An Introduction to Machine Learning

L1: Basics and Probability Theory

Alexander J. Smola

Statistical Machine Learning Program
Canberra, ACT 0200 Australia
Alex.Smola@nicta.com.au

Machine Learning Summer School 2008

Overview

L1: Machine learning and probability theory
Introduction to pattern recognition, classification, regression, novelty detection, probability theory, Bayes rule, inference

L2: Density estimation and Parzen windows
Nearest Neighbor, Kernels density estimation, Silverman's rule, Watson Nadaraya estimator, crossvalidation

L3: Perceptron and Kernels
Hebb's rule, perceptron algorithm, convergence, feature maps, kernels

L4: Support Vector estimation
Geometrical view, dual problem, convex optimization, kernels

L5: Support Vector estimation
Regression, Quantile regression, Novelty detection, ν-trick

L6: Structured Estimation
Sequence annotation, web page ranking, path planning, implementation and optimization

L1 Introduction to Machine Learning

Data

- Texts, images, vectors, graphs

What to do with data

- Unsupervised learning (clustering, embedding, etc.)
- Classification, sequence annotation
- Regression, autoregressive models, time series
- Novelty detection

What is not machine learning

- Artificial intelligence
- Rule based inference

Statistics and probability theory

- Probability of an event
- Dependence, independence, conditional probability
- Bayes rule, Hypothesis testing

Outline

1. **Data**

2. Data Analysis
   - Unsupervised Learning
   - Supervised Learning

Data

Vectors

- Collections of features
  e.g. height, weight, blood pressure, age, ...
- Can map categorical variables into vectors

Matrices

- Images, Movies
- Remote sensing and satellite data (multispectral)

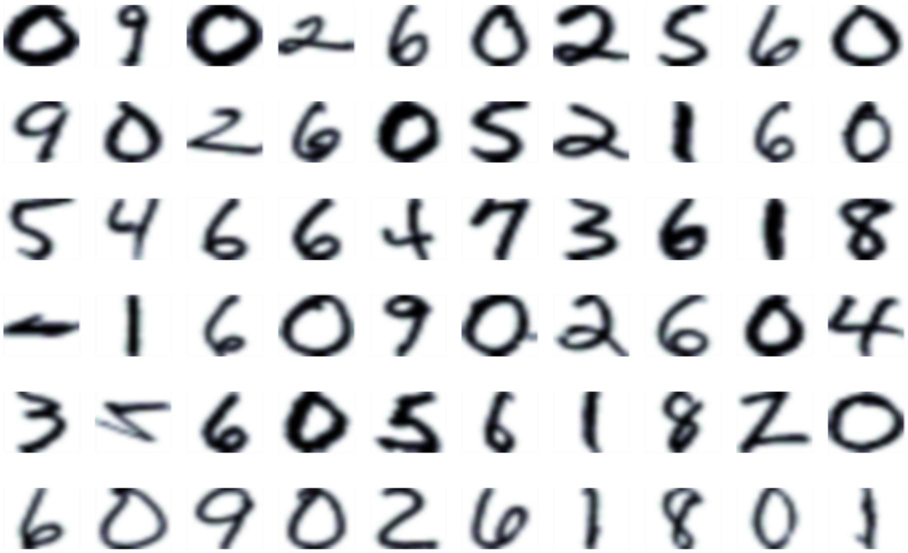
Strings

- Documents
- Gene sequences

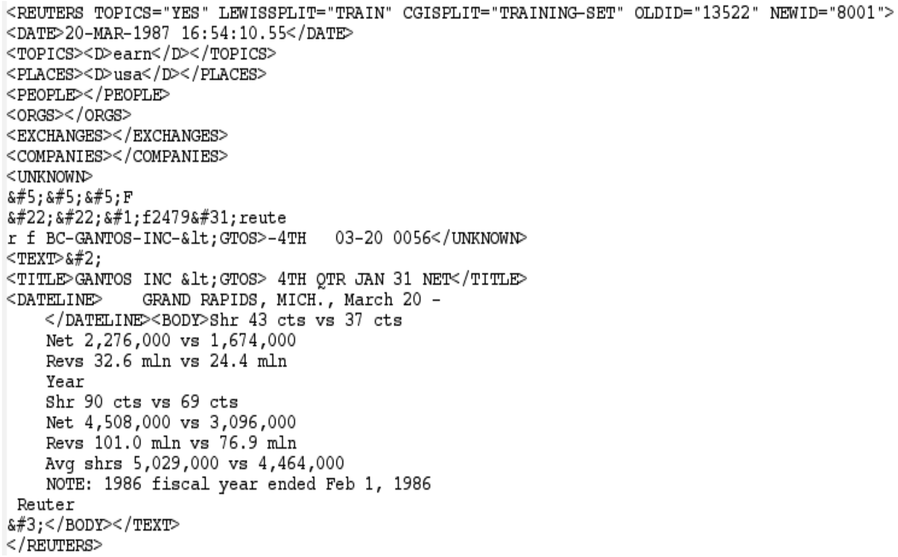
Structured Objects

- XML documents
- Graphs

Optical Character Recognition



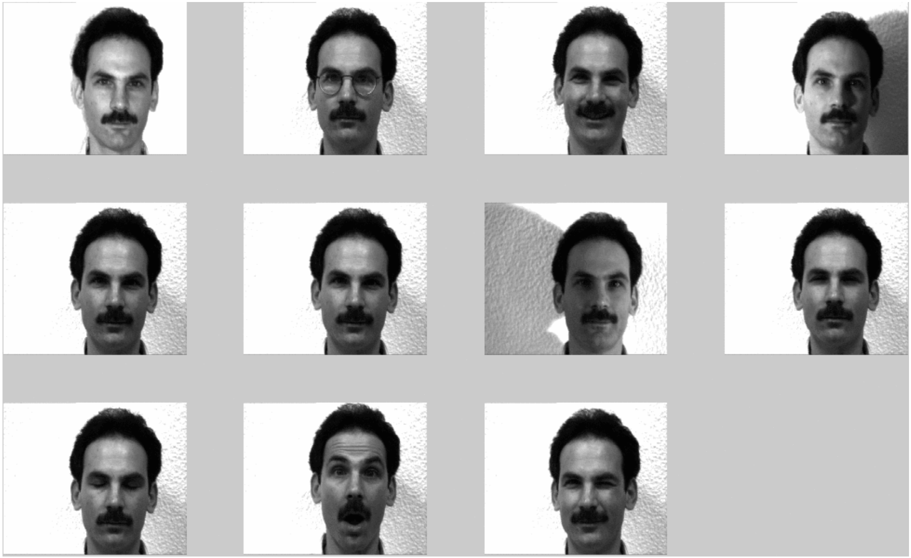
Reuters Database



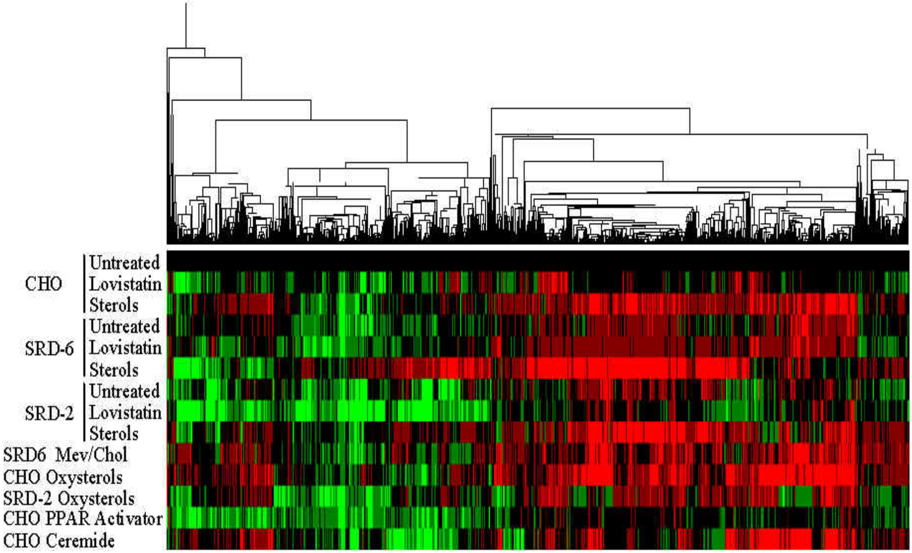
```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="13522" NEWID="8001">
<DATE>20-MAR-1987 16:54:10.55</DATE>
<TOPICS><D>earn</D></TOPICS>
<PLACES><D>usa</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;F
&#22;&#22;&#1;f2479&#31;reute
r f BC-GANTOS-INC-&lt;GTOS>-4TH   03-20 0056</UNKNOWN>
<TEXT>&#2;
<TITLE>GANTOS INC &lt;GTOS> 4TH QTR JAN 31 NET</TITLE>
<DATELINE>    GRAND RAPIDS, MICH., March 20 -
    </DATELINE><BODY>Shr 43 cts vs 37 cts
    Net 2,276,000 vs 1,674,000
    Revs 32.6 mln vs 24.4 mln
    Year
    Shr 90 cts vs 69 cts
    Net 4,508,000 vs 3,096,000
    Revs 101.0 mln vs 76.9 mln
    Avg shrs 5,029,000 vs 4,464,000
    NOTE: 1986 fiscal year ended Feb 1, 1986
 Reuter
&#3;</BODY></TEXT>
</REUTERS>
```

Faces

More Faces



Microarray Data



Biological Sequences

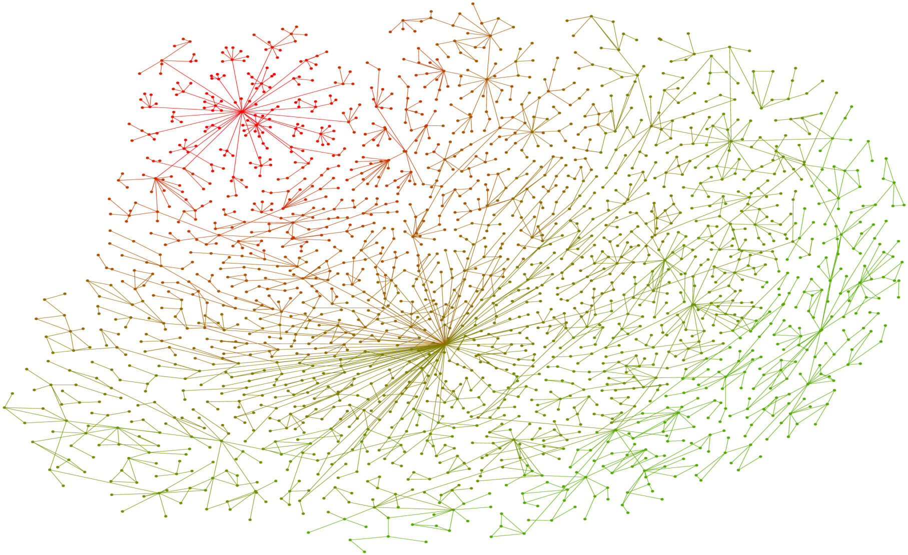
Goal

Estimate function of protein based on sequence information.

Example Data

```
>0_d1vcaa2 2.1.1.4.1 (1-90) N-terminal domain of vascular cell adhesion molecule-1 (VCAM-1) [human (Homo sapiens)]
FKIETTPESRYLAQIGDSVSLTCSTTGCESPFFSWRTQIDSPLNGKVTNEGTTSTLTMNPVSFGNEHSYL
CTATCESRKLEKGIQVEIYS
>0_d1zxq_2 2.1.1.4.2 (1-86) N-terminal domain of intracellular adhesion molecule-2, ICAM-2 [human (Homo sapiens)]
KVFEVHVRPKKLAVEPKGSLEVNCSTTCNQPEVGGLETSLNKILLDEQAQWKHYLVSNISHDTVLQCHFT
CSGKQESMNSNVSVYQ
>0_d1tlk__ 2.1.1.4.3 Telokin [turkey (Meleagris gallopavo)]
VAEEKPHVKPYFTKTILDMDVVEGSAARFDCKVEGYPDPEVMWFKDDNPVKESRHFQIDYDEEGNCSLTI
SEVCGDDDAKYTCKAVNSLGEATCTAELLVETM
>0_d2ncm__ 2.1.1.4.4 N-terminal domain of neural cell adhesion molecule (NCAM) [human (Homo sapiens)]
RVLQVDIVPSQGEISVGESKFFLCQVAGDAKDKDISWFSPNGEKLSPNQQRISVVWNDDDSSTLTIYNAN
IDDAGIYKCVVTAEDGTQSEATVNVKIFQ
 >0_d1tnm__ 2.1.1.4.5 Titin [Human (Homo sapiens), module M5]
RILTKPRSMTVYEGESARFSCDTDGEPVPTVTWLRKGQVLSTSARHQVTTTKYKSTFEISSVQASDEGNY
SVVVENSEGKQEAEFTLTIQK
>0_d1wiu__ 2.1.1.4.6 Twitchin [Nematode (Caenorhabditis elegans)]
LKPKILTASRKIKIKAGFTHNLEVDFIGAPDPTATWTVGDSGAALAPELLVDAKSSTTSIFFPSAKRADS
GNYKLKVKNELGEDEAIFEVIVQ
>0_d1koa_1 2.1.1.4.6 (351-447) Twitchin [Nematode (Caenorhabditis elegans)]
QPRFIVKPYGTEVGEGQSANFYCRVIASSPPVVTWHKDDRELKQSVKYMKRYNGNDYGLTINRVKGDDKG
EYTVRAKNSYGTKEEIVFLNVTRHSEP
```

Graphs



Missing Variables

Incomplete Data

- Measurement devices may fail
  E.g. dead pixels on camera, microarray, forms incomplete, . . .
- Measuring things may be expensive
  diagnosis for patients
- Data may be censored

How to fix it

- Clever algorithms (not this course . . .)
- **Simple mean imputation**
  Substitute in the average from other observations
- Works amazingly well (for starters) . . .

Mini Summary

Data Types

- Vectors (feature sets, microarrays, HPLC)
- Matrices (photos, dynamical systems, controllers)
- Strings (texts, biological sequences)
- Structured documents (XML, HTML, collections)
- Graphs (web, gene networks, tertiary structure)

Problems and Opportunities

- Data may be incomplete (use mean imputation)
- Data may come from different sources (adapt model)
- Data may be biased (e.g. it is much easier to get blood samples from university students for cheap).
- Problem may be ill defined, e.g. “find information.” (get information about what user really needs)
- Environment may react to intervention (butterfly portfolios in stock markets)

Outline

1. Data
2. **Data Analysis**
   - Unsupervised Learning
   - Supervised Learning

What to do with data

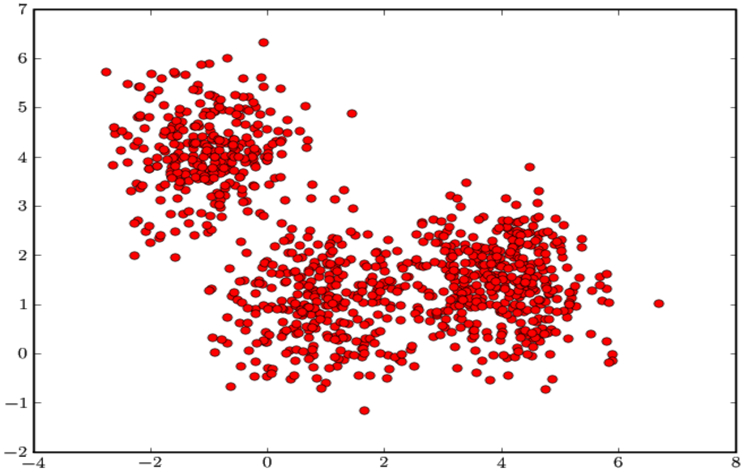
Unsupervised Learning

- Find clusters of the data
- Find low-dimensional representation of the data (e.g. unroll a swiss roll, find structure)
- Find interesting directions in data
- Interesting coordinates and correlations
- Find novel observations / database cleaning

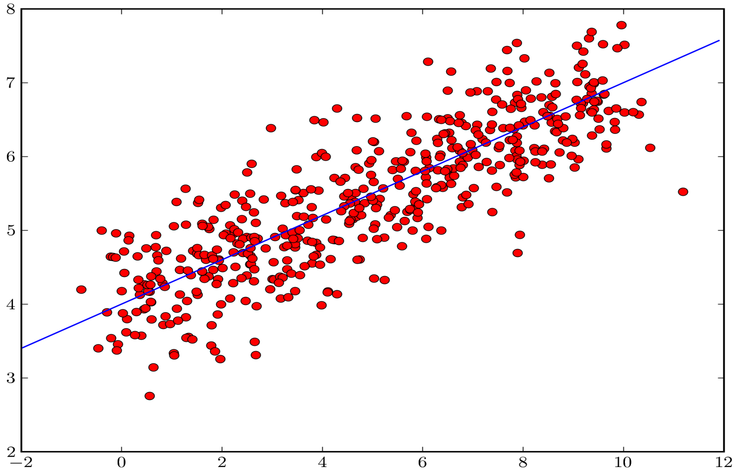
Supervised Learning

- Classification (distinguish apples from oranges)
- Speech recognition
- Regression (tomorrow's stock value)
- Predict time series
- Annotate strings

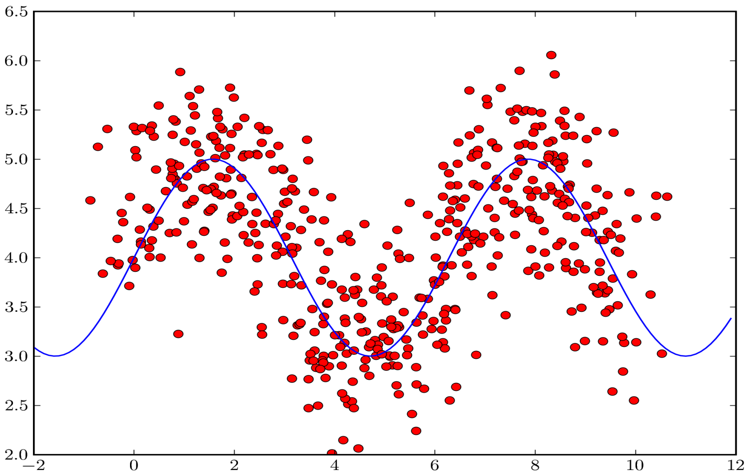
Clustering



Principal Components



Linear Subspace



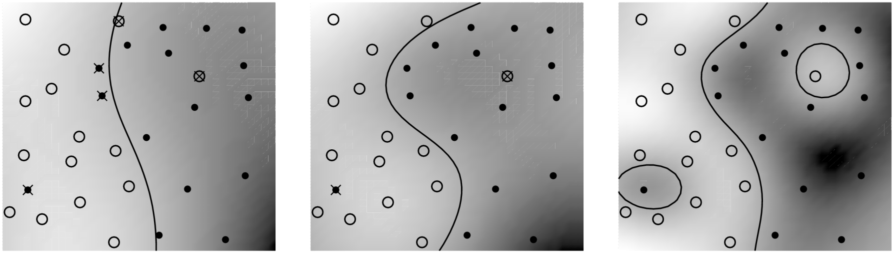
Classification

Data

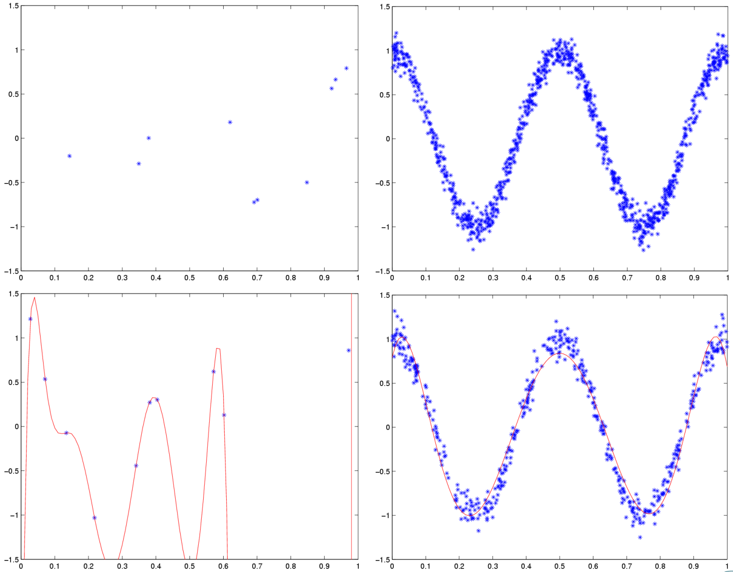
Pairs of observations (x_i, y_i) drawn from distribution e.g., (blood status, cancer), (credit transactions, fraud), (sound profile of jet engine, defect)

Goal

Estimate $y \in \{\pm 1\}$ **given** x at a new location. Or find a function $f(x)$ that does the trick.



Regression



Regression

Data

Pairs of observations (x_i, y_i) generated from some joint distribution $\Pr(x, y)$, e.g.,

- market index, SP100
- fab parfameters, yield
- user profile, price

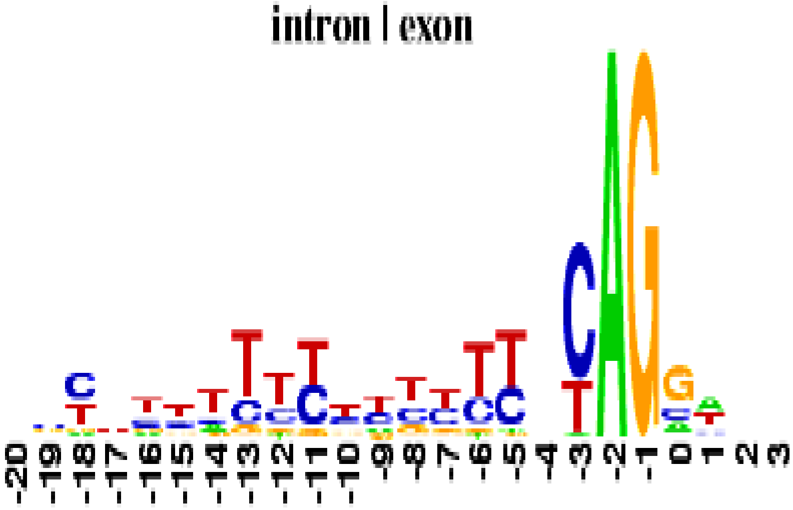
Task

Estimate y, given x, such that some loss $c(x, y, f(x))$ is minimized.

Examples

- Quadratic error between y and $f(x)$, i.e. $c(x, y, f(x)) = \frac{1}{2}(y - f(x))^2$.
- Absolute value, i.e., $c(x, y, f(x)) = |y - f(x))|$.

Annotating Strings



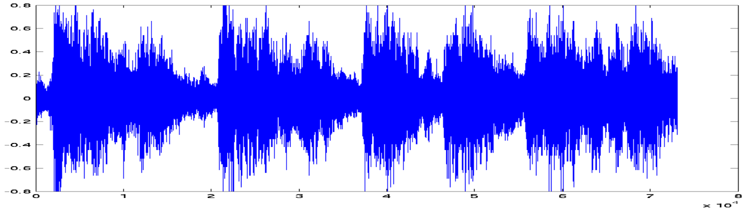
Annotating Audio

Goal

- Possible meaning of an audio sequence
- Give confidence measure

Example (from Australian Prime Minister's speech)

- a stray alien
- Australian



Novelty Detection

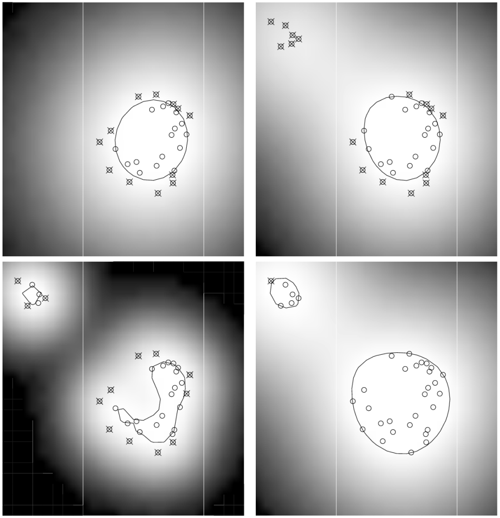
Data

Observations x_i from some $\mathrm{P}(x)$, e.g.,

- network usage patterns
- handwritten digits
- alarm sensors
- factory status

Task

Find unusual events, clean database, distinguish typical examples.



What Machine Learning is not

Logic

- If A meets B and B meets C, does A know C?
- Rule satisfaction
- Logical rules from data

Artificial Intelligence

- Understanding of the world
- Meet *Sunny* from *I, Robot*
- Go and get me a bottle of beer
  Robot need not *understand* what it is doing.

Biology and Neuroscience

- Understand the brain by building neural networks?!?
- Model brain and build good systems with that
- Get inspiration from biology but no requirement to build systems like that (e.g. jet planes don’t flap wings)

Eliza

Rule based AI dialog system, vintage 1970

I am the psychotherapist. Please, describe your problems.

I am here to learn about giving good talks

Why do you say that?

Because I'm interested in learning that

Is it because you are interested in learning that that you came to me?

Yes, indeed.

Earlier you said you are interested in learning that?

Sure, you stupid computer, this is what I want!

Maybe your plans have something to do with this.

Dammit, sure, that's what i'm trying to say.

I don't understand.

Because you're stupid.

Does it bother you that I am stupid?

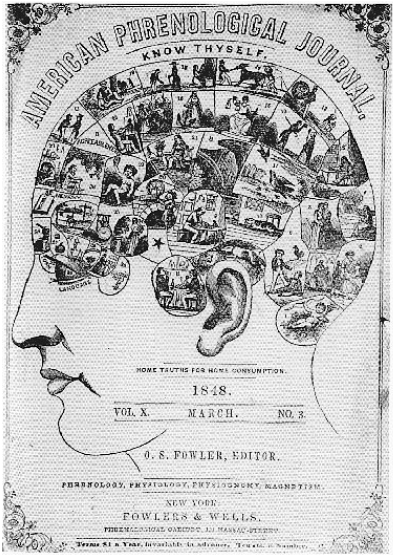
Indeed, it does.

You mentioned that you are interested in learning that?

That's true. And now I want to relax a bit before giving a talk.

Try it out on Emacs using `M-x doctor`.

How the brain doesn’t work


AMERICAN PHRENOLOGICAL JOURNAL.
KNOW THYSELF.
LANGUAGE
HOME TRUTHS FOR HOME CONSUMPTION.
1848.
VOL. X.
MARCH.
NO. 3.
O. S. FOWLER, EDITOR.
PHRENOLOGY, PHYSIOLOGY, PHYSIOGNOMY, MAGNETISM.
NEW YORK:
FOWLERS & WELLS,
PHRENOLOGICAL CABINET, 131 NASSAU-STREET.
Terms $1 a Year, invariably in advance.

Mini Summary

Structure Extraction

- Clustering
- Low-dimensional subspaces
- Low-dimensional representation of data

Novelty Detection

- Find typical observations (Joe Sixpack)
- Find highly unusual ones (oddball)
- Database cleaning

Supervised Learning

- Regression
- Classification
- Preference relationships (recommender systems)

Statistics and Probability Theory

Why do we need it?

- We deal with **uncertain events**
- Need mathematical formulation for probabilities
- Need to estimate probabilities from data
  (e.g. for coin tosses, we only observe number of heads and tails, not whether the coin is really fair).

How do we use it?

- Statement about probability that an object is an apple (rather than an orange)
- Probability that two things happen at the same time
- Find unusual events (= low density events)
- Conditional events
  (e.g. what happens if A, B, and C are true)

Probability

Basic Idea

We have events in a space of possible outcomes. Then $\Pr(X)$ tells us how likely is that an event $x \in X$ will occur.

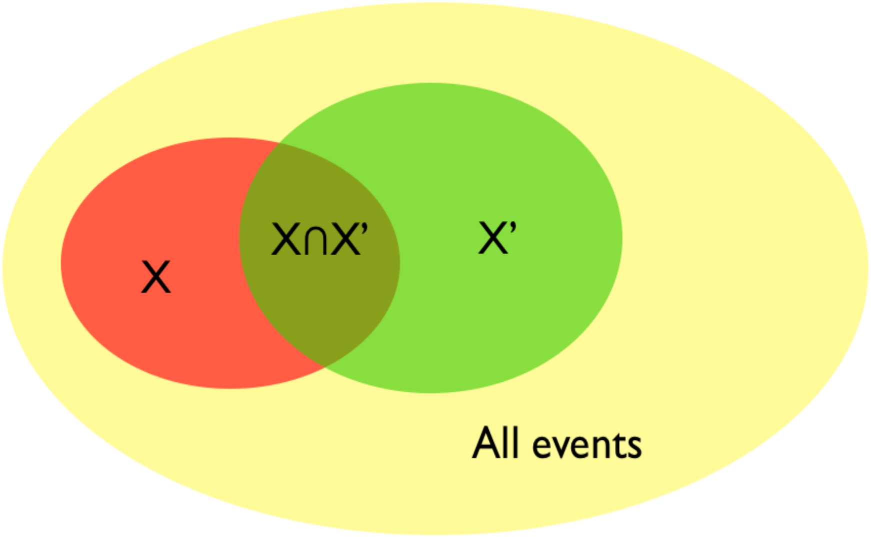
Basic Axioms

- $\Pr(X) \in [0, 1]$ for all $X \subseteq \mathfrak{X}$
- $\Pr(\mathfrak{X}) = 1$
- $\Pr\left(\cup_i X_i\right) = \sum_i \Pr(X_i)$ if $X_i \cap X_j = \emptyset$ for all $i \neq j$

Simple Corollary

$$\Pr(X \cup Y) = \Pr(X) + \Pr(Y) - \Pr(X \cap Y)$$

Example



Multiple Variables

Two Sets

Assume that x and y are drawn from a probability measure on the product space of $\mathcal{X}$ and $\mathcal{Y}$. Consider the space of events $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

Independence

If x and y are independent, then for all $X \subset \mathcal{X}$ and $Y \subset \mathcal{Y}$

$$\Pr(X, Y) = \Pr(X) \cdot \Pr(Y).$$

Independent Random Variables

	Y Outcome: boy	Y Outcome: girl
X Astrologist's Prediction: boy	0.25	0.25
X Astrologist's Prediction: girl	0.25	0.25

Dependent Random Variables

Y
Outcome

X
Physician's
Prediction

	0.49	0.01
	0.01	0.49

Bayes Rule

Dependence and Conditional Probability

Typically, knowing x will tell us something about y (think regression or classification). We have

$$\Pr(Y|X)\Pr(X) = \Pr(Y,X) = \Pr(X|Y)\Pr(Y).$$

- Hence $\Pr(Y,X) \leq \min(\Pr(X), \Pr(Y))$.

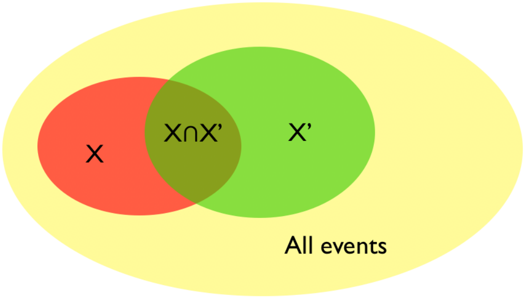
Bayes Rule

$$\Pr(X|Y) = \frac{\Pr(Y|X)\Pr(X)}{\Pr(Y)}.$$

Proof using conditional probabilities

$$\Pr(X, Y) = \Pr(X|Y)\Pr(Y) = \Pr(Y|X)\Pr(X)$$

Example



$$\Pr(X \cap X') = \Pr(X|X')\Pr(X') = \Pr(X'|X)\Pr(X)$$

AIDS Test

How likely is it to have AIDS if the test says so?

- Assume that roughly 0.1% of the population is infected.

$$p(X = \text{AIDS}) = 0.001$$

- The AIDS test reports positive for all infections.

$$p(Y = \text{test positive} | X = \text{AIDS}) = 1$$

- The AIDS test reports positive for 1% healthy people.

$$p(Y = \text{test positive} | X = \text{healthy}) = 0.01$$

We use Bayes rule to infer $\Pr(\text{AIDS}|\text{test positive})$ via

$$\frac{\Pr(Y|X)\Pr(X)}{\Pr(Y)} = \frac{\Pr(Y|X)\Pr(X)}{\Pr(Y|X)\Pr(X) + \Pr(Y|\mathcal{X}\backslash X)\Pr(\mathcal{X}\backslash X)}$$
$$= \frac{1 \cdot 0.001}{1 \cdot 0.001 + 0.01 \cdot 0.999} = 0.091$$

Eye Witness

Evidence from an Eye-Witness

A witness is 90% certain that a certain customer committed the crime. There were 20 people in the bar ...

Would you convict the person?

- Everyone is presumed innocent until proven guilty:

$$p(X = \text{guilty}) = 1/20$$

- Eyewitness has equal confusion probability

$$p(Y = \text{eyewitness identifies}|X = \text{guilty}) = 0.9$$
$$\text{and } p(Y = \text{eyewitness identifies}|X = \text{not guilty}) = 0.1$$

Bayes Rule

$$\Pr(X|Y) = \frac{0.9 \cdot 0.05}{0.9 \cdot 0.05 + 0.1 \cdot 0.95} = 0.3213 = 32\%$$

But most judges would convict him anyway ...

Improving Inference

Follow up on the AIDS test:

The doctor performs a followup via a conditionally independent test which has the following properties:

- The second test reports positive for 90% infections.
- The AIDS test reports positive for 5% healthy people.

$$\Pr(\mathrm{T1}, \mathrm{T2}|\mathrm{Health}) = \Pr(\mathrm{T1}|\mathrm{Health}) \Pr(\mathrm{T2}|\mathrm{Health}).$$

A bit more algebra reveals (assuming that both tests are independent): $\frac{0.01 \cdot 0.05 \cdot 0.999}{0.01 \cdot 0.05 \cdot 0.999 + 1 \cdot 0.9 \cdot 0.001} = 0.357.$

Conclusion:

Adding extra observations can improve the confidence of the test considerably.

Different Contexts

Hypothesis Testing:

- Is solution A or B better to solve the problem (e.g. in manufacturing)?
- Is a coin tainted?
- Which parameter setting should we use?

Sensor Fusion:

- Evidence from sensors A and B (AIDS test 1 and 2).
- We have different types of data.

More Data:

- We obtain two sets of data — we get more confident
- Each observation can be seen as an additional test

Mini Summary

Probability theory

- Basic tools of the trade
- Use it to model uncertain events

Dependence and Independence

- Independent events don't convey any information about each other.
- Dependence is what we exploit for estimation
- Leads to Bayes rule

Testing

- Prior probability matters
- Combining tests improves outcomes
- Common sense can be misleading

Summary

Data

Vectors, matrices, strings, graphs, . . .

What to do with data

Unsupervised learning (clustering, embedding, etc.), Classification, sequence annotation, Regression, . . .

Random Variables

Dependence, Bayes rule, hypothesis testing

An Introduction to Machine Learning

L2: Instance Based Estimation

Alexander J. Smola

Statistical Machine Learning Program
Canberra, ACT 0200 Australia
Alex.Smola@nicta.com.au

Machine Learning Summer School 2008

L2 Instance Based Methods

Nearest Neighbor Rules

Density estimation

- empirical frequency, bin counting
- priors and Laplace rule

Parzen windows

- Smoothing out the estimates
- Examples

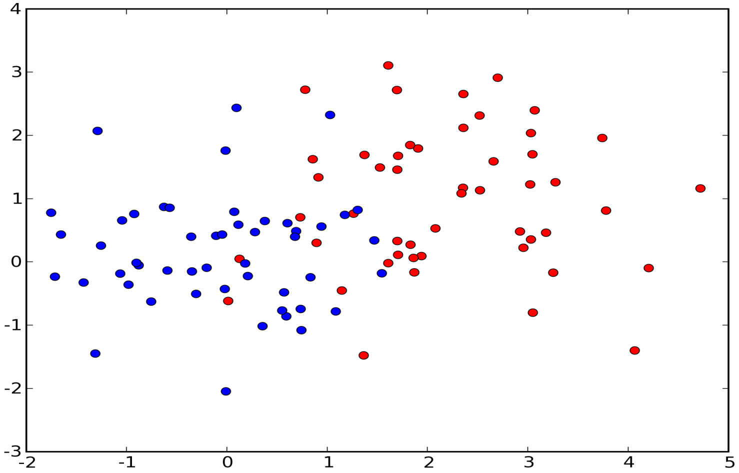
Adjusting parameters

- Cross validation
- Silverman's rule

Classification and regression with Parzen windows

- Watson-Nadaraya estimator

Binary Classification



Nearest Neighbor Rule

Goal

Given some data x_i, want to classify using class label y_i.

Solution

Use the label of the nearest neighbor.

Modified Solution (classification)

Use the label of the majority of the k nearest neighbors.

Modified Solution (regression)

Use the value of the average of the k nearest neighbors.

Key Benefits

- Basic algorithm is **very simple**.
- Can use arbitrary similarity measures
- Will eventually converge to the best possible result.

Problems

- Slow and inefficient when we have lots of data.
- Not very smooth estimates.

Python Pseudocode

Nearest Neighbor Classifier

```
from pylab import *
from numpy import *

...  load data ...

xnorm = sum(x**2)
xtestnorm = sum(xtest**2)

dists = (-2.0*dot(x.transpose(), xtest) + xtestnorm).transpose() + xnorm

labelindex = dists.argmin(axis=1)
```

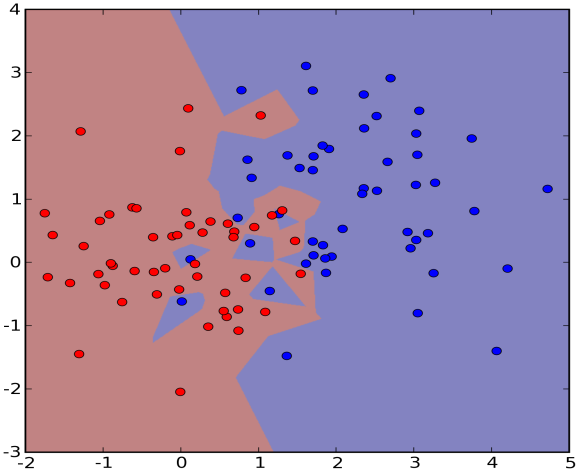
k-Nearest Neighbor Classifier

```
sortargs = dists.argsort(axis=1)
k = 7
ytest = sign(mean(y[sortargs[:,0:k]], axis=1))
```

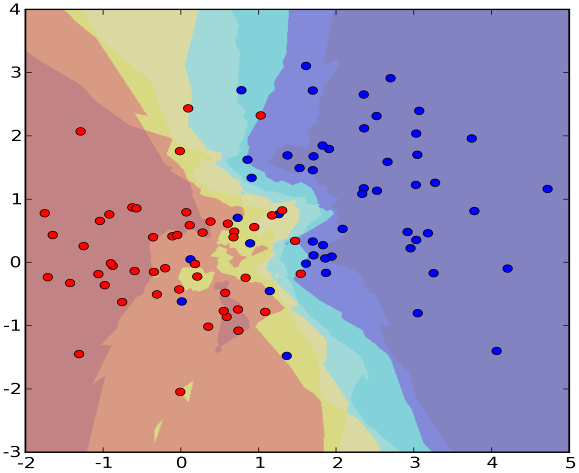
Nearest Neighbor Regression

just drop `sign(...)`

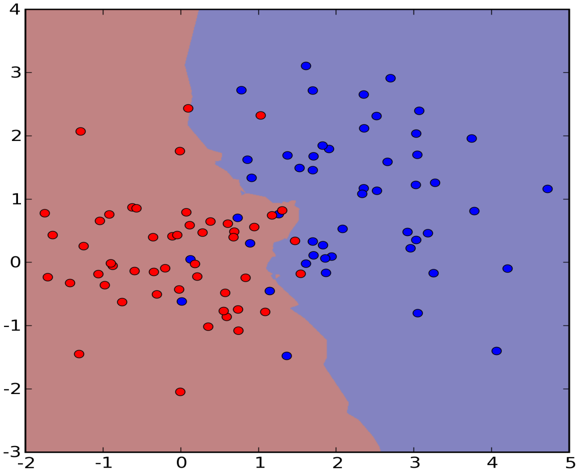
Nearest Neighbor



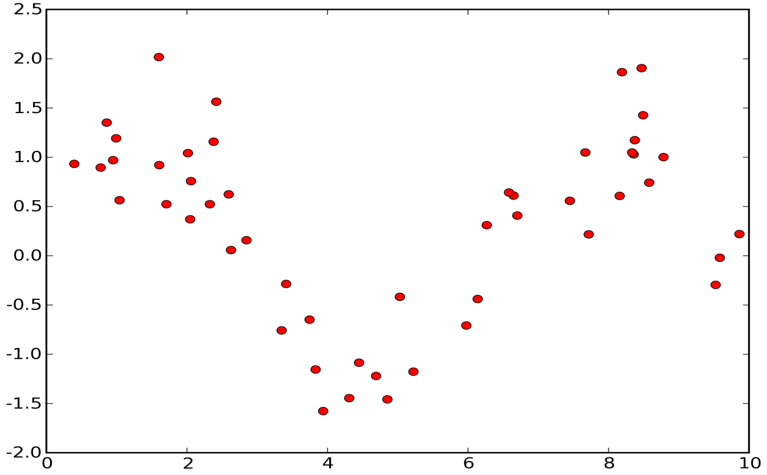
7 Nearest Neighbors



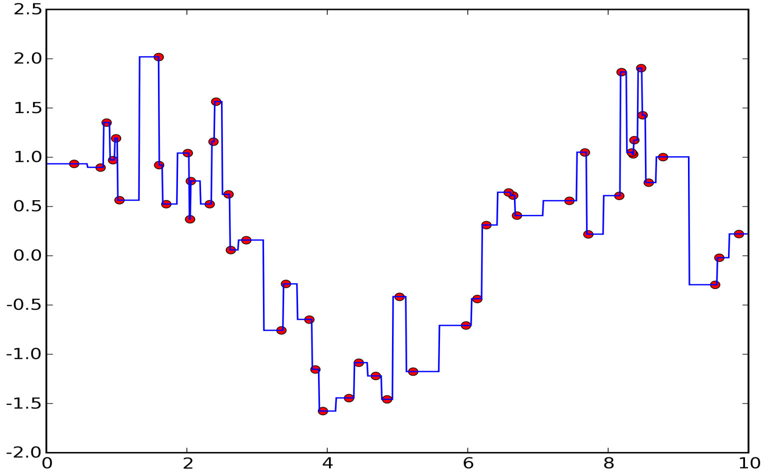
7 Nearest Neighbors



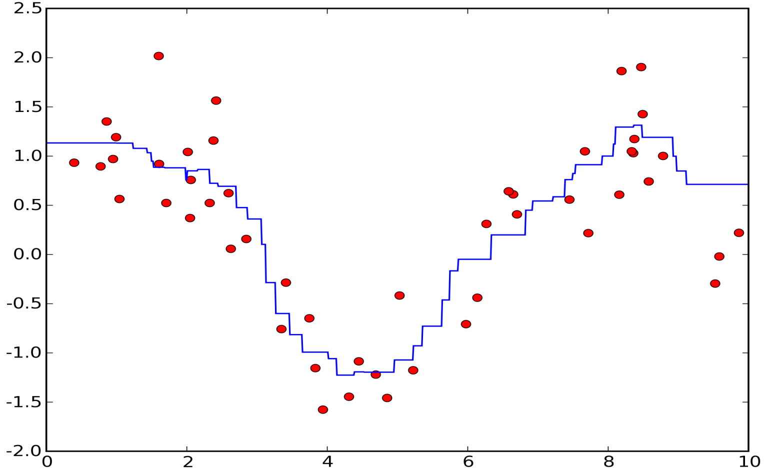
Regression Problem



Nearest Neighbor Regression



7 Nearest Neighbors Regression



Mini Summary

Nearest Neighbor Rule

Predict same label as nearest neighbor

***k*-Nearest Neighbor Rule**

Average estimates over k neighbors

Details

- Easy to implement
- No training required
- Slow if lots of training data
- Not so great performance

Estimating Probabilities from Data

Rolling a dice:

Roll the dice many times and count how many times each side comes up. Then assign empirical probability estimates according to the frequency of occurrence.

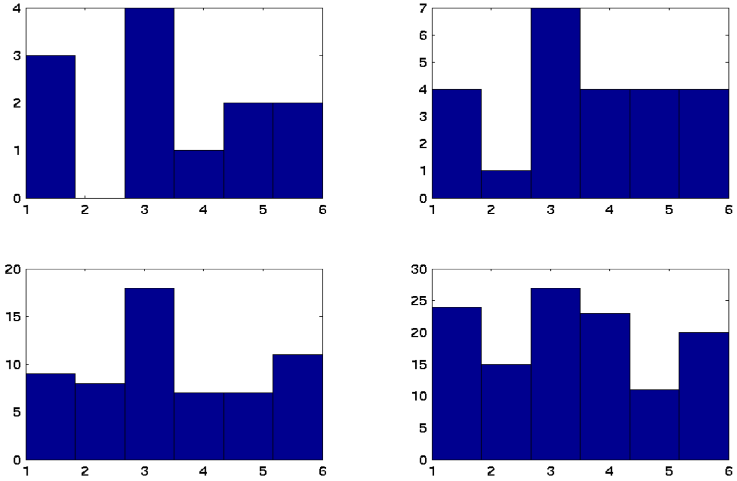
$$\hat{\Pr}(i) = \frac{\#\text{occurrences of } i}{\#\text{trials}}$$

Maximum Likelihood Estimation:

Find parameters such that the observations are *most likely* given the current set of parameters.

This does not check whether the parameters are plausible!

Practical Example



Properties of MLE

Hoeffding's Bound

The probability estimates converge exponentially fast

$$\Pr\{|\pi_i - p_i| > \epsilon\} \leq 2\exp(-2m\epsilon^2)$$

Problem

- For small ϵ this can still take a very long time. In particular, for a fixed confidence level δ we have

$$\delta = 2\exp(-2m\epsilon^2) \Longrightarrow \epsilon = \sqrt{\frac{-\log\delta + \log 2}{2m}}$$

- The above bound holds only for single π_i, **but not uniformly over all i**.

Improved Approach

If we know something about π_i, we should use this extra information: use priors.

Priors to the Rescue

Big Problem

Only sampling *many times* gets the parameters right.

Rule of Thumb

We need at least 10-20 times as many observations.

Conjugate Priors

Often we know what we should expect. Using a conjugate prior helps. We **insert fake additional data** which we assume that it comes from the prior.

Conjugate Prior for Discrete Distributions

- Assume we see u_i additional observations of class i.

$$\pi_i = \frac{\#\text{occurrences of } i + u_i}{\#\text{trials} + \sum_j u_j}.$$

- Assuming that the dice is even, set $u_i = m_0$ for all $1 \leq i \leq 6$. For $u_i = 1$ this is the **Laplace Rule**.

Example: Dice

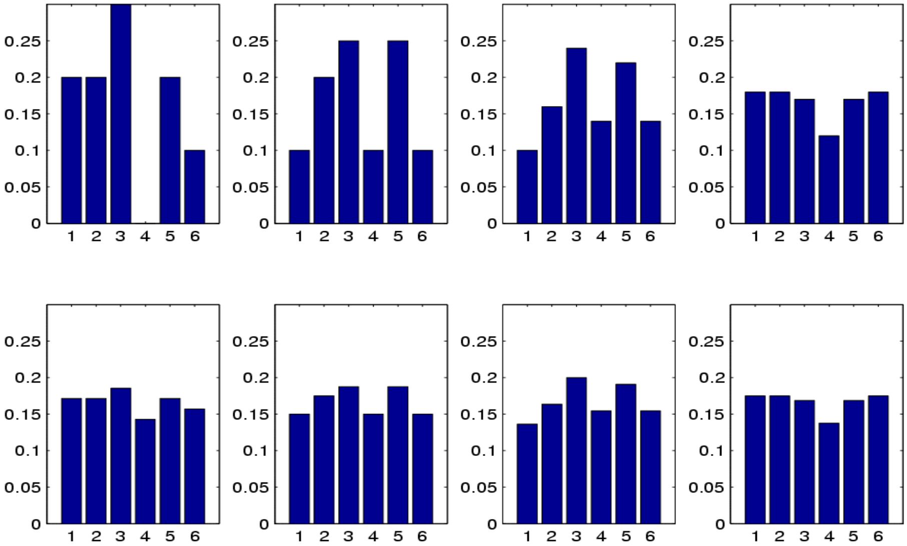
20 tosses of a dice

Outcome	1	2	3	4	5	6
Counts	3	6	2	1	4	4
MLE	0.15	0.30	0.10	0.05	0.20	0.20
MAP ($m_0 = 6$)	0.25	0.27	0.12	0.08	0.19	0.19
MAP ($m_0 = 100$)	0.16	0.19	0.16	0.15	0.17	0.17

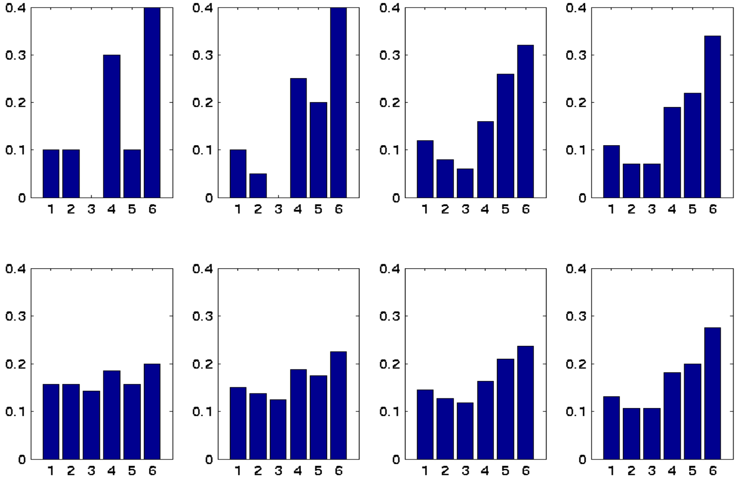
Consequences

- Stronger prior brings the estimate closer to uniform distribution.
- More robust against outliers
- But: Need more data to detect deviations from prior

Correct dice



Tainted dice



Mini Summary

Maximum Likelihood Solution

- Count number of observations per event
- Set probability to empirical frequency of occurrence.

Maximum a Posteriori Solution

- We have a good guess about solution
- Use conjugate prior
- Corresponds to inventing extra data
- Set probability to take additional observations into account

Big Guns: Hoeffding and friends

- Use uniform convergence and tail bounds
- Exponential convergence for fixed scale
- Only sublinear convergence, when fixed confidence.

Extension

- Works also for other estimates, such as means and covariance matrices.

Density Estimation

Data

Continuous valued random variables.

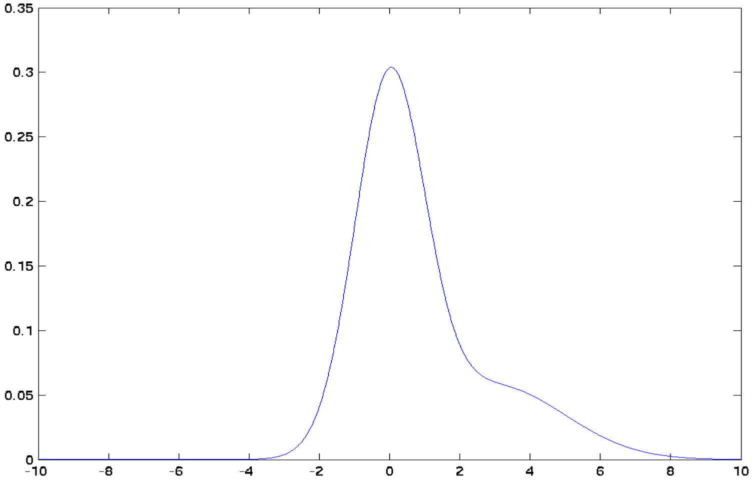
Naive Solution

Apply the bin-counting strategy to the continuum. That is, we discretize the domain into bins.

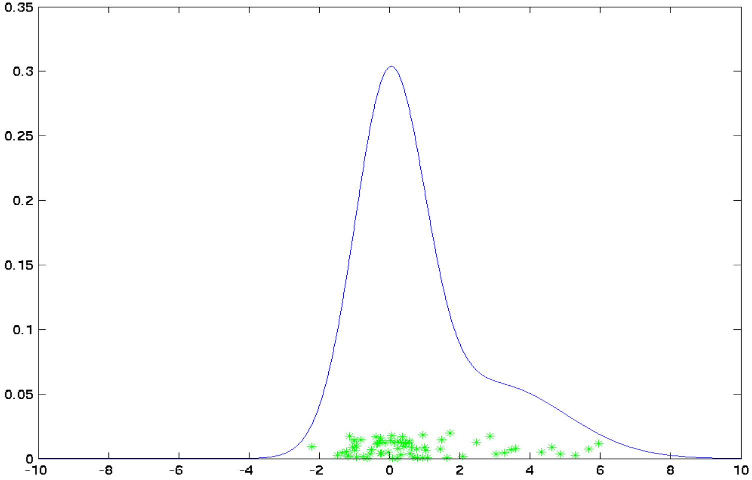
Problems

- We need lots of data to fill the bins
- In more than one dimension the number of bins grows exponentially:
- Assume 10 bins per dimension, so we have 10 in $\mathbb{R}^1$
- 100 bins in $\mathbb{R}^2$
- 10^{10} bins (10 billion bins) in $\mathbb{R}^{10}$ …

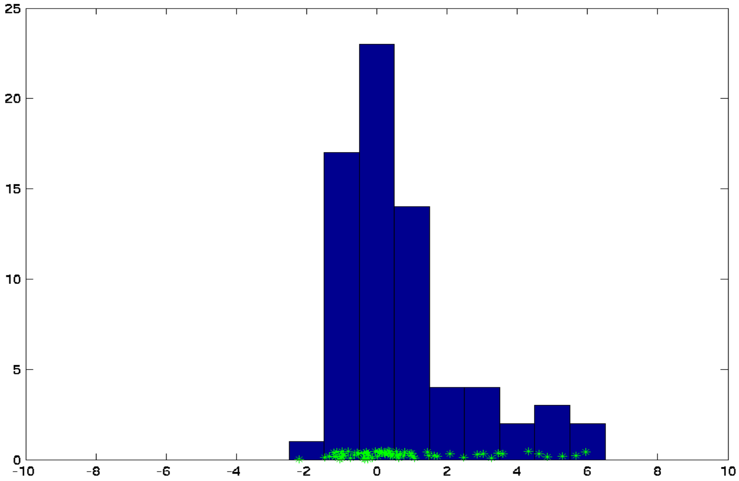
Mixture Density



Sampling from $p(x)$



Bin counting



Parzen Windows

Naive approach

Use the empirical density

$$p_{\mathrm{emp}}(x) = \frac{1}{m}\sum_{i=1}^{m}\delta(x, x_i).$$

which has a delta peak for every observation.

Problem

What happens when we see slightly different data?

Idea

Smear out p_{emp} by convolving it with a kernel $k(x, x')$. Here $k(x, x')$ satisfies

$$\int_{\mathcal{X}} k(x, x')dx' = 1 \text{ for all } x \in \mathcal{X}.$$

Parzen Windows

Estimation Formula

Smooth out p_{emp} by convolving it with a kernel $k(x, x')$.

$$p(x) = \frac{1}{m} \sum_{i=1}^{m} k(x_i, x)$$

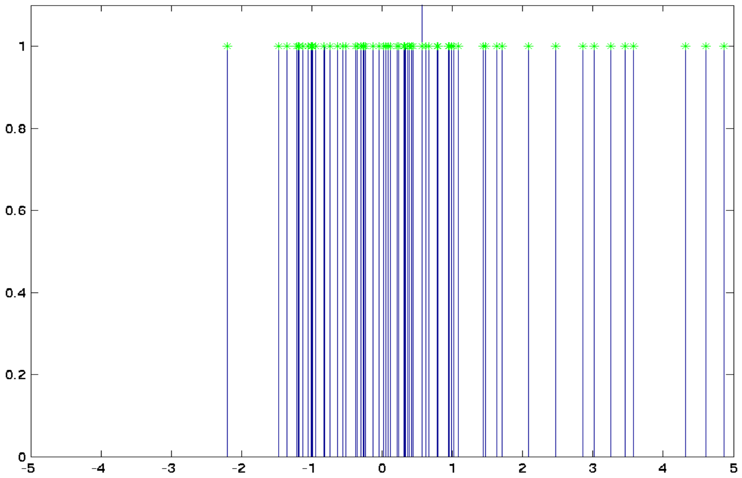
Adjusting the kernel width

- Range of data should be adjustable
- Use kernel function $k(x, x')$ which is a proper kernel.
- Scale kernel by radius r. This yields

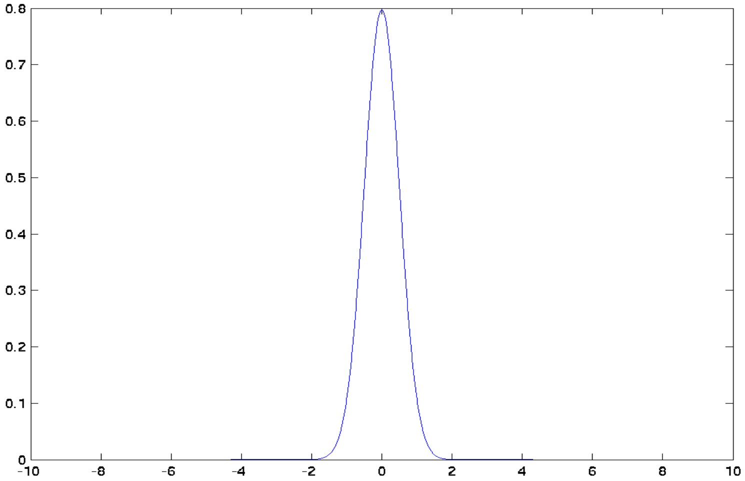
$$k_r(x, x') := r^n k(rx, rx')$$

Here n is the dimensionality of x.

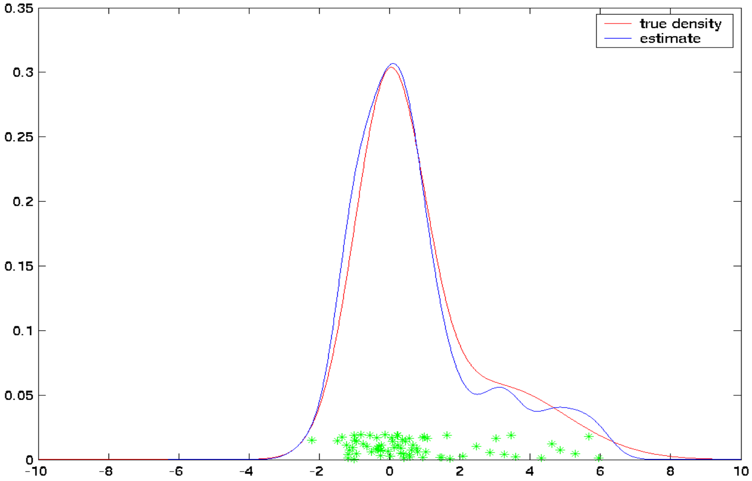
Discrete Density Estimate



Smoothing Function



Density Estimate



Examples of Kernels

Gaussian Kernel

$$k(x, x') = (2\pi\sigma^2)^{\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}\|x - x'\|^2\right)$$

Laplacian Kernel

$$k(x, x') = \lambda^n 2^{-n} \exp\left(-\lambda\|x - x'\|_1\right)$$

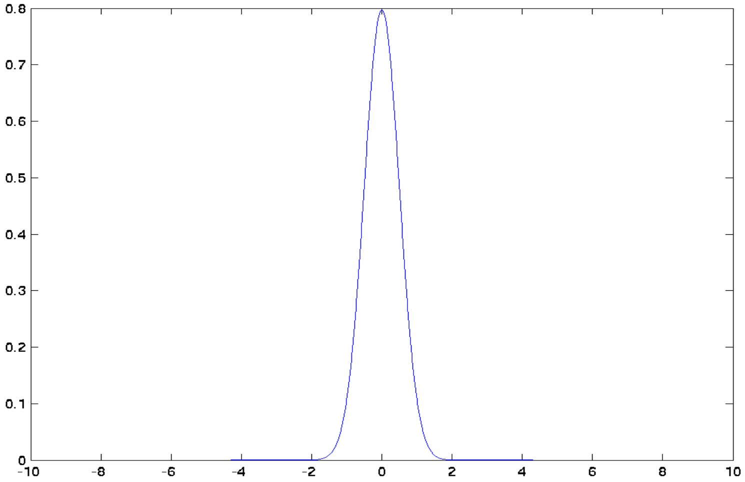
Indicator Kernel

$$k(x, x') = 1_{[-0.5,0.5]}(x - x')$$

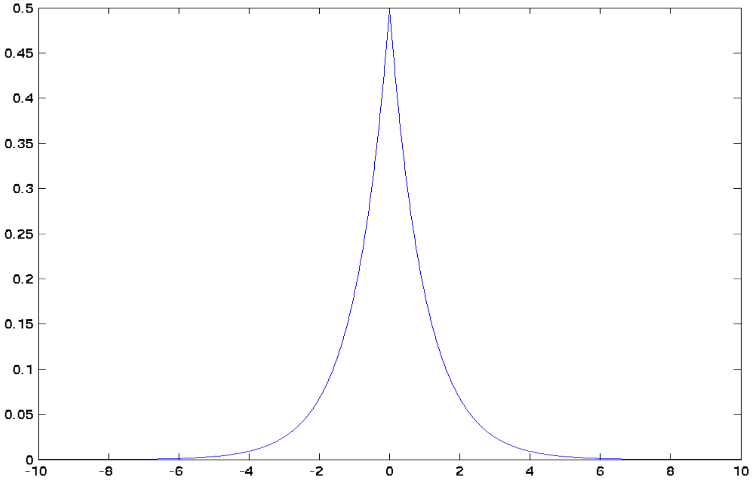
Important Issue

Width of the kernel is usually much more important than **type**.

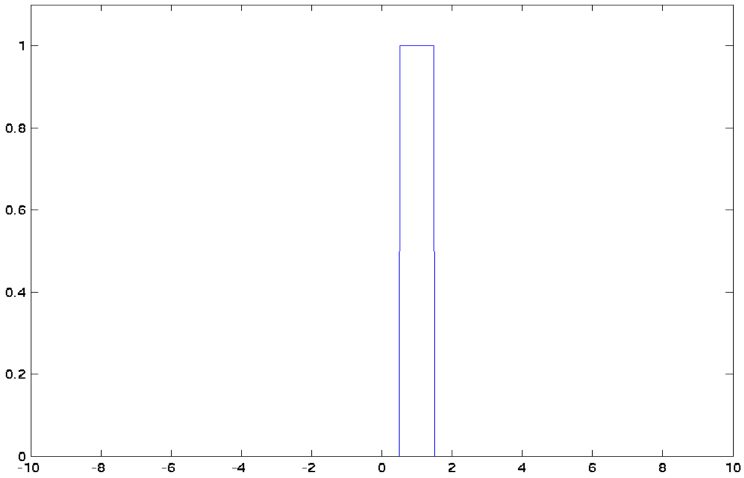
Gaussian Kernel



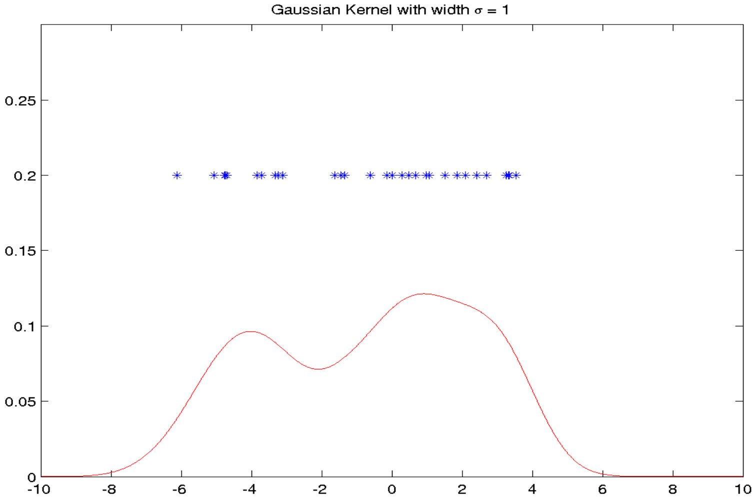
Laplacian Kernel



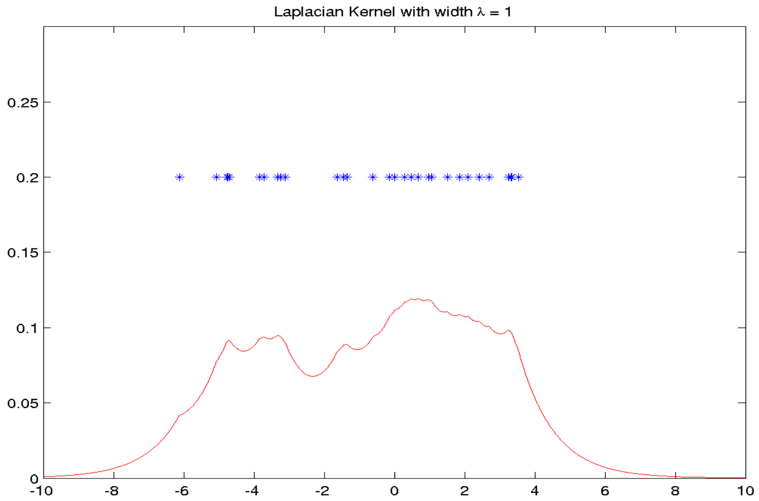
Indicator Kernel



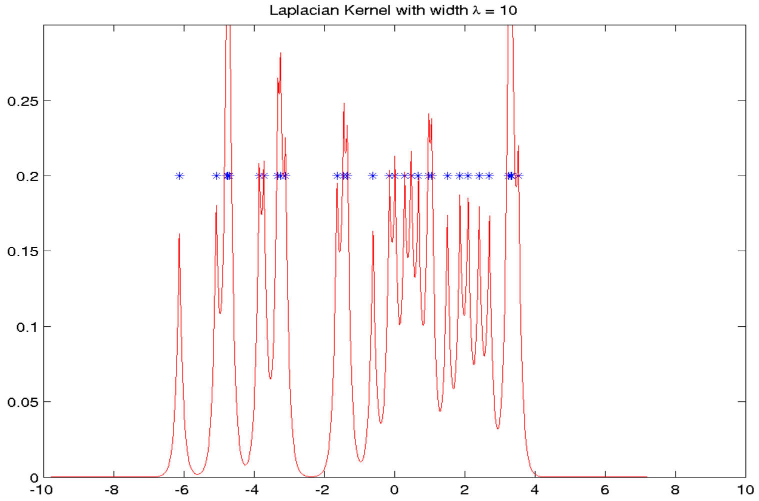
Gaussian Kernel



Laplacian Kernel



Laplacian Kernel



Selecting the Kernel Width

Goal

We need a method for adjusting the kernel width.

Problem

The likelihood keeps on increasing as we narrow the kernels.

Reason

The likelihood estimate we see is distorted (we are being overly optimistic through optimizing the parameters).

Possible Solution

Check the performance of the density estimate on an unseen part of the data. This can be done e.g. by

- Leave-one-out crossvalidation
- Ten-fold crossvalidation

Expected log-likelihood

What we really want

- A parameter such that in expectation the likelihood of the data is maximized

$$p_r(X) = \prod_{i=1}^{m} p_r(x_i)$$

or equivalently $$\frac{1}{m}\log p_r(X) = \frac{1}{m}\sum_{i=1}^{m}\log p_r(x_i).$$

- However, if we optimize r for the seen data, we will always overestimate the likelihood.

Solution: Crossvalidation

- Test on unseen data
- Remove a fraction of data from X, say X', estimate using $X \backslash X'$ and test on X'.

Crossvalidation Details

Basic Idea

Compute $p(X'|\theta(X\backslash X'))$ for various subsets of X and average over the corresponding log-likelihoods.

Practical Implementation

Generate subsets $X_i \subset X$ and compute the log-likelihood estimate

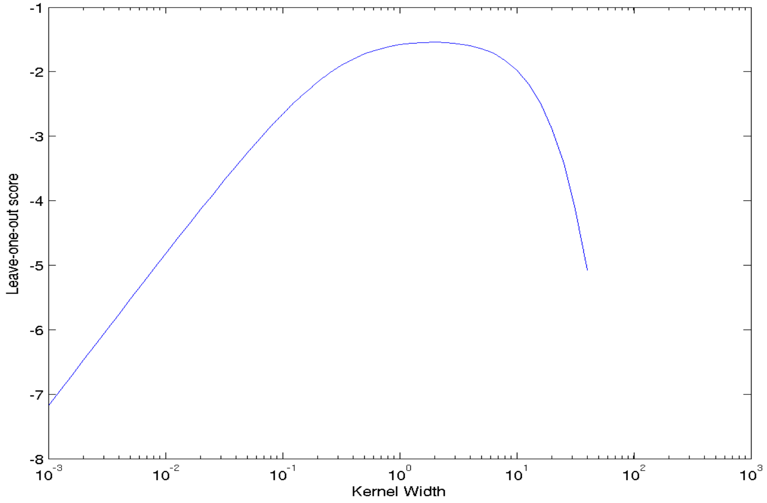
$$\frac{1}{n}\sum_{i}^{n}\frac{1}{|X_i|}\log p(X_i|\theta(X|\backslash X_i))$$

Pick the parameter which maximizes the above estimate.

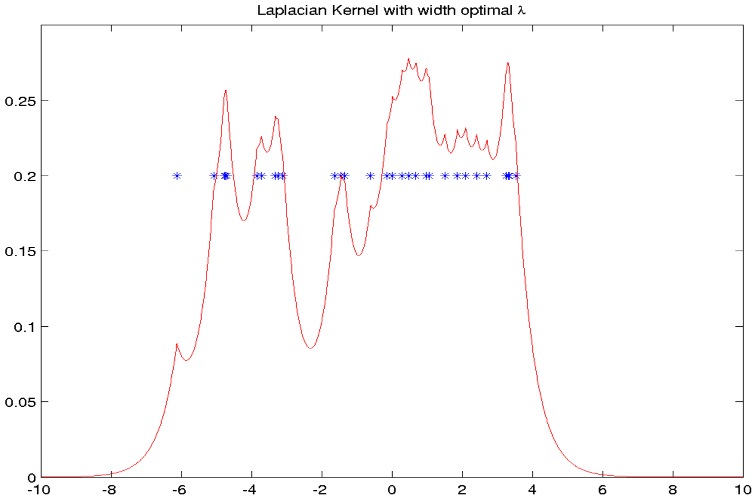
Special Case: Leave-one-out Crossvalidation

$$p_{X\backslash x_i}(x_i) = \frac{m}{m-1}p_X(x_i) - \frac{1}{m-1}k(x_i, x_i)$$

Cross Validation



Best Fit ($\lambda = 1.9$)



Mini Summary

Discrete Density

- Bin counting
- Problems for continuous variables
- Really big problems for variables in high dimensions (curse of dimensionality)

Parzen Windows

- Smooth out discrete density estimate.
- Smoothing kernel integrates to 1 (allows for similar observations to have some weight).
- Density estimate is average over kernel functions
- Scale kernel to accommodate spacing of data

Tuning it

- Cross validation
- Expected log-likelihood

Application: Novelty Detection

Goal

Find the least likely observations x_i from a dataset X. Alternatively, identify low-density regions, given X.

Idea

Perform density estimate $p_X(x)$ and declare all x_i with $p_X(x_i) < p_0$ as novel.

Algorithm

Simply compute $f(x_i) = \sum_j k(x_i, x_j)$ for all i and sort according to their magnitude.

Applications

Network Intrusion Detection

Detect whether someone is trying to hack the network, downloading tons of MP3s, or doing anything else *unusual* on the network.

Jet Engine Failure Detection

You can't destroy jet engines just to see *how* they fail.

Database Cleaning

We want to find out whether someone stored bogus information in a database (typos, etc.), mislabelled digits, ugly digits, bad photographs in an electronic album.

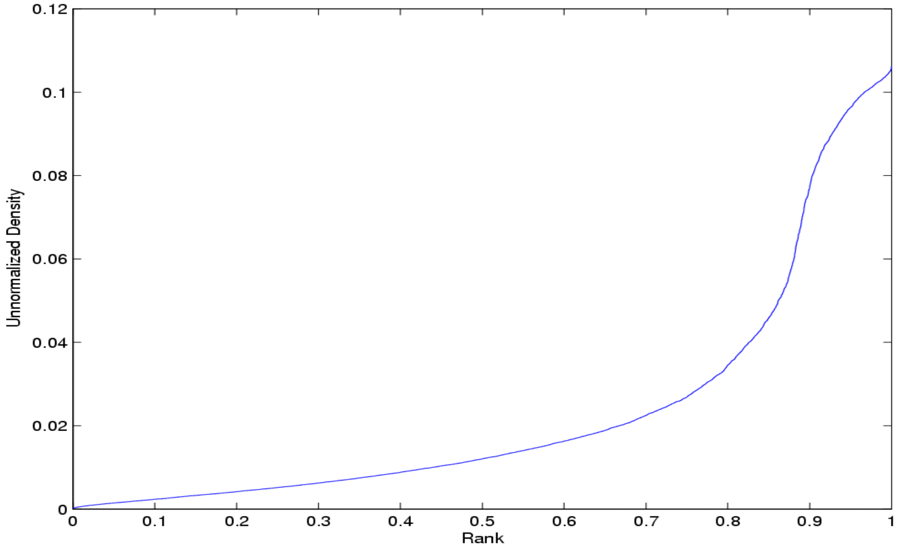
Fraud Detection

Credit Cards, Telephone Bills, Medical Records

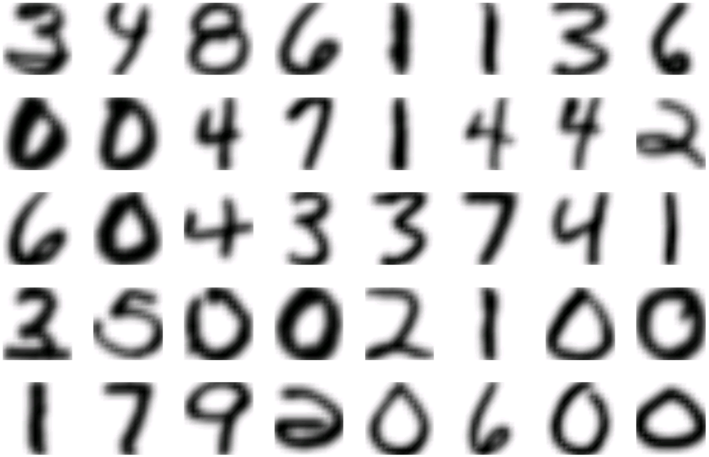
Self calibrating alarm devices

Car alarms (adjusts itself to where the car is parked), home alarm (furniture, temperature, windows, etc.)

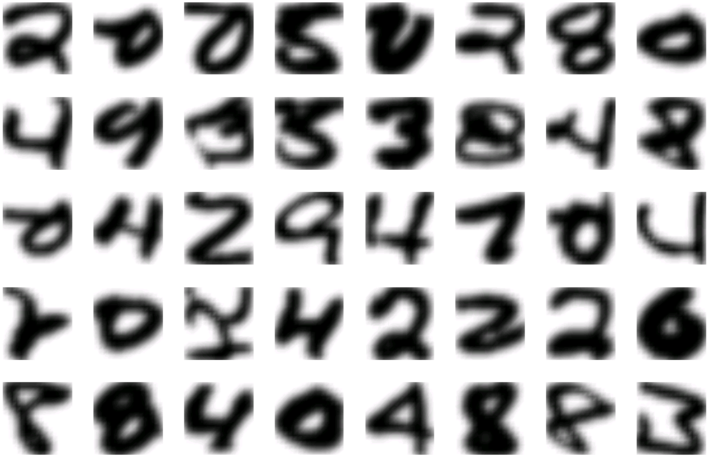
Order Statistic of Densities



Typical Data



Outliers



Silverman's Automatic Adjustment

Problem

One 'width fits all' does not work well whenever we have regions of high and of low density.

Idea

Adjust width such that neighbors of a point are included in the kernel at a point. More specifically, adjust range h_i to yield

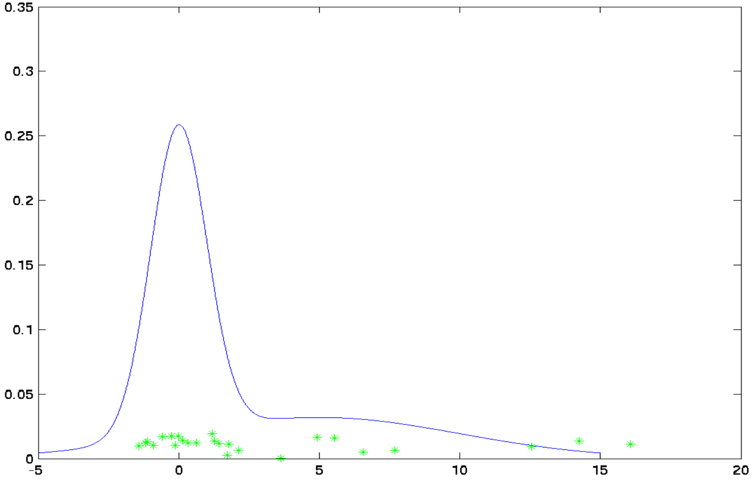
$$h_i = \frac{r}{k} \sum_{x_j \in \mathrm{NN}(x_i, k)} \|x_j - x_i\|$$

where $\mathrm{NN}(x_i, k)$ is the set of k nearest neighbors of x_i and r is typically chosen to be 0.5.

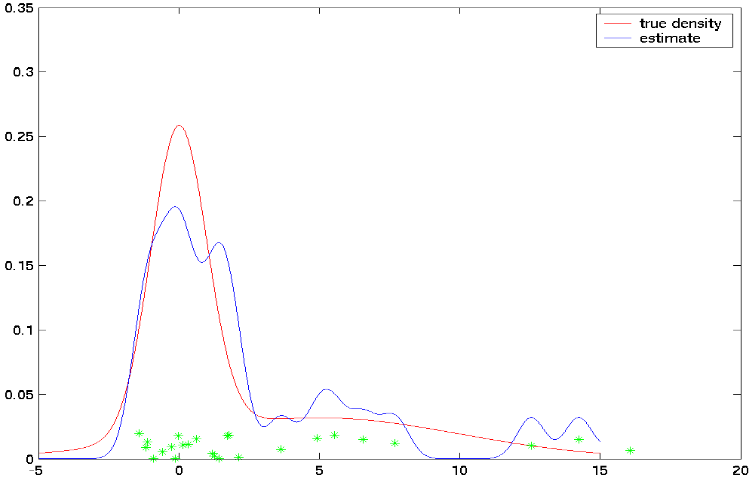
Result

State of the art density estimator, regression estimator and classifier.

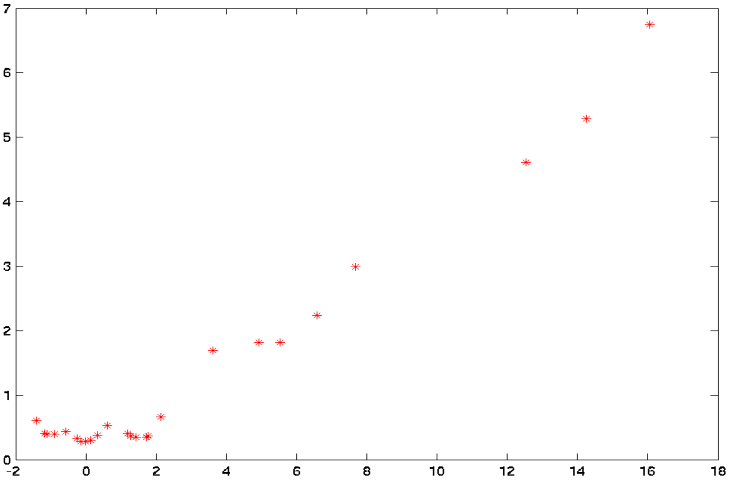
Sampling from $p(x)$



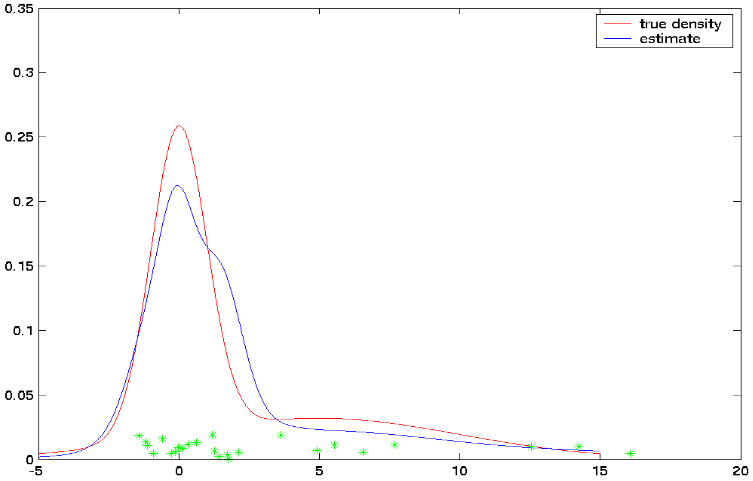
Uneven Scales



Neighborhood Scales



Adjusted Width



Watson-Nadaraya Estimator

Goal

Given pairs of observations (x_i, y_i) with $y_i \in \{\pm 1\}$ find estimator for conditional probability $\Pr(y|x)$.

Idea

Use definition $p(x, y) = p(y|x)p(x)$ and estimate both $p(x)$ and $p(x, y)$ using Parzen windows. Using Bayes rule this yields

$$\Pr(y = 1|x) = \frac{P(y = 1, x)}{P(x)} = \frac{m^{-1} \sum_{y_i=1} k(x_i, x)}{m^{-1} \sum_i k(x_i, x)}$$

Bayes optimal decision

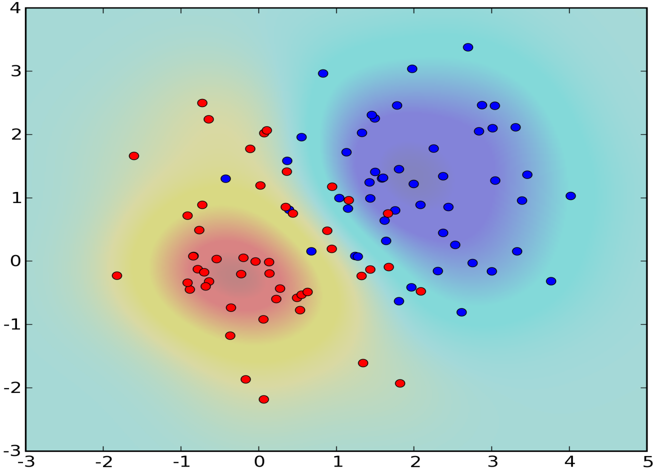
We want to classify $y = 1$ for $\Pr(y = 1|x) > 0.5$. This is equivalent to checking the sign of

$$\Pr(y = 1|x) - \Pr(y = -1|x) \propto \sum_i y_i k(x_i, x)$$

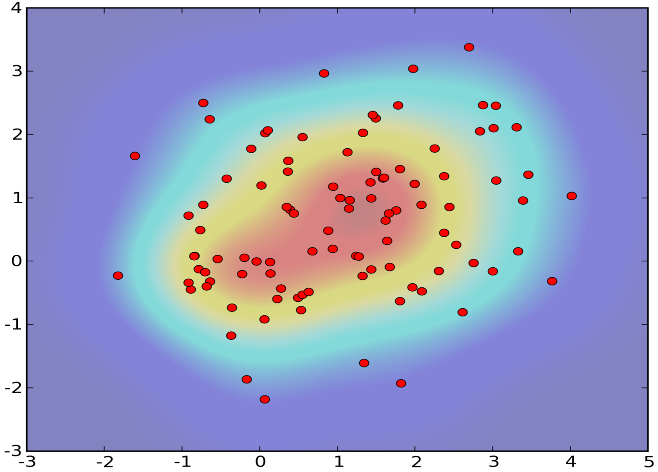
Python Pseudocode

```
# Kernel function
import elefant.kernels.vector
k = elefant.kernels.vector.CGaussKernel(1)

# Compute difference between densities
ytest = k.Expand(xtest, x, y)

# Compute density estimate (up to scalar)
density = k.Expand(xtest, x, ones(x.shape[0]))
```

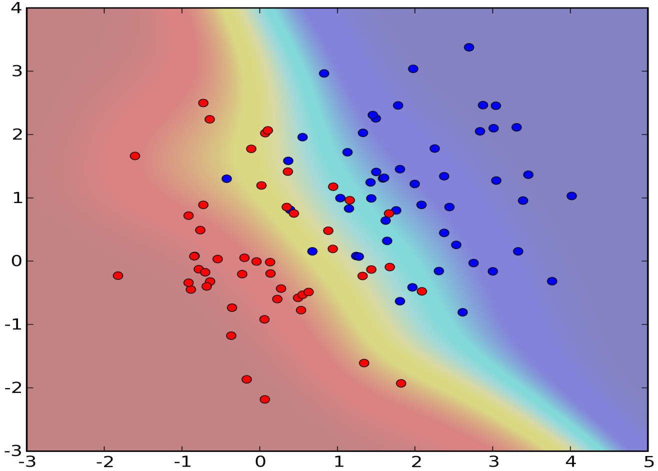
Parzen Windows Classifier



Parzen Windows Density Estimate



Parzen Windows Conditional



Watson Nadaraya Regression

Decision Boundary

Picking $y = 1$ or $y = -1$ depends on the sign of

$$\Pr(y = 1|x) - \Pr(y = -1|x) = \frac{\sum_i y_i k(x_i, x)}{\sum_i k(x_i, x)}$$

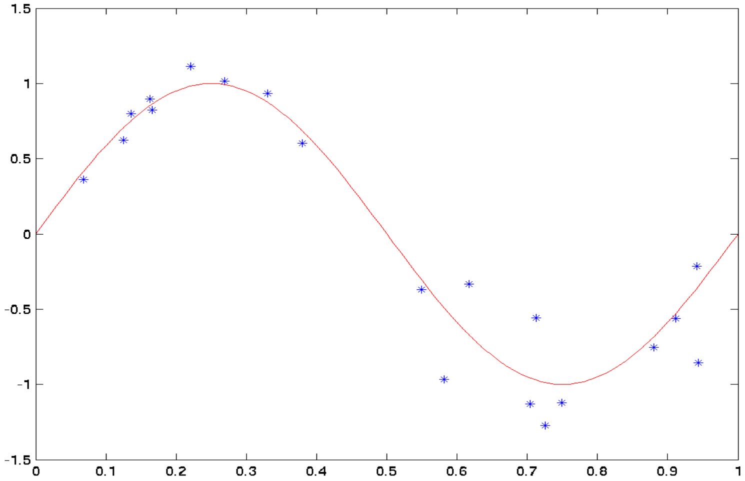
Extension to Regression

- Use the same equation for regression. This means that

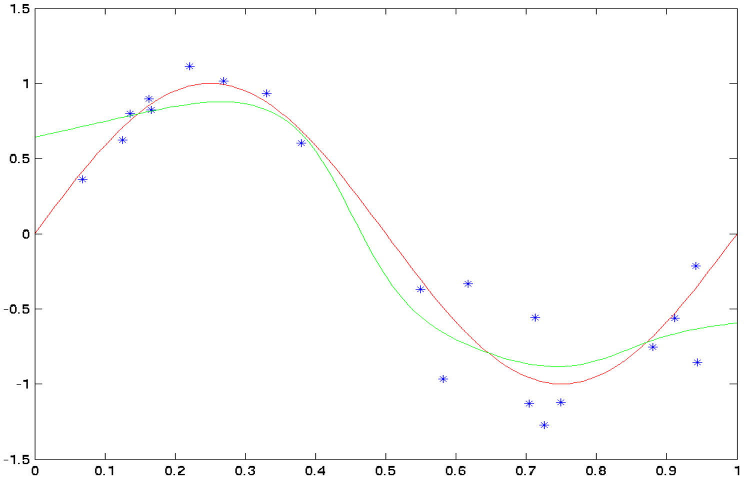
$$f(x) = \frac{\sum_i y_i k(x_i, x)}{\sum_i k(x_i, x)}$$

  where now $y_i \in \mathbb{R}$.
- We get a locally weighted version of the data

Regression Problem



Watson Nadaraya Regression



Mini Summary

Novelty Detection

- Observations in low-density regions are special (outliers).
- Applications to database cleaning, network security, etc.

Adaptive Kernel Width (Silverman's Trick)

- Kernels wide wherever we have low density

Watson Nadaraya Estimator

- Conditional density estimate
- Difference between class means (in feature space)
- Same expression works for regression, too

Summary

Density estimation

- empirical frequency, bin counting
- priors and Laplace rule

Parzen windows

- Smoothing out the estimates
- Examples

Adjusting parameters

- Cross validation
- Silverman's rule

Classification and regression with Parzen windows

- Watson-Nadaraya estimator
- Nearest neighbor classifier

An Introduction to Machine Learning

L3: Perceptron and Kernels

Alexander J. Smola

Statistical Machine Learning Program
Canberra, ACT 0200 Australia
Alex.Smola@nicta.com.au

Machine Learning Summer School 2008

L3 Perceptron and Kernels

Hebb's rule

- positive feedback
- perceptron convergence rule

Hyperplanes

- Linear separability
- Inseparable sets

Features

- Explicit feature construction
- Implicit features via kernels

Kernels

- Examples
- Kernel perceptron

Biology and Learning

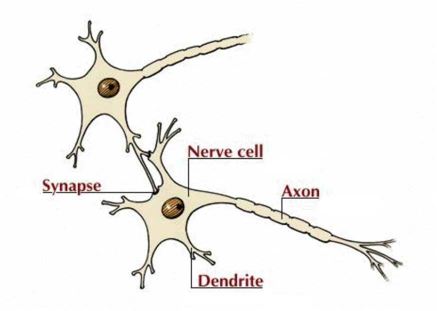
Basic Idea

- Good behavior should be rewarded, bad behavior punished (or not rewarded).
  This improves the fitness of the system.
- Example: hitting a tiger should be rewarded . . .
- Correlated events should be combined.
- Example: Pavlov's salivating dog.

Training Mechanisms

- Behavioral modification of individuals (learning):
  Successful behavior is rewarded (e.g. food).
- Hard-coded behavior in the genes (instinct):
  The wrongly coded animal dies.

Neurons



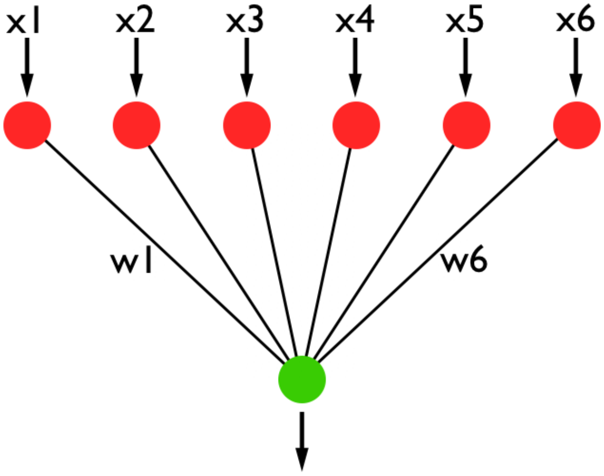
Soma
Cell body. Here the signals are combined ("CPU").

Dendrite
Combines the inputs from several other nerve cells ("input bus").

Synapse
Interface between two neurons ("connector").

Axon
This may be up to 1m long and will transport the activation signal to nerve cells at different locations ("output cable").

Perceptron



$$f(x) = w1\ x1 + \ldots + w6\ x6$$

Perceptrons

Weighted combination

- The output of the neuron is a linear combination of the inputs (from the other neurons via their axons) rescaled by the synaptic weights.
- Often the output does not directly correspond to the activation level but is a monotonic function thereof.

Decision Function

- At the end the results are combined into

$$f(x) = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right).$$

Separating Half Spaces

Linear Functions

An abstract model is to assume that

$$f(x) = \langle w, x \rangle + b$$

where $w, x \in \mathbb{R}^m$ and $b \in \mathbb{R}$.

Biological Interpretation

The weights w_i correspond to the synaptic weights (activating or inhibiting), the multiplication corresponds to the processing of inputs via the synapses, and the summation is the combination of signals in the cell body (soma).

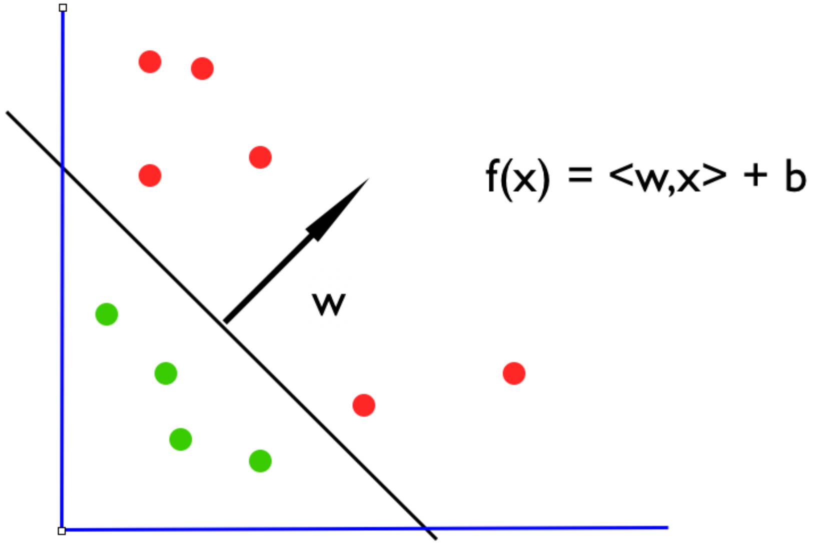
Applications

Spam filtering (e-mail), echo cancellation (old analog overseas cables)

Learning

Weights are “plastic” — adapted via the training data.

Linear Separation



Perceptron Algorithm

```
argument:  $X := \{x_1, \ldots, x_m\} \subset \mathcal{X}$ (data)
           $Y := \{y_1, \ldots, y_m\} \subset \{\pm 1\}$ (labels)
function $(w, b) = \mathrm{Perceptron}(X, Y)$
    initialize $w, b = 0$
    repeat
        Pick $(x_i, y_i)$ from data
        if $y_i(w \cdot x_i + b) \leq 0$ then
            $w' = w + y_i x_i$
            $b' = b + y_i$
    until $y_i(w \cdot x_i + b) > 0$ for all $i$
end
```

Interpretation

Algorithm

- Nothing happens if we classify (x_i, y_i) correctly
- If we see incorrectly classified observation we update (w, b) by $y_i(x_i, 1)$.
- Positive reinforcement of observations.

Solution

- Weight vector is linear combination of observations x_i:

$$w \longleftarrow w + y_i x_i$$

- Classification can be written in terms of dot products:

$$w \cdot x + b = \sum_{j \in E} y_j x_j \cdot x + b$$

Theoretical Analysis

Incremental Algorithm

Already while the perceptron is learning, we can use it.

Convergence Theorem (Rosenblatt and Novikoff)

Suppose that there exists a $\rho > 0$, a weight vector w^* satisfying $\|w^*\| = 1$, and a threshold b^* such that

$$y_i \left(\langle w^*, x_i\rangle + b^*\right) \geq \rho \text{ for all } 1 \leq i \leq m.$$

Then the hypothesis maintained by the perceptron algorithm converges to a linear separator after no more than

$$\frac{(b^{*2} + 1)(R^2 + 1)}{\rho^2}$$

updates, where $R = \max_i \|x_i\|$.

Proof, Part I

Starting Point

We start from $w_1 = 0$ and $b_1 = 0$.

Step 1: Bound on the increase of alignment

Denote by w_i the value of w at step i (analogously b_i).

$$\text{Alignment: } \langle (w_i, b_i), (w^*, b^*) \rangle$$

For error in observation (x_i, y_i) we get

$$\begin{aligned}
&\langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle \\
&= \langle \left[(w_j, b_j) + y_i(x_i, 1) \right], (w^*, b^*) \rangle \\
&= \langle (w_j, b_j), (w^*, b^*) \rangle + y_i \langle (x_i, 1) \cdot (w^*, b^*) \rangle \\
&\geq \langle (w_j, b_j), (w^*, b^*) \rangle + \rho \\
&\geq j\rho.
\end{aligned}$$

Alignment increases with number of errors.

Proof, Part II

Step 2: Cauchy-Schwartz for the Dot Product

$$\begin{aligned}\langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle &\leq \|(w_{j+1}, b_{j+1})\| \, \|(w^*, b^*)\| \\ &= \sqrt{1 + (b^*)^2} \|(w_{j+1}, b_{j+1})\|\end{aligned}$$

Step 3: Upper Bound on $\|(w_j, b_j)\|$

If we make a mistake we have

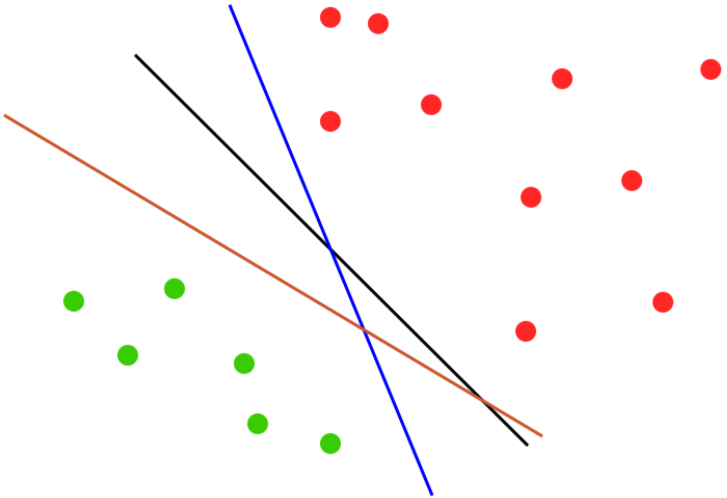
$$\begin{aligned}\|(w_{j+1}, b_{j+1})\|^2 &= \|(w_j, b_j) + y_i(x_i, 1)\|^2 \\ &= \|(w_j, b_j)\|^2 + 2y_i \langle (x_i, 1), (w_j, b_j) \rangle + \|(x_i, 1)\|^2 \\ &\leq \|(w_j, b_j)\|^2 + \|(x_i, 1)\|^2 \\ &\leq j(R^2 + 1).\end{aligned}$$

Step 4: Combination of first three steps

$$j\rho \leq \sqrt{1 + (b^*)^2} \|(w_{j+1}, b_{j+1})\| \leq \sqrt{j(R^2 + 1)((b^*)^2 + 1)}$$

Solving for j proves the theorem.

Solutions of the Perceptron



Interpretation

Learning Algorithm

We perform an update only if we make a mistake.

Convergence Bound

- Bounds the maximum number of mistakes **in total**. We will make at most $(b^{*2} + 1)(R^1 + 1)/\rho^2$ mistakes in the case where a "correct" solution w^*, b^* exists.
- This also bounds the expected error (if we know ρ, R, and $|b^*|$).

Dimension Independent

Bound does not depend on the dimensionality of $\mathcal{X}$.

Sample Expansion

We obtain w as a **linear combination** of x_i.

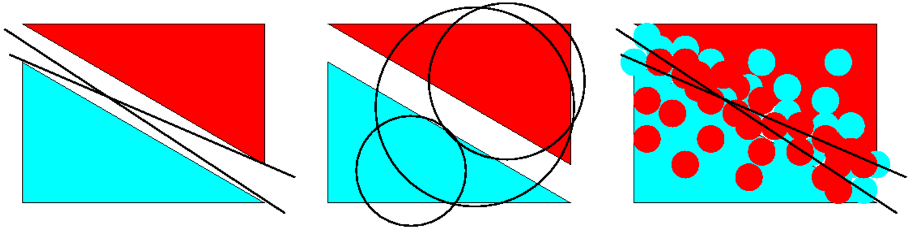
Realizable and Non-realizable Concepts

Realizable Concept

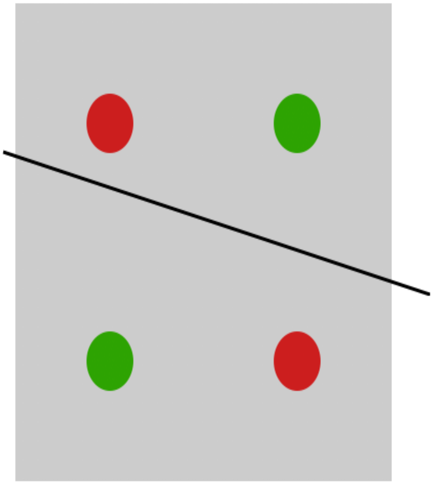
Here some w^*, b^* exists such that y is generated by $y = \operatorname{sgn}(\langle w^*, x\rangle + b)$. In general realizable means that the exact functional dependency is included in the class of admissible hypotheses.

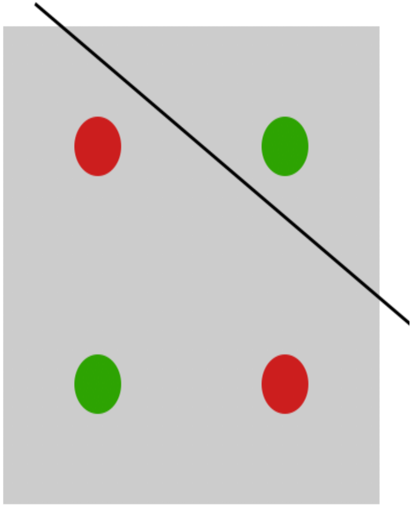
Unrealizable Concept

In this case, the exact concept does not exist or it is not included in the function class.



The XOR Problem





Mini Summary

Perceptron

- Separating halfspaces
- Perceptron algorithm
- Convergence theorem
- Only depends on margin, dimension independent

Pseudocode

```
for i in range(m):
    ytest = numpy.dot(w, x[:,i]) + b
    if ytest * y[i] <= 0:
        w += y[i] * x[:,i]
        b += y[i]
```

Nonlinearity via Preprocessing

Problem

Linear functions are often too simple to provide good estimators.

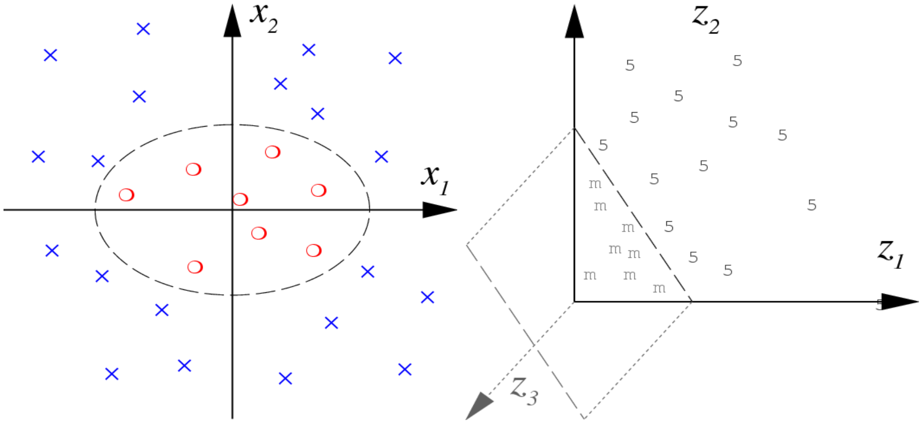
Idea

- Map to a higher dimensional feature space via $\Phi : x \to \Phi(x)$ and solve the problem there.
- Replace every $\langle x, x' \rangle$ by $\langle \Phi(x), \Phi(x') \rangle$ in the perceptron algorithm.

Consequence

- We have nonlinear classifiers.
- Solution lies in the choice of features $\Phi(x)$.

Nonlinearity via Preprocessing



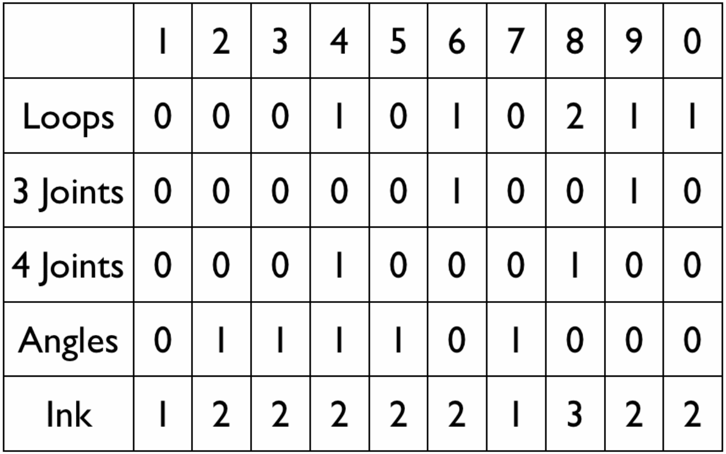
Features

Quadratic features correspond to circles, hyperbolas and ellipsoids as separating surfaces.

Constructing Features

Idea

Construct features manually. E.g. for OCR we could use



	1	2	3	4	5	6	7	8	9	0
Loops	0	0	0	1	0	1	0	2	1	1
3 Joints	0	0	0	0	0	1	0	0	1	0
4 Joints	0	0	0	1	0	0	0	1	0	0
Angles	0	1	1	1	1	0	1	0	0	0
Ink	1	2	2	2	2	2	1	3	2	2

More Examples

Two Interlocking Spirals

If we transform the data (x_1, x_2) into a radial part $(r = \sqrt{x_1^2 + x_2^2})$ and an angular part $(x_1 = r \cos \phi,$ $x_1 = r \sin \phi)$, the problem becomes much easier to solve (we only have to distinguish different stripes).

Japanese Character Recognition

Break down the images into strokes and recognize it from the latter (there's a predefined order of them).

Medical Diagnosis

Include physician's comments, knowledge about unhealthy combinations, features in EEG, . . .

Suitable Rescaling

If we observe, say the weight and the height of a person, rescale to zero mean and unit variance.

Perceptron on Features

```
argument: $X := \{x_1, \ldots, x_m\} \subset \mathcal{X}$ (data)
          $Y := \{y_1, \ldots, y_m\} \subset \{\pm 1\}$ (labels)
function $(w, b) = \mathrm{Perceptron}(X, Y, \eta)$
    initialize $w, b = 0$
    repeat
        Pick $(x_i, y_i)$ from data
        if $y_i(w \cdot \Phi(x_i) + b) \leq 0$ then
            $w' = w + y_i \Phi(x_i)$
            $b' = b + y_i$
    until $y_i(w \cdot \Phi(x_i) + b) > 0$ for all $i$
end
```

Important detail

$$w = \sum_j y_j \Phi(x_j) \text{ and hence } f(x) = \sum_j y_j (\Phi(x_j) \cdot \Phi(x)) + b$$

Problems with Constructing Features

Problems

- Need to be an expert in the domain (e.g. Chinese characters).
- Features may not be robust (e.g. postman drops letter in dirt).
- Can be expensive to compute.

Solution

- Use shotgun approach.
- Compute many features and hope a good one is among them.
- Do this efficiently.

Polynomial Features

Quadratic Features in $\mathbb{R}^2$

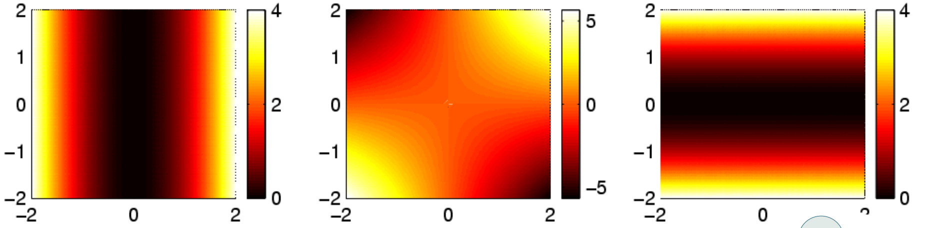
$$\Phi(x) := \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

Dot Product

$$\begin{aligned}\langle\Phi(x), \Phi(x')\rangle &= \left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right), \left({x_1'}^2, \sqrt{2}x_1'x_2', {x_2'}^2\right)\right\rangle \\ &= \langle x, x'\rangle^2.\end{aligned}$$

Insight

Trick works for any polynomials of order d via $\langle x, x'\rangle^d$.



Kernels

Problem

- Extracting features can sometimes be very costly.
- Example: second order features in 1000 dimensions. This leads to 5005 numbers. For higher order polynomial features much worse.

Solution

Don't compute the features, try to compute dot products implicitly. For some features this works …

Definition

A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric function in its arguments for which the following property holds

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \text{ for some feature map } \Phi.$$

If $k(x, x')$ is much cheaper to compute than $\Phi(x)$ …

Polynomial Kernels in $\mathbb{R}^n$

Idea

- We want to extend $k(x, x') = \langle x, x' \rangle^2$ to

$$k(x, x') = (\langle x, x' \rangle + c)^d \text{ where } c \geq 0 \text{ and } d \in \mathbb{N}.$$

- Prove that such a kernel corresponds to a dot product.

Proof strategy

Simple and straightforward: compute the explicit sum given by the kernel, i.e.

$$k(x, x') = (\langle x, x' \rangle + c)^d = \sum_{i=0}^{m} \binom{d}{i} (\langle x, x' \rangle)^i c^{d-i}$$

Individual terms $(\langle x, x' \rangle)^i$ are dot products for some $\Phi_i(x)$.

Kernel Perceptron

```
argument:  $X := \{x_1, \ldots, x_m\} \subset \mathcal{X}$ (data)
           $Y := \{y_1, \ldots, y_m\} \subset \{\pm 1\}$ (labels)
function $f = \mathrm{Perceptron}(X, Y, \eta)$
    initialize $f = 0$
    repeat
        Pick $(x_i, y_i)$ from data
        if $y_i f(x_i) \leq 0$ then
            $f(\cdot) \leftarrow f(\cdot) + y_i k(x_i, \cdot) + y_i$
    until $y_i f(x_i) > 0$ for all $i$
end
```

Important detail

$$w = \sum_j y_j \Phi(x_j) \text{ and hence } f(x) = \textstyle\sum_j y_j k(x_j, x) + b.$$

Are all $k(x, x')$ good Kernels?

Computability

We have to be able to compute $k(x, x')$ efficiently (much cheaper than dot products themselves).

"Nice and Useful" Functions

The features themselves have to be useful for the learning problem at hand. Quite often this means smooth functions.

Symmetry

Obviously $k(x, x') = k(x', x)$ due to the symmetry of the dot product $\langle \Phi(x), \Phi(x') \rangle = \langle \Phi(x'), \Phi(x) \rangle$.

Dot Product in Feature Space

Is there always a Φ such that k really is a dot product?

Mercer's Theorem

The Theorem

For any symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which is square integrable in $\mathcal{X} \times \mathcal{X}$ and which satisfies

$$\int_{\mathcal{X} \times \mathcal{X}} k(x, x') f(x) f(x') dx dx' \geq 0 \text{ for all } f \in L_2(\mathcal{X})$$

there exist $\phi_i : \mathcal{X} \to \mathbb{R}$ and numbers $\lambda_i \geq 0$ where

$$k(x, x') = \sum_i \lambda_i \phi_i(x) \phi_i(x') \text{ for all } x, x' \in \mathcal{X}.$$

Interpretation

Double integral is continuous version of vector-matrix-vector multiplication. For positive semidefinite matrices

$$\sum_i \sum_j k(x_i, x_j) \alpha_i \alpha_j \geq 0$$

Properties of the Kernel

Distance in Feature Space

Distance between points in feature space via

$$\begin{aligned} d(x,x')^2 :=& \|\Phi(x)-\Phi(x')\|^2 \\ =& \langle \Phi(x),\Phi(x)\rangle - 2\langle \Phi(x),\Phi(x')\rangle + \langle \Phi(x'),\Phi(x')\rangle \\ =& k(x,x) - 2k(x,x') + k(x',x') \end{aligned}$$

Kernel Matrix

To compare observations we compute dot products, so we study the matrix K given by

$$K_{ij} = \langle \Phi(x_i),\Phi(x_j)\rangle = k(x_i,x_j)$$

where x_i are the training patterns.

Similarity Measure

The entries K_{ij} tell us the overlap between $\Phi(x_i)$ and $\Phi(x_j)$, so $k(x_i,x_j)$ is a similarity measure.

Properties of the Kernel Matrix

K is Positive Semidefinite

Claim: $\alpha^\top K \alpha \geq 0$ for all $\alpha \in \mathbb{R}^m$ and all kernel matrices $K \in \mathbb{R}^{m \times m}$. Proof:

$$\begin{aligned} \sum_{i,j}^{m} \alpha_i \alpha_j K_{ij} &= \sum_{i,j}^{m} \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j) \rangle \\ &= \left\langle \sum_{i}^{m} \alpha_i \Phi(x_i), \sum_{j}^{m} \alpha_j \Phi(x_j) \right\rangle = \left\| \sum_{i=1}^{m} \alpha_i \Phi(x_i) \right\|^2 \end{aligned}$$

Kernel Expansion

If w is given by a linear combination of $\Phi(x_i)$ we get

$$\langle w, \Phi(x) \rangle = \left\langle \sum_{i=1}^{m} \alpha_i \Phi(x_i), \Phi(x) \right\rangle = \sum_{i=1}^{m} \alpha_i k(x_i, x).$$

A Counterexample

A Candidate for a Kernel

$$k(x, x') = \begin{cases} 1 & \text{if } \|x - x'\| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

This is symmetric and gives us some information about the proximity of points, yet it is not a proper kernel . . .

Kernel Matrix

We use three points, $x_1 = 1, x_2 = 2, x_3 = 3$ and compute the resulting “kernelmatrix” K. This yields

$$K = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \text{ and eigenvalues } (\sqrt{2}-1)^{-1}, 1 \text{ and } (1-\sqrt{2}).$$

as eigensystem. Hence k is not a kernel.

Some Good Kernels

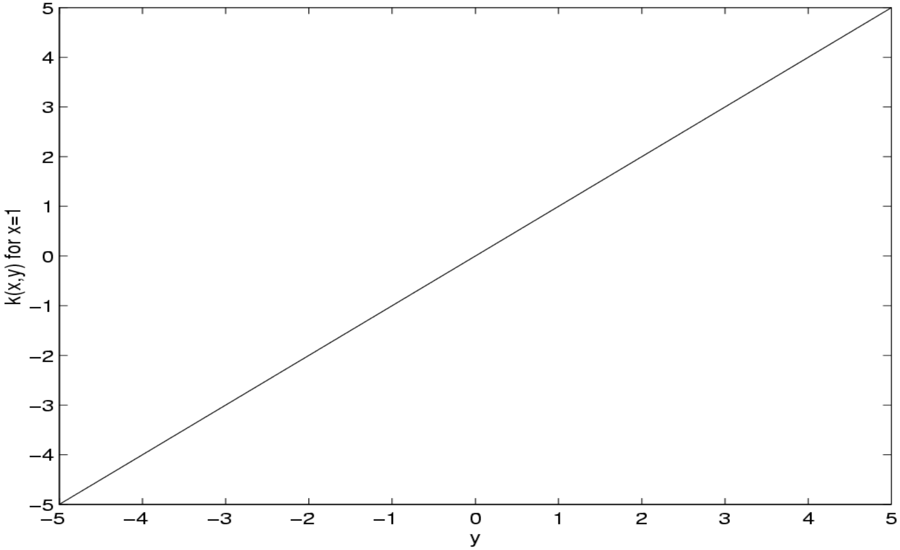
Examples of kernels $k(x, x')$

Linear	$\langle x, x' \rangle$
Laplacian RBF	$\exp(-\lambda \|x - x'\|)$
Gaussian RBF	$\exp\left(-\lambda \|x - x'\|^2\right)$
Polynomial	$(\langle x, x' \rangle + c \rangle)^d, c \geq 0, \; d \in \mathbb{N}$
B-Spline	$B_{2n+1}(x - x')$
Cond. Expectation	$\mathbf{E}_c[p(x\|c)p(x'\|c)]$

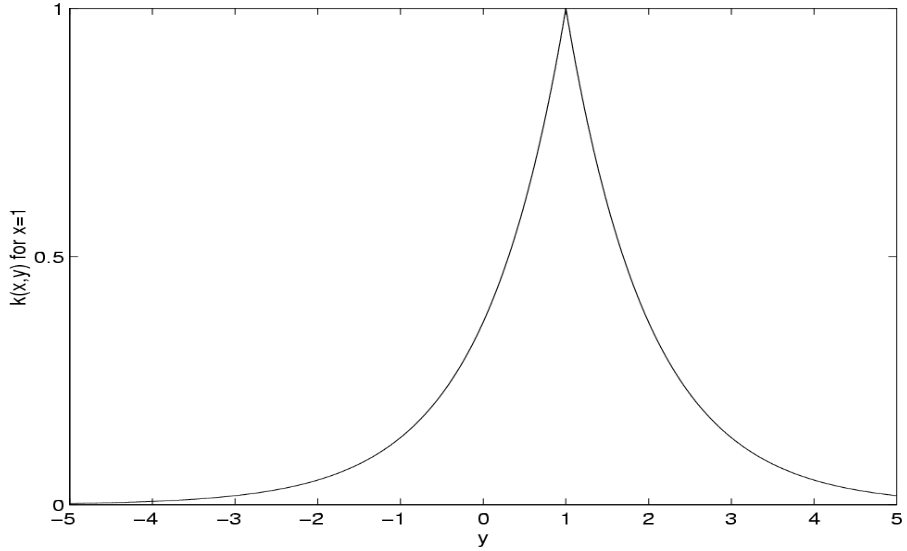
Simple trick for checking Mercer's condition

Compute the Fourier transform of the kernel and check that it is nonnegative.

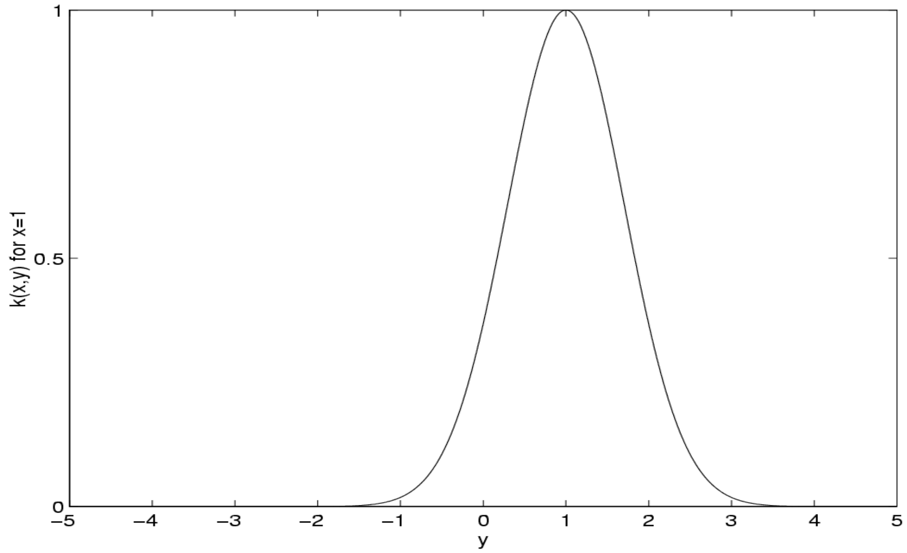
Linear Kernel



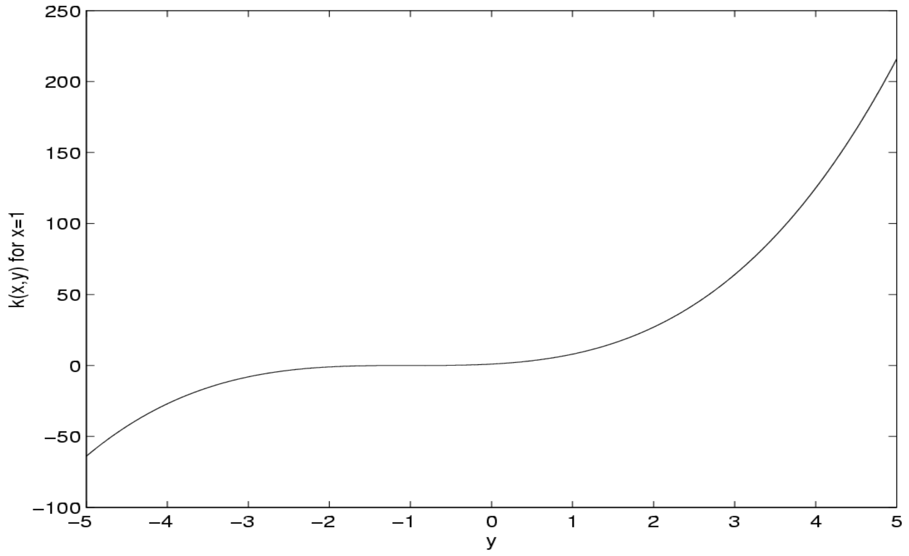
Laplacian Kernel



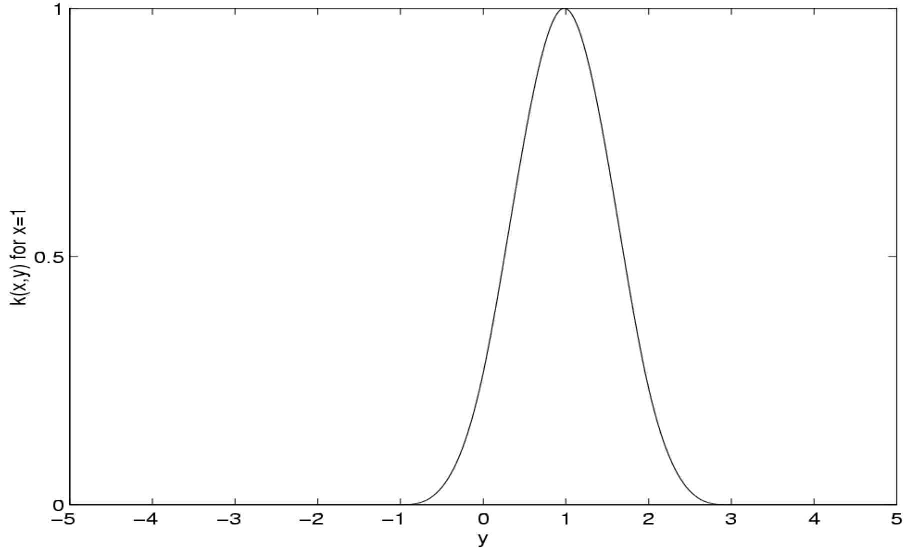
Gaussian Kernel



Polynomial (Order 3)



B_3-Spline Kernel



Mini Summary

Features

- Prior knowledge, expert knowledge
- Shotgun approach (polynomial features)
- Kernel trick $k(x, x') = \langle \phi(x), \phi(x') \rangle$
- Mercer's theorem

Applications

- Kernel Perceptron
- Nonlinear algorithm automatically by query-replace

Examples of Kernels

- Gaussian RBF
- Polynomial kernels

Summary

Hebb's rule

- positive feedback
- perceptron convergence rule, kernel perceptron

Features

- Explicit feature construction
- Implicit features via kernels

Kernels

- Examples
- Mercer's theorem

An Introduction to Machine Learning

L4: Support Vector Classification

Alexander J. Smola

Statistical Machine Learning Program
Canberra, ACT 0200 Australia
Alex.Smola@nicta.com.au

Machine Learning Summer School 2008

L4 Support Vector Classification

Support Vector Machine

- Problem definition
- Geometrical picture
- Optimization problem

Optimization Problem

- Hard margin
- Convexity
- Dual problem
- Soft margin problem

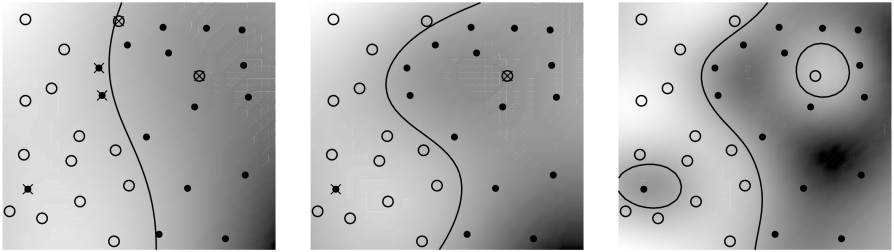
Classification

Data

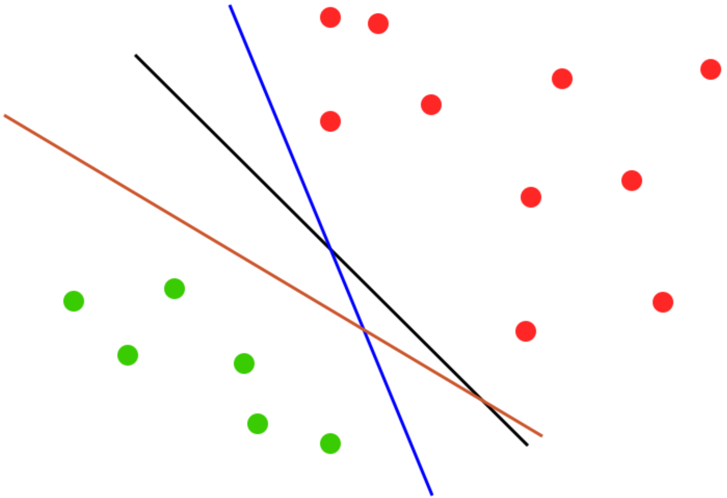
Pairs of observations (x_i, y_i) generated from some distribution $\mathrm{P}(x, y)$, e.g., (blood status, cancer), (credit transaction, fraud), (profile of jet engine, defect)

Task

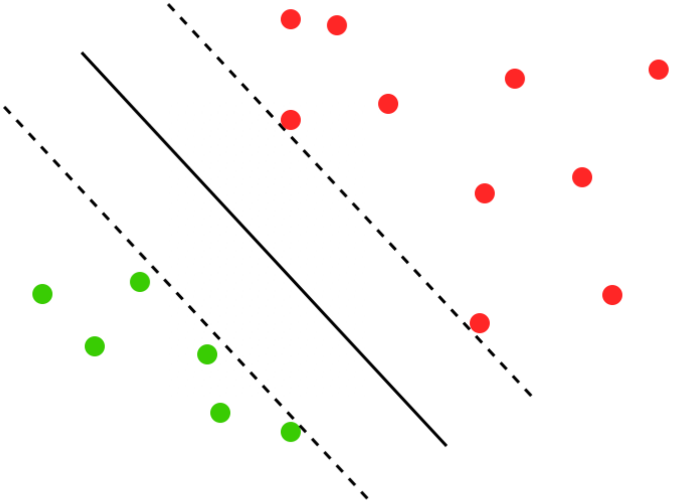
- Estimate y given x at a new location.
- Modification: find a function $f(x)$ that does the task.



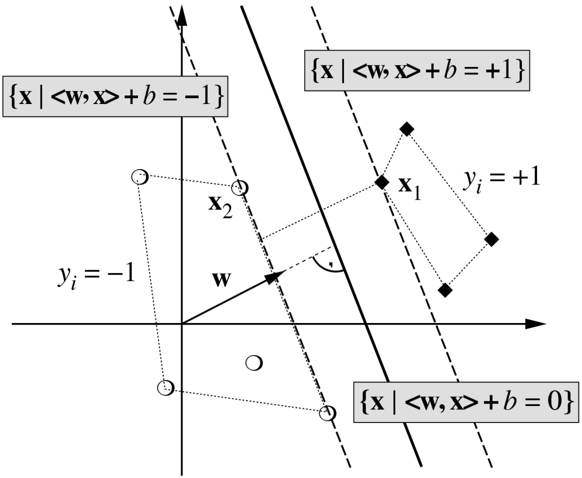
So Many Solutions



One to rule them all . . .



Optimal Separating Hyperplane



$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = -1\}$
$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = +1\}$
$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$
$\mathbf{x}_1$
$\mathbf{x}_2$
$y_i = +1$
$y_i = -1$
$\mathbf{w}$

Note:

$$\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = +1$$
$$\langle \mathbf{w}, \mathbf{x}_2 \rangle + b = -1$$

$$\Rightarrow \quad \langle \mathbf{w}, (\mathbf{x}_1 - \mathbf{x}_2) \rangle = 2$$

$$\Rightarrow \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}_1 - \mathbf{x}_2) \right\rangle = \frac{2}{\|\mathbf{w}\|}$$

Optimization Problem

Margin to Norm

- Separation of sets is given by $\frac{2}{\|w\|}$ so maximize that.
- Equivalently minimize $\frac{1}{2}\|w\|$.
- Equivalently minimize $\frac{1}{2}\|w\|^2$.

Constraints

- Separation with margin, i.e.

$$\begin{aligned} \langle w, x_i \rangle + b &\geq 1 && \text{if } y_i = 1 \\ \langle w, x_i \rangle + b &\leq -1 && \text{if } y_i = -1 \end{aligned}$$

- Equivalent constraint

$$y_i(\langle w, x_i \rangle + b) \geq 1$$

Optimization Problem

Mathematical Programming Setting

Combining the above requirements we obtain

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 \\ &\text{subject to} && y_i(\langle w, x_i\rangle + b) - 1 \geq 0 \text{ for all } 1 \leq i \leq m \end{aligned}$$

Properties

- Problem is convex
- Hence it has unique minimum
- Efficient algorithms for solving it exist

Lagrange Function

Objective Function $\frac{1}{2}\|w\|^2$.

Constraints $c_i(w, b) := 1 - y_i(\langle w, x_i \rangle + b) \leq 0$

Lagrange Function

$$\begin{aligned} L(w, b, \alpha) &= \text{PrimalObjective} + \sum_i \alpha_i c_i \\ &= \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\langle w, x_i \rangle + b)) \end{aligned}$$

Saddle Point Condition

Derivatives of L with respect to w and b must vanish.

Support Vector Machines

Optimization Problem

$$\text{minimize } \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle - \sum_{i=1}^{m}\alpha_i$$

$$\text{subject to } \sum_{i=1}^{m}\alpha_i y_i = 0 \text{ and } \alpha_i \geq 0$$

Support Vector Expansion

$$w = \sum_i \alpha_i y_i x_i \text{ and hence } f(x) = \sum_{i=1}^{m}\alpha_i y_i \langle x_i, x\rangle + b$$

Kuhn Tucker Conditions

$$\alpha_i(1 - y_i(\langle x_i, x\rangle + b)) = 0$$

Proof (optional)

Lagrange Function

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\langle w, x_i \rangle + b))$$

Saddlepoint condition

$$\partial_w L(w, b, \alpha) = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \iff w = \sum_{i=1}^{m} \alpha_i y_i x_i$$

$$\partial_b L(w, b, \alpha) = -\sum_{i=1}^{m} \alpha_i y_i x_i = 0 \iff \sum_{i=1}^{m} \alpha_i y_i = 0$$

To obtain the dual optimization problem we have to substitute the values of w and b into L. Note that the dual variables α_i have the constraint $\alpha_i \geq 0$.

Proof (optional)

Dual Optimization Problem

After substituting in terms for b, w the Lagrange function becomes

$$-\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j\langle x_i, x_j\rangle + \sum_{i=1}^{m}\alpha_i$$

$$\text{subject to } \sum_{i=1}^{m}\alpha_i y_i = 0 \text{ and } \alpha_i \geq 0 \text{ for all } 1 \leq i \leq m$$

Practical Modification

Need to **maximize** dual objective function. Rewrite as

$$\text{minimize } \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j\langle x_i, x_j\rangle - \sum_{i=1}^{m}\alpha_i$$

subject to the above constraints.

Support Vector Expansion

Solution in $$w = \sum_{i=1}^{m} \alpha_i y_i x_i$$

- w is given by a linear combination of training patterns x_i. **Independent of the dimensionality of x.**
- w depends on the Lagrange multipliers α_i.

Kuhn-Tucker-Conditions

- At optimal solution $\text{Constraint} \cdot \text{Lagrange Multiplier} = 0$
- In our context this means

$$\alpha_i (1 - y_i(\langle w, x_i \rangle + b)) = 0.$$

Equivalently we have

$$\alpha_i \neq 0 \Longrightarrow y_i \left(\langle w, x_i \rangle + b\right) = 1$$

Only points at the decision boundary can contribute to the solution.

Mini Summary

Linear Classification

- Many solutions
- Optimal separating hyperplane
- Optimization problem

Support Vector Machines

- Quadratic problem
- Lagrange function
- Dual problem

Interpretation

- Dual variables and SVs
- SV expansion
- Hard margin and infinite weights

Kernels

Nonlinearity via Feature Maps

Replace x_i by $\Phi(x_i)$ in the optimization problem.

Equivalent optimization problem

$$\text{minimize } \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i\alpha_j y_i y_j k(x_i, x_j) - \sum_{i=1}^{m} \alpha_i$$

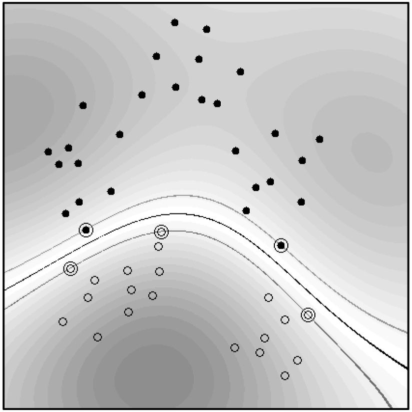
$$\text{subject to } \sum_{i=1}^{m} \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0$$

Decision Function

$$w = \sum_{i=1}^{m} \alpha_i y_i \Phi(x_i) \text{ implies}$$

$$f(x) = \langle w, \Phi(x)\rangle + b = \sum_{i=1}^{m} \alpha_i y_i k(x_i, x) + b.$$

Examples and Problems



Advantage

Works well when the data is noise free.

Problem

Already a single wrong observation can ruin everything — we require $y_i f(x_i) \geq 1$ for all i.

Idea

Limit the influence of individual observations by making the constraints less stringent (introduce slacks).

Optimization Problem (Soft Margin)

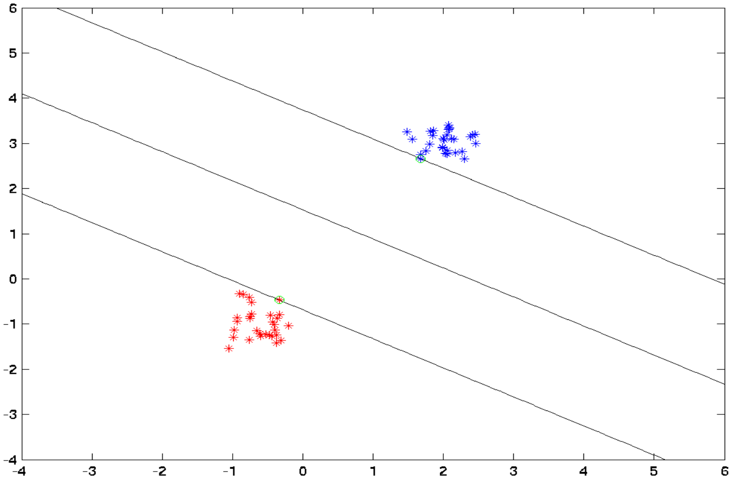
Recall: Hard Margin Problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 \\ \text{subject to} & y_i(\langle w, x_i\rangle + b) - 1 \geq 0 \end{array}$$

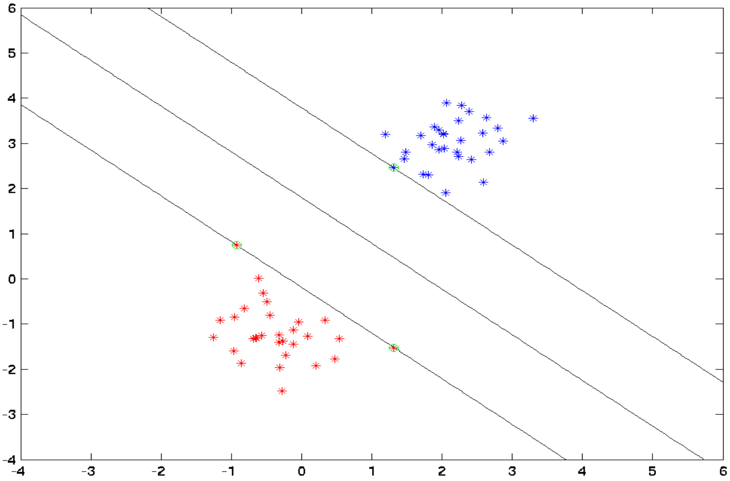
Softening the Constraints

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} & y_i(\langle w, x_i\rangle + b) - 1 + \xi_i \geq 0 \text{ and } \xi_i \geq 0 \end{array}$$

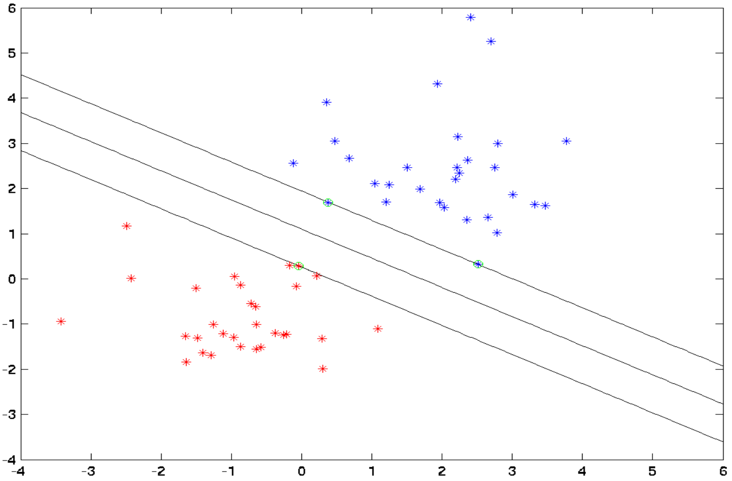
Linear SVM $C = 50$



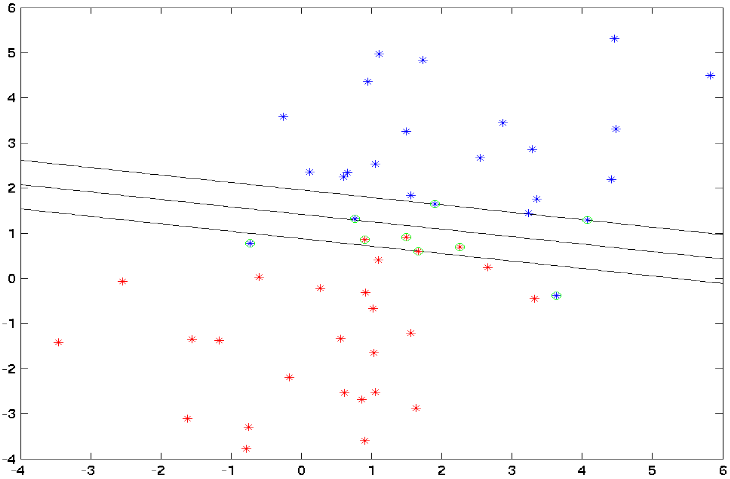
Linear SVM $C = 50$



Linear SVM $C = 50$



Linear SVM $C = 50$



Insights

Changing C

- For clean data C doesn't matter much.
- For noisy data, large C leads to narrow margin (SVM tries to do a good job at separating, even though it isn't possible)

Noisy data

- Clean data has few support vectors
- Noisy data leads to data in the margins
- More support vectors for noisy data

Python pseudocode

SVM Classification

```
import elefant.kernels.vector
# linear kernel
k = elefant.kernels.vector.CLinearKernel()
# Gaussian RBF kernel
k = elefant.kernels.vector.CGaussKernel(rbf)

import elefant.estimation.svm.svmclass as
svmclass
svm = svmclass.SVC(C, kernel=k)

alpha, b = svm.Train(x, y)
ytest = svm.Test(xtest)
```

Dual Optimization Problem

Optimization Problem

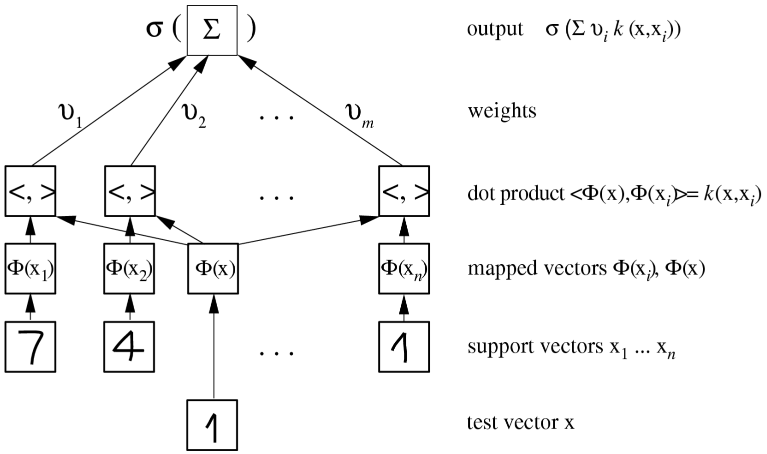
$$\text{minimize } \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i\alpha_j y_i y_j k(x_i, x_j) - \sum_{i=1}^{m} \alpha_i$$

$$\text{subject to } \sum_{i=1}^{m} \alpha_i y_i = 0 \text{ and } C \geq \alpha_i \geq 0 \text{ for all } 1 \leq i \leq m$$

Interpretation

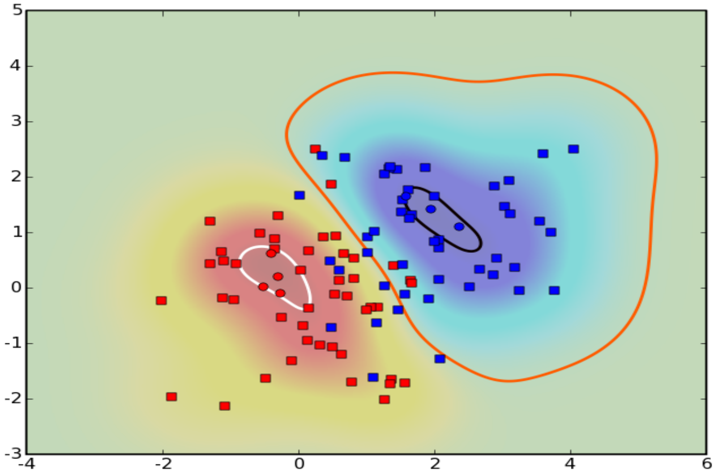
- Almost same optimization problem as before
- Constraint on weight of each α_i (bounds influence of pattern).
- Efficient solvers exist (more about that tomorrow).

SV Classification Machine

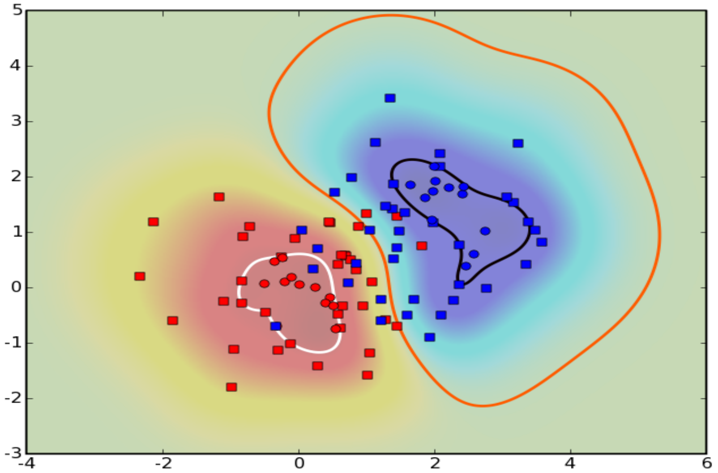


$\sigma(\Sigma)$
υ_1
υ_2
. . .
υ_m
<, >
$\Phi(x_1)$
$\Phi(x_2)$
$\Phi(x)$
$\Phi(x_n)$
output $\sigma(\Sigma\, \upsilon_i\, k(x,x_i))$
weights
dot product $<\Phi(x),\Phi(x_i)> = k(x,x_i)$
mapped vectors $\Phi(x_i)$, $\Phi(x)$
support vectors $x_1 \ldots x_n$
test vector x

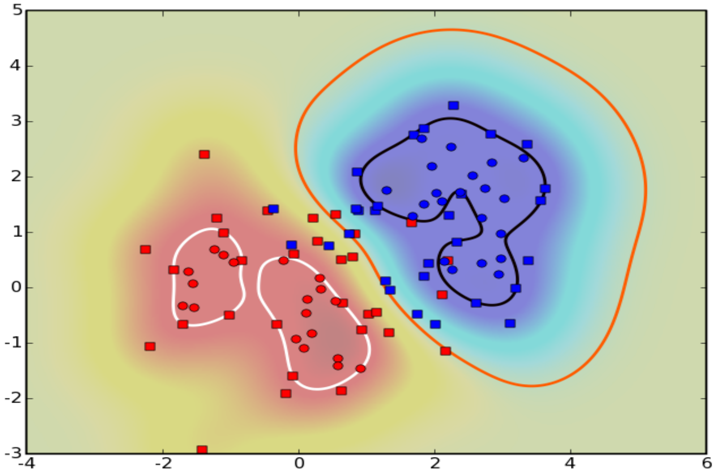
Gaussian RBF with $C = 0.1$



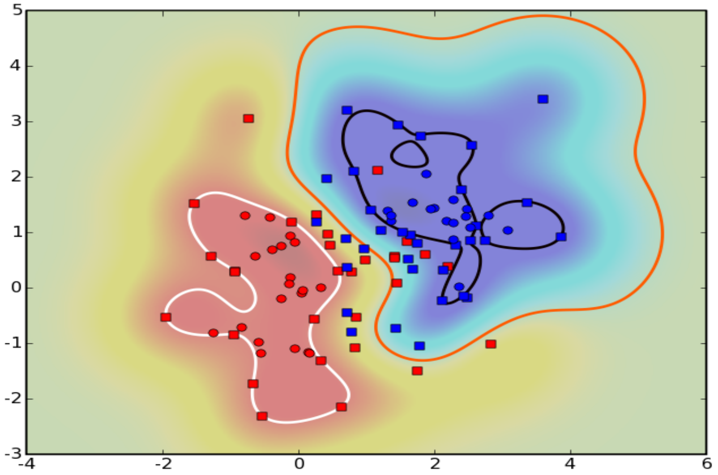
Gaussian RBF with $C = 0.2$



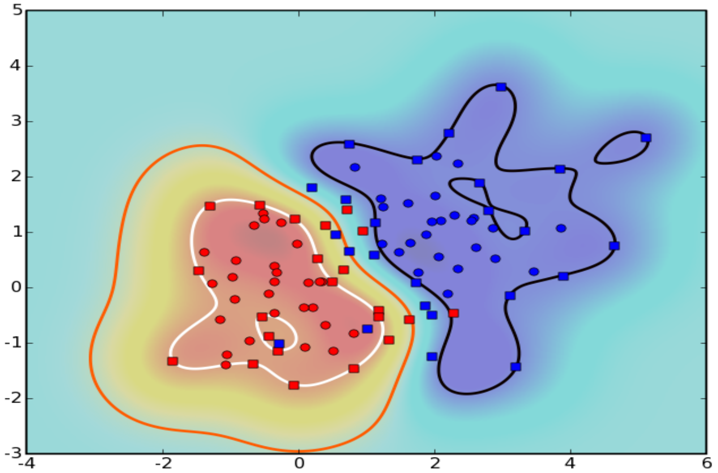
Gaussian RBF with $C = 0.4$



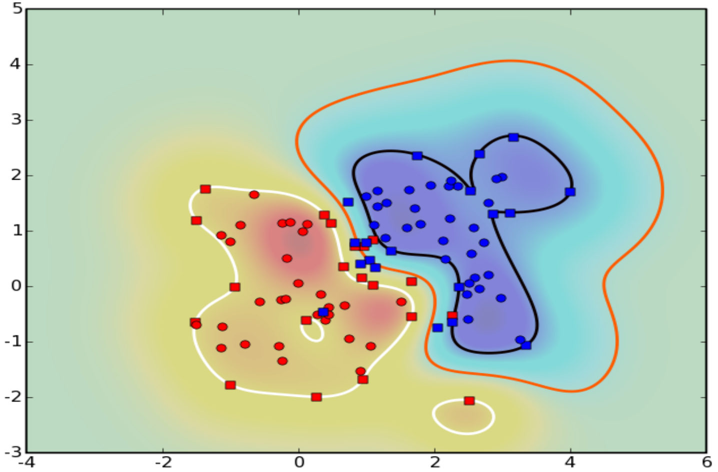
Gaussian RBF with $C = 0.8$



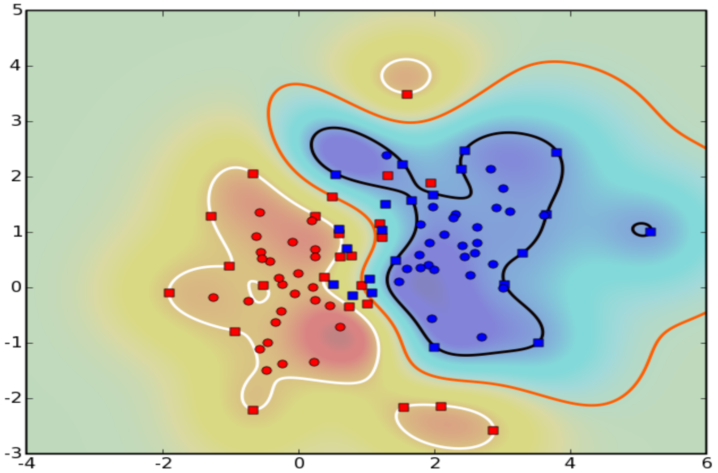
Gaussian RBF with $C = 1.6$



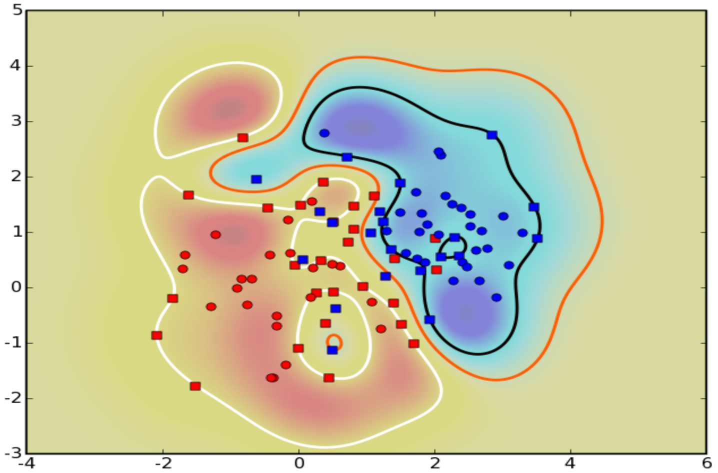
Gaussian RBF with $C = 3.2$



Gaussian RBF with $C = 6.4$



Gaussian RBF with $C = 12.8$



Insights

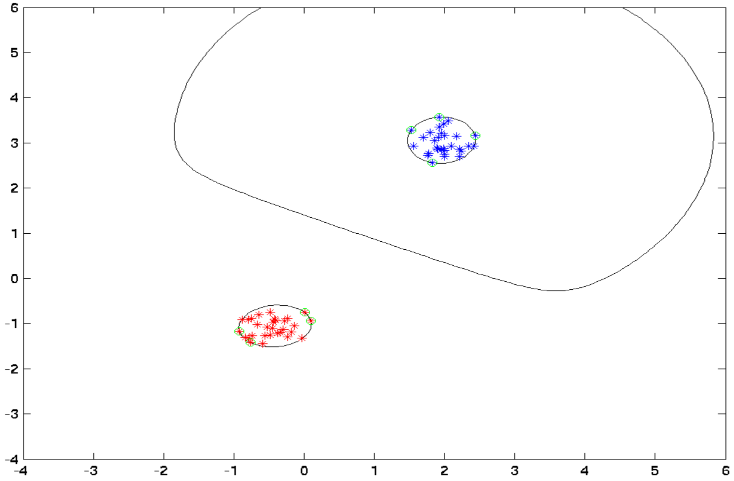
Changing C

- For clean data C doesn't matter much.
- For noisy data, large C leads to more complicated margin (SVM tries to do a good job at separating, even though it isn't possible)
- Overfitting for large C

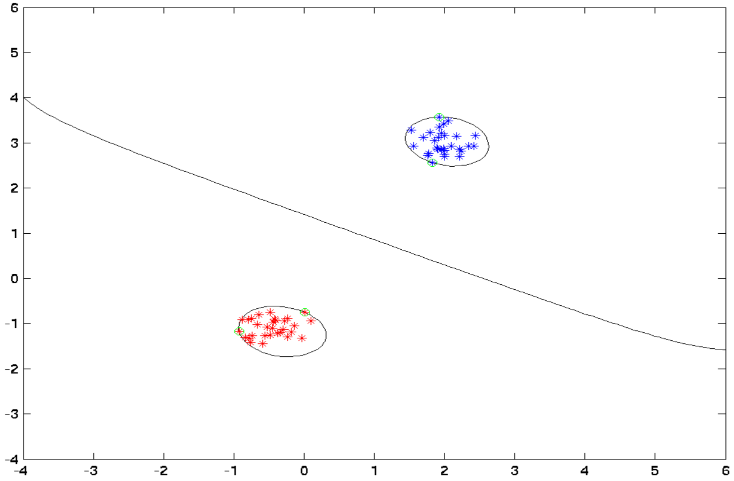
Noisy data

- Clean data has few support vectors
- Noisy data leads to data in the margins
- More support vectors for noisy data

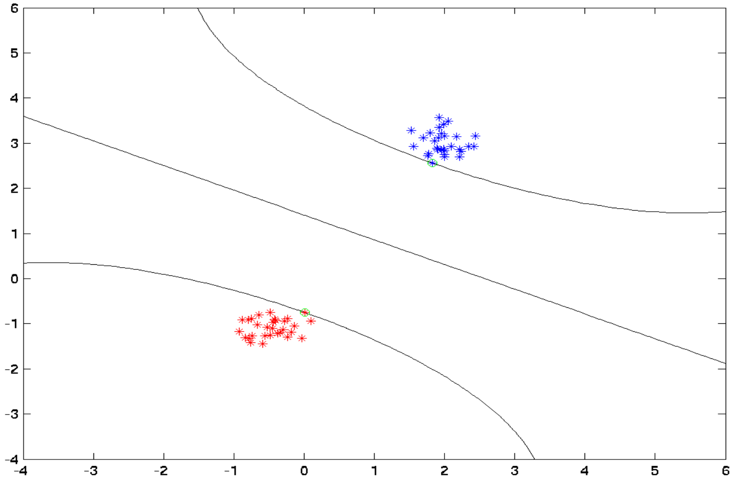
Gaussian RBF with $\sigma = 1$



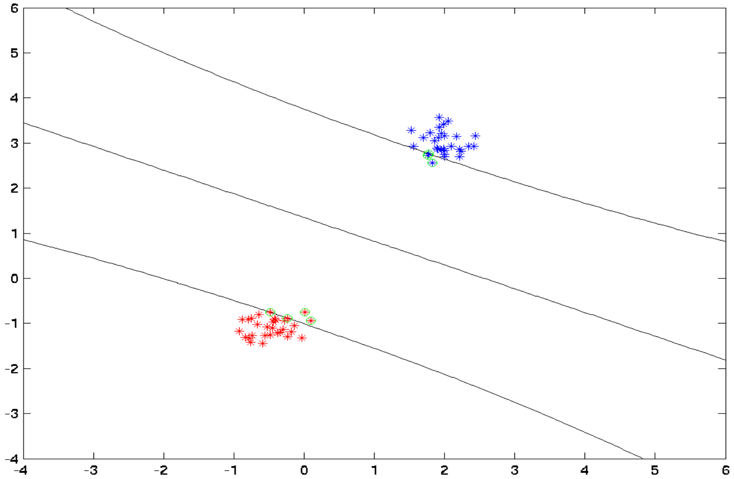
Gaussian RBF with $\sigma = 2$



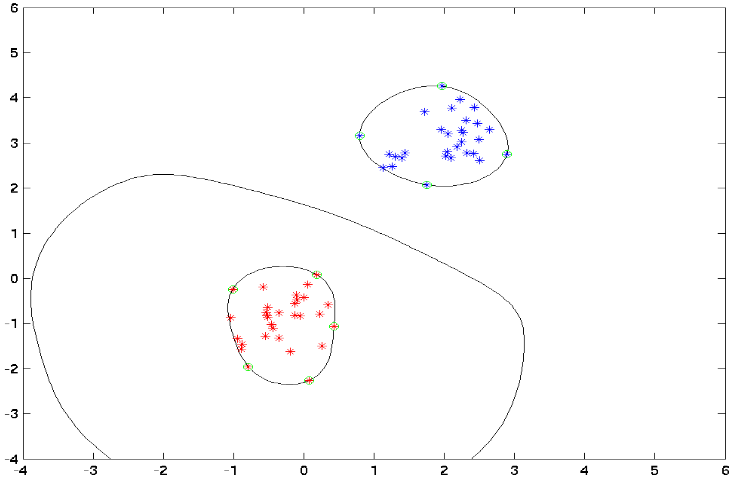
Gaussian RBF with $\sigma = 5$



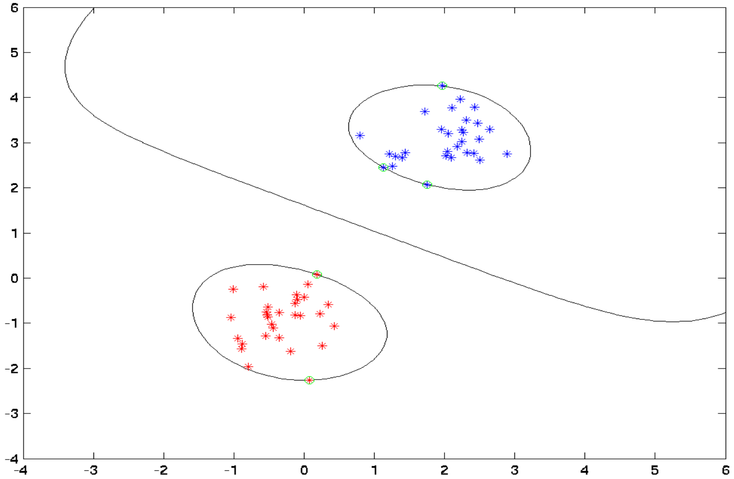
Gaussian RBF with $\sigma = 10$



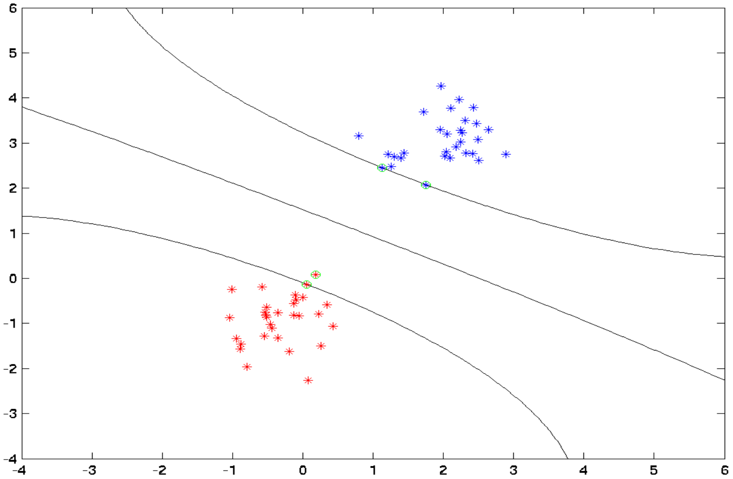
Gaussian RBF with $\sigma = 1$



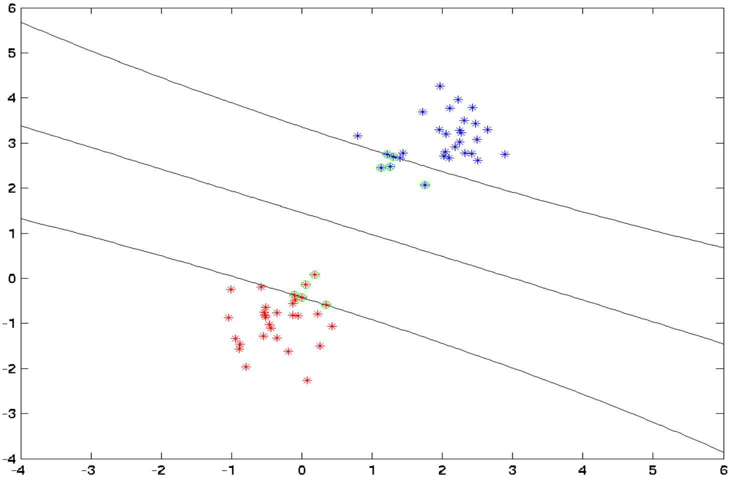
Gaussian RBF with $\sigma = 2$



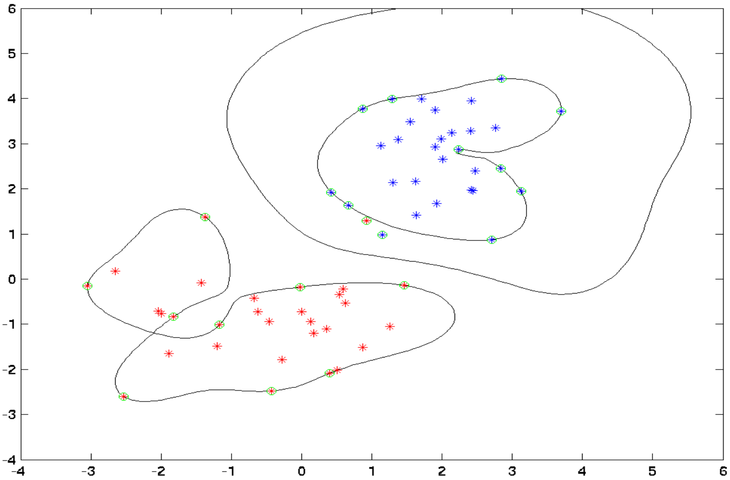
Gaussian RBF with $\sigma = 5$



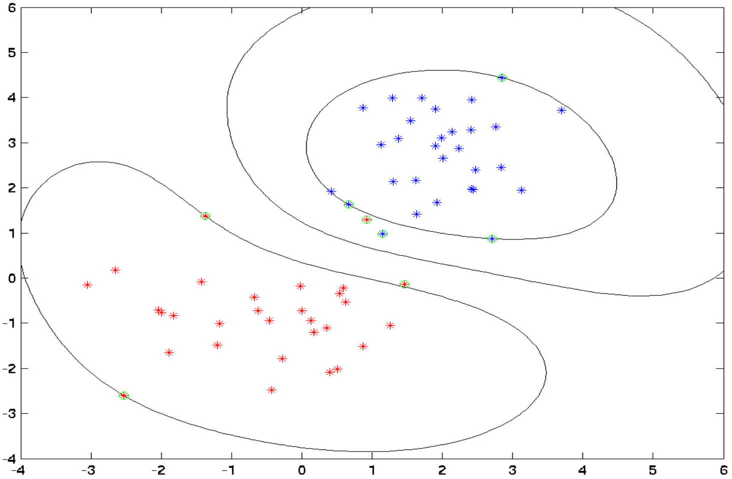
Gaussian RBF with $\sigma = 10$



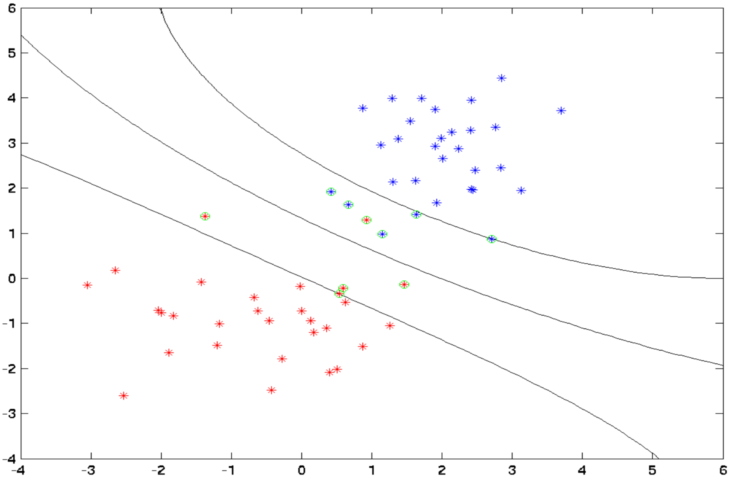
Gaussian RBF with $\sigma = 1$



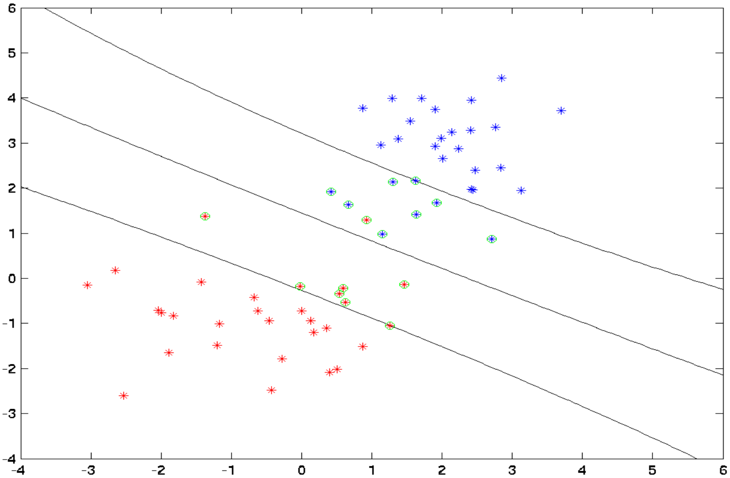
Gaussian RBF with $\sigma = 2$



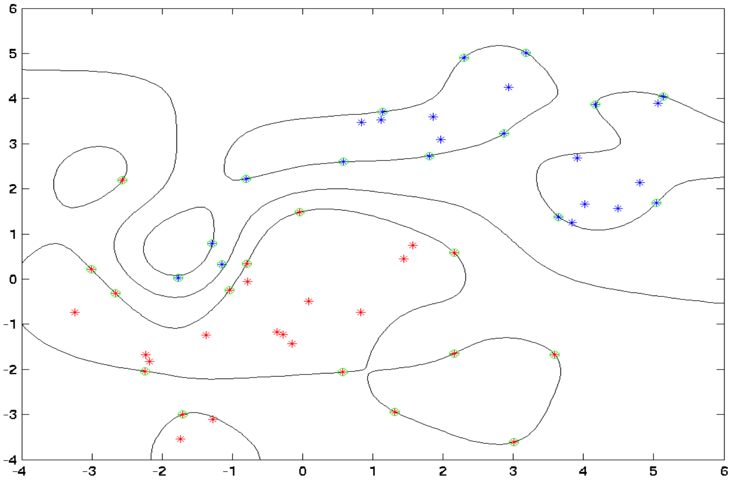
Gaussian RBF with $\sigma = 5$



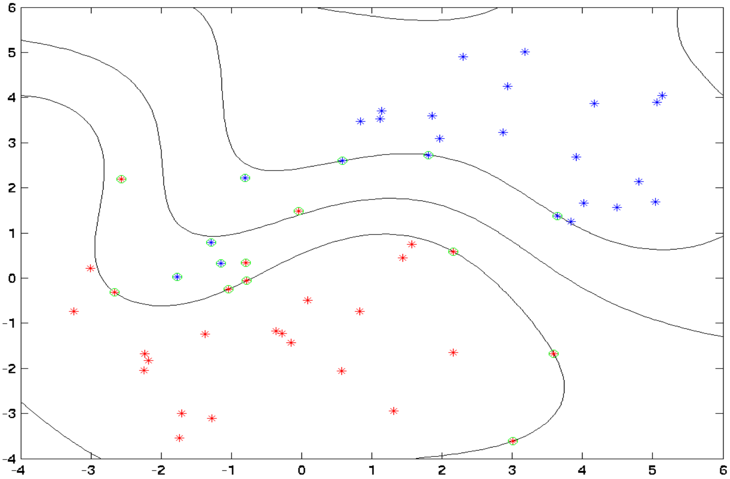
Gaussian RBF with $\sigma = 10$



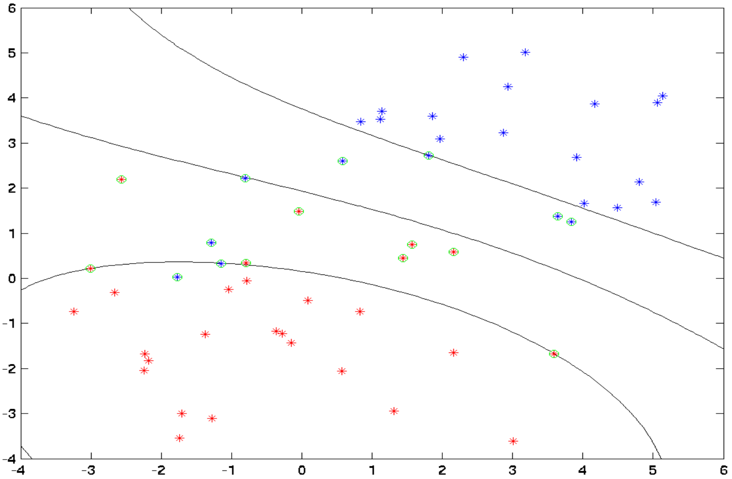
Gaussian RBF with $\sigma = 1$



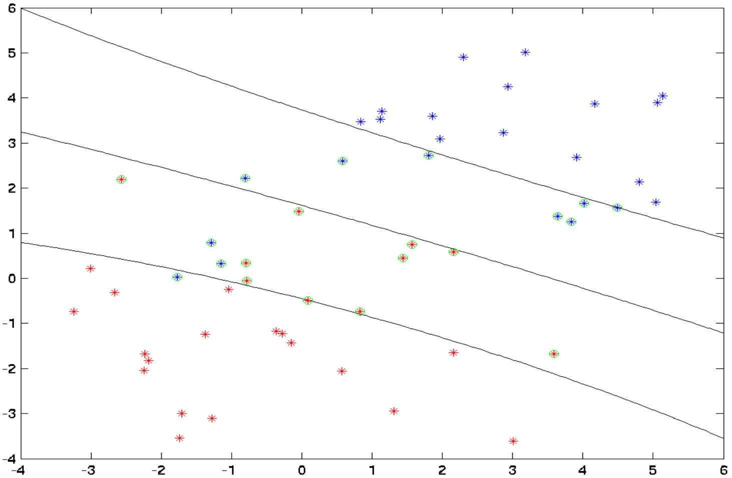
Gaussian RBF with $\sigma = 2$



Gaussian RBF with $\sigma = 5$



Gaussian RBF with $\sigma = 10$



Insights

Changing σ

- For clean data σ doesn't matter much.
- For noisy data, small σ leads to more complicated margin (SVM tries to do a good job at separating, even though it isn't possible)
- Lots of overfitting for small σ

Noisy data

- Clean data has few support vectors
- Noisy data leads to data in the margins
- More support vectors for noisy data

Summary

Support Vector Machine

- Problem definition
- Geometrical picture
- Optimization problem

Optimization Problem

- Hard margin
- Convexity
- Dual problem
- Soft margin problem

An Introduction to Machine Learning

L5: Novelty Detection and Regression

Alexander J. Smola

Statistical Machine Learning Program
Canberra, ACT 0200 Australia
Alex.Smola@nicta.com.au

Machine Learning Summer School 2008

L5 Novelty Detection and Regression

Novelty Detection

- Basic idea
- Optimization problem
- Stochastic Approximation
- Examples

Regression

- Additive noise
- Regularization
- Examples
- SVM Regression
- Quantile Regression

Resources

Books

- V. Vapnik, The Nature of Statistical Learning Theory, 1995
- V. Vapnik, Statistical Learning Theory, 1998
- N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines, 2000
- J. Shawe-Taylor and N. Cristianini, Kernel Methods for Pattern Analysis, 2004
- B. Schölkopf and A. J. Smola, Learning with Kernels, 2002
- R. Herbrich, Learning Kernel Classifiers: Theory and Algorithms, 2002

Web Resources

- Machine Learning Summer School
  http://www.mlss.cc
- Kernel Machines
  http://www.kernel-machines.org

Resources

Software

- SVMLight (T. Joachims, Cornell)
- LibSVM (C. Lin, NTU Taipei)
- SVMLin (V. Simdhani, U Chicago)
- SVMTorch (S. Bengio, Martigny)
- PLearn (P. Vincent, Montreal)
- Elefant (K. Gawande, NICTA)
- WEKA (Waikato)
- R (Vienna, other places)

More Course Material

- http://sml.nicta.com.au/$\sim$smola/

Conferences

Neural Information Processing Systems (NIPS)
Best ML conference, cutting edge, proof of concept!

International Conference on Machine Learning (ICML)
Solid machine learning work, less cutting edge, more detail.

Uncertainty in Artificial Intelligence (UAI)
Mainly graphical models and probabilistic reasoning.

Computational Learning Theory (COLT)
The main theory conference. Not applied!

Knowledge Discovery and Data Mining (KDD)
Data mining meets machine learning. Applications rule.

American Association on Artificial Intelligence (AAAI)
Classical AI conference. Markov models and graphical models.

Journals

Journal of Machine Learning Research (JMLR)
Prime ML Journal

Machine Learning Journal (MLJ)
Editorial from MLJ started JMLR . . .

IEEE Pattern Analysis and Machine Intelligence (PAMI)
Classical Pattern Recognition

IEEE Information Theory
Prime Theory Journal

Neural Computation
Neuroscience meets learning

Annals of Statistics
Prime Statistics Journal

Statistics and Computing
Algorithms

Novelty Detection

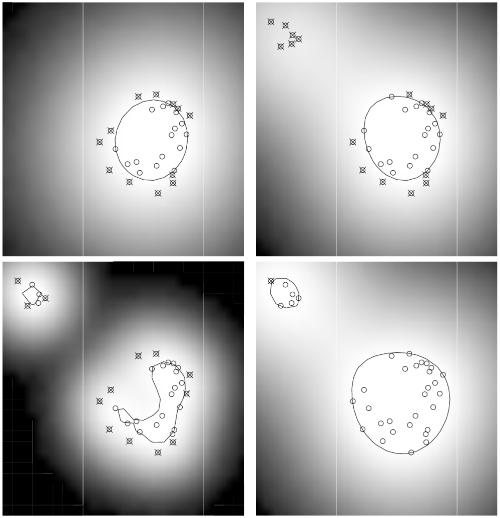
Data

Observations (x_i) generated from some $\mathrm{P}(x)$, e.g.,

- network usage patterns
- handwritten digits
- alarm sensors
- factory status

Task

Find unusual events, clean database, distinguish typical examples.



Applications

Network Intrusion Detection

Detect whether someone is trying to hack the network, downloading tons of MP3s, or doing anything else *unusual* on the network.

Jet Engine Failure Detection

You can't destroy jet engines just to see *how* they fail.

Database Cleaning

We want to find out whether someone stored bogus information in a database (typos, etc.), mislabelled digits, ugly digits, bad photographs in an electronic album.

Fraud Detection

Credit Cards, Telephone Bills, Medical Records

Self calibrating alarm devices

Car alarms (adjusts itself to where the car is parked), home alarm (furniture, temperature, windows, etc.)

Novelty Detection via Densities

Key Idea

- Novel data is one that we don't see frequently.
- It must lie in low density regions.

Step 1: Estimate density

- Observations $x_1, \ldots, x_m$
- Density estimate via Parzen windows

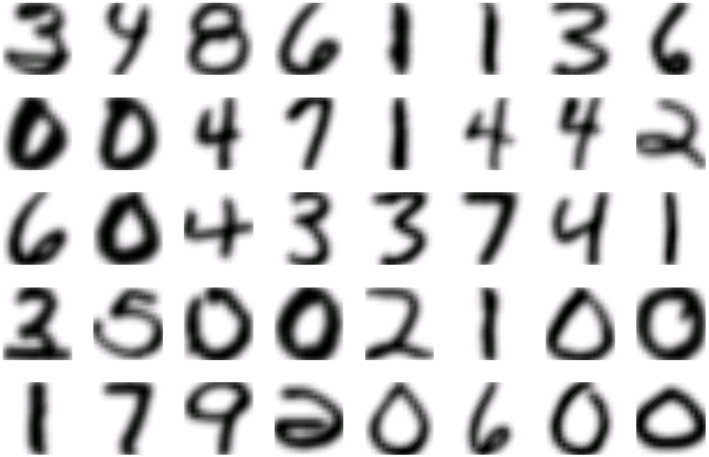
Step 2: Thresholding the density

- Sort data according to density and use it for rejection
- Practical implementation: compute

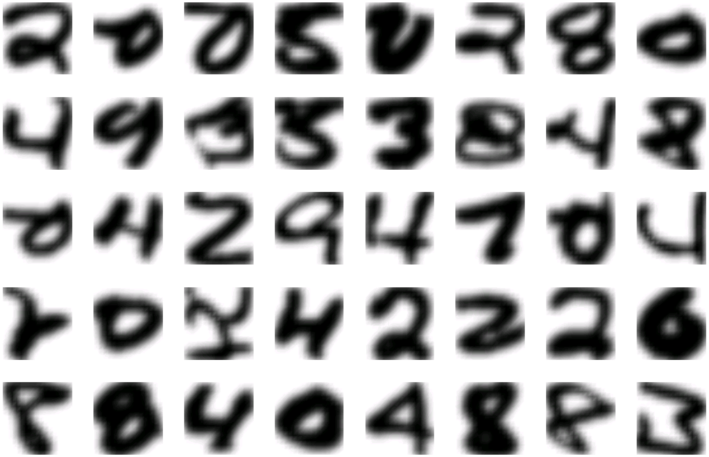
$$p(x_i) = \frac{1}{m} \sum_j k(x_i, x_j) \text{ for all } i$$

  and sort according to magnitude.
- Pick smallest $p(x_i)$ as novel points.

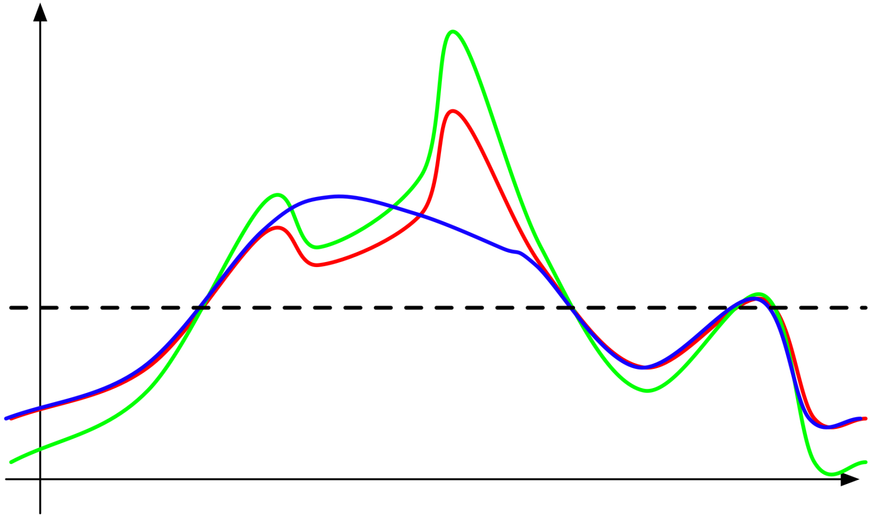
Typical Data



Outliers



A better way . . .



A better way . . .

Problems

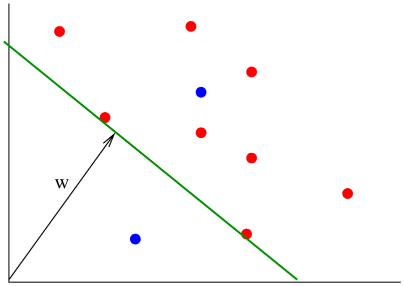
- We do not care about estimating the density properly in regions of high density (waste of capacity).
- We only care about the relative density for thresholding purposes.
- We want to eliminate a certain fraction of observations and tune our estimator specifically for this fraction.

Solution

- Areas of low density can be approximated as the **level set** of an auxiliary function. No need to estimate $p(x)$ directly — use proxy of $p(x)$.
- Specifically: find $f(x)$ such that x is novel if $f(x) \leq c$ where c is some constant, i.e. $f(x)$ describes the amount of novelty.

Maximum Distance Hyperplane

Idea Find hyperplane, given by $f(x) = \langle w, x \rangle + b = 0$ that has **maximum distance from origin** yet is still closer to the origin than the observations.



Hard Margin

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & \langle w, x_i \rangle \geq 1 \end{aligned}$$

Soft Margin

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & \langle w, x_i \rangle \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

The ν-Trick

Problem

- Depending on C, the number of novel points will vary.
- We would like to **specify the fraction** ν beforehand.

Solution

Use hyperplane separating data from the origin

$$H := \{x | \langle w, x \rangle = \rho\}$$

where the threshold ρ is **adaptive**.

Intuition

- Let the hyperplane shift by shifting ρ
- Adjust it such that the 'right' number of observations is considered novel.
- Do this automatically

The ν-Trick

Primal Problem

$$\text{minimize } \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \xi_i - m\nu\rho$$

$$\text{where } \langle w, x_i \rangle - \rho + \xi_i \geq 0$$

$$\xi_i \geq 0$$

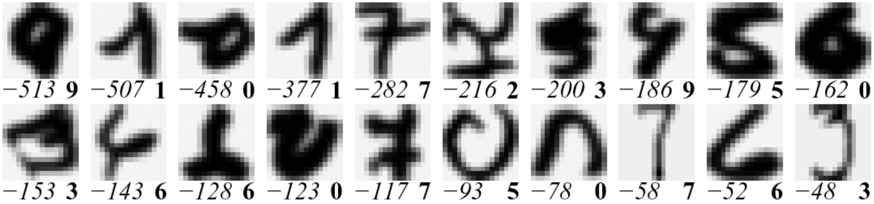
Dual Problem

$$\text{minimize } \frac{1}{2}\sum_{i=1}^{m} \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$\text{where } \alpha_i \in [0, 1] \text{ and } \sum_{i=1}^{m} \alpha_i = \nu m.$$

Similar to SV classification problem, use standard optimizer for it.

USPS Digits



- Better estimates since we only optimize in low density regions.
- Specifically tuned for small number of outliers.
- Only estimates of a level-set.
- For $\nu = 1$ we get the Parzen-windows estimator back.

A Simple Online Algorithm

Objective Function

$$\frac{1}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max(0, \rho - \langle w, \phi(x_i)\rangle) - \nu\rho$$

Stochastic Approximation

$$\frac{1}{2}\|w\|^2 \max(0, \rho - \langle w, \phi(x_i)\rangle) - \nu\rho$$

Gradient

$$\partial_w[\ldots] = \begin{cases} w - \phi(x_i) & \text{if } \langle w, \phi(x_i)\rangle < \rho \\ w & \text{otherwise} \end{cases}$$

$$\partial_\rho[\ldots] = \begin{cases} (1 - \nu) & \text{if } \langle w, \phi(x_i)\rangle < \rho \\ -\nu & \text{otherwise} \end{cases}$$

Practical Implementation

Update in coefficients

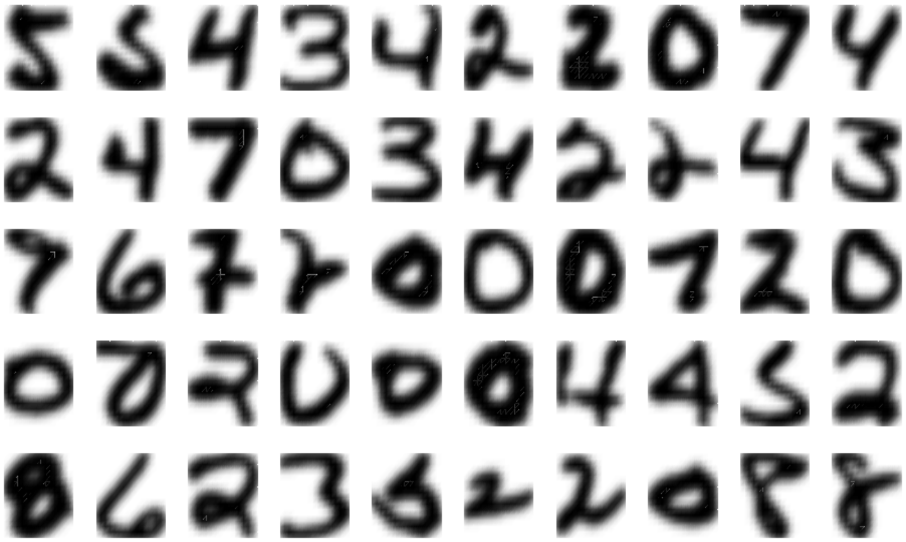
$$\alpha_j \leftarrow (1-\eta)\alpha_j \text{ for } j \neq i$$

$$\alpha_i \leftarrow \begin{cases} \eta_i & \text{if } \sum_{j=1}^{i-1} \alpha_i k(x_i, x_j) < \rho \\ 0 & \text{otherwise} \end{cases}$$

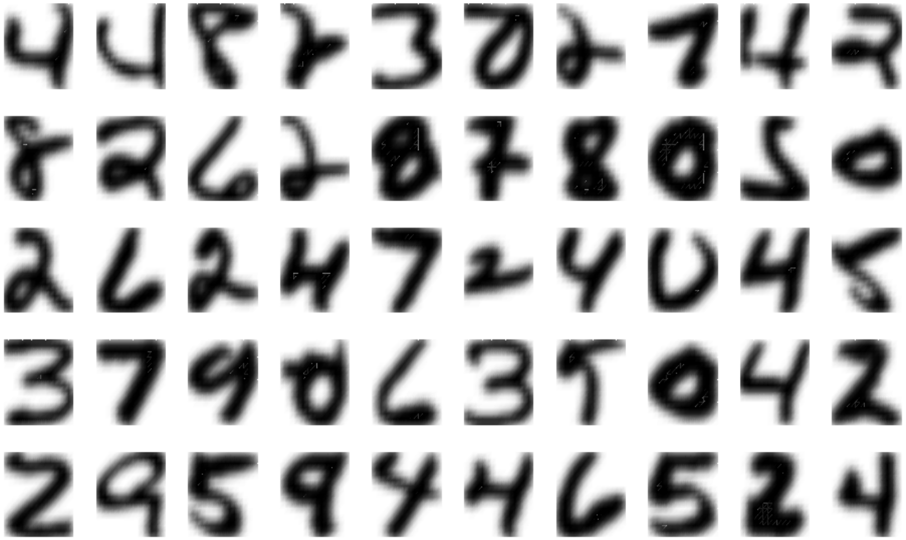
$$\rho = \begin{cases} \rho + \eta(\nu - 1) & \text{if } \sum_{j=1}^{i-1} \alpha_i k(x_i, x_j) < \rho \\ \rho + \eta\nu & \text{otherwise} \end{cases}$$

Using learning rate η.

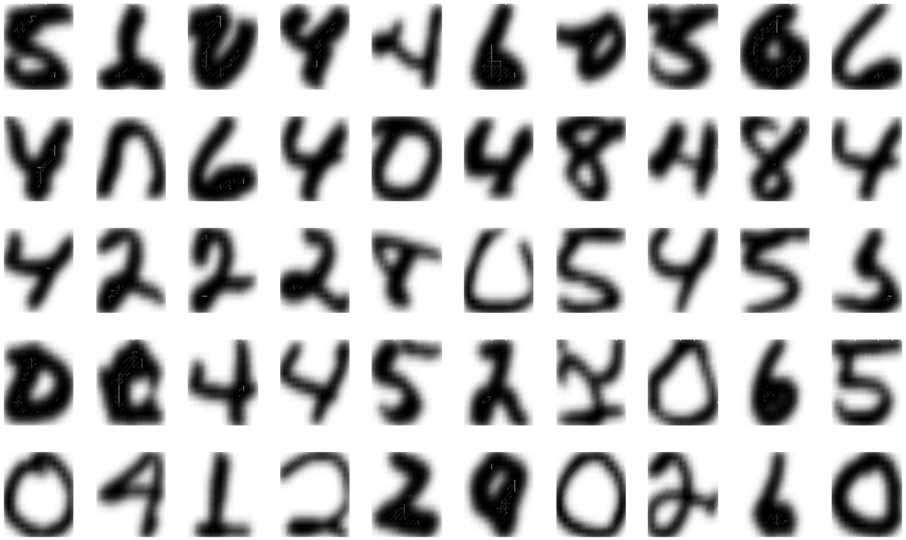
Online Training Run



Worst Training Examples



Worst Test Examples



Mini Summary

Novelty Detection via Density Estimation

- Estimate density e.g. via Parzen windows
- Threshold it at level and pick low-density regions as novel

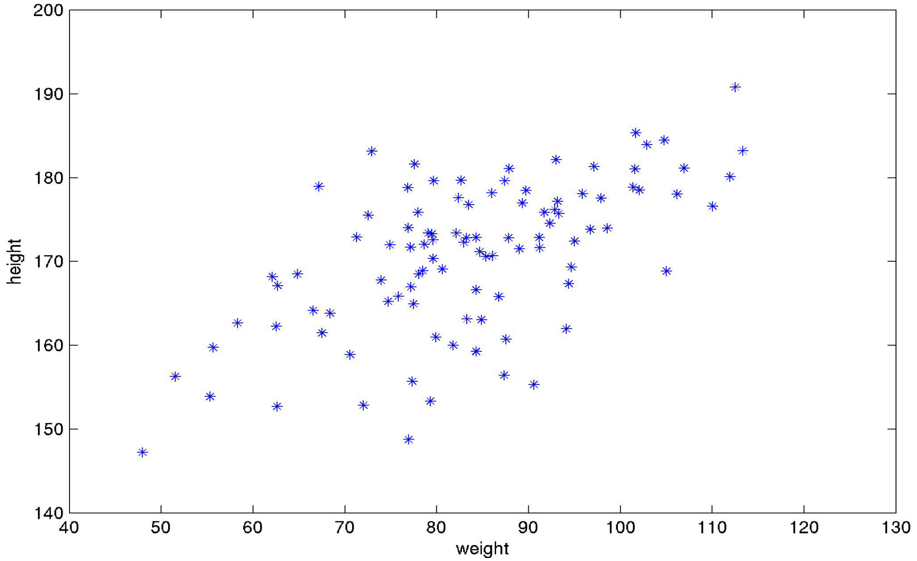
Novelty Detection via SVM

- Find halfspace bounding data
- Quadratic programming solution
- Use existing tools

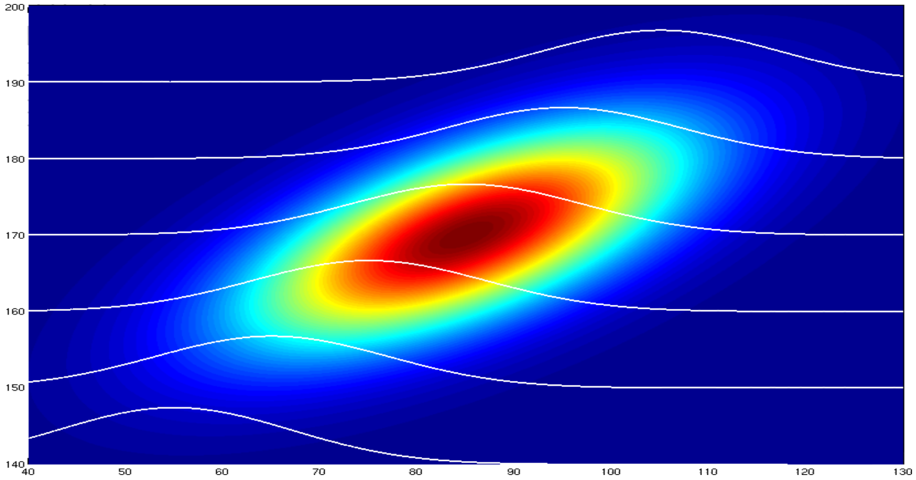
Online Version

- Stochastic gradient descent
- Simple update rule: keep data if novel, but only with fraction ν and adjust threshold.
- Easy to implement

A simple problem



Inference



$$p(\text{weight}|\text{height}) = \frac{p(\text{height}, \text{weight})}{p(\text{height})} \propto p(\text{height}, \text{weight})$$

Bayesian Inference HOWTO

Joint Probability

We have distribution over y and y', given training and test data x, x'.

Bayes Rule

This gives us the conditional probability via

$$p(y, y'|x, x') = p(y'|y, x, x')p(y|x) \text{ and hence}$$
$$p(y'|y) \propto p(y, y'|x, x') \text{ for fixed } y.$$

Normal Distribution in $\mathbb{R}^n$

Normal Distribution in $\mathbb{R}$

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

Normal Distribution in $\mathbb{R}^n$

$$p(x) = \frac{1}{\sqrt{(2\pi)^n \det \Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

Parameters

- $\mu \in \mathbb{R}^n$ is the mean.
- $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance **matrix**.
- Σ has only nonnegative eigenvalues:
  The variance is of a random variable is never negative.

Gaussian Process Inference

Our Model

We assume that all y_i are related, as given by some covariance matrix K. More specifically, we assume that $\mathrm{Cov}(y_i, y_j)$ is given by two terms:

- A general correlation term, parameterized by $k(x_i, x_j)$
- An additive noise term, parameterized by $\delta_{ij}\sigma^2$.

Practical Solution

Since $y'|y \sim \mathcal{N}(\tilde{\mu}, \tilde{K})$, we only need to collect all terms in $p(t, t')$ depending on t' by matrix inversion, hence

$$\tilde{K} = K_{y'y'} - K_{yy'}^{\top} K_{yy}^{-1} K_{yy'} \text{ and } \tilde{\mu} = \mu' + K_{yy'}^{\top} \underbrace{\left[K_{yy}^{-1}(y - \mu)\right]}_{\text{independent of } y'}$$

Key Insight

We can use this for regression of y' given y.

Some Covariance Functions

Observation

Any function k leading to a symmetric matrix with nonnegative eigenvalues is a valid covariance function.

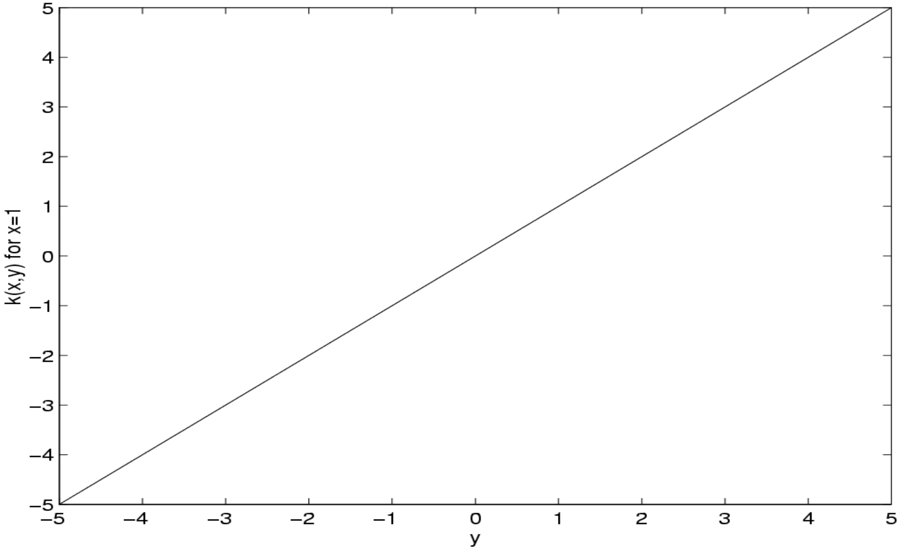
Necessary and sufficient condition (Mercer's Theorem)

k needs to be a nonnegative integral kernel.

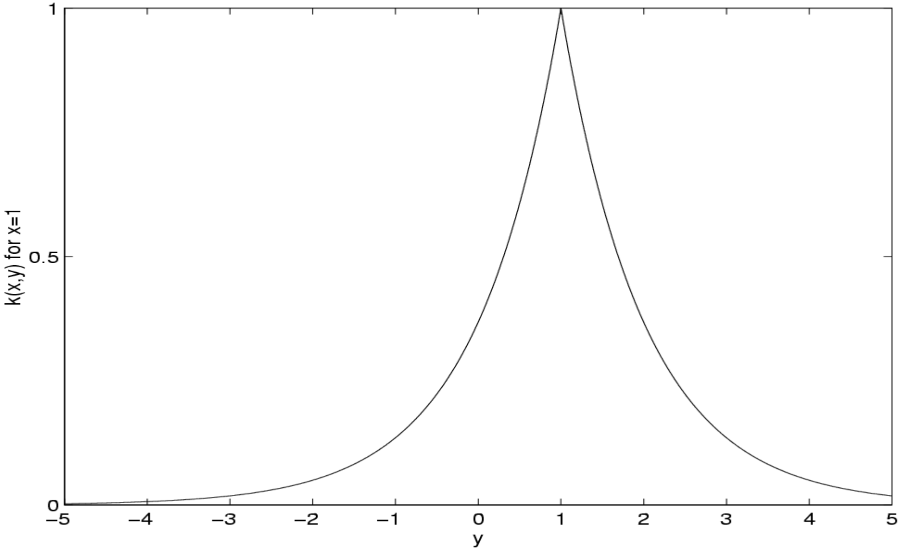
Examples of kernels $k(x, x')$

Linear	$\langle x, x' \rangle$
Laplacian RBF	$\exp\left(-\lambda \|x - x'\|\right)$
Gaussian RBF	$\exp\left(-\lambda \|x - x'\|^2\right)$
Polynomial	$\left(\langle x, x' \rangle + c \rangle\right)^d, c \geq 0,\ d \in \mathbb{N}$
B-Spline	$B_{2n+1}(x - x')$
Cond. Expectation	$\mathbf{E}_c[p(x\|c)p(x'\|c)]$

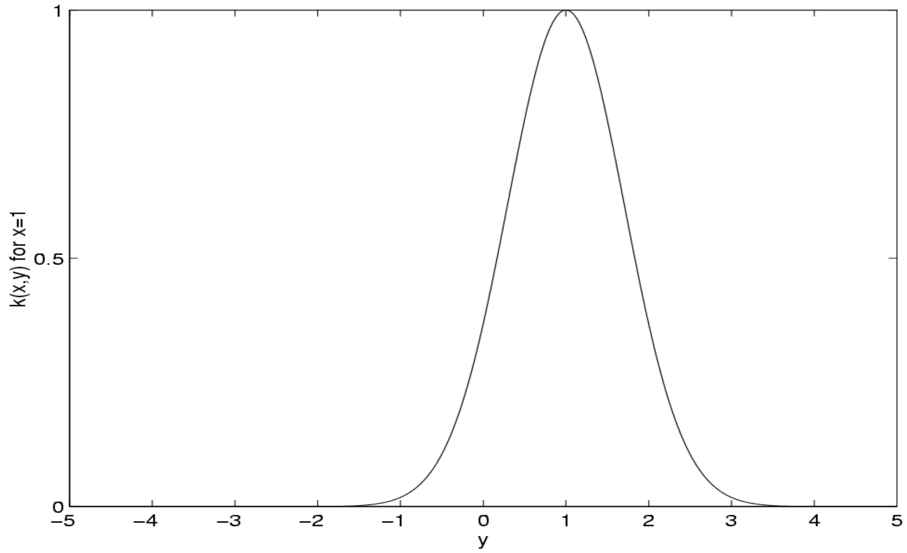
Linear Covariance



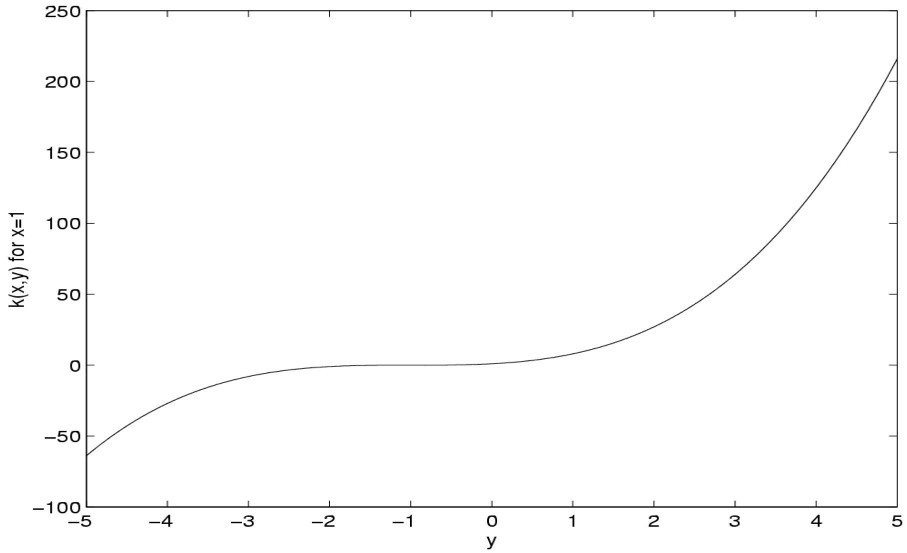
Laplacian Covariance



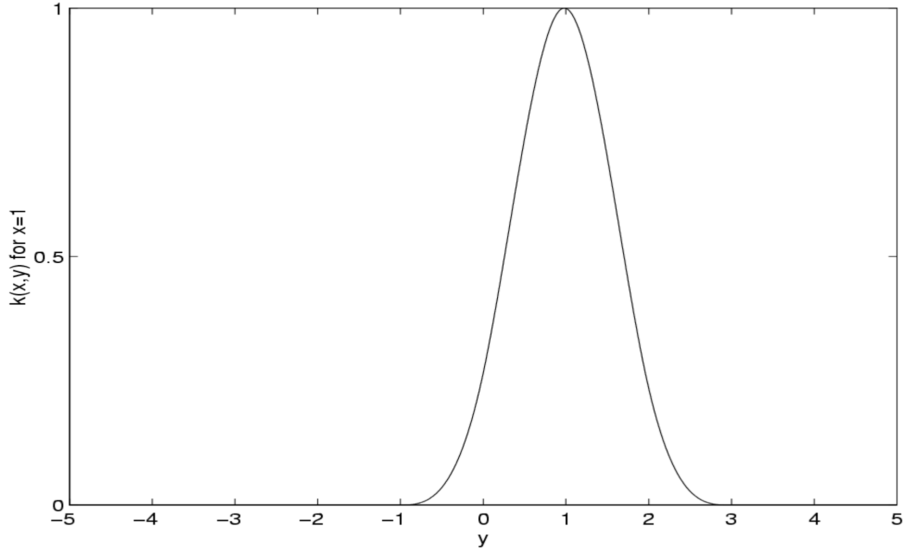
Gaussian Covariance



Polynomial (Order 3)



B_3-Spline Covariance



Gaussian Processes and Kernels

Covariance Function

- Function of two arguments
- Leads to matrix with nonnegative eigenvalues
- Describes correlation between pairs of observations

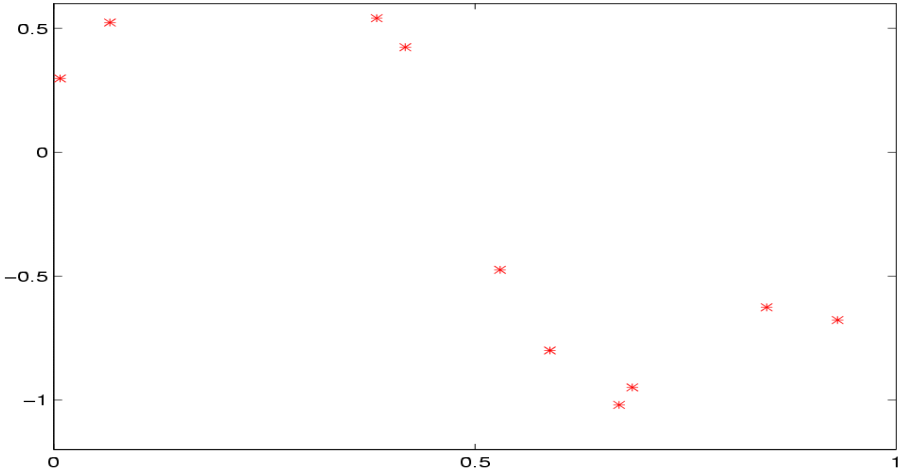
Kernel

- Function of two arguments
- Leads to matrix with nonnegative eigenvalues
- Similarity measure between pairs of observations

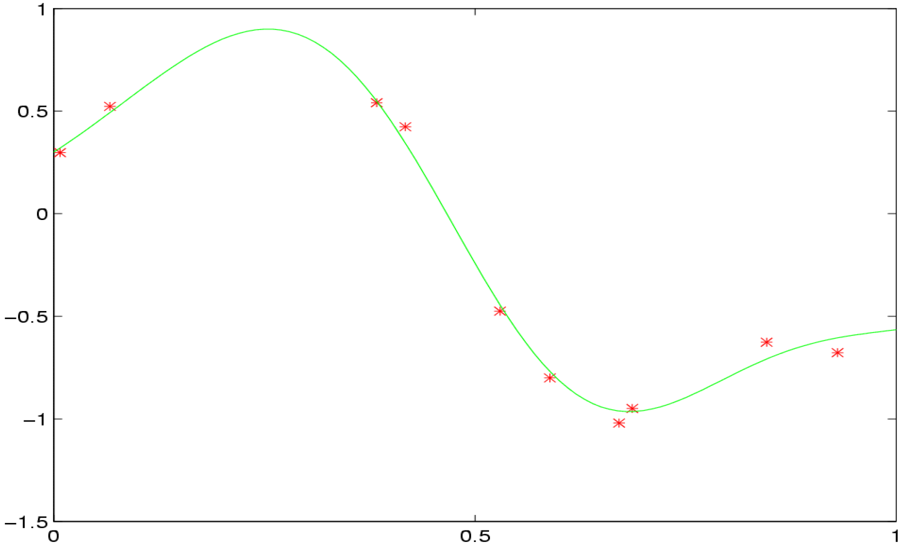
Lucky Guess

- We suspect that kernels and covariance functions are the same . . .

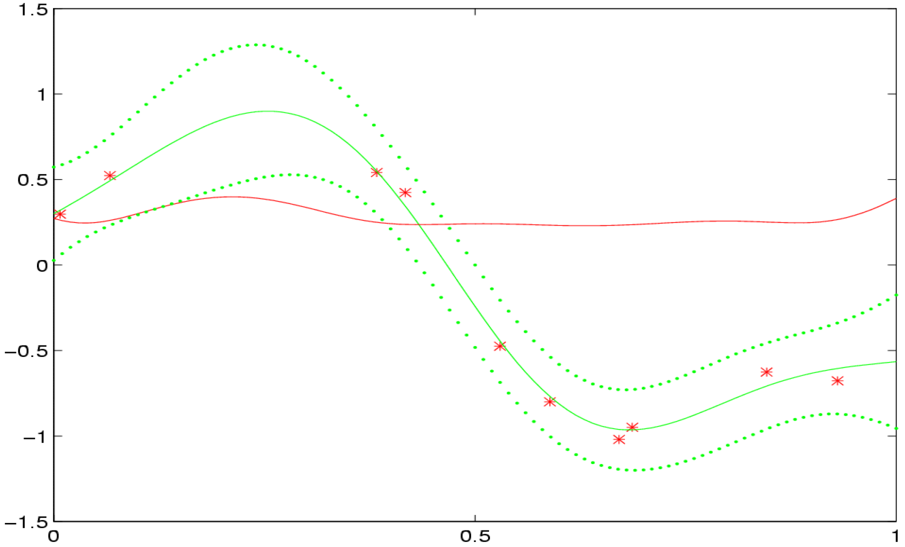
Training Data



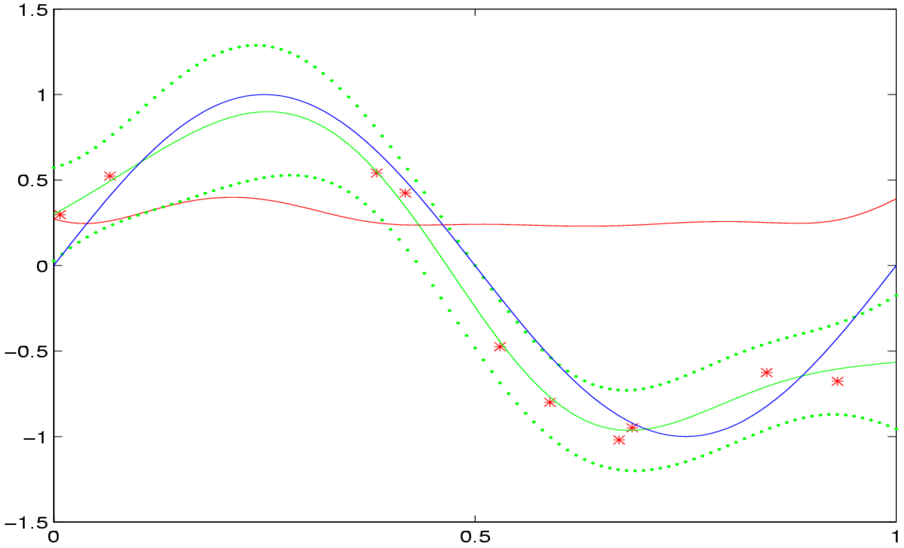
Mean $\vec{k}^\top(x)(K + \sigma^2\mathbf{1})^{-1}y$



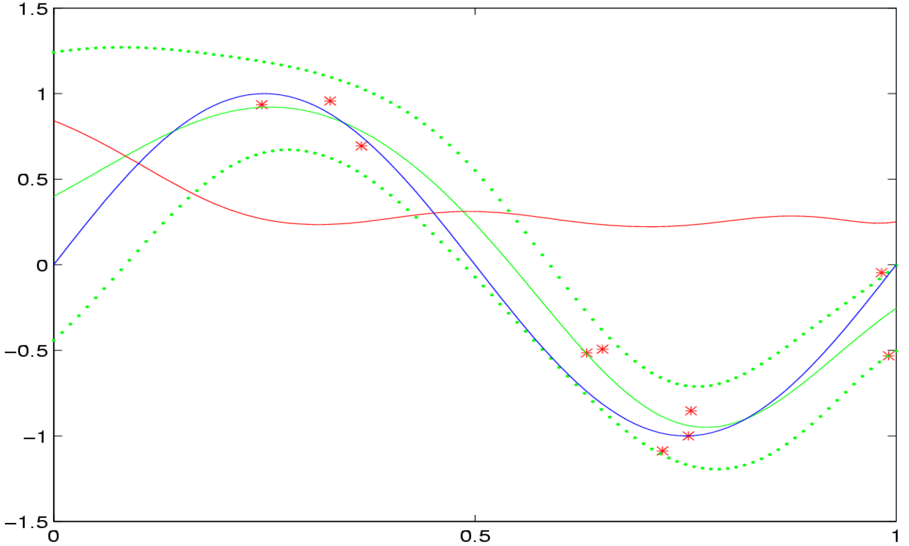
Variance $k(x,x) + \sigma^2 - \vec{k}^\top(x)(K + \sigma^2\mathbf{1})^{-1}\vec{k}(x)$



Putting everything together . . .



Another Example



The ugly details

Covariance Matrices

- Additive noise

$$K = K_{\text{kernel}} + \sigma^2 \mathbf{1}$$

- Predictive mean and variance

$$\tilde{K} = K_{y'y'} - K_{yy'}^\top K_{yy}^{-1} K_{yy'} \text{ and } \tilde{\mu} = K_{yy'}^\top K_{yy}^{-1} y$$

Pointwise prediction

$$\begin{aligned} K_{yy} &= K + \sigma^2 \mathbf{1} \\ K_{y'y'} &= k(x, x) + \sigma^2 \\ K_{yy'} &= (k(x_1, x), \ldots, k(x_m, x)) \end{aligned}$$

Plug this into the mean and covariance equations.

Mini Summary

Gaussian Process

- Like function, just random
- Mean and covariance determine the process
- Can use it for estimation

Regression

- Jointly normal model
- Additive noise to deal with error in measurements
- Estimate for mean and uncertainty

Support Vector Regression

Loss Function

Given y, find $f(x)$ such that the loss $l(y, f(x))$ is minimized.

- Squared loss $(y - f(x))^2$.
- Absolute loss $|y - f(x)|$.
- ϵ-insensitive loss $\max(0, |y - f(x)| - \epsilon)$.
- Quantile regression loss $\max(\tau(y - f(x)), (1 - \tau)(f(x) - y))$.

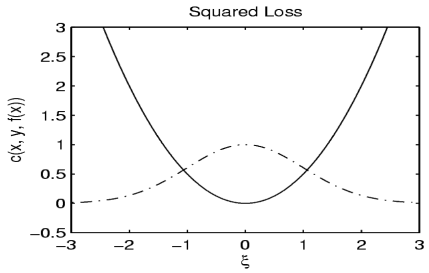
Expansion

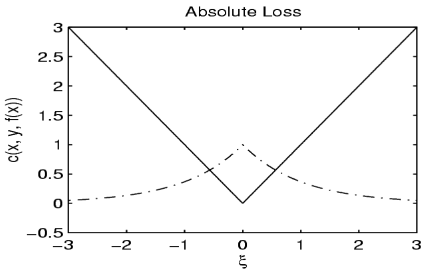
$$f(x) = \langle \phi(x), w \rangle + b$$

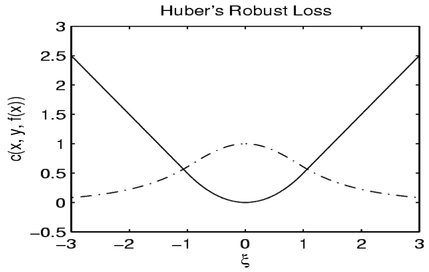
Optimization Problem

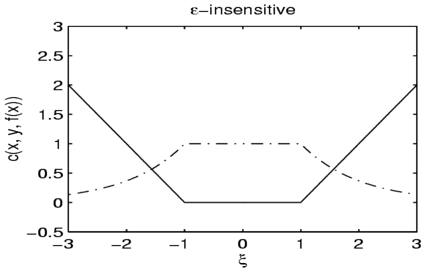
$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} l(y_i, f(x_i)) + \frac{\lambda}{2} \|w\|^2$$

Regression loss functions









Summary

Novelty Detection

- Basic idea
- Optimization problem
- Stochastic Approximation
- Examples

LMS Regression

- Additive noise
- Regularization
- Examples
- SVM Regression

An Introduction to Machine Learning

L6: Structured Estimation

Alexander J. Smola

Statistical Machine Learning Program
Canberra, ACT 0200 Australia
Alex.Smola@nicta.com.au

Machine Learning Summer School 2008

L6 Structured Estimation

Multiclass Estimation

- Margin Definition
- Optimization Problem
- Dual Problem

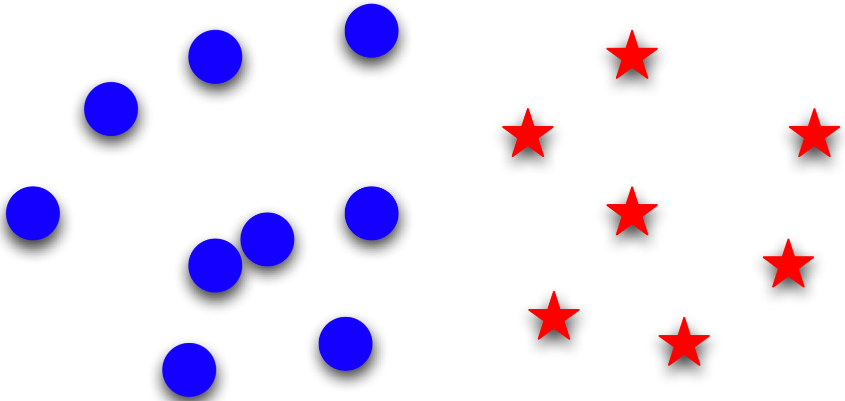
Max-Margin-Markov Networks

- Feature map
- Column generation and SVMStruct
- Application to sequence annotation

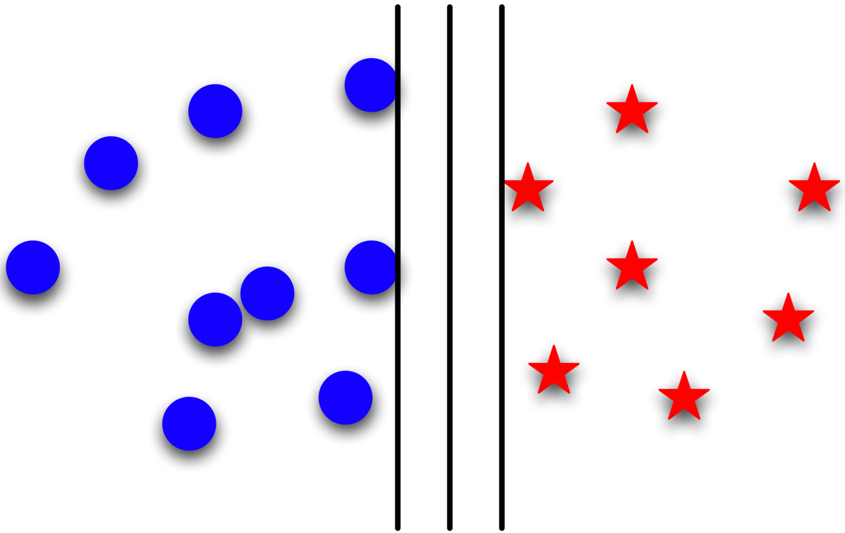
Web Page Ranking

- Ranking Measures
- Linear assignment problems
- Examples

Binary Classification



Binary Classification



Multiclass Classification

Goal

Given x_i and $y_i \in \{1, \ldots, N\}$, define a margin.

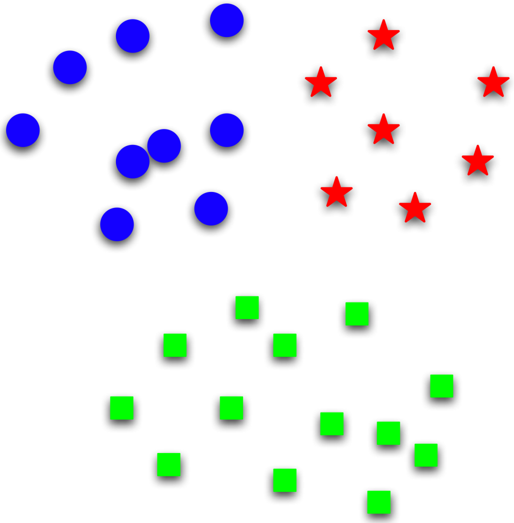
Binary Classification

$$\begin{aligned} \text{for } y_i = 1 \ \ \langle x_i, w \rangle &\geq 1 + \langle x_i, -w \rangle \\ \text{for } y_i = -1 \ \ \langle x_i, -w \rangle &\geq 1 + \langle x_i, w \rangle \end{aligned}$$

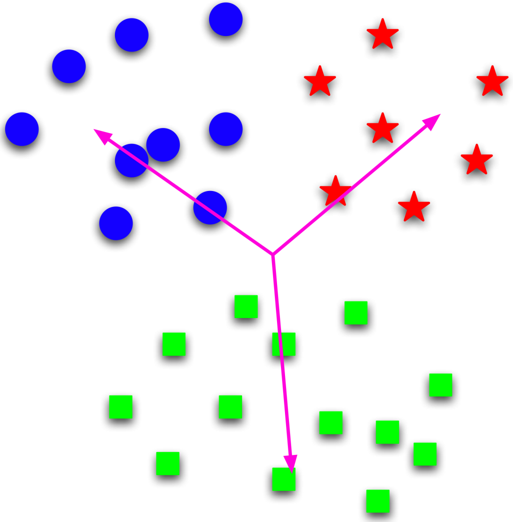
Multiclass Classification

$$\langle x_i, w_y \rangle \geq 1 + \langle x_i, w_{y'} \rangle \ \text{ for all } y' \neq y.$$

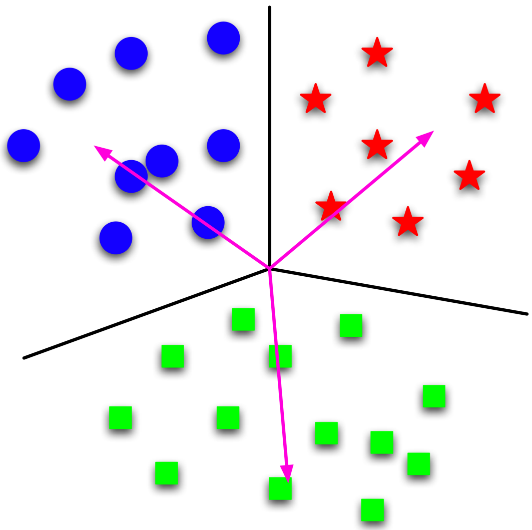
Multiclass Classification



Multiclass Classification



Multiclass Classification



Structured Estimation

Key Idea

Combine x and y into **one** feature vector $\phi(x, y)$.

Large Margin Condition and Slack

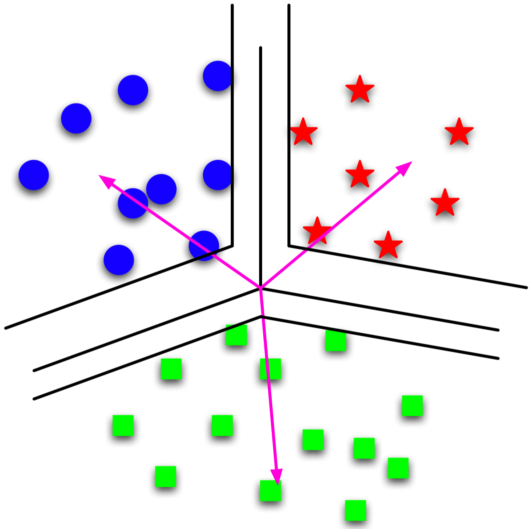
$$\langle \Phi(x, y), w \rangle \geq \Delta(y, y') + \langle \Phi(x, y'), w \rangle - \xi \text{ for all } y' \neq y.$$

- $\Delta(y, y')$ is the cost of misclassifying y for y'.
- $\xi \geq 0$ is as a slack variable.

$$\underset{w,\xi}{\text{minimize}} \ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i$$

subject to $\langle \Phi(x_i, y_i) - \Phi(x_i, y'), w \rangle \geq \Delta(y_i, y') - \xi_i$ for all $y' \neq y_i$.

Multiclass Margin



Dual Problem

Quadratic Program

$$\underset{\alpha}{\text{minimize}} \ \frac{1}{2} \sum_{i,j,y,y'} \alpha_{iy} \alpha_{jy'} K_{iy,jy'} - \sum_{i,y} \alpha_{iy} \Delta(y_i, y)$$

$$\text{subject to } \sum_{y} \alpha_{iy} \leq C \text{ and } \alpha_{iy} \geq 0.$$

Here $K_{iy,jy'} = \langle \phi(x_i, y_i) - \phi(x_i, y), \phi(x_j, y_j) - \phi(x_j, y') \rangle$.

$$w = \sum_{i,y} \alpha_{iy} \left(\phi(x_i, y_i) - \phi(x_i, y) \right).$$

Solving It

- Use SVMStruct (by Thorsten Joachims)
- Column generation (subset optimization). At optimality:

$$\alpha_{iy} \left[\langle \phi(x_i, y_i) - \phi(x_i, y), w \rangle - \Delta(y_i, y) \right] = 0$$

Pick (i, y) pairs for which this doesn't hold.

Implementing It

Start

Use an existing structured SVM solver, e.g. SVMStruct.

Loss Function

Define a loss function $\Delta(y, y')$ for your problem.

Feature Map

Define a suitable feature map $\phi(x, y)$. More examples later.

Column Generator

Implement algorithm which maximizes

$$\langle \phi(x_i, y), w \rangle + \Delta(y_i, y)$$

Mini Summary

Multiclass Margin

- Joint Feature Map
- Relative margin using misclassification error
- Binary classification a special case

Optimization Problem

- Convex Problem
- Can be solved using existing packages
- Column generation
- Joint feature map

Named Entity Tagging

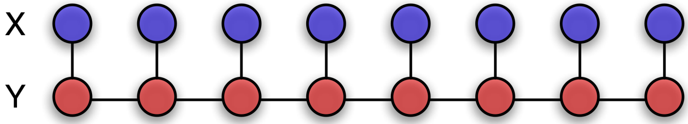
Goal

Given a document, i.e. a sequence of words, find those words which correspond to named entities.

Interaction

Adjacent labels will influence which words get tagged.

`President Bush was hiding behind the bush.`



Joint Feature Map

$$\phi(x, y) = \left[\sum_{i=1}^{l} y_i \phi(x_i), \sum_{i=1}^{l} y_i y_{i+1} \right]$$

Estimation and Column Generation

Loss Function

Count how many of the labels are wrong, i.e. $\Delta(y, y') = \|y - y'\|_1$.

Estimation

Find sequence y maximizing $\langle \phi(x, y), w \rangle$, that is

$$\sum_{i=1}^{l} y_i \langle \phi(x_i), w_1 \rangle + y_i y_{i+1} w_2$$

For column generation additional term $\sum_{i=1}^{l} |y_i - y_i'|$.

Dynamic Programming

We are maximizing a function $\sum_{i=1}^{l} f(y_i, y_{i+1})$.

Dynamic Programming

Background

Generalized distributive law, Viterbi, Shortest path

Key Insight

To maximize $\sum_{i=1}^{l} f(y_i, y_{i+1})$, once we've picked $y_j = 1$ the problems on either side become independent. In equations

$$
\begin{aligned}
&\underset{y}{\text{maximize}} \sum_{i=1}^{l} f(y_i, y_{i+1}) \\
&= \underset{y_2,\ldots,y_l}{\text{maximize}} \left[\sum_{i=2}^{l} f(y_i, y_{i+1}) + \underbrace{\underset{y_1}{\text{maximize}}\, f(y_1, y_2)}_{:=g_2(y_2)} \right] \\
&= \underset{y_3,\ldots,y_l}{\text{maximize}} \left[\sum_{i=3}^{l} f(y_i, y_{i+1}) + \underbrace{\underset{y_2}{\text{maximize}}\, f(y_2, y_3) + g_2(y_2)}_{:=g_3(y_3)} \right]
\end{aligned}
$$

Implementing It

Forward Pass

- Compute recursion

$$g_{i+1}(y_{i+1}) := \underset{y_i}{\text{maximize}}\, f(y_i, y_{i+1}) + g_i(y_i)$$

- Store best answers

$$y_i(y_{i+1}) := \underset{y_i}{\text{argmax}}\, f(y_i, y_{i+1}) + g_i(y_i)$$

Backward Pass

After computing the last term y_l, solve recursion $y_i(y_{i+1})$.

Cost

- Linear time for forward and backward pass
- Linear storage

Extensions

Fancy Feature Maps
Can use more complicated interactions between words and labels.

Fancy Labels
More sophisticated than binary labels. E.g. tag for place, person, organization, etc.

Fancy Structures
Rather than linear structure, have a 2D structure. Annotate images.

Mini Summary

Named Entity Tagging

- Sequence of words, find named entities
- Can be written as a structured estimation problem
- Feature map decomposes into separate terms

Dynamic Programming

- Objective function a sum of adjacent terms
- Same as Viterbi algorithm
- Linear time and space

Web Page Ranking

Goal

Given a set of documents d_i and a query q, find ranking of documents such that most relevant documents come first.

Data

At training time, we have ratings of pages $y_i \in \{0, 5\}$.

Scoring Function

Discounted cumulative gain. That is, we gain more if we rank relevant pages highly, namely

$$\mathrm{DCG}(\pi, y) = \sum_{i,j} \pi_{ij} \frac{2^{y_i} + 1}{\log(j + 1)}.$$

π is a permutation matrix (exactly one entry per row / column is 1, rest is 0).

From Scores to Losses

Goal

We need a loss function, not a performance score.

Idea

Use performance relative to the best as loss score.

Practical Implementation

Instead of $\mathrm{DCG}(\pi, y)$ use $\Delta(\mathbf{1}, \pi) = \mathrm{DCG}(1, y) - \mathrm{DCG}(\pi, y)$.

Feature map ...

Goal

Find w such that $\langle w, \phi(d_i, q)\rangle$ gives us a score (like PageRank, but we want to learn it from data).

Joint feature map

- Need to map $q, \{d_1, \ldots, d_l\}$ and π into feature space.
- Want to get sort operation at test time from $\langle \phi(q, D, \pi), w\rangle$.

Solution

$$\phi(q, D, \pi) = \sum_{i,j} \pi_{ij} c_i \phi(q, d_j) \text{ where } c_i \text{ is decreasing.}$$

Consequence

$\sum_{i,j} \pi_{ij} c_i \langle \phi(q, d_j), w\rangle$ is maximized by sorting documents along c_i, i.e. in descending order.

Sorting

Unsorted: score is 57

c_i	1	2	3	4	5
Page ranks	3	2	3	9	1

Sorted: score is 71

c_i	1	2	3	4	5
Page ranks	1	2	3	3	9

This is also known as the Polya-Littlewood-Hardy inequality

Column Generation

Goal

Efficiently find permutation which maximizes

$$\langle \phi(q, D, \pi), w \rangle + \Delta(\mathbf{1}, \pi)$$

Optimization Problem

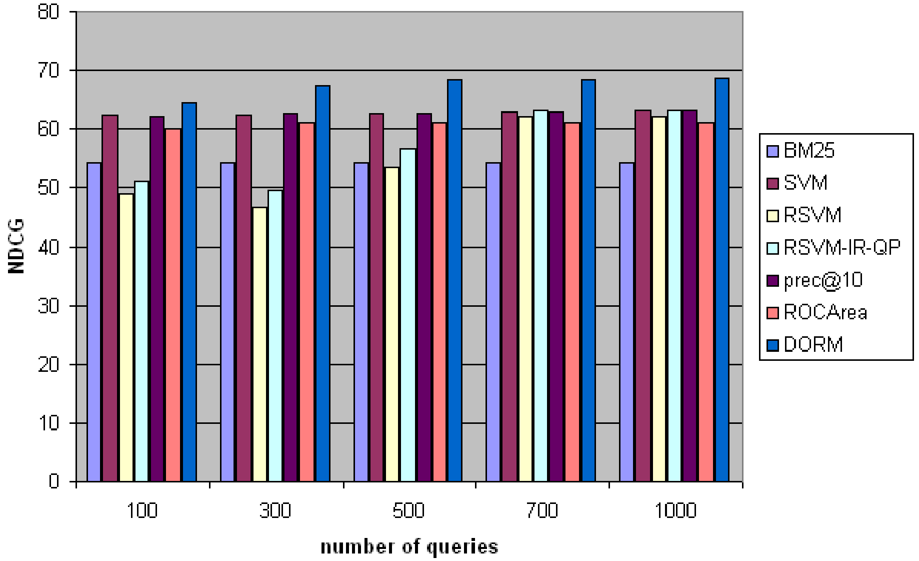
$$\underset{\pi}{\text{maximize}} \sum_{i,j} \pi_{ij} \left[c_i \langle \phi(d_j, q), w \rangle + \frac{2^{y_i} + 1}{\log(j+1)} \right]$$

This is a linear assignment problem. Efficient codes exist (Hungarian marriage algorithm) to solve this in $O(l^3)$ time.

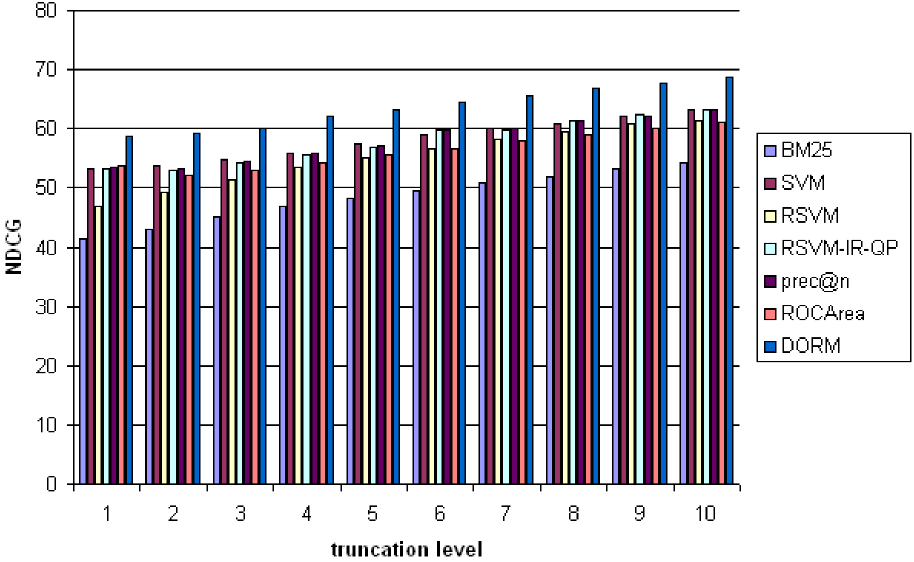
Putting everything together

- Use existing SVM solver (e.g. SVMStruct)
- Implement column generator for training
- Design sorting kernel

NDCG Optimization



NDCG Optimization



Mini Summary

Ranking Problem

- Web page ranking (documents with relevance score)
- Multivariate performance score
- Hard to optimize directly

Feature Map

- Maps permutations and data jointly into feature space
- Simple sort operation at test time

Column Generation

- Linear assignment problem
- Integrate in structured SVM solver

Summary

Structured Estimation

- Basic idea
- Optimization problem

Named Entity Tagging

- Annotation of a sequence
- Joint featuremap
- Dynamic programming

Ranking

- Multivariate performance score
- Linear assignment problem