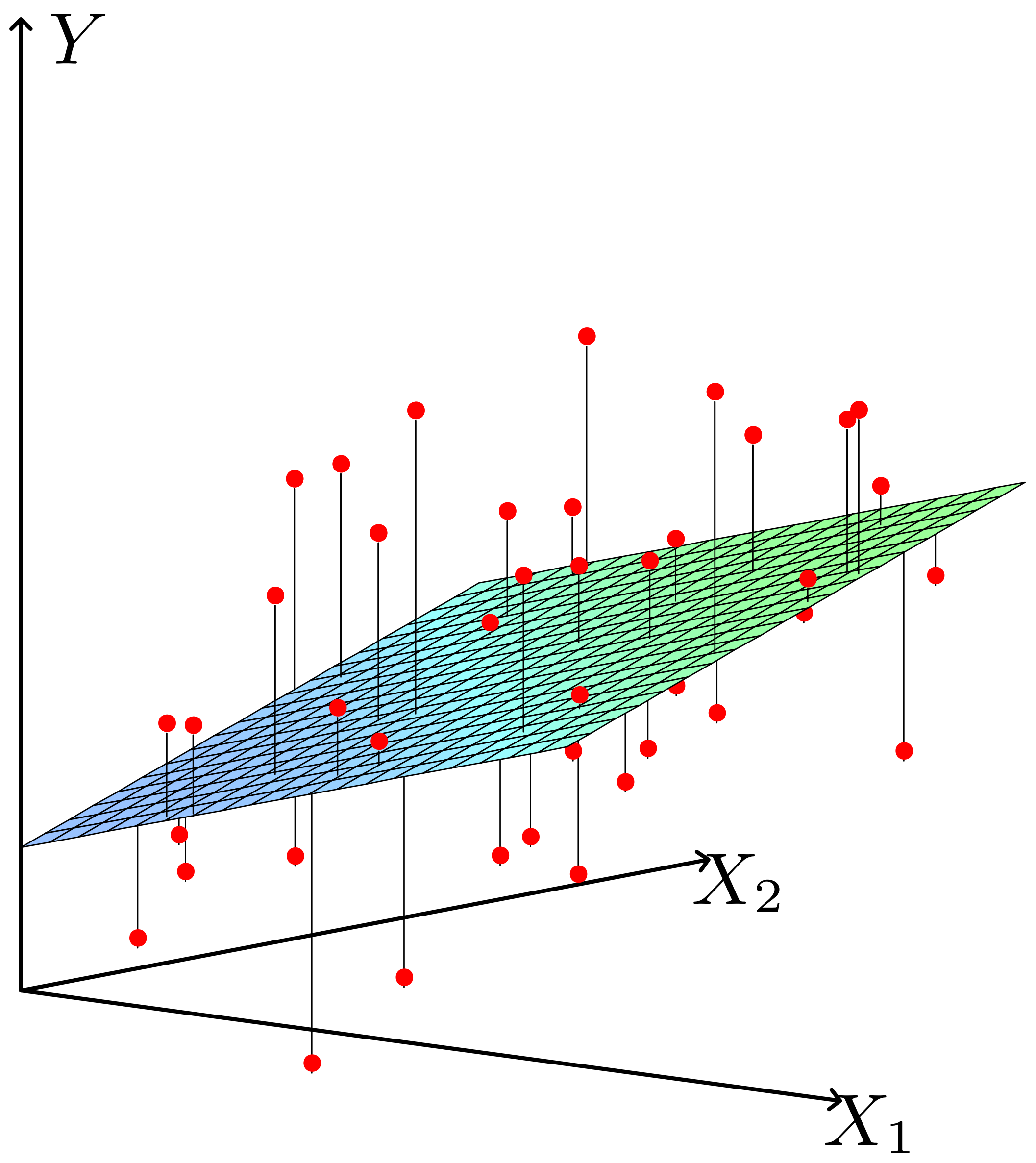
Batch supervised learning

Given: example inputs and targets (training set)
Task: predict targets for new inputs (test set)

Examples:

— classification (binary and multi-class)
— (real-valued) regression
— ordinal regression
— Poisson regression
— ranking(?)

Linear regression



Find linear function

$$\hat{\mathbf{y}} = X\mathbf{w}$$

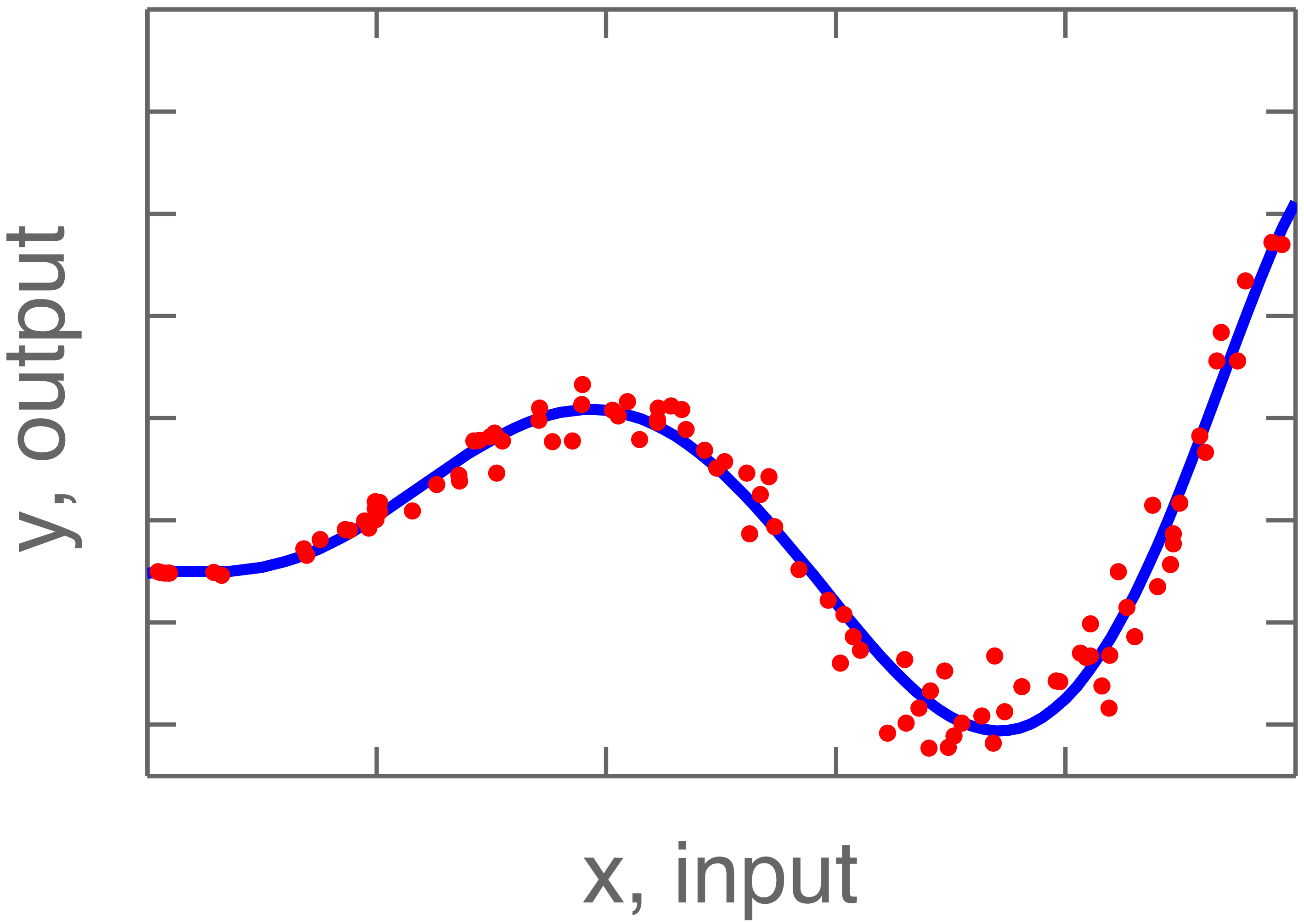
that minimizes sum of squared residuals from $\mathbf{y}$.

Matlab/Octave:

```
ww = X \ yy
```

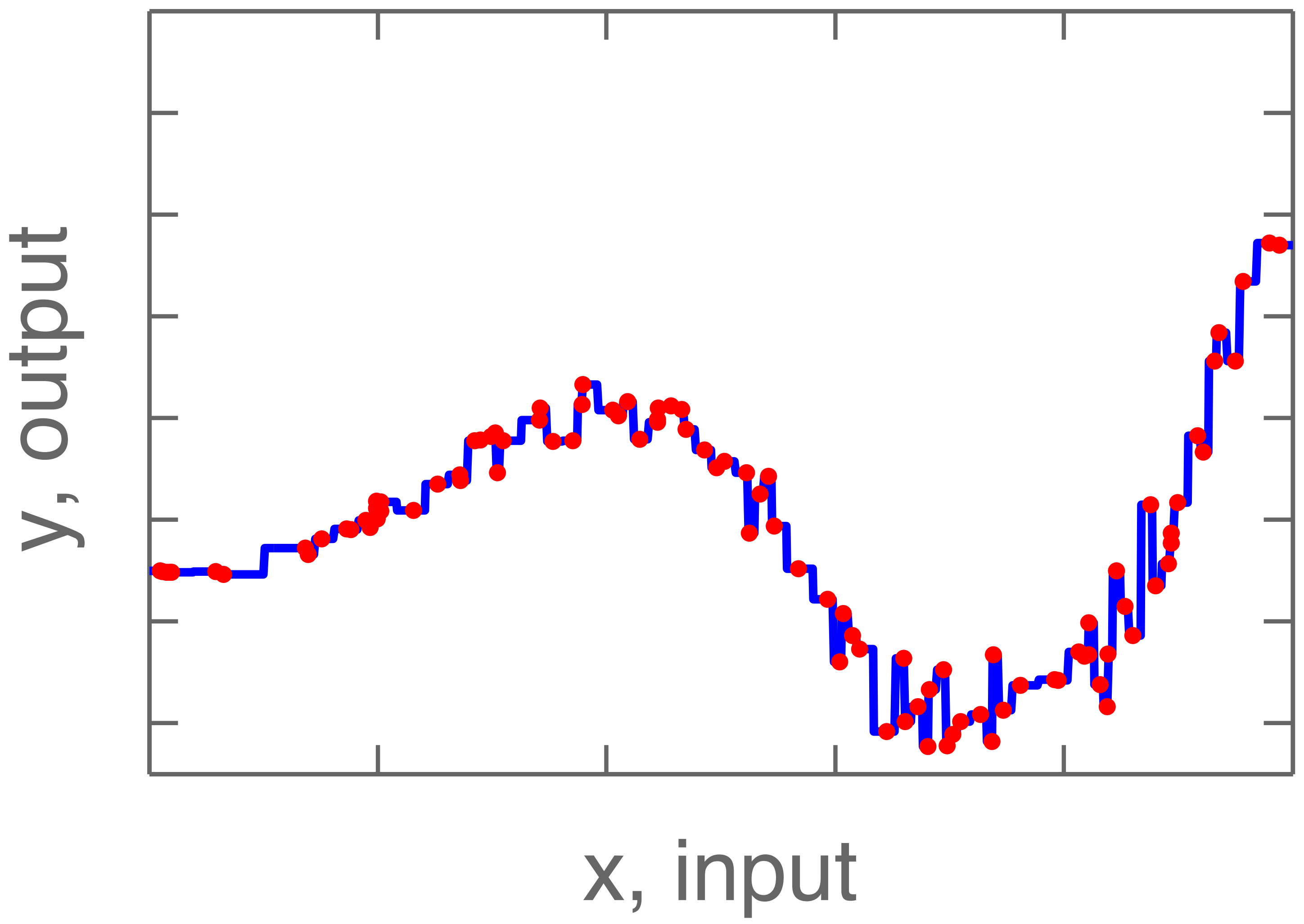
Elements of Statistical Learning (2nd Ed.) © Hastie, Tibshirani & Friedman 2009

Linear regression (with features)



```
X = [ones(N, 1), xx, xx.^2, xx.^3, xx.^4, xx.^5, xx.^6];
Xnew = [ones(N, 1), xnew, xnew.^2, xnew.^3, xnew.^4, xnew.^5, xnew.^6];
ww = X \ yy;
ynew = Xnew * ww;
```

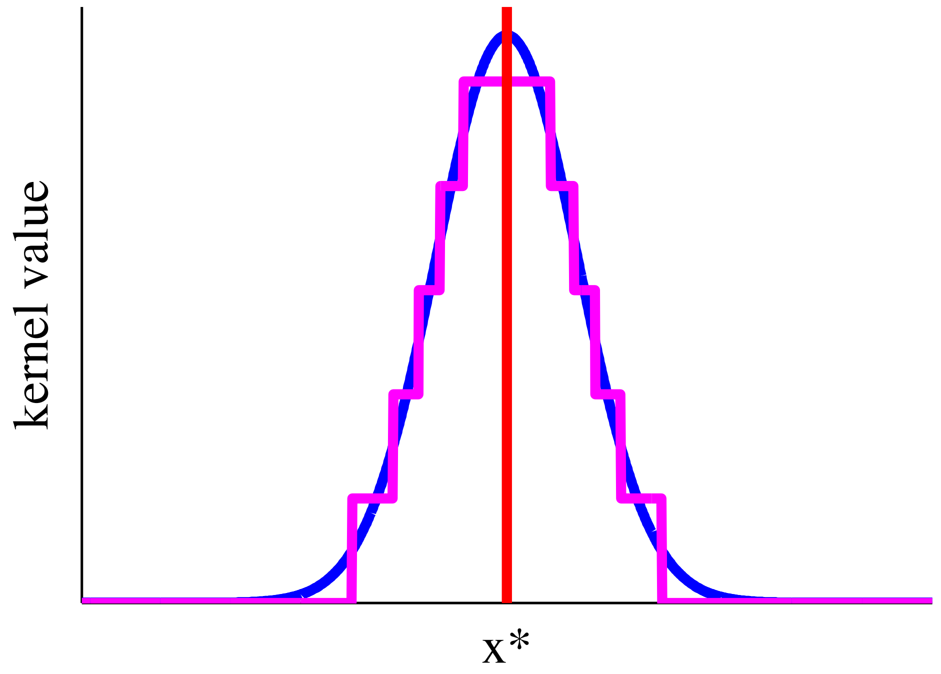
Neighbour-based regression



Take height from nearest input

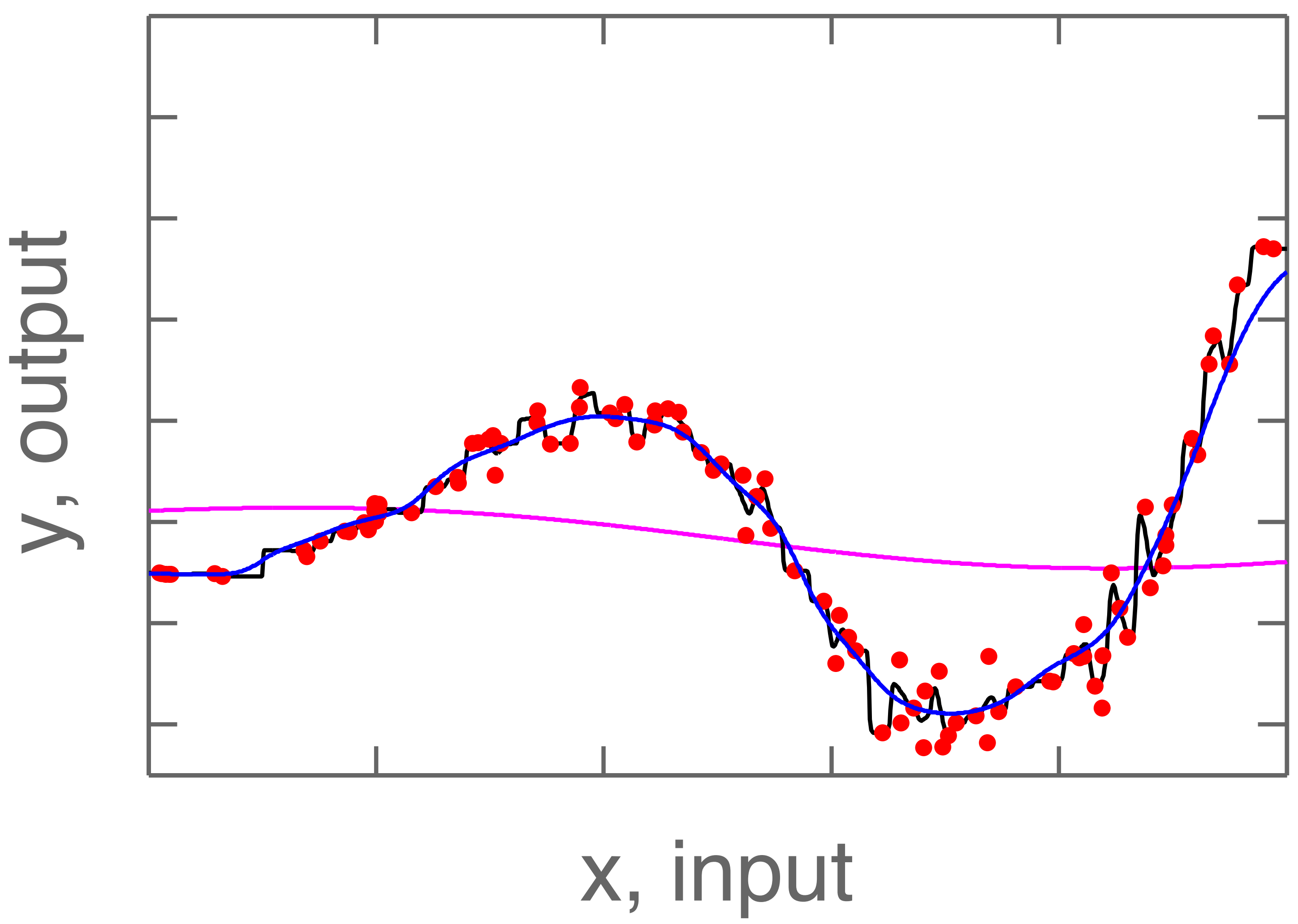
Simplest kernel smoothing

Weight points in proportion to a *kernel*:



$$\hat{y}_* = \sum_i w_i y_i, \quad \text{where } w_i = k(x_*, x_i) / \sum_j k(x_*, x_j)$$

Simplest kernel smoothing

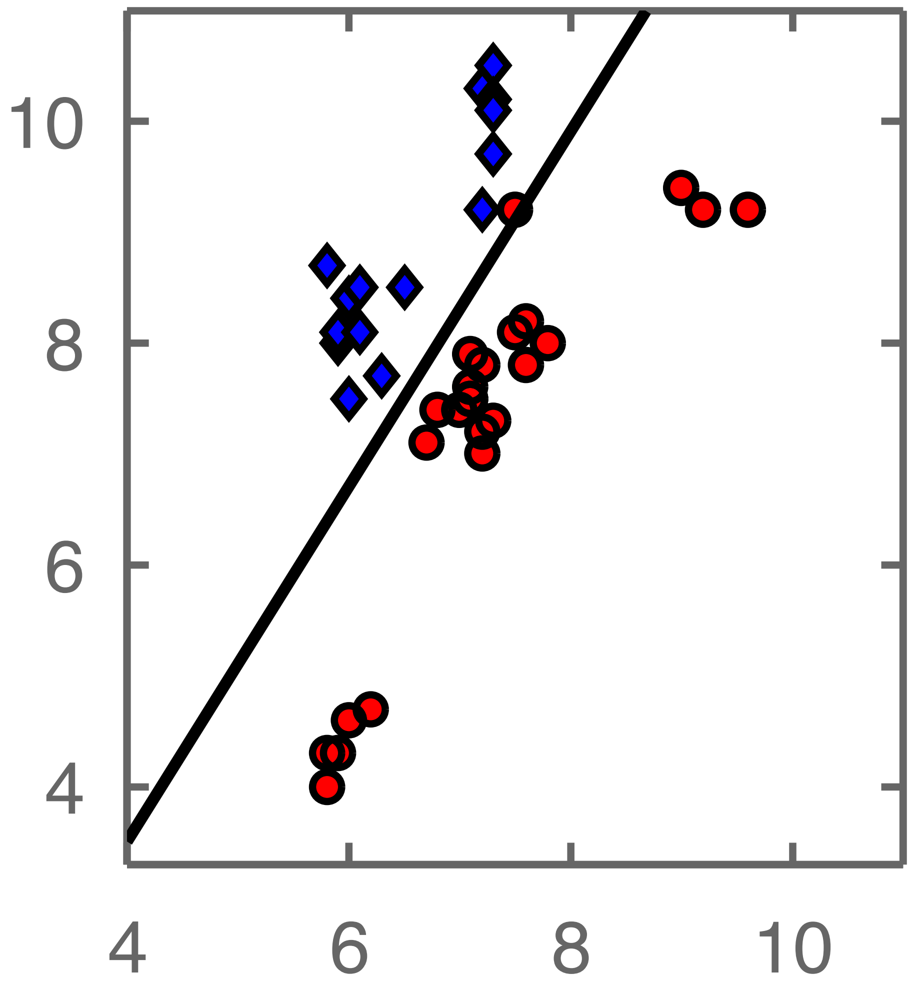


$$\hat{y}_* = \sum_i w_i y_i, \quad \text{where } w_i = k(x_*, x_i) / \sum_j k(x_*, x_j)$$

Least squares classifier

Why not run regressors on binary targets?

```
% fit yy = xx*ww
% by least squares
ww = xx \ yy;
```

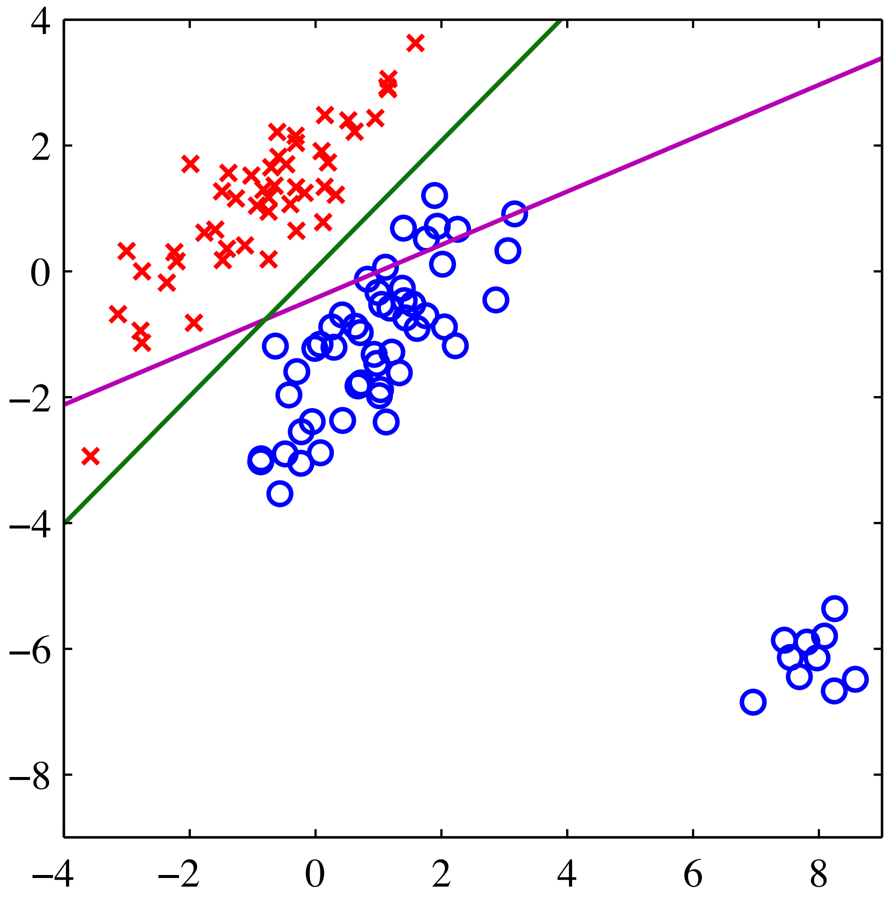


Line shows `xx*ww == 0`. ww(3) is a bias: xx(:,3) = 1

Least squares classifier (2)

Why not run regressors on binary targets?

```
% fit yy = xx*ww
% by least squares
ww = xx \ yy;
```



PRML C. M. Bishop (2006)

Magenta line shows `xx*ww == 0`. ww(3) is a bias: xx(:,3) = 1

Roadmap

Supervised learning:

Many ways of mapping inputs $\longrightarrow$ outputs

How do we choose what to do?

How do we know if we are doing well?

Later: a little on practical issues,
other ideas in machine learning.

Algorithm's Objective/Cost

Formal objective for algorithms:

— Minimize a cost function

— Or maximize an objective function

Proving convergence:

— does objective monotonically improve?

Considering alternatives:

— does another algorithm score better?

Loss functions

We want to specify the objective of an algorithm

One idea: consider *loss function* $L(\hat{y}(\mathbf{x}_*); y_*)$

Would be good to minimize loss at test time

Minimizing empirical loss might be a reasonable proxy: $\sum_i L(\hat{y}(\mathbf{x}_i); y_i)$

Choosing loss functions

Motivated by the application:

0–1 error, achieving a tolerance, business cost

Computational convenience:

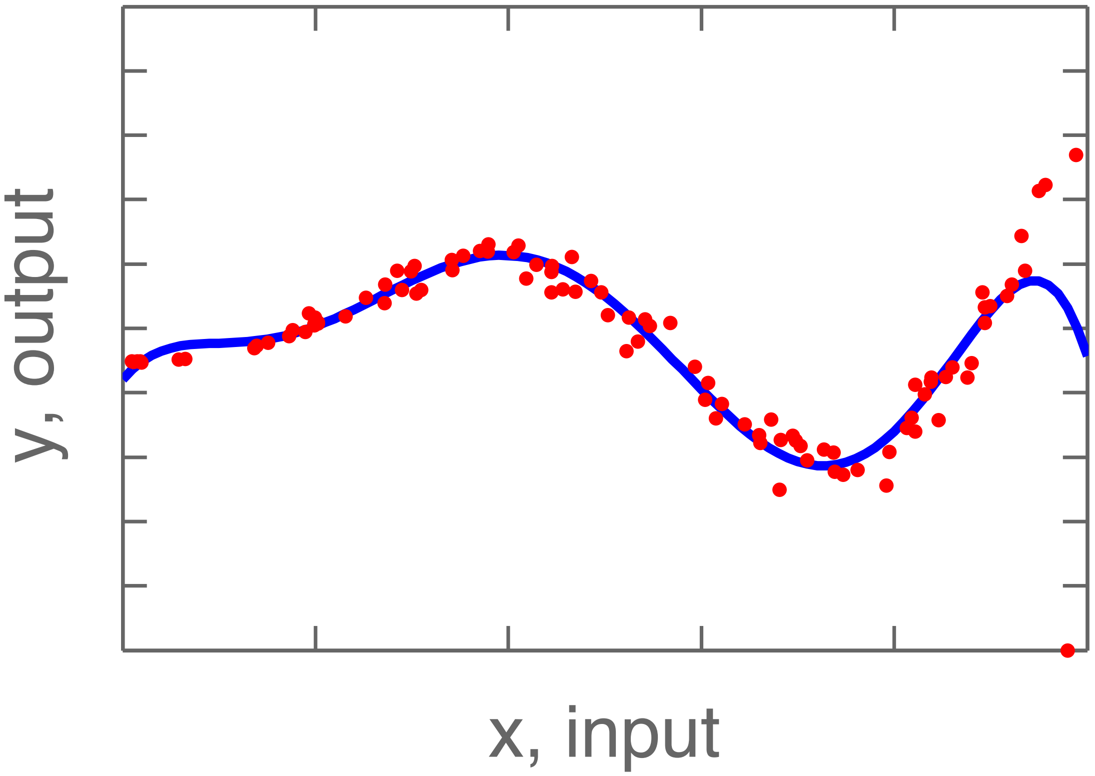
Differentiability, convexity

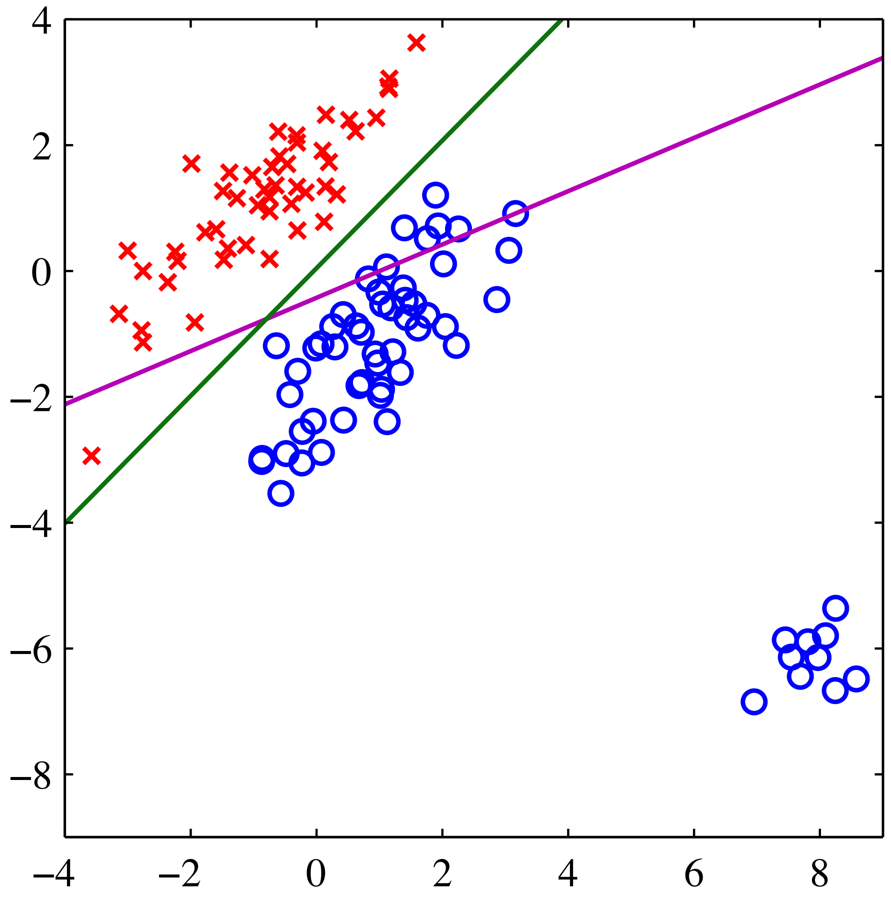
Beware of loss being dominated by artifacts:

— Outliers

— Unbalanced classes

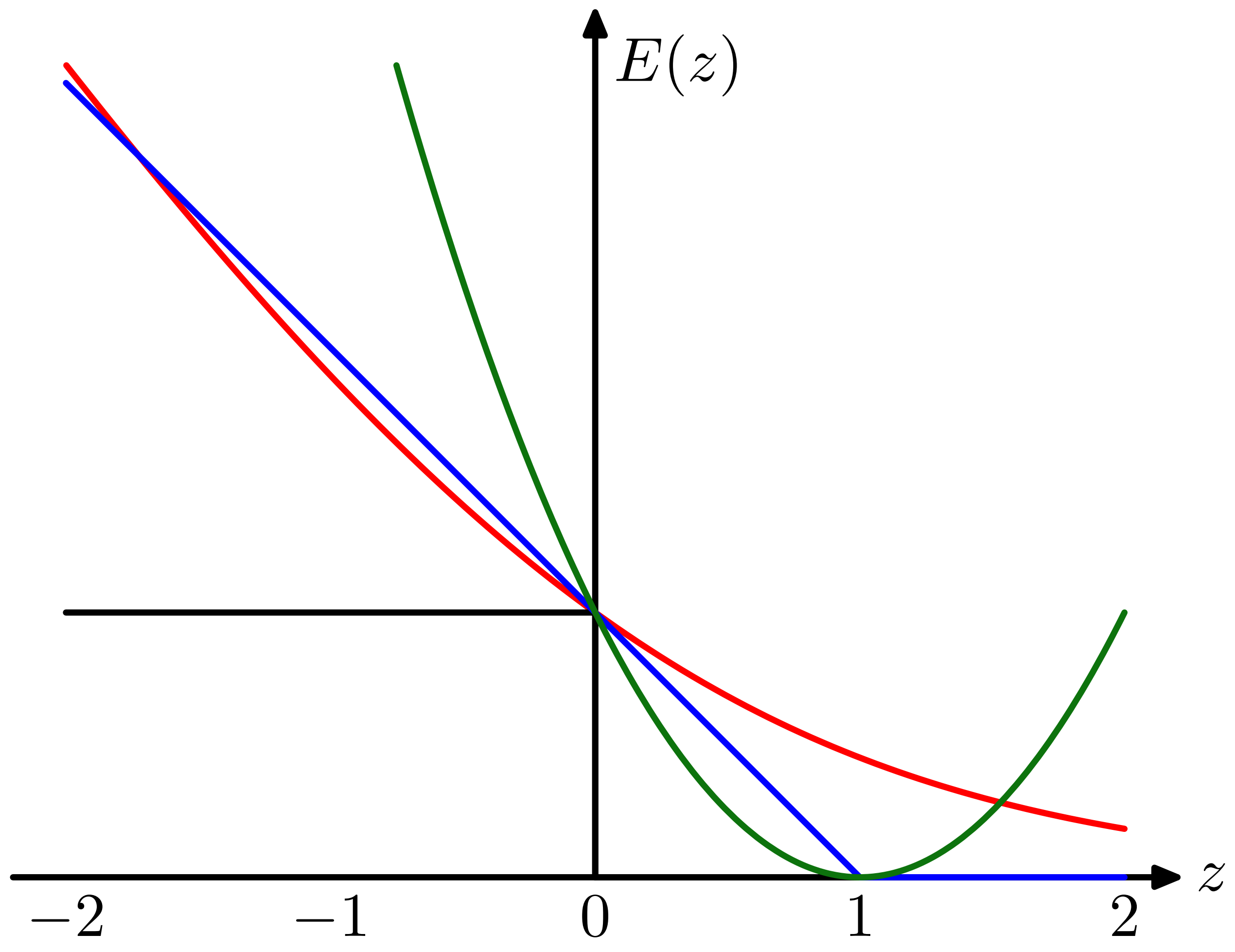
Squared error





$$z = \mathbf{w}^\top \mathbf{x}, \qquad \hat{y} = \mathrm{sgn}(z)$$

Comparing loss functions



$$z = \mathbf{w}^\top \mathbf{x}$$

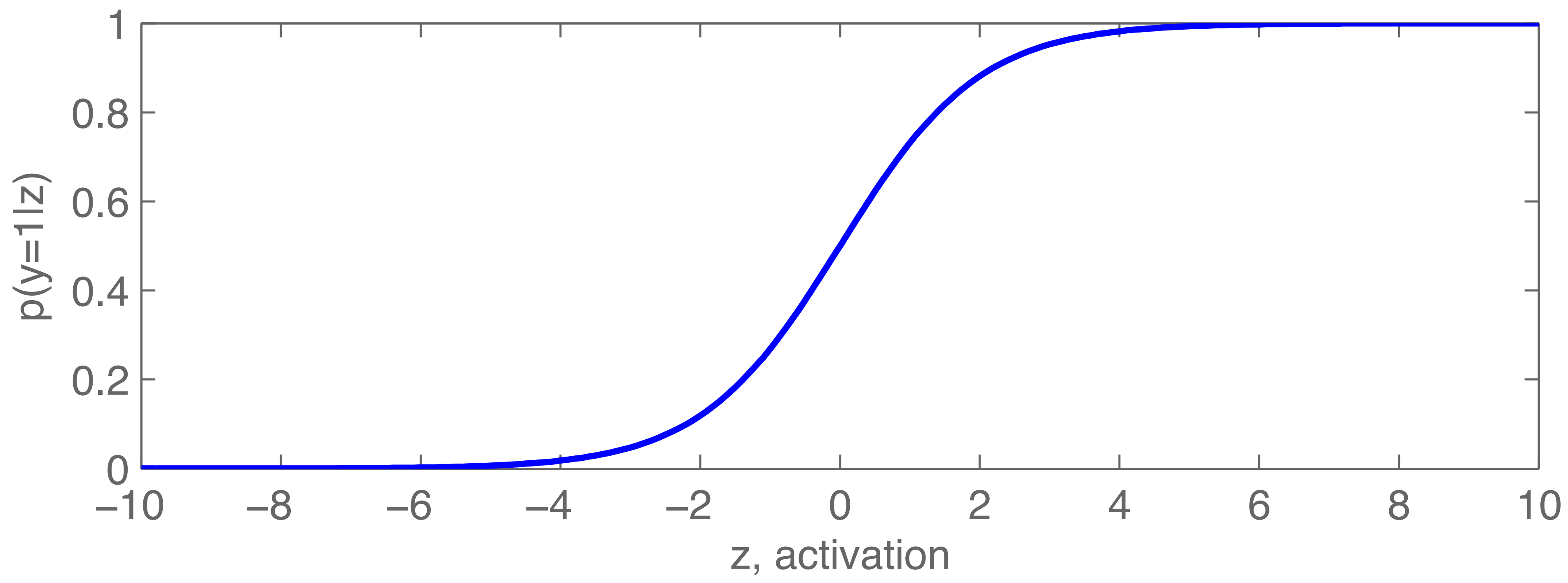
Pattern Recognition and Machine Learning C. M. Bishop (2006)

Logistic regression

Probabilistic model for labels:

$$P(y\,|\,\mathbf{x},\mathbf{w}) = \frac{1}{1+e^{-y(\mathbf{w}^\top\mathbf{x})}},$$

activation $z = \mathbf{w}^\top\mathbf{x}$



Logistic regression

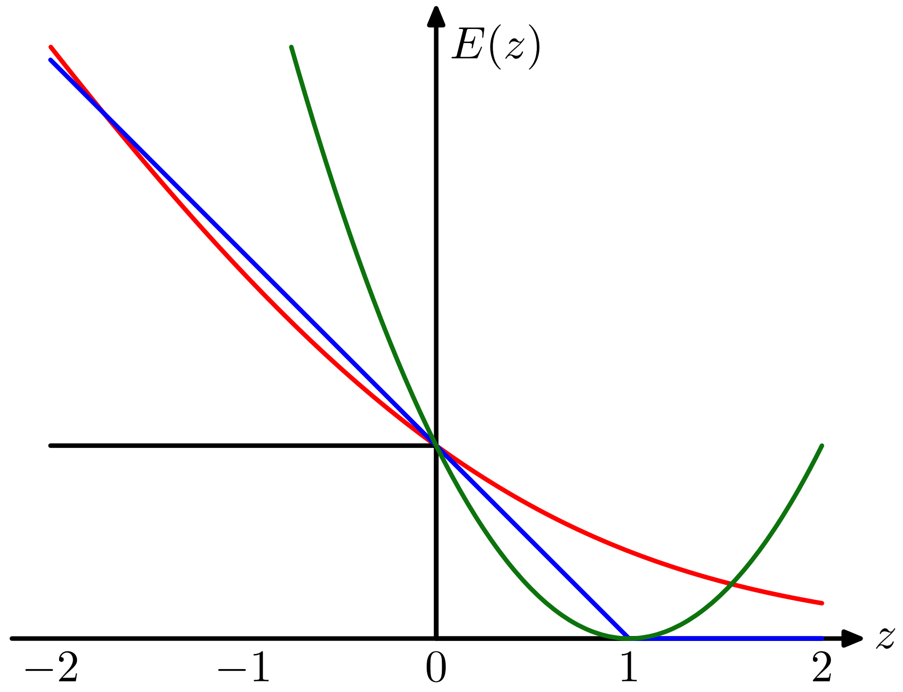
Maximize the likelihood of w:

$$P(\mathbf{y}\,|\,X,\mathbf{w}) = \prod_i \frac{1}{1+e^{-y_i(\mathbf{w}^\top\mathbf{x}_i)}}$$

$$\log P(\mathbf{y}\,|\,X,\mathbf{w}) = -\sum_i \log(1+e^{-y_i(\mathbf{w}^\top\mathbf{x}_i)})$$

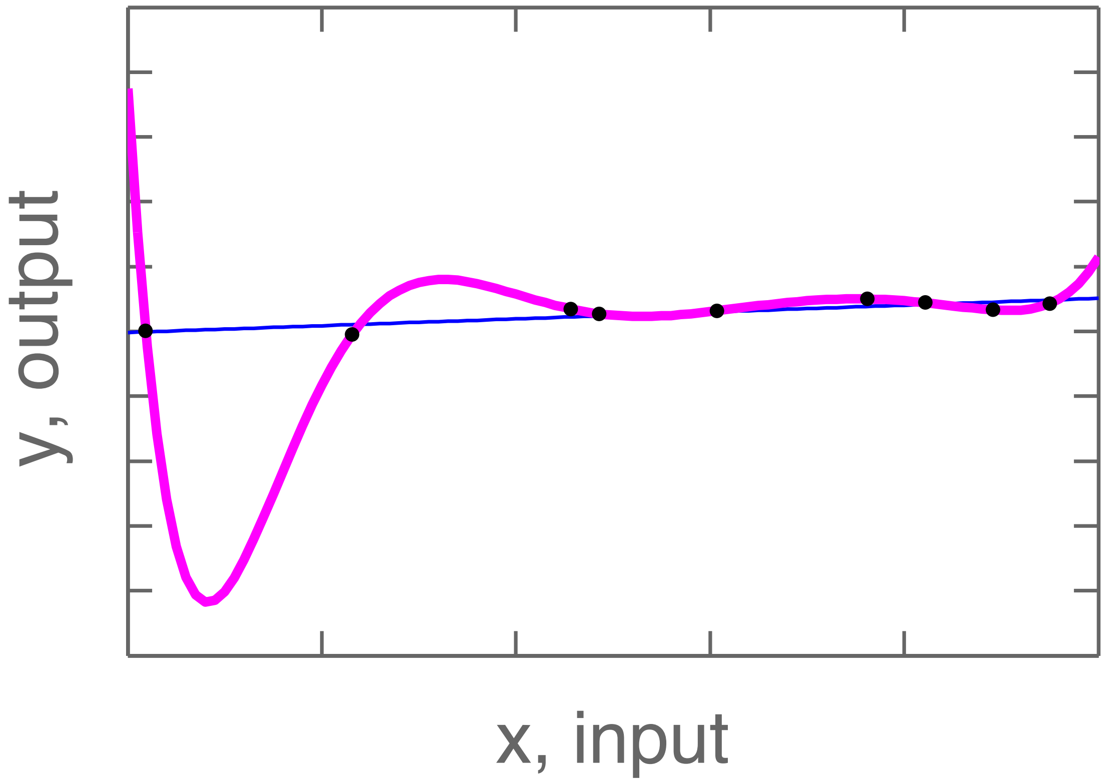
Pay loss:

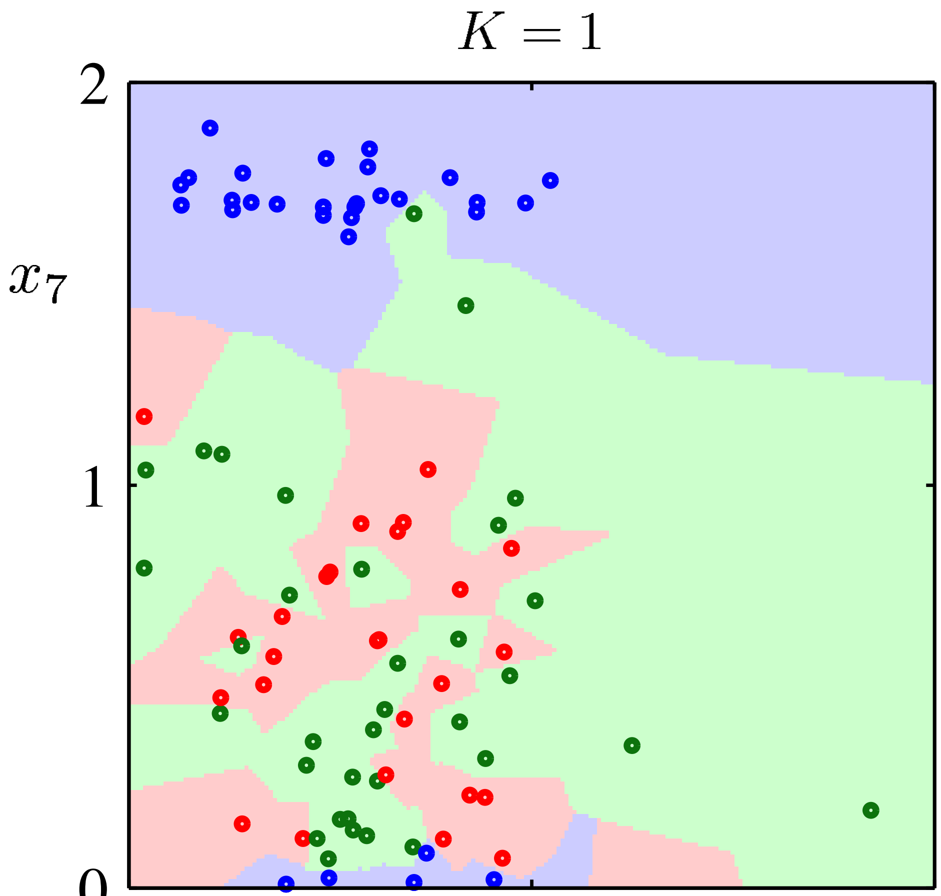
$$\log(1+e^{-y_i(\mathbf{w}^\top\mathbf{x}_i)})$$



Over fitting

We can make the empirical loss zero:

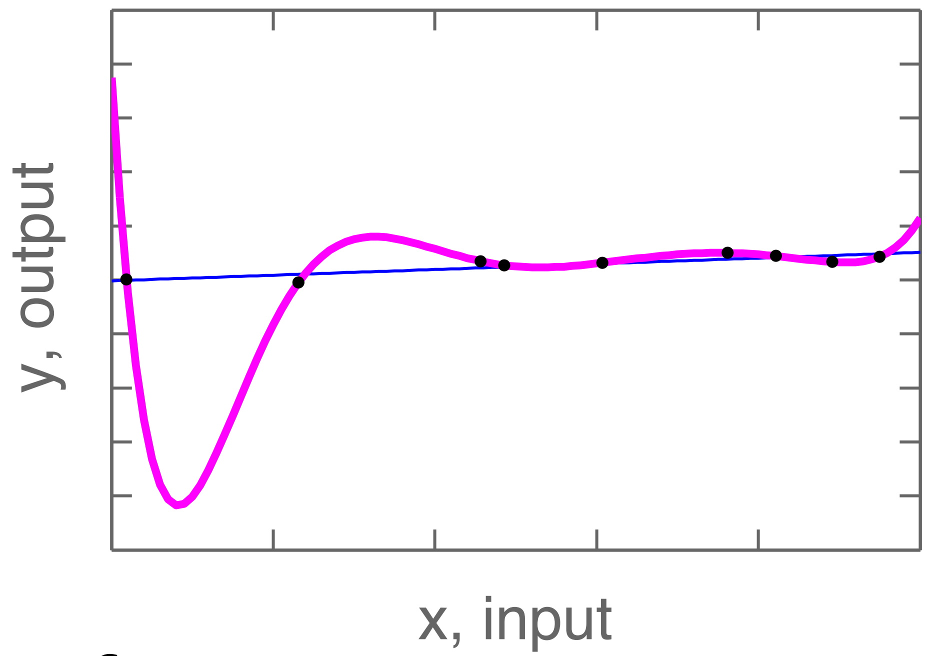




Generalization

Want to do well on future, unknown data

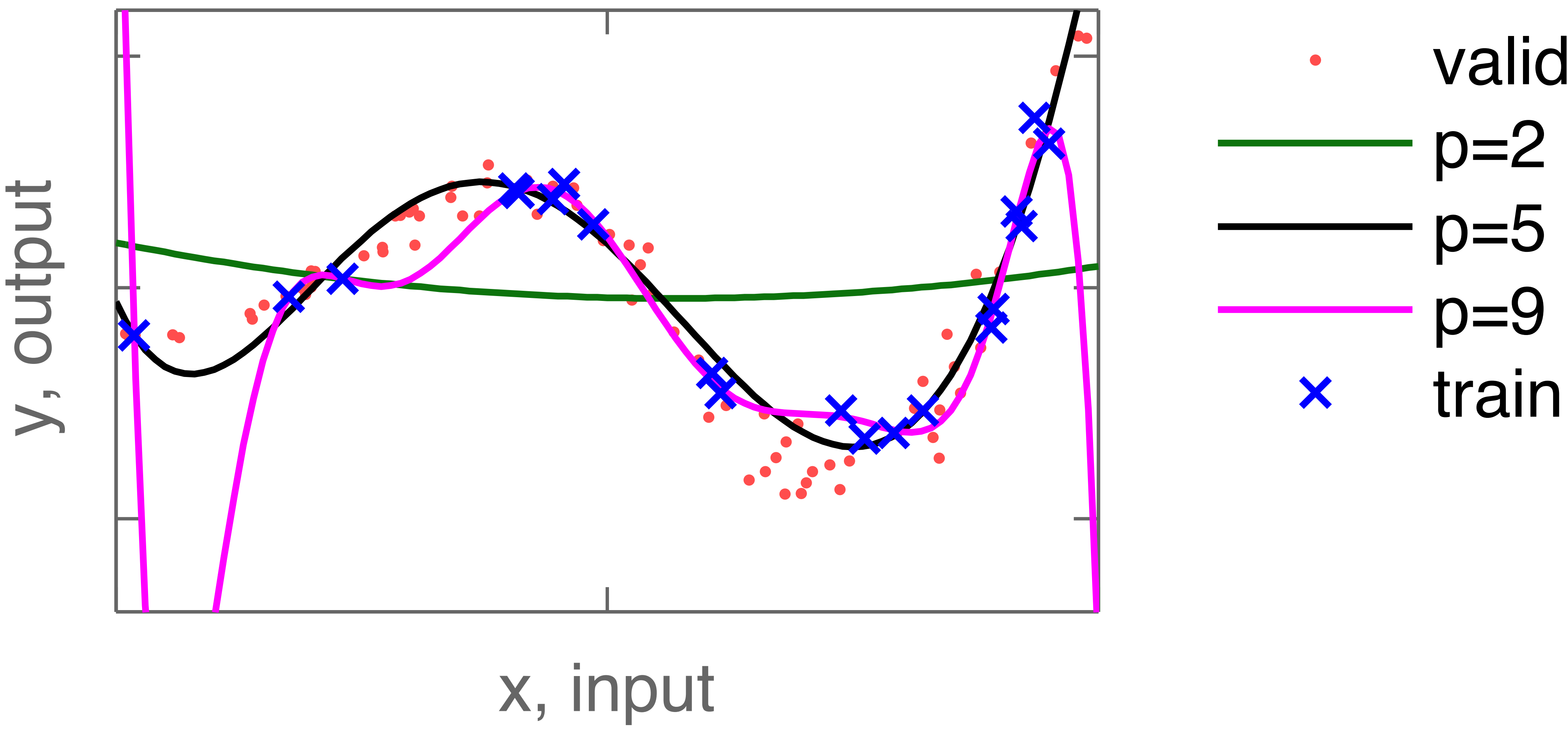
There is no one right way to proceed



Multiple theoretical points of view

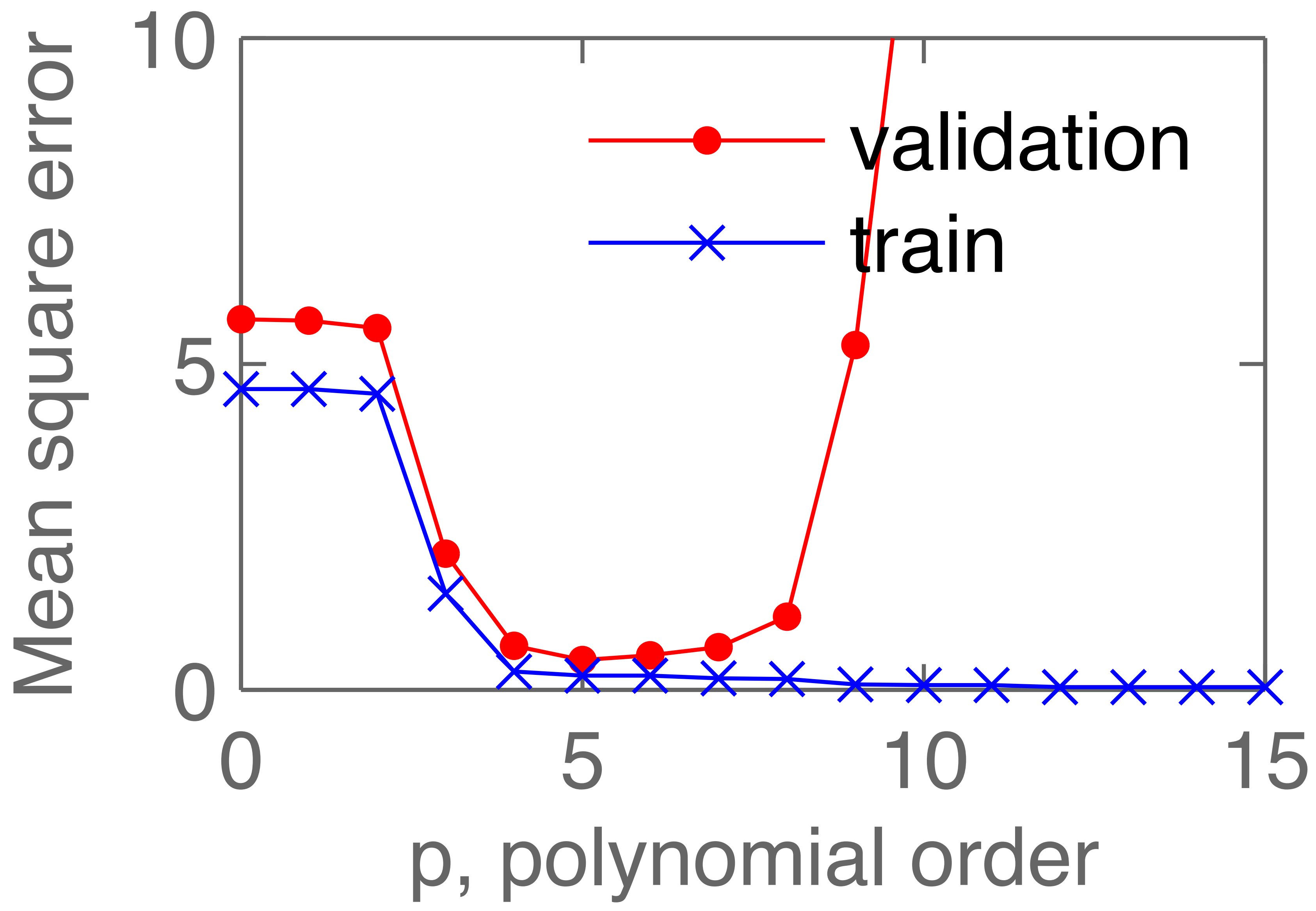
First, a simple practical procedure

Validation set



```
X = [x.^0, x.^1, x.^2, ..., x.^p];
```

Learning curves



Using validation sets

Can overfit the validation set:

— Always hold out a true “test set”
for final evaluation

Validation set is not used (much) for fitting:

— make training set as large as you can
— consider multiple folds/rounds

Early stopping

Validation set used to estimate
test error for fitted models.

Tracking a validation set is also used during fitting of a single model:

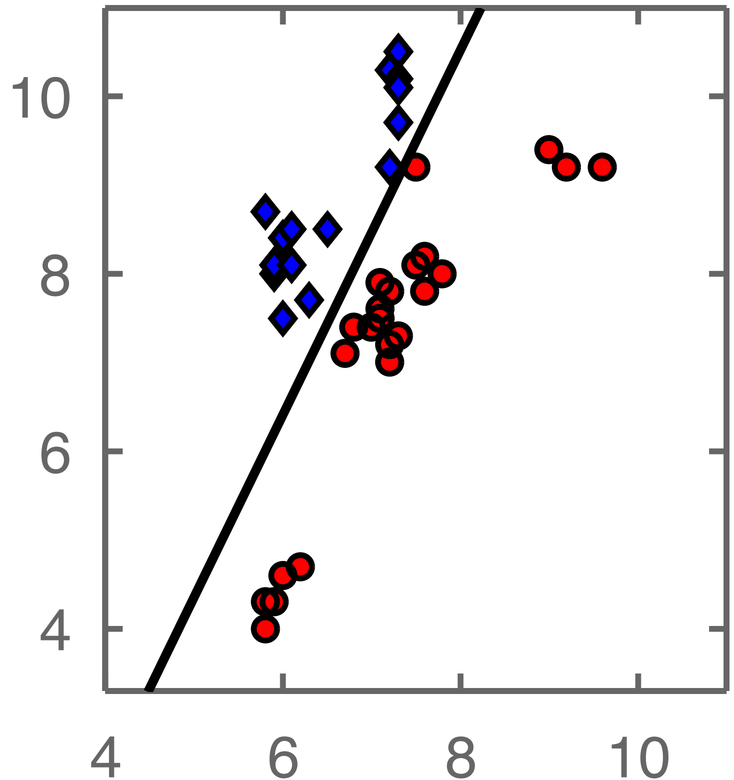
— *Ad hoc*, that is, a hack
— depends on optimizer
— can work annoyingly well
— sometimes fast compared to alternatives

Regularization

We don't need the validation set to know some fits are silly. Regularization discourages solutions we don't like

Shrinkage or **weight decay**:
make parameters closer to zero.

Early stopping can be similar:
stop before weights grow big



Regularization

Maximize a penalized objective:

Penalize $\mathcal{F}$ = negative training error, or training probability

$\mathcal{F}(\mathbf{w}; X, \mathbf{y}) - \lambda \sum_i |w_i|$, $\quad L_1$, (e.g., 'lasso')

$\mathcal{F}(\mathbf{w}; X, \mathbf{y}) - \lambda \sum_i w_i^2$, $\quad L_2$, (e.g., 'ridge regression')

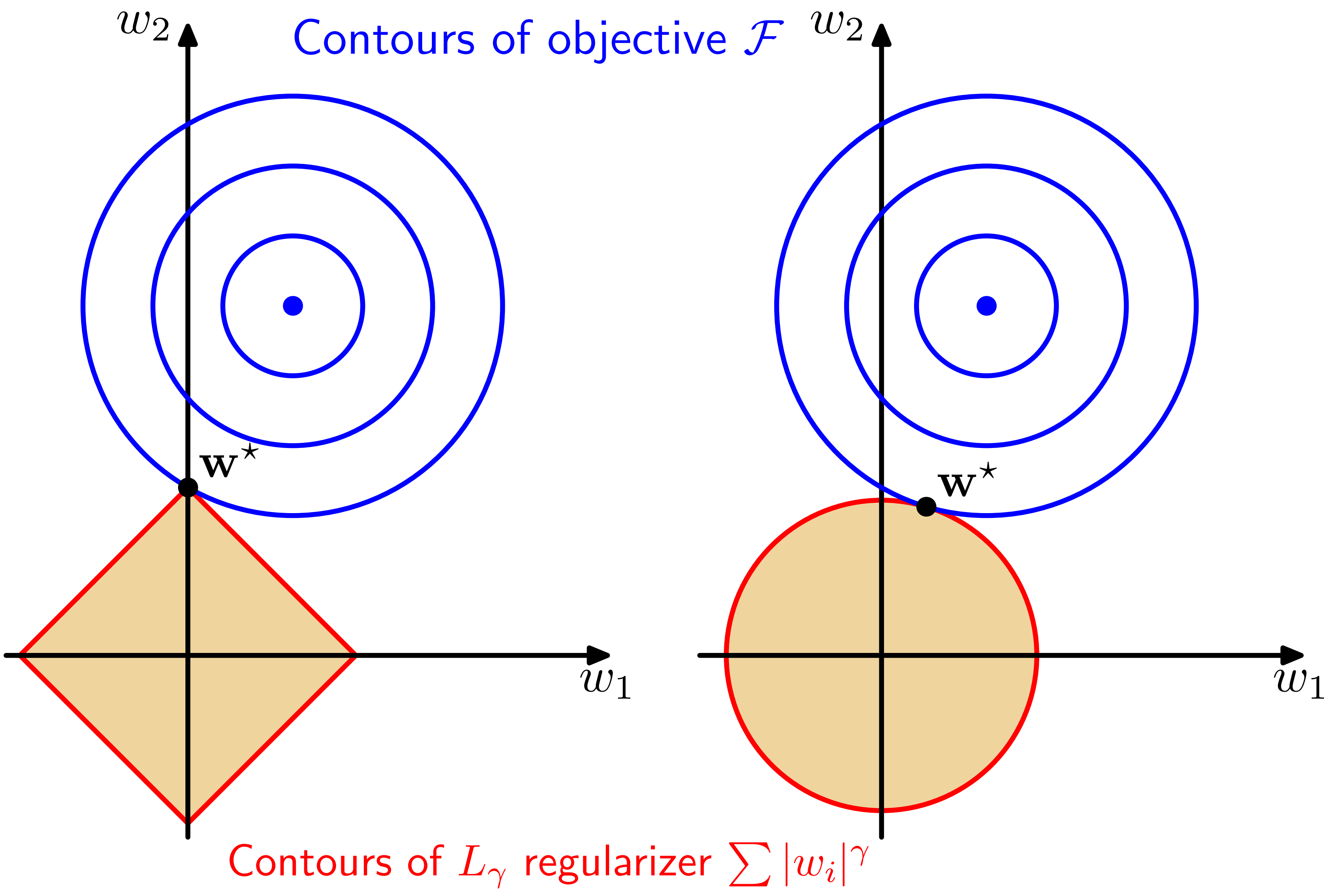
Equivalently, constrain parameters:

Maximize $\mathcal{F}(\mathbf{w}; X, \mathbf{y})$ such that $\sum_i |w_i| = t$ or $\sum_i w_i^2 = t$

One–one correspondence between t and λ

λ is a Lagrange multiplier for the constrained problem.

Different regularizers


w_2
Contours of objective $\mathcal{F}$
w_2
$\mathbf{w}^\star$
$\mathbf{w}^\star$
w_1
w_1
Contours of L_γ regularizer $\sum |w_i|^\gamma$

Pattern Recognition and Machine Learning C. M. Bishop (2006)

Regularization constants

How do we pick λ or t?

— based on validation

— based on bounds on generalization error

— based on "empirical Bayes"

Or given a well-defined procedure, reinterpret λ and do something else.
Fully Bayesian methods "integrate out" λ.

Learning theory

Compute bounds on generalization error

Based on held-out validation set:

— easy to construct

— easy to interpret theoretically

— harder to extend to K-fold validation

Based on training set performance:

— philosophically pleasing and interesting

— bounds can be very loose

The Bayesian view

Why do we use training data?
Because there are things we don't know.
There are also things we do know (regularization)

Under certain assumptions, probabilities and probability calculus (e.g., Bayes rule) are *the* way to combine prior beliefs and observations.

Bayesian procedure

Specify possible worlds by putting a *prior* distribution over parameters of a model $p(\mathbf{w})$

Given data $\mathcal{D} = \{X, \mathbf{y}\}$ we have a *posterior*:

$$p(\mathbf{w}\,|\,\mathcal{D}) \;=\; \frac{p(\mathcal{D}\,|\,\mathbf{w})\,P(\mathbf{w})}{P(\mathcal{D})} \;\propto\; p(\mathcal{D}\,|\,\mathbf{w})\,P(\mathbf{w})$$

Minimize expected loss:

$$\hat{y} = \operatorname*{arg\,min}_{y} \sum_{y_*} L(y; y_*)\, p(y_*\,|\,\mathbf{x}_*, \mathcal{D}), \qquad \text{cf “Bayes classifier”}$$

$$= \operatorname*{arg\,min}_{y} \sum_{y_*} L(y; y_*) \int p(y_*\,|\,\mathbf{w})\, p(\mathbf{w}\,|\,\mathcal{D})\; \mathrm{d}\mathbf{w}$$

Bayesian connections

Approximating the integral can involve maximizing:

$$\log p(\mathbf{w}\,|\,\mathcal{D}) = \log p(\mathcal{D}\,|\,\mathbf{w}) + \log p(\mathbf{w}) + \text{const.}$$

where $\log p(\mathbf{w}) = -\lambda \sum_i w_i^2$, if $p(\mathbf{w}) = \mathcal{N}(0, 2\lambda)$

$\Rightarrow$ Optimize L2-regularized log-likelihood.

However, *not* tied to $-$ve log-prob loss.
The loss we care about is optimized later.

Bayesian advantages

In principle the solution to any problem

(e.g., outliers, data corruption, missing inputs)

is solved by modelling it and probability calculus.

Don't know λ?

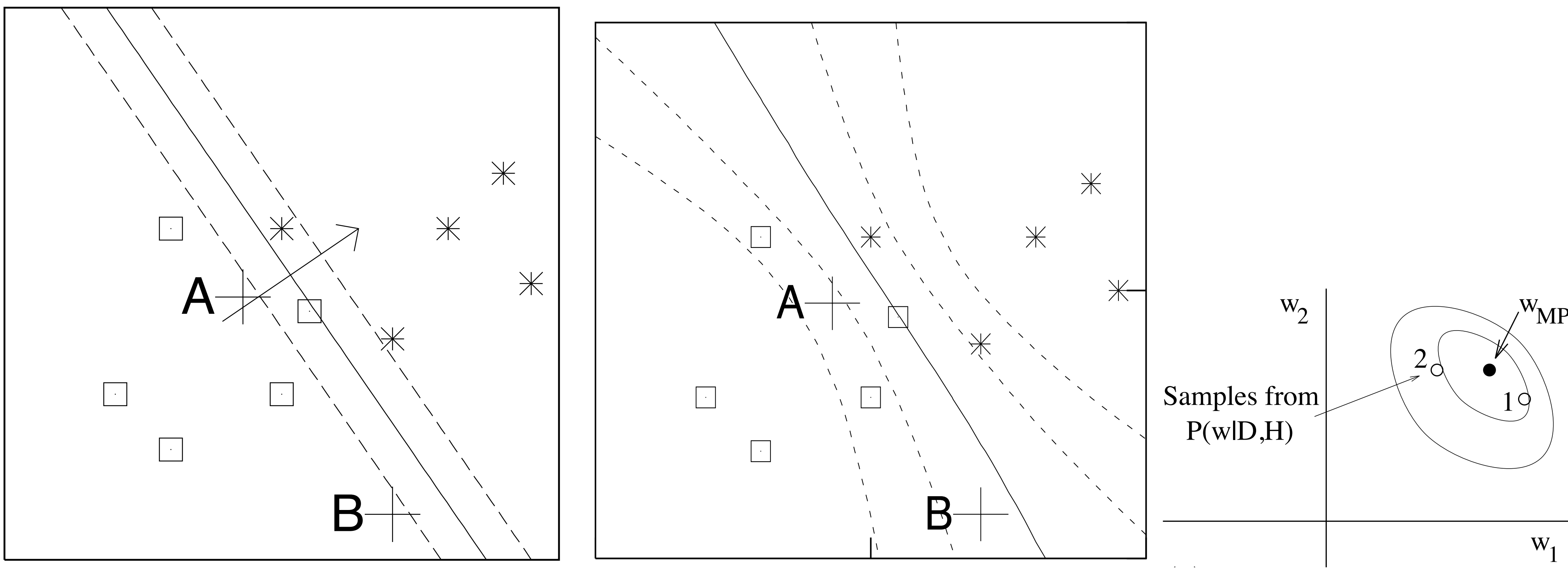
$$\hat{y} = \arg\min_y \sum_{y_*} L(y; y_*) \int p(y_* \,|\, \mathbf{w})\, p(\mathbf{w} \,|\, \mathcal{D})\; \mathrm{d}\mathbf{w}$$

$$= \arg\min_y \sum_{y_*} L(y; y_*) \int p(y_* \,|\, \mathbf{w}) \int p(\mathbf{w} \,|\, \mathcal{D}, \lambda)\, p(\lambda \,|\, \mathcal{D})\; \mathrm{d}\lambda\; \mathrm{d}\mathbf{w}$$

Might maximize $p(\lambda \,|\, \mathcal{D})$ instead of integrating over λ.

"Empirical Bayes" / Type-II maximum likelihood maximizes $p(\mathcal{D} \,|\, \lambda)$

Bayesian logistic regression



Information theory, inference, and learning algorithms, David J. C. MacKay (2003)
Logistic regression is referred to as a single-neuron classifier

Bayesian computations

Given a model, the Bayesian solution is a purely mechanical manipulation of probabilities.
(given a little practice)

The required integrals can be formidable.

Huge research effort in **approximate inference**:
— variational objectives
— Monte Carlo sampling

Bayesian models

We can fit arbitrary rules for $\mathbf{x} \rightarrow y$
and estimate future losses.

To take posterior beliefs seriously,
need to start with sensible prior beliefs.

Need flexible distributions over models and priors that are “sensible enough”.

Graphical models and non-parametric Bayesian methods offer useful machinery.

Supervised learning

Overview:

— identify task and loss function

— build flexible model of mapping $\mathbf{x} \rightarrow y$

— use data to set all the free parameters

Some form of regularization is crucial

Large, complicated models are useful in applications.

— penalize empirical score

— Bayesian methods

Pre-processing

Back to thinking about applications

Given a dataset of example inputs and outputs, will what we have seen so far work?

What's $\mathbf{x}$?

Rescaling and centering

Centering:

— Might have all features at 1000 ± 10.

— Hard to predict ball park of bias.

— Subtract mean from all input features.

Rescaling:

— I might measure heights in cm or m.

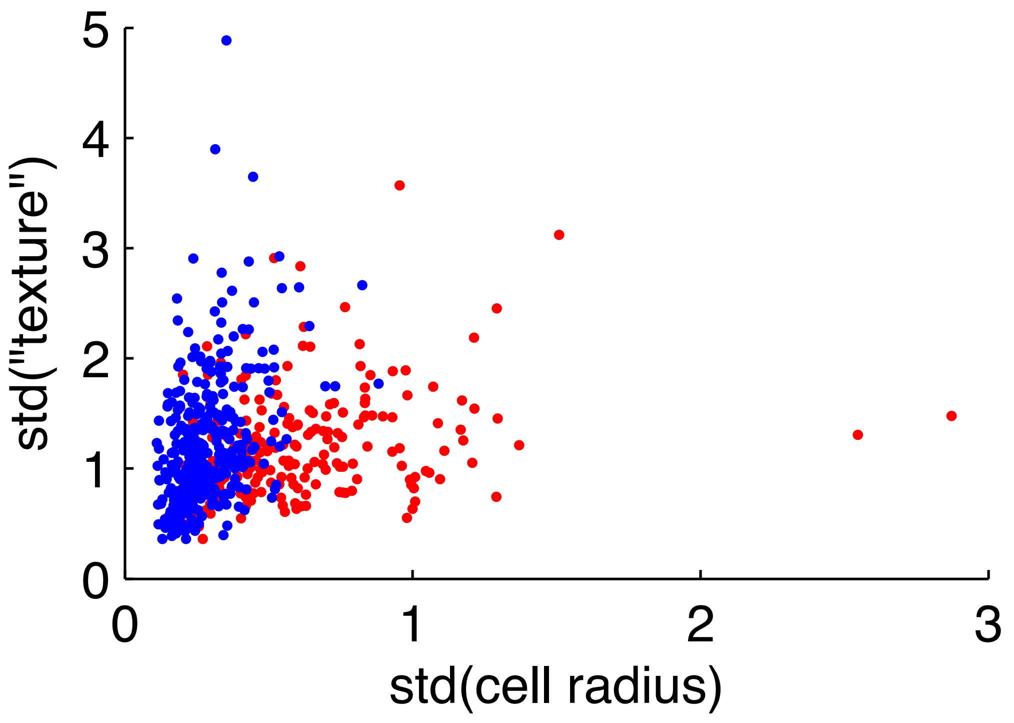
— Rescale inputs to have unit variance

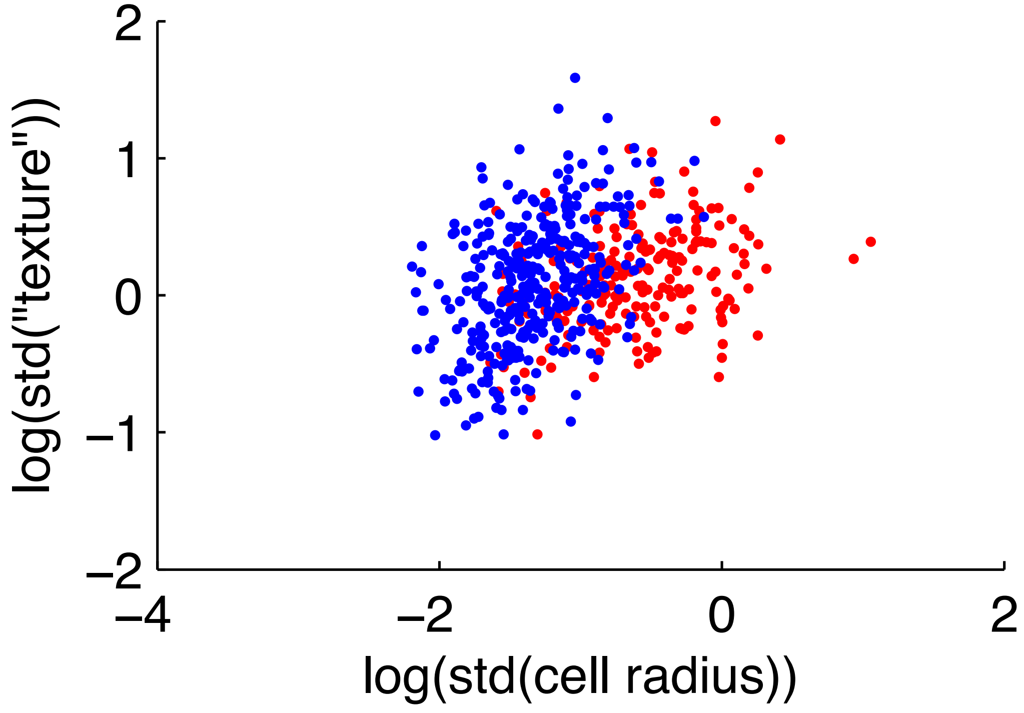
. . . or interquartile range.

Similarly center and scale real-valued outputs

Care at test time: apply same scaling to inputs and reverse scaling to predictions.

Log-transform +ve inputs





(Wisconsin breast cancer data from the UCI repository)

Positive quantities are often highly skewed

The log-domain is often much more natural

Encoding attributes

Categorical variables:

- A study has three individuals.
- Three different colours
- Possible encoding: `100, 010, 001`

Ordinal variables:

- Movie rating, 1–3 stars
- Tissue anomaly rating, expert scores in 1–3
- Possible encoding: `00, 10, 11`

Basis functions features

Toy examples used polynomials: $x, x^2, x^3 \ldots$
Often a bad choice

Polynomials of sparse binary features make sense:

$$x_1x_2,\ x_1x_3,\ \cdots,\ x_1x_2x_3$$

Other options:

— Radial basis functions: $\exp(-|\mathbf{x}-\boldsymbol{\mu}|^2/h^2)$
— Sigmoids: $1/(1+\exp(-\mathbf{v}^\top\mathbf{x}))$
— Fourier, wavelets, . . .

Feature engineering

The difference in performance in an application can depend more on the original features than the algorithm.

Working out clever ways of making features from complex objects like images can be worthwhile, is hard, and isn't always respected. . .

The SIFT story

Taken from: `http://www.lecun.com/ex/pamphlets/publishing-models.html`

Many of us have horror stories about how some of our best papers have been rejected by conferences. Perhaps the best case in point of the last few years is David Lowe's work on the SIFT method. After years of being rejected from conference[s] starting in 1997, the journal version published in 2004 went on to become the most highly cited paper in all of engineering sciences in 2005.

David Lowe relates the story:

I did submit papers on earlier versions of SIFT to both ICCV and CVPR (around 1997/98) and both were rejected. I then added more of a systems flavor and the paper was published at ICCV 1999, but just as a poster. By then I had decided the computer vision community was not interested, so I applied for a patent and intended to promote it just for industrial applications.

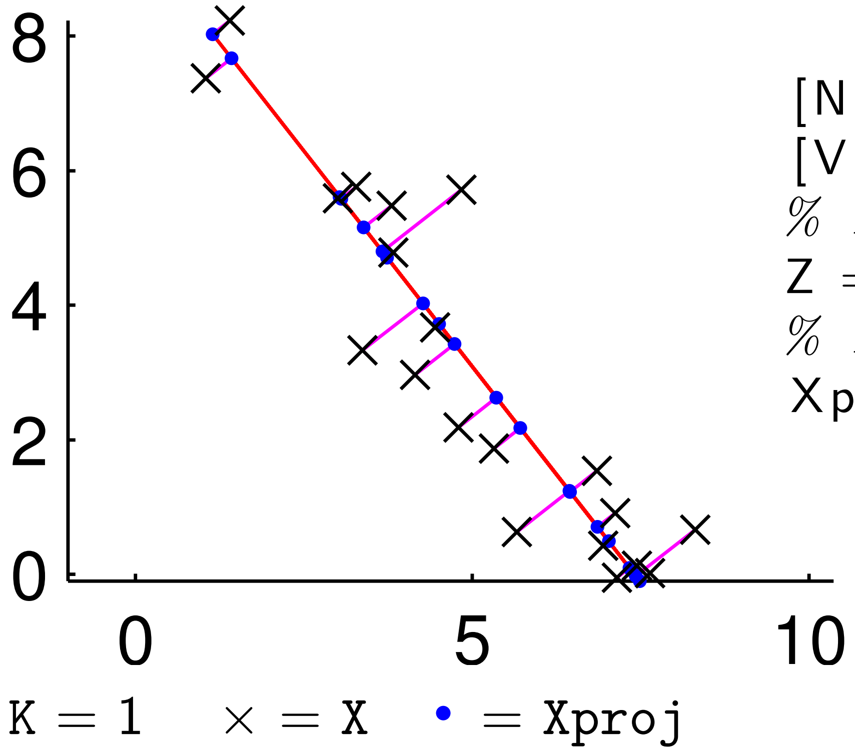
Creating extra data

A dirty trick: create more training ‘data’ by corrupting examples in the real training set.

Changes could respect invariances that would be difficult or burdensome to model directly.

PCA — Principal Components

Can reduce dimensionality of inputs with PCA



```
[N,D] = size(X);
[V,S] = svds(cov(X,1), K);
% Project X onto K principal eigenvectors
Z = (X - repmat(mean(X),N,1))*V; % NxK
% Project back into D-dims (for figure)
Xproj = Z*V' + repmat(mean(X),N,1); % NxD
```

Projects onto a linear subspace minimizing square error such that new features are uncorrelated.

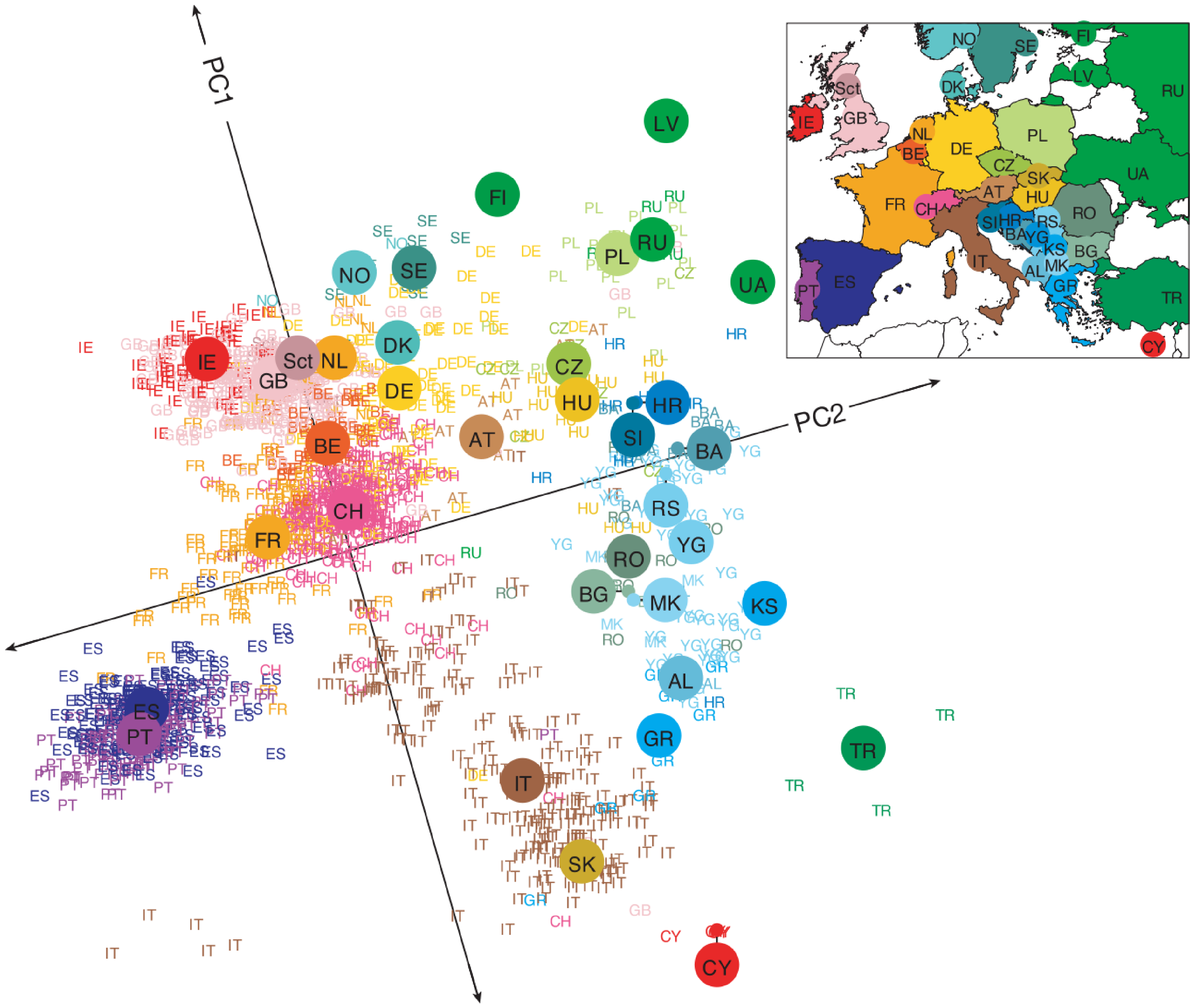
PCA applied to DNA data

Novembre et al. (2008) — doi:10.1038/nature07331

After selection of both individuals and features:

1,387 individuals
197,146 single nucleotide polymorphisms (SNPs)

Use PCA to reduce features to two(!) dimensions


PC1
PC2
LV
FI
RU
PL
UA
NO
SE
DK
IE
Sct
GB
NL
DE
CZ
HU
HR
SI
BA
AT
BE
CH
FR
RS
YG
RO
BG
MK
KS
AL
GR
TR
ES
PT
IT
SK
CY

Visualization

Visualization may suggest issues, transformations, features for a particular problem.
The insights could be results in themselves.

Artifacts that could cause problems:
    outliers, thresholding, kinks, . . .

How noisy is the data?
How relevant are features?

Did you remember to hold out your test set?

Things to try

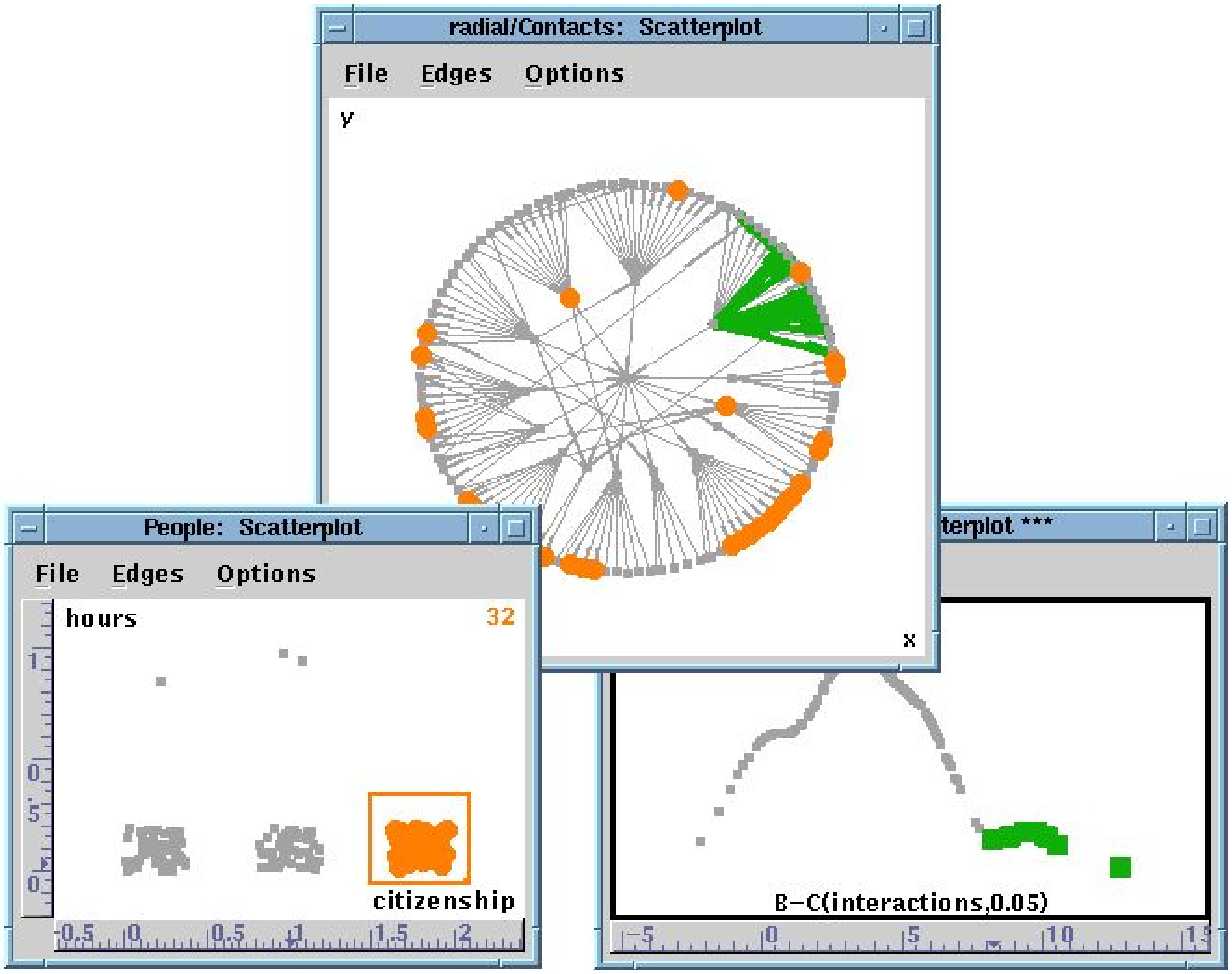
Histograms of individual features

Scatter plots of pairs of features

Scatter plots of PCA, and random projections

Colour plots using $\mathbf{y}$, or plot against it

Tours, linked plots, more



http://www.ggobi.org/

Swayne, Buja and Lang (2004)

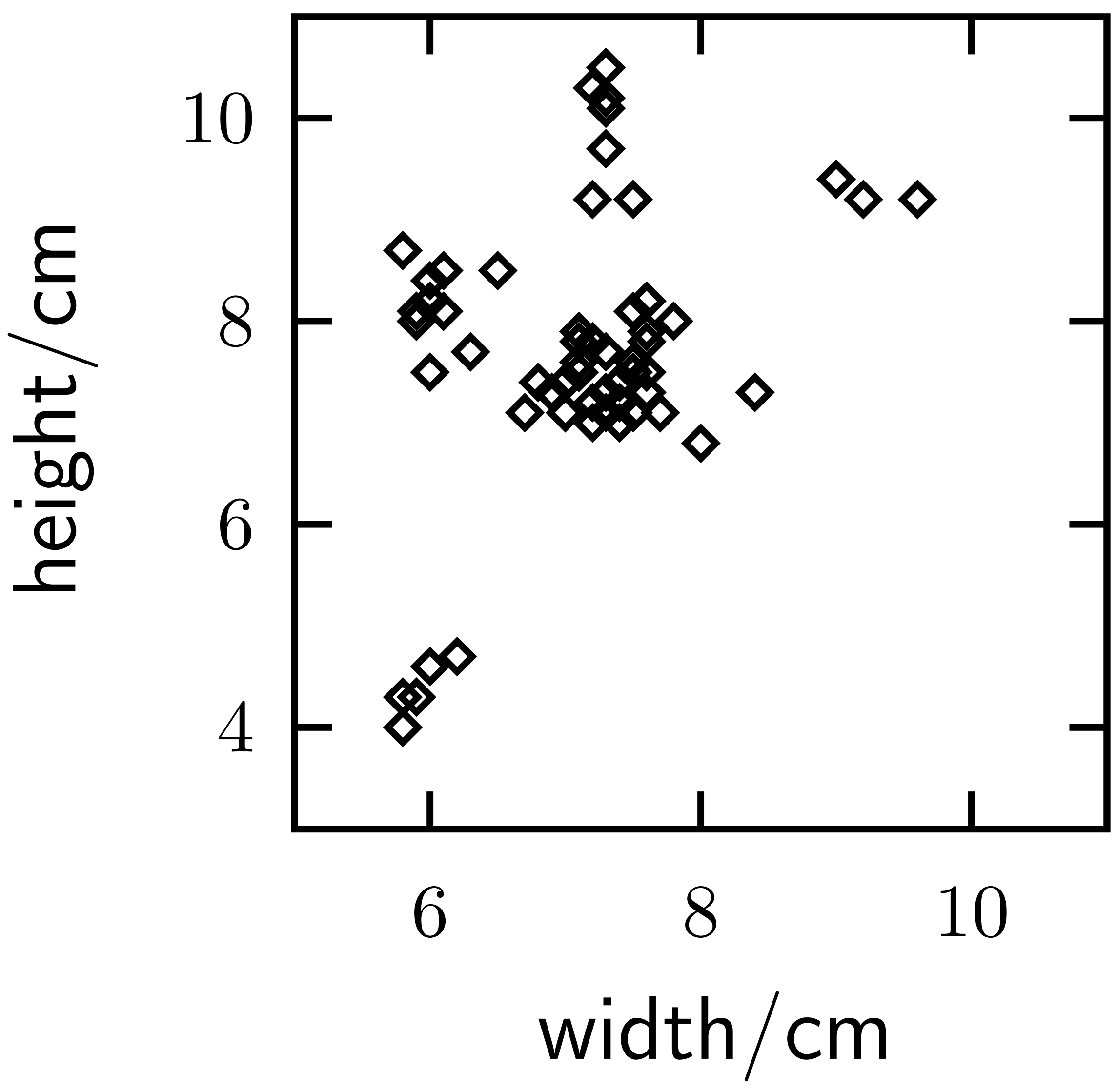
Unsupervised Learning

Learn structure in data automatically without a supervisor providing targets

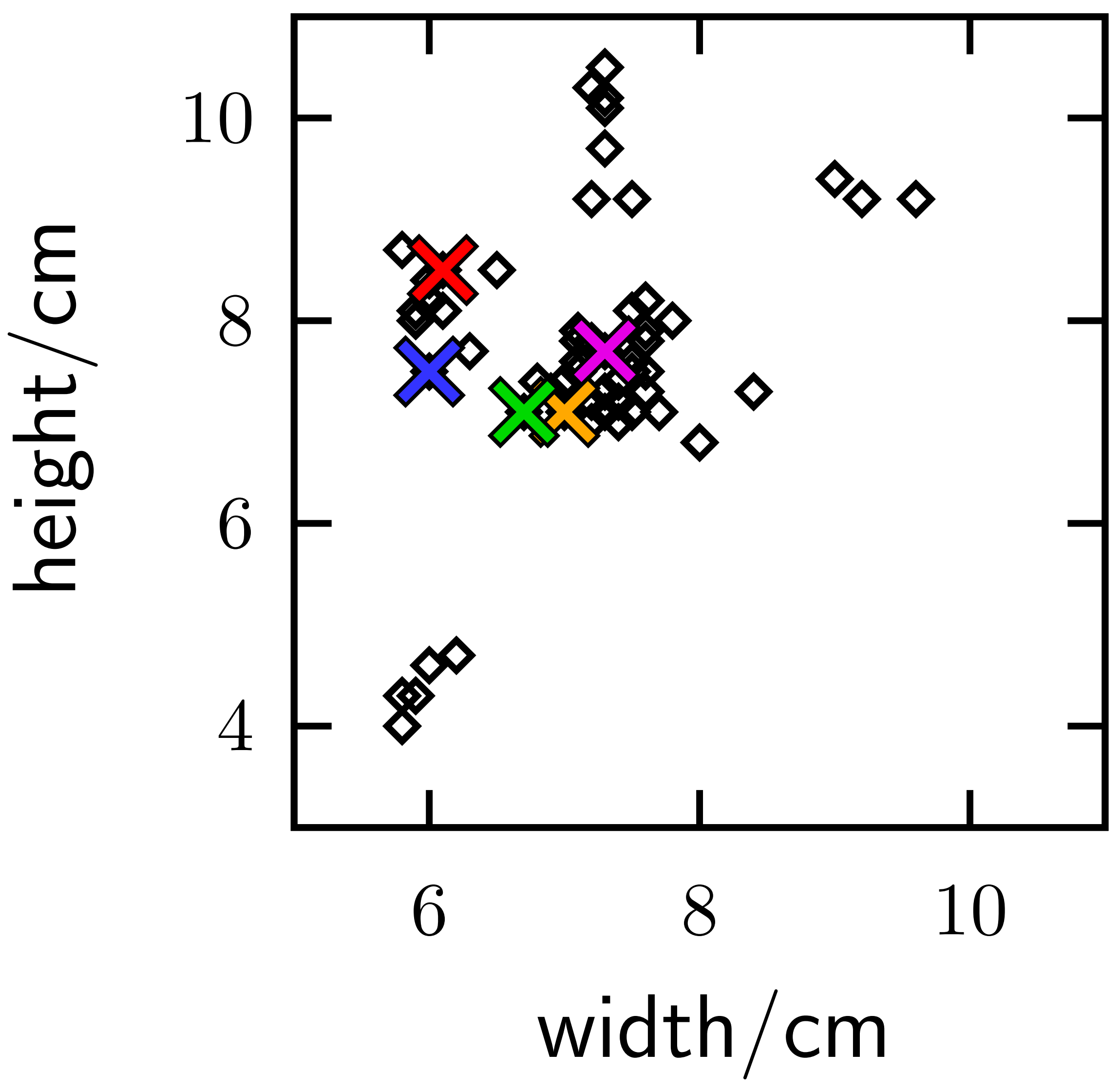
Much less well defined.

Human tweaking often important,
but we would always like computers
to do more by themselves.

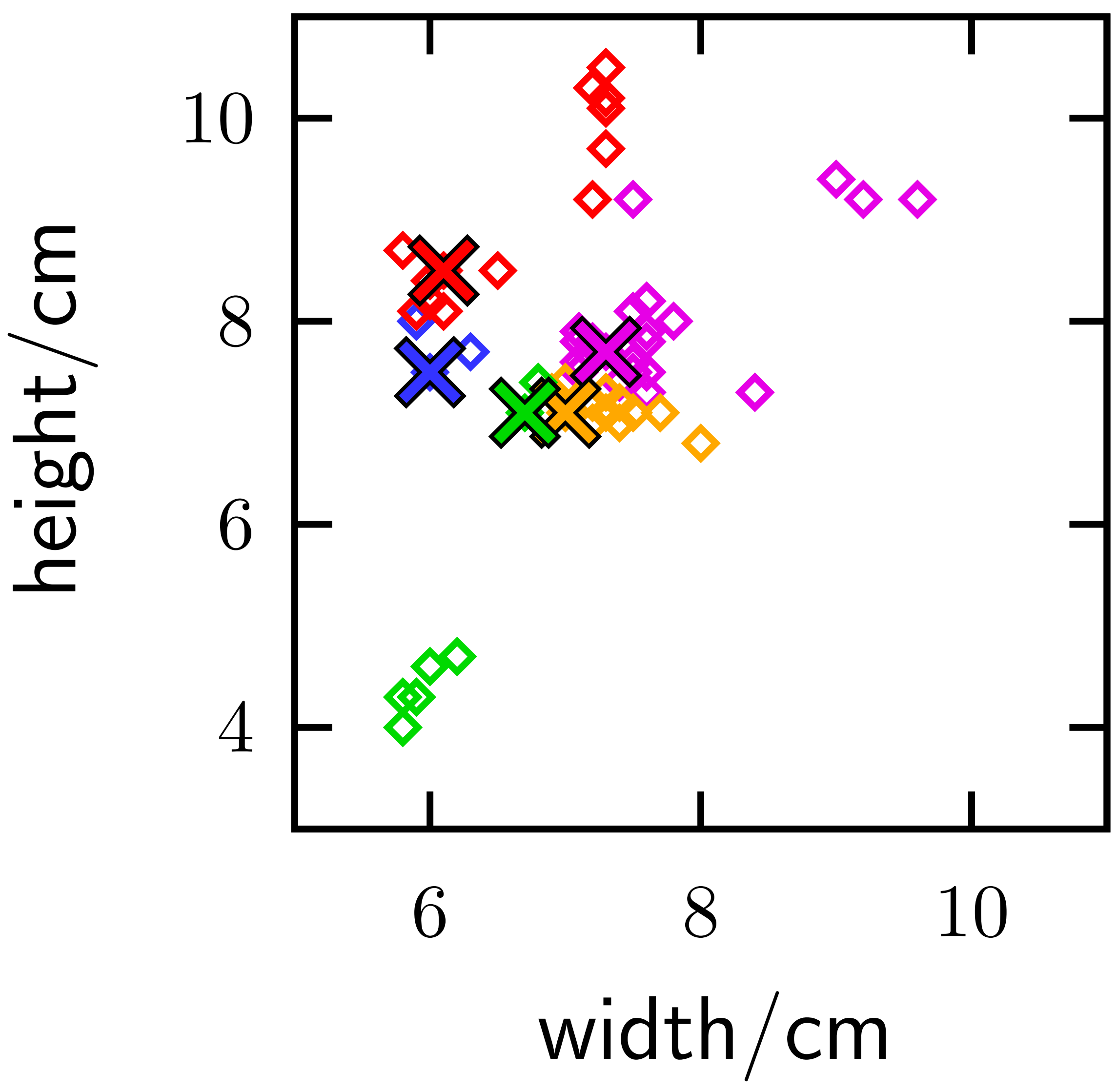
I see clusters



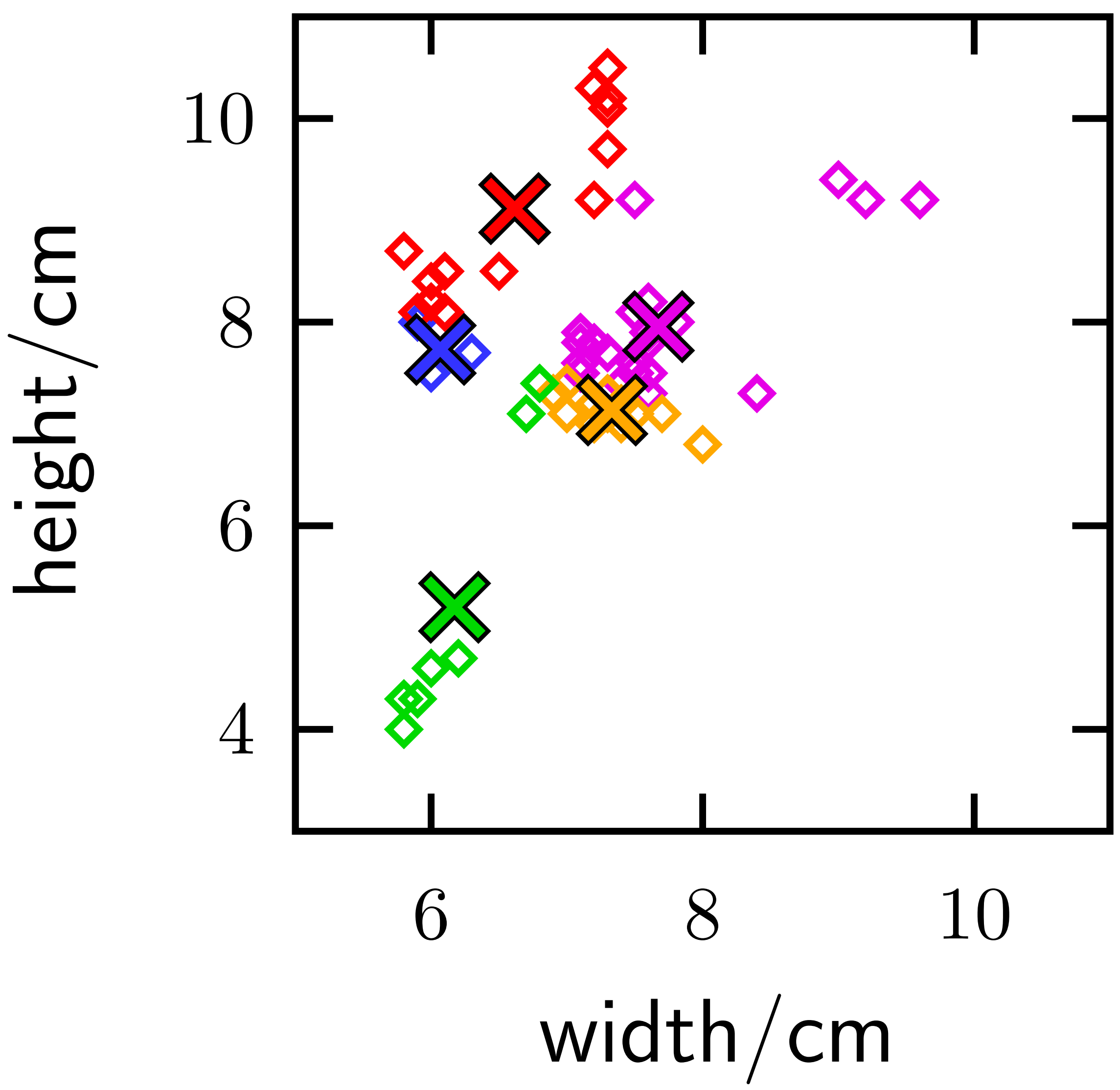
K-means demo



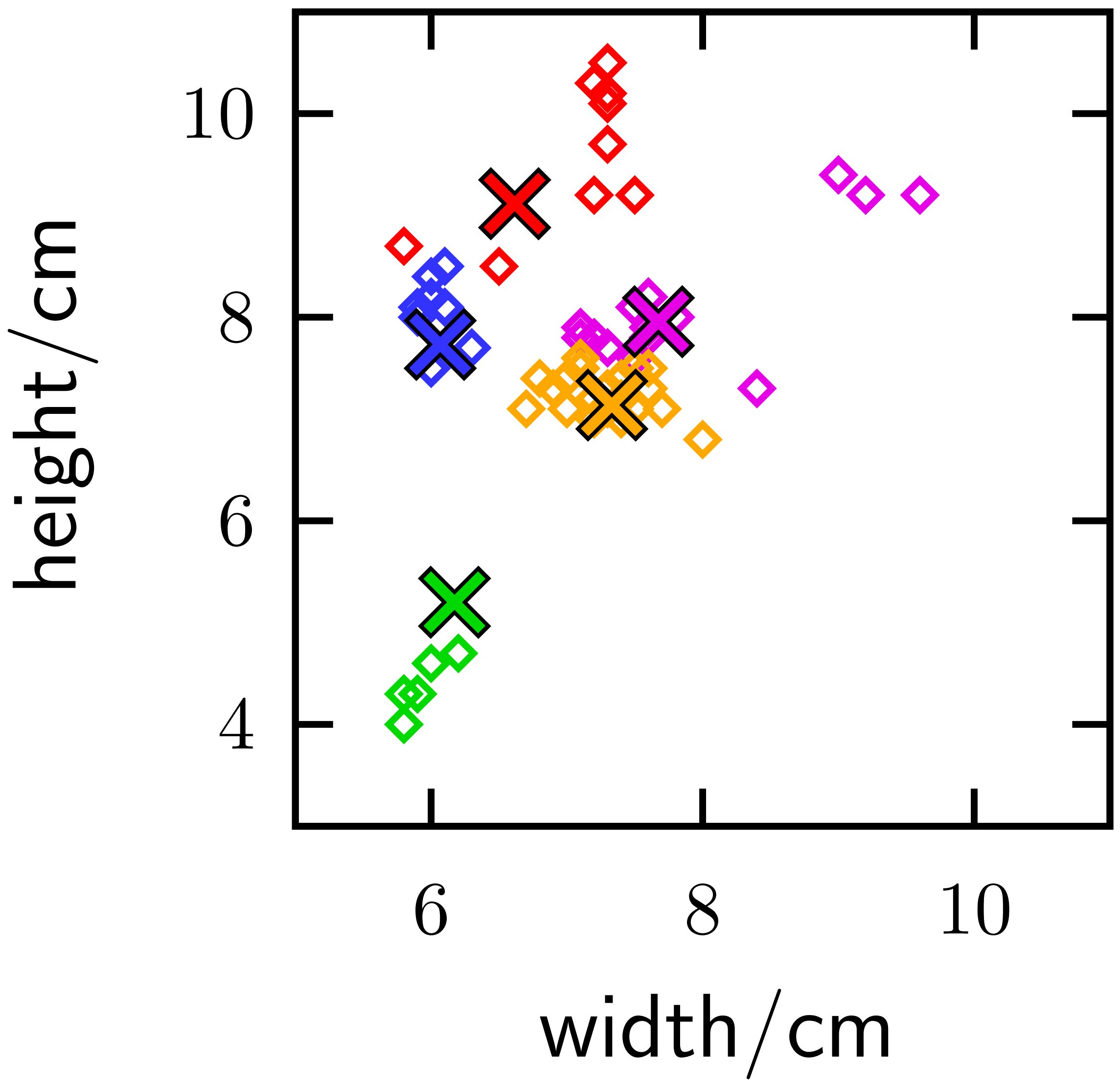
K-means demo



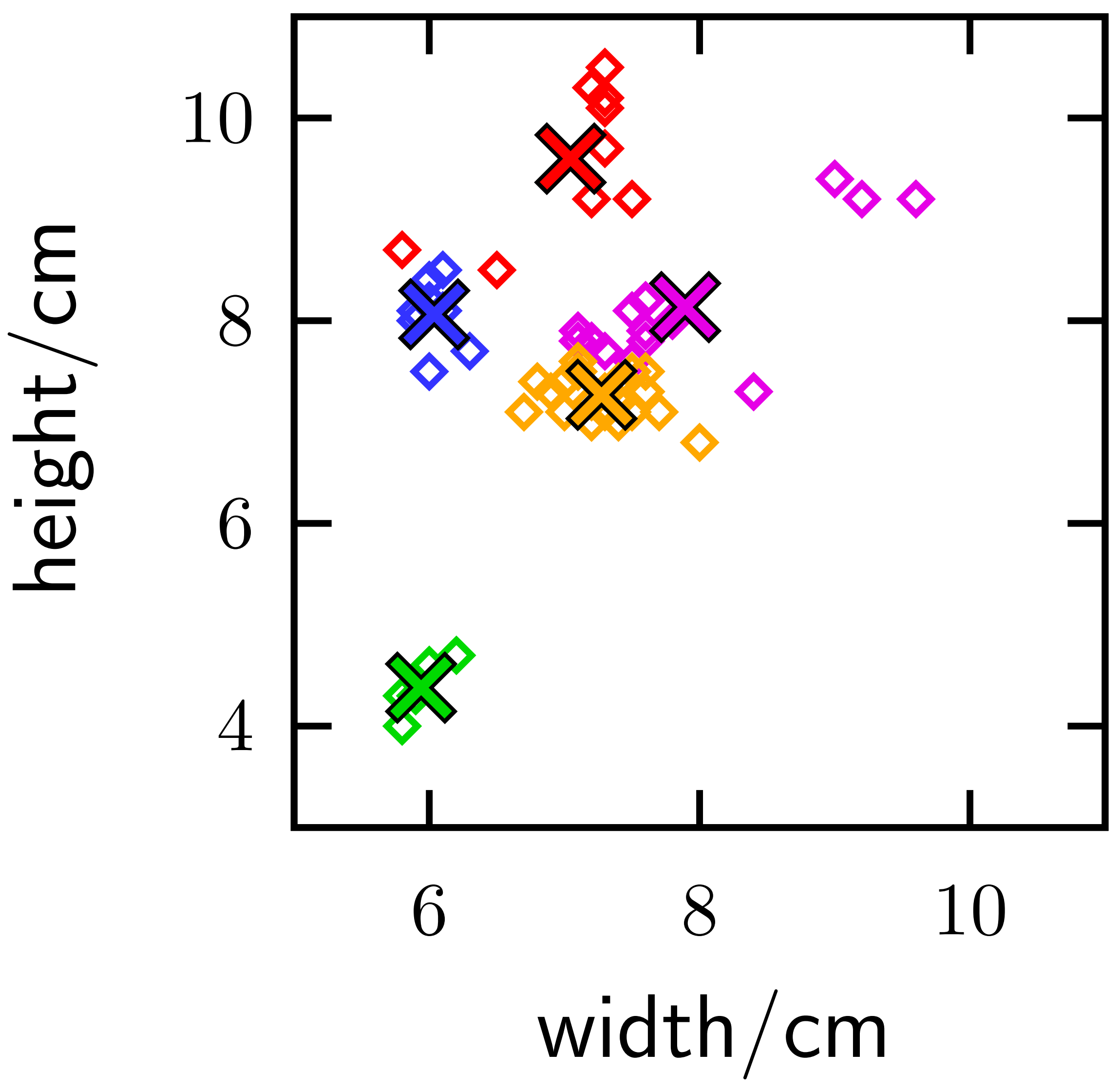
K-means demo



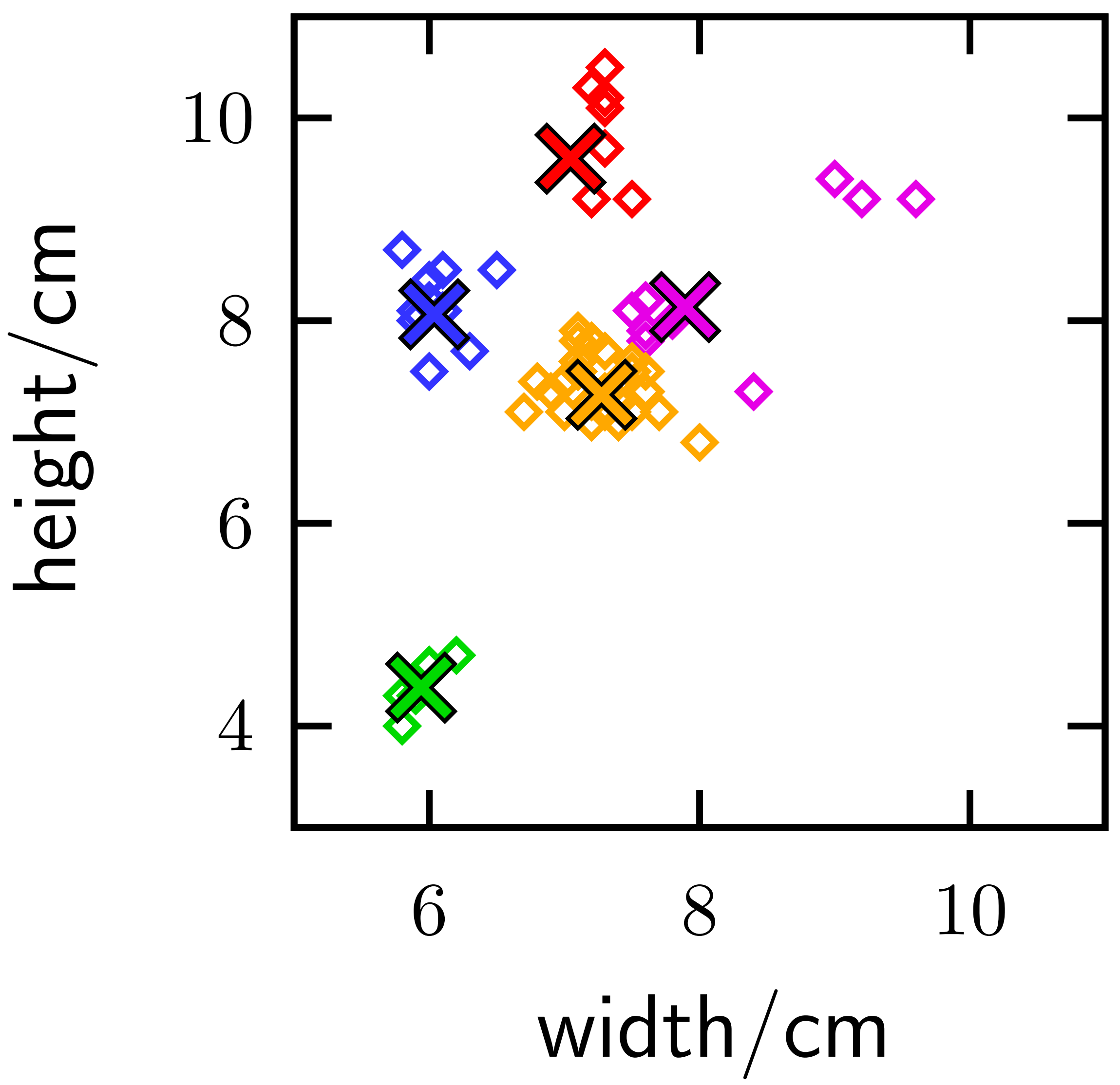
K-means demo



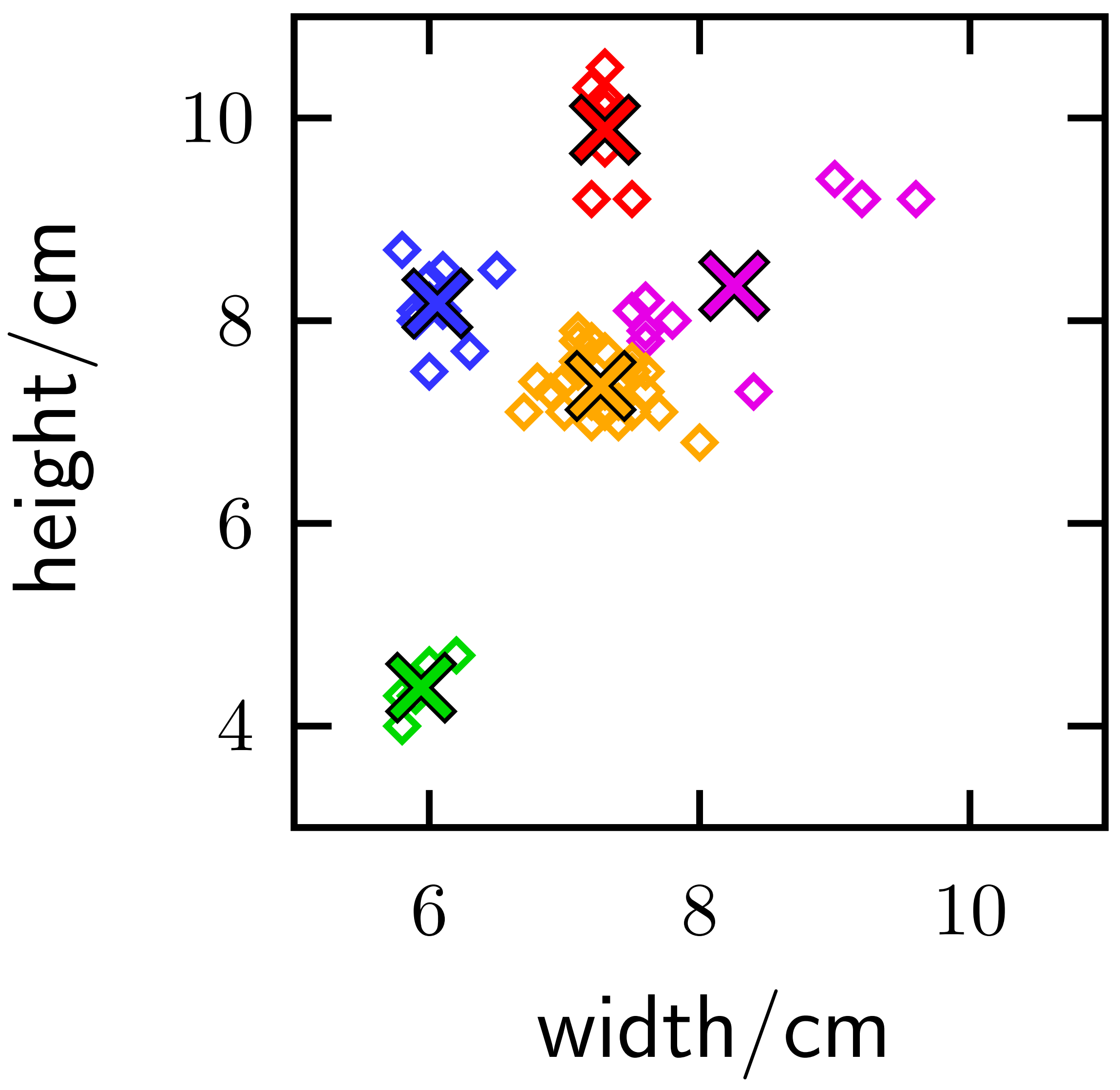
K-means demo



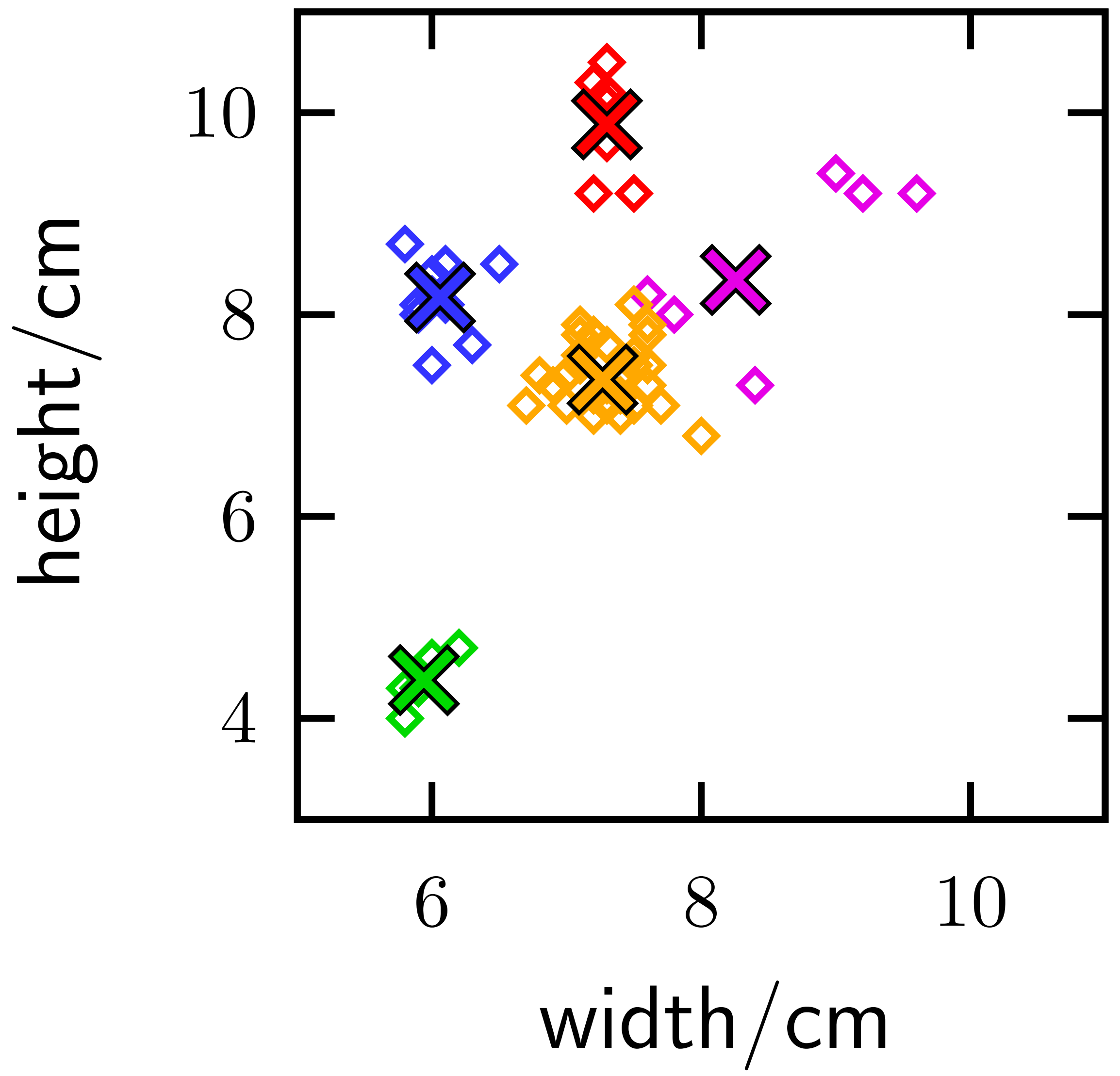
K-means demo



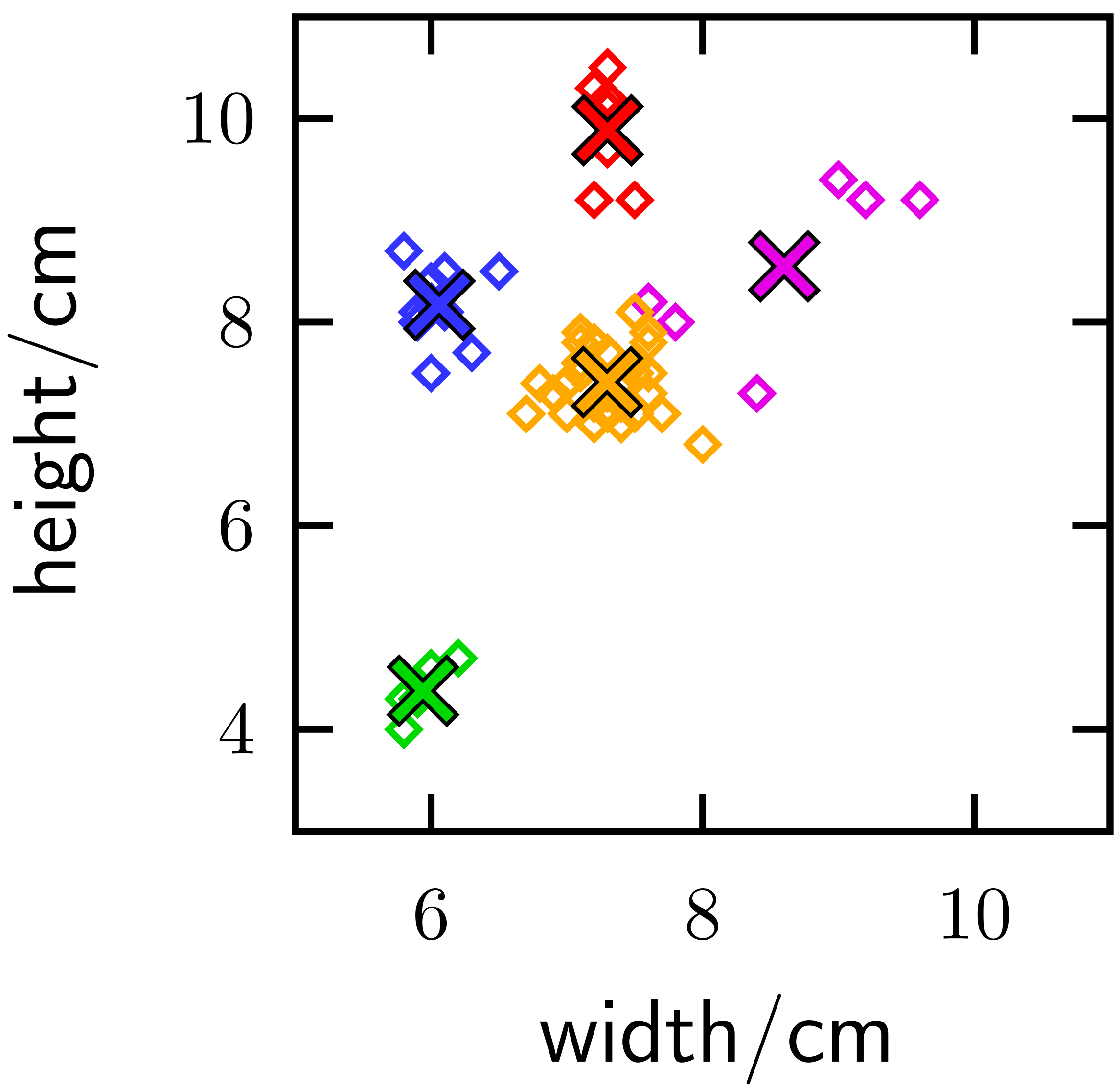
K-means demo



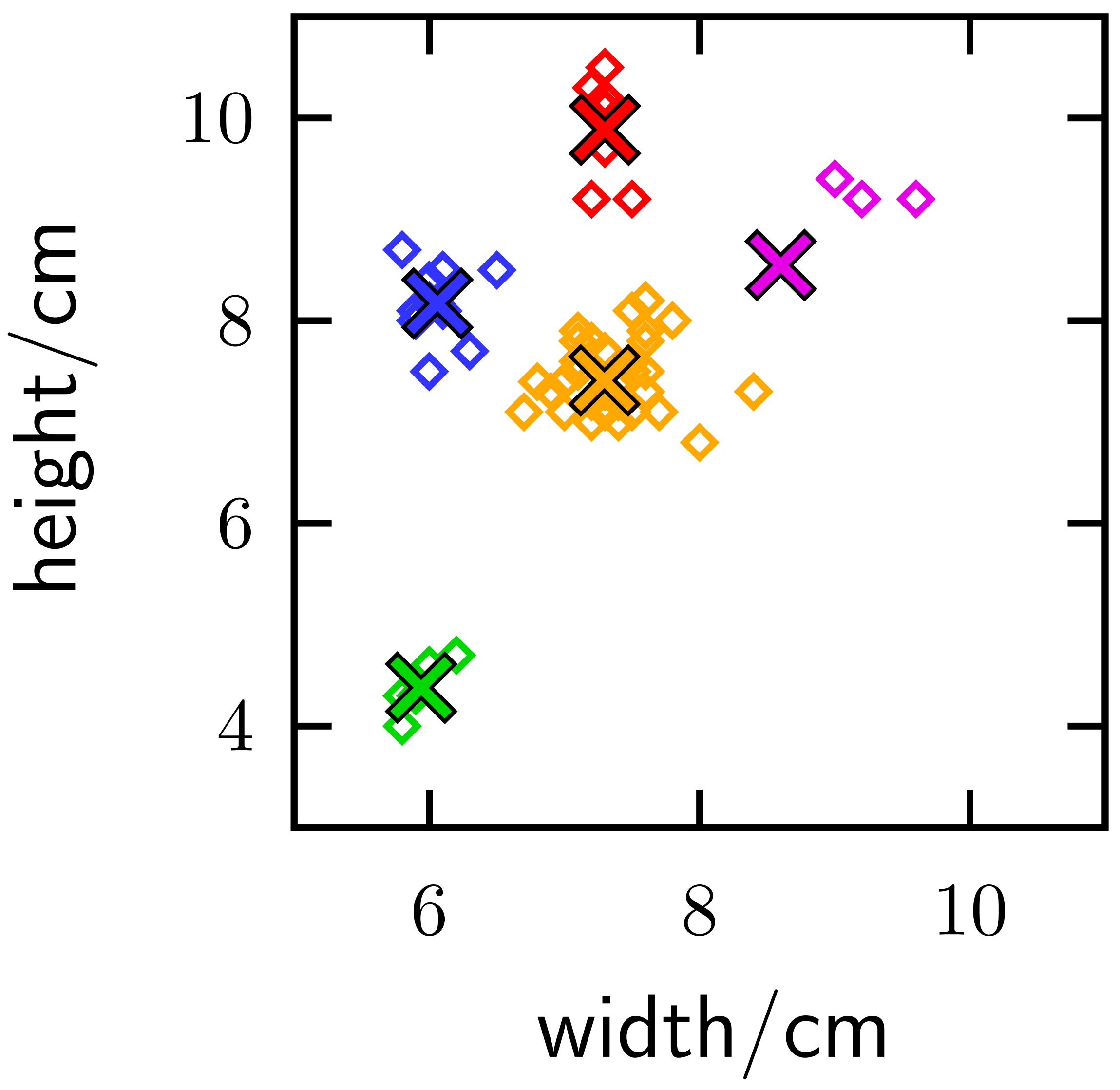
K-means demo



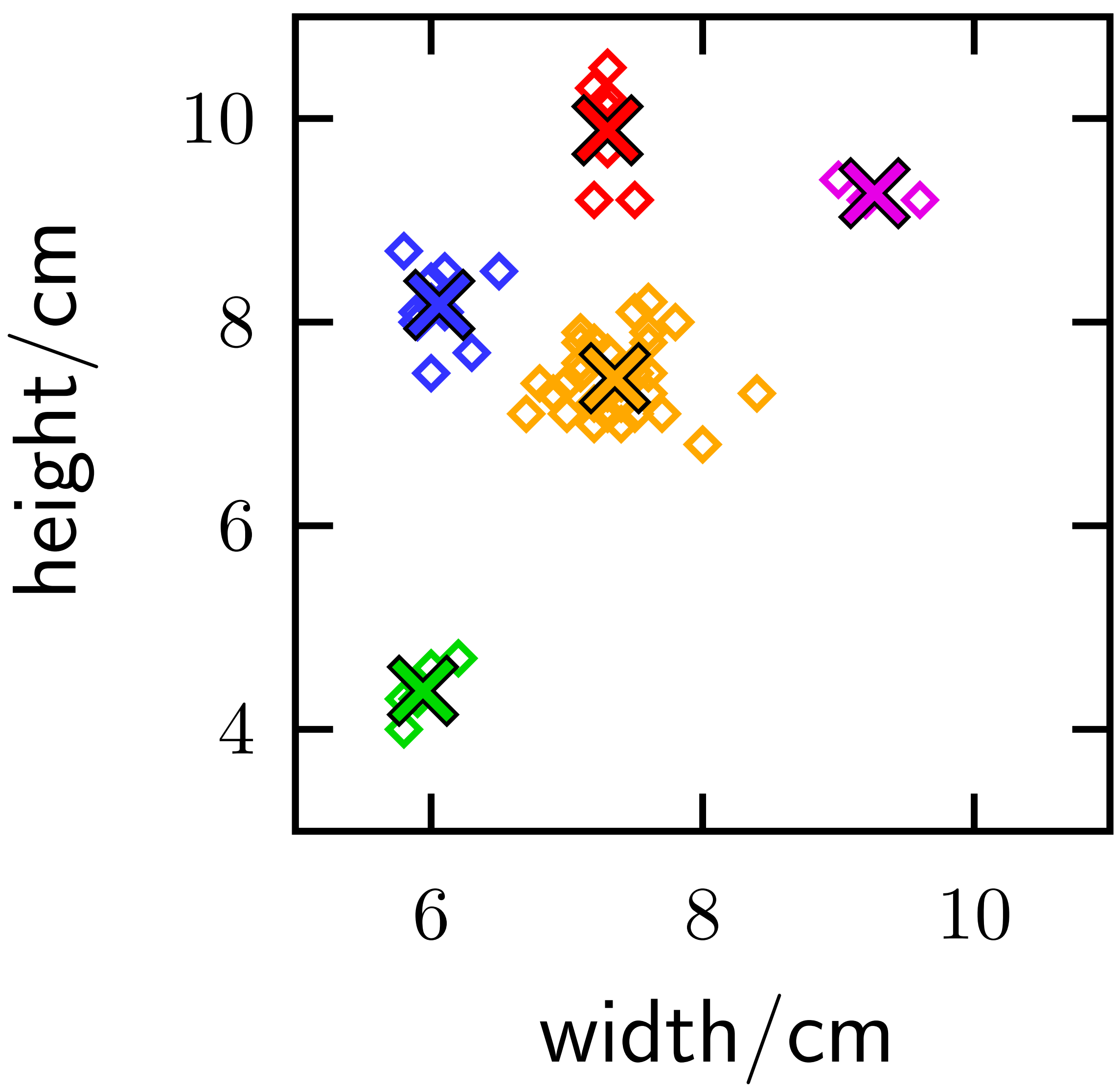
K-means demo



K-means demo

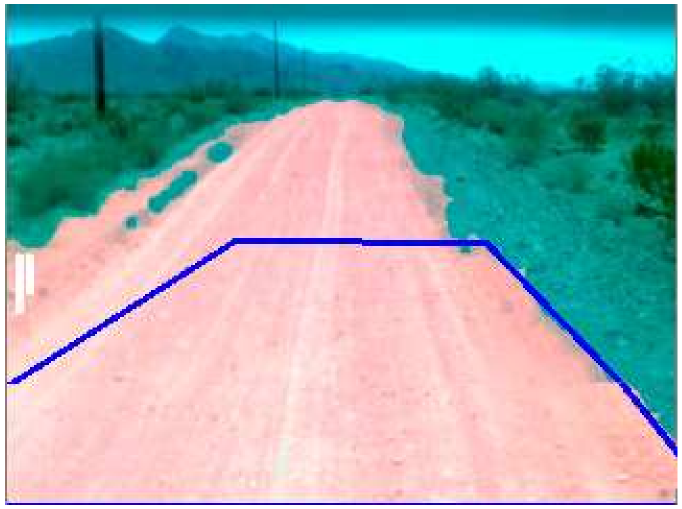


K-means demo



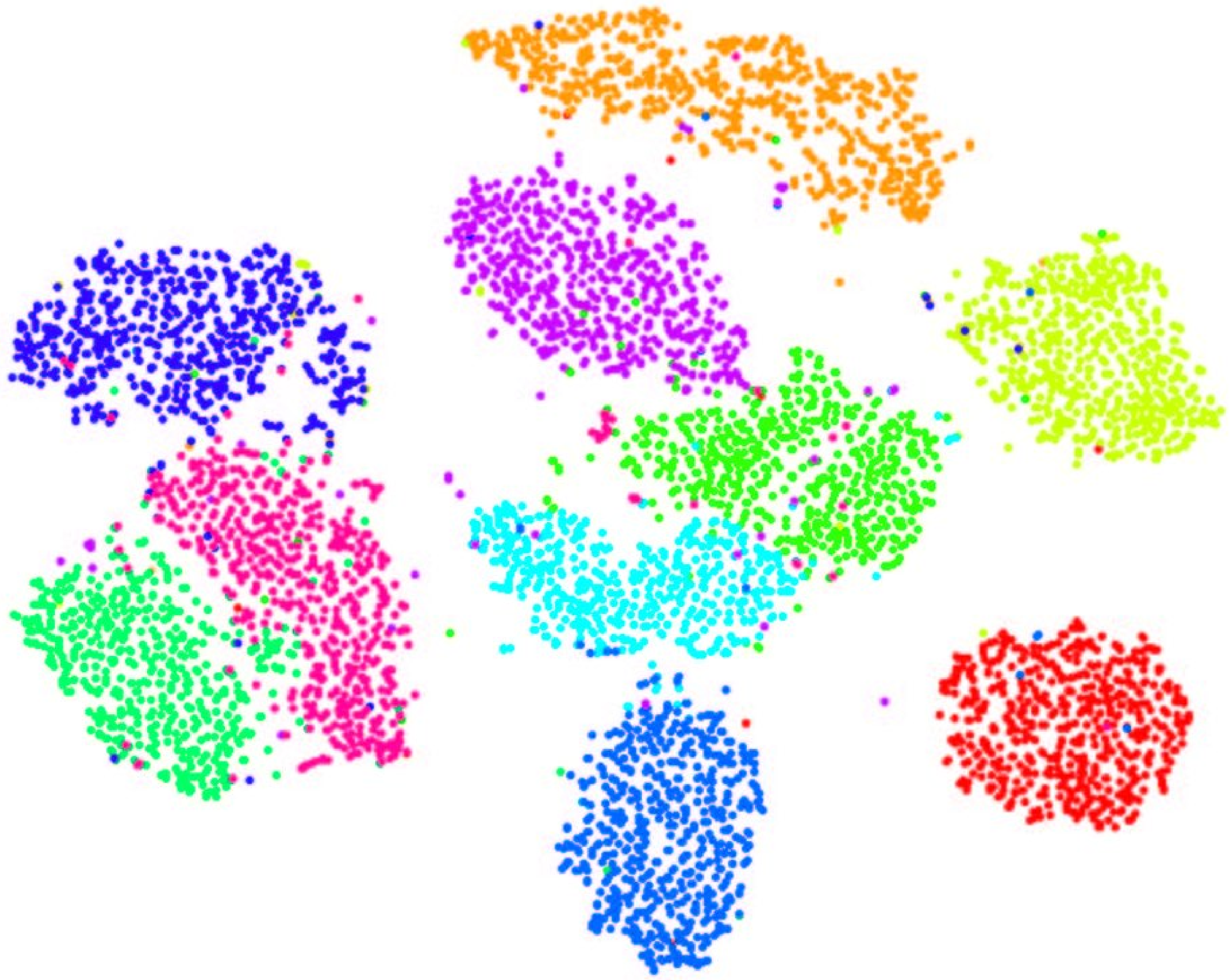
Mixtures of Gaussians

MoG is a probabilistic, fancier K-means



Dahlkamp et al. (2006)
`http://robots.stanford.edu/talks/stanley/`

Dimensionality reduction



t-SNE on MNIST digits, Van der Maaten and Hinton (2008)

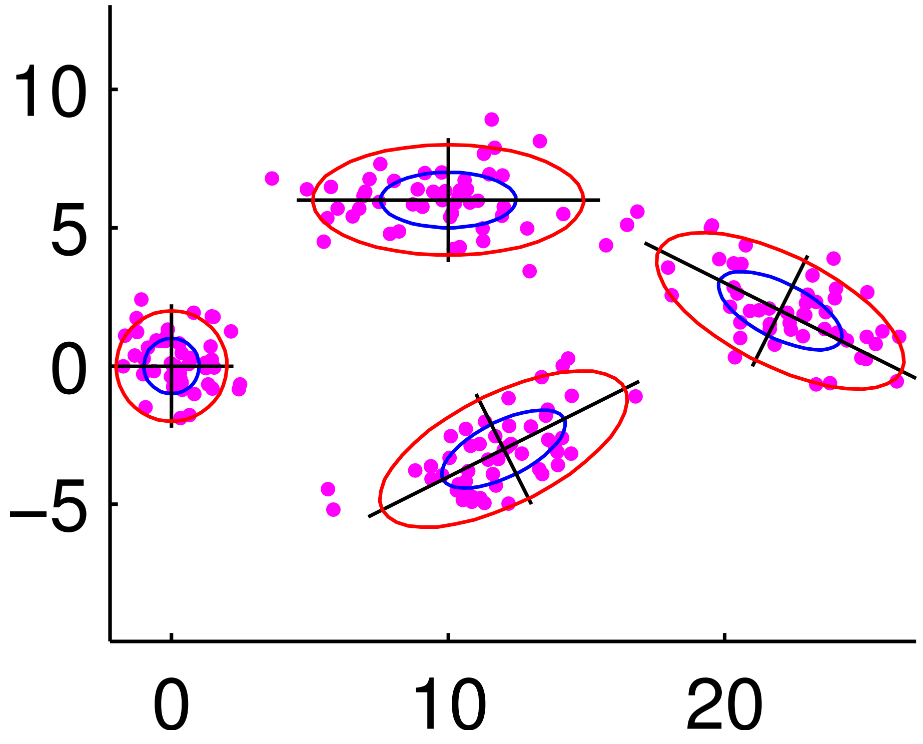
See also: `http://www.cs.toronto.edu/~hinton/turian.png`

Density estimation

One principle for unsupervised learning:

Model joint probability of all data

$$p(X \,|\, \mathbf{w}) \quad \text{or} \quad p(\mathbf{y}, X \,|\, \mathbf{w}) = p(\mathbf{y} \,|\, X, \mathbf{w}_d)\, p(X \,|\, \mathbf{w}_g)$$



Jointly modelling/regularizing $\mathbf{w}_d$, $\mathbf{w}_g$ uses data more fully

“Missing data”

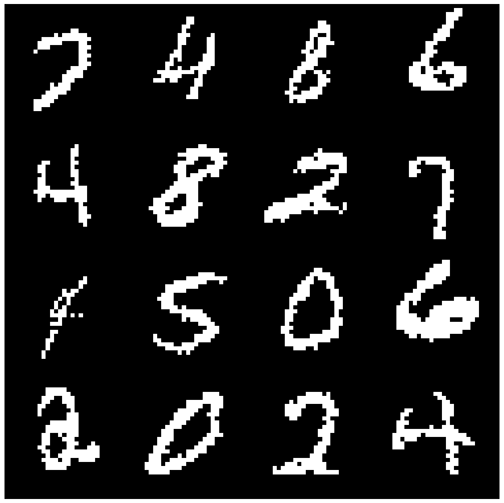
Having a fully generative model of $(\mathbf{x}, y)$ allows probabilistic/Bayesian inference to deal with any type of unknown

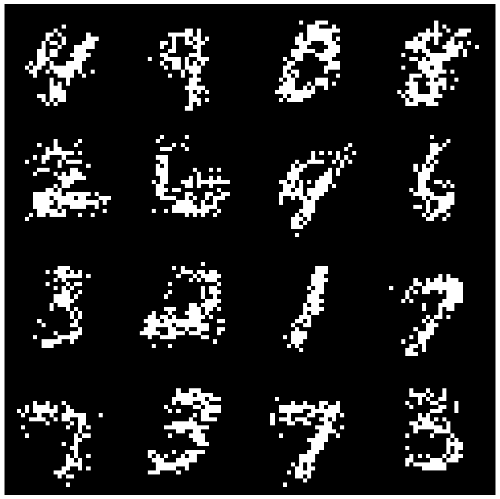
The varieties of “missing data”:

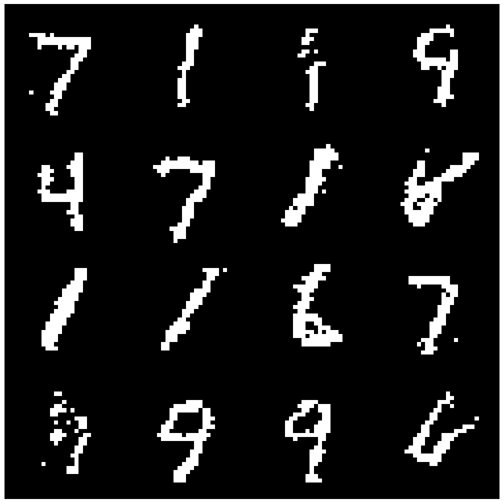
— features missing at random
— data censored for a reason
— truncated data
— unlabelled data

Underfitting

The main problem with unsupervised or part-supervised learning of structure is **underfitting**.


Data samples


MoB samples


RBM samples

More underfitting

Even non-parametric methods can underfit

Recommender systems:

— Amazon

— Netflix

Netflix prize dataset:

100,480,507 ratings; 480,189 users; 17,770 movies

Building richer models

Learn features and relationships between variables: neural networks, graphical models.

Model combination: boosting, cascading, mixtures of experts, products of experts, stacking models (deep learning)

Note: Bayesian model averaging, does inference for the model you have (simple or otherwise)

Summary

Machine learning uses data to refine flexible computer programs

We seek formalize learning objectives that allow for fitting the richness present in real data

There are many useful tools out there, but huge practical and theoretical challenges to enjoy

What I haven't covered

Lots more

Other settings: transductive learning, semi-supervised learning, online learning

Reinforcement learning, games

Time series, structured outputs, relational learning and many other specialist settings

A rant about least squares

Whenever a person eagerly inquires if my computer can solve a set of 300 equations in 300 unknowns. . . The odds are all too high that our inquiring friend. . . has collected a set of experimental data and is now attempting to fit a 300-parameter model to it—by Least Squares! The sooner this guy can be eased out of your office, the sooner you will be able to get back to useful work—but these chaps are persistent. . . you end up by getting angry and throwing the guy out of your office.

There is usually a reasonable procedure. Unfortunately, it is undramatic, laborious, and requires thought—which most of these charlatans avoid like the plague. They should merely fit a five-parameter model, then a six-parameter one. . . Somewhere along the line—and it will be much closer to 15 parameters than to 300—the significant improvement will cease and the fitting operation is over. There is no system of 300 equations, no 300 parameters, and no glamor.

The computer center's director must prevent the looting of valuable computer time by these would-be fitters of many parameters. The task is not a pleasant one, but the legitimate computer users have rights, too. . . the impasse finally has to be broken by violence—which therefore might as well be used in the very beginning.

Numerical methods that (mostly) work, Forman S. Acton (1990, original edition 1970)