An Introduction to Machine Learning with Kernels

Lecture 1

Alexander J. Smola
Alex.Smola@nicta.com.au

Statistical Machine Learning Program
National ICT Australia, Canberra

NATIONAL ICT AUSTRALIA

Day 1

Machine learning and probability theory

Introduction to pattern recognition, classification, regression, novelty detection, probability theory, Bayes rule, inference

Density estimation and Parzen windows

Kernels and density estimation, Silverman's rule, Watson Nadaraya estimator, crossvalidation

Perceptron and kernels

Hebb's rule, perceptron algorithm, convergence, feature maps, kernel trick, examples

Support Vector classification

Geometrical view, dual problem, convex optimization, kernels and SVM

Day 2

Text analysis and bioinformatics

Text categorization, biological sequences, kernels on strings, efficient computation, examples

Optimization

Sequential minimal optimization, convex subproblems, convergence, SVMLight, SimpleSVM

Regression and novelty detection

SVM regression, regularized least mean squares, adaptive margin width, novel observations

Practical tricks

Crossvalidation, ν-trick, median trick, data scaling, smoothness and kernels

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

Data

Vectors

- Collections of features (e.g. height, weight, blood pressure, age, . . .)
- Can map categorical variables into vectors (useful for mixed objects)

Matrices

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

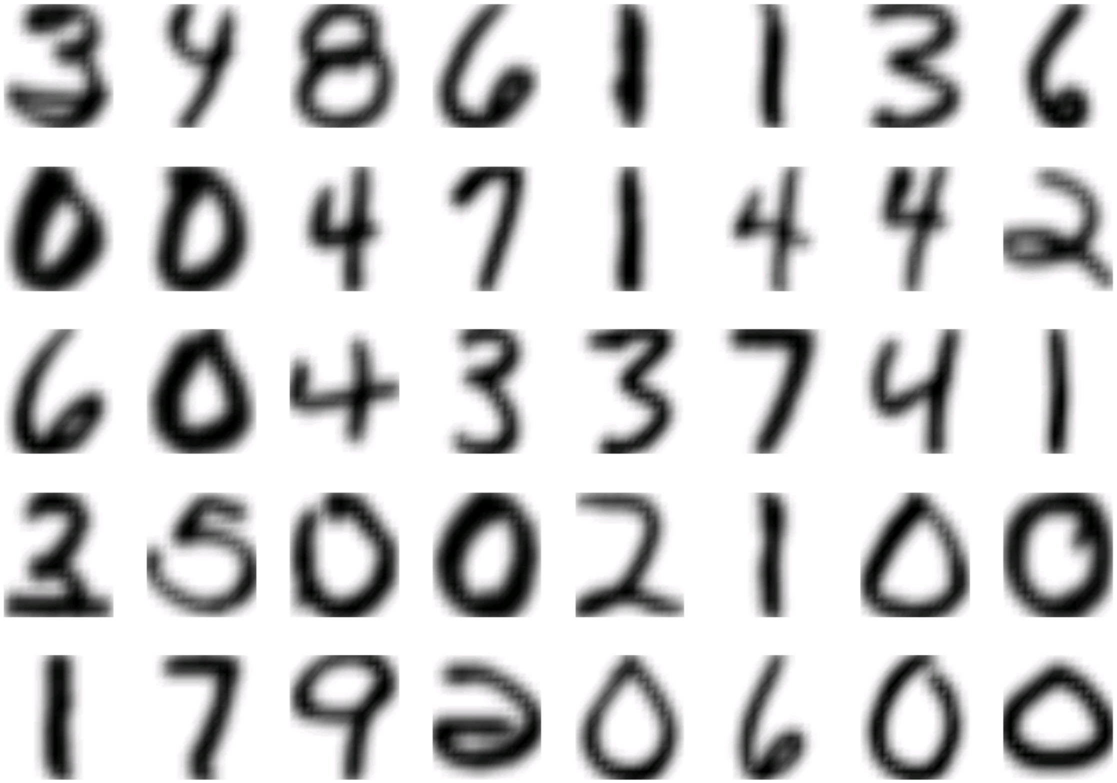
Strings

- Documents
- Gene sequences

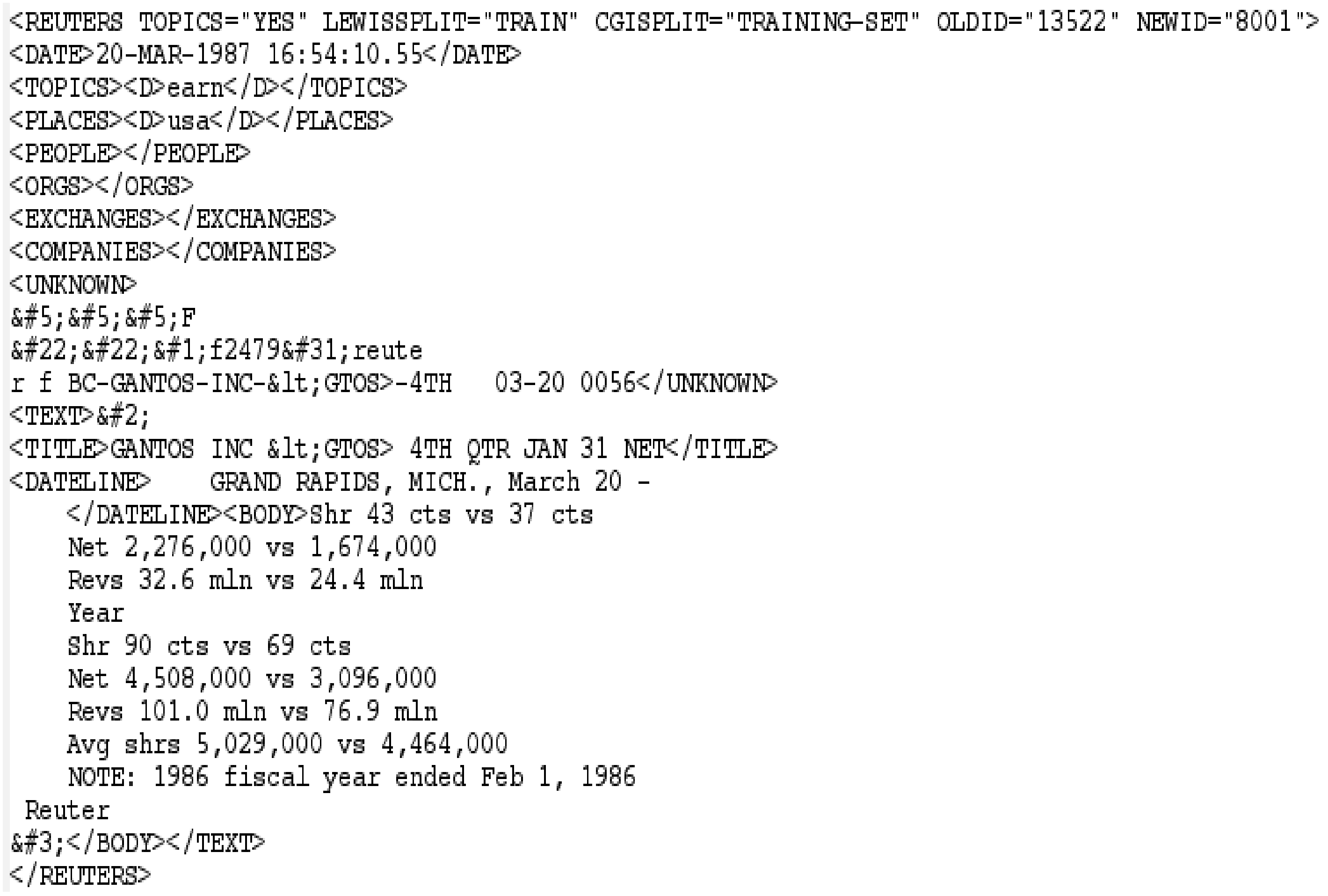
Structured Objects

- XML documents
- Graphs

Optical Character Recognition

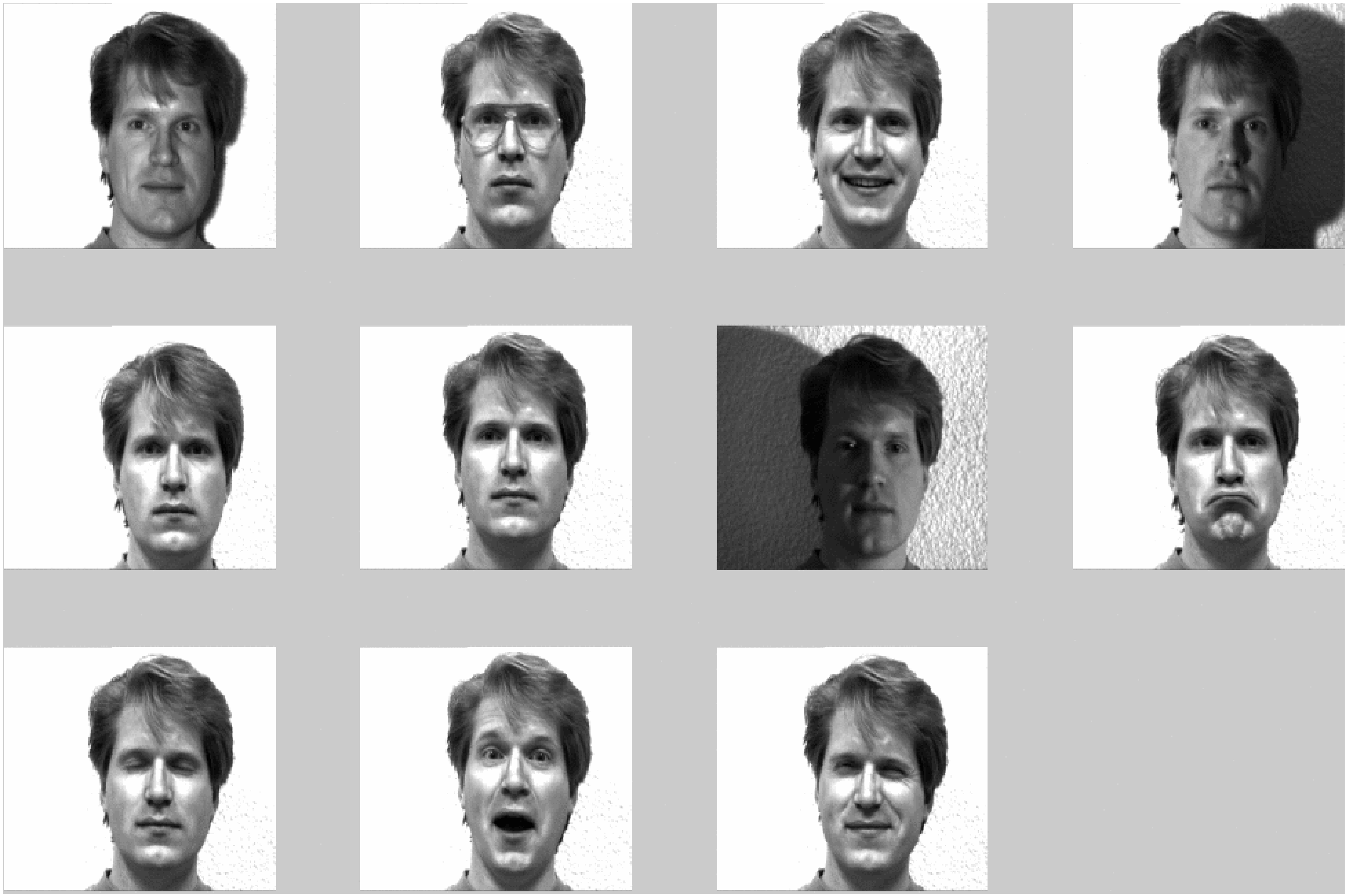


Reuters Database



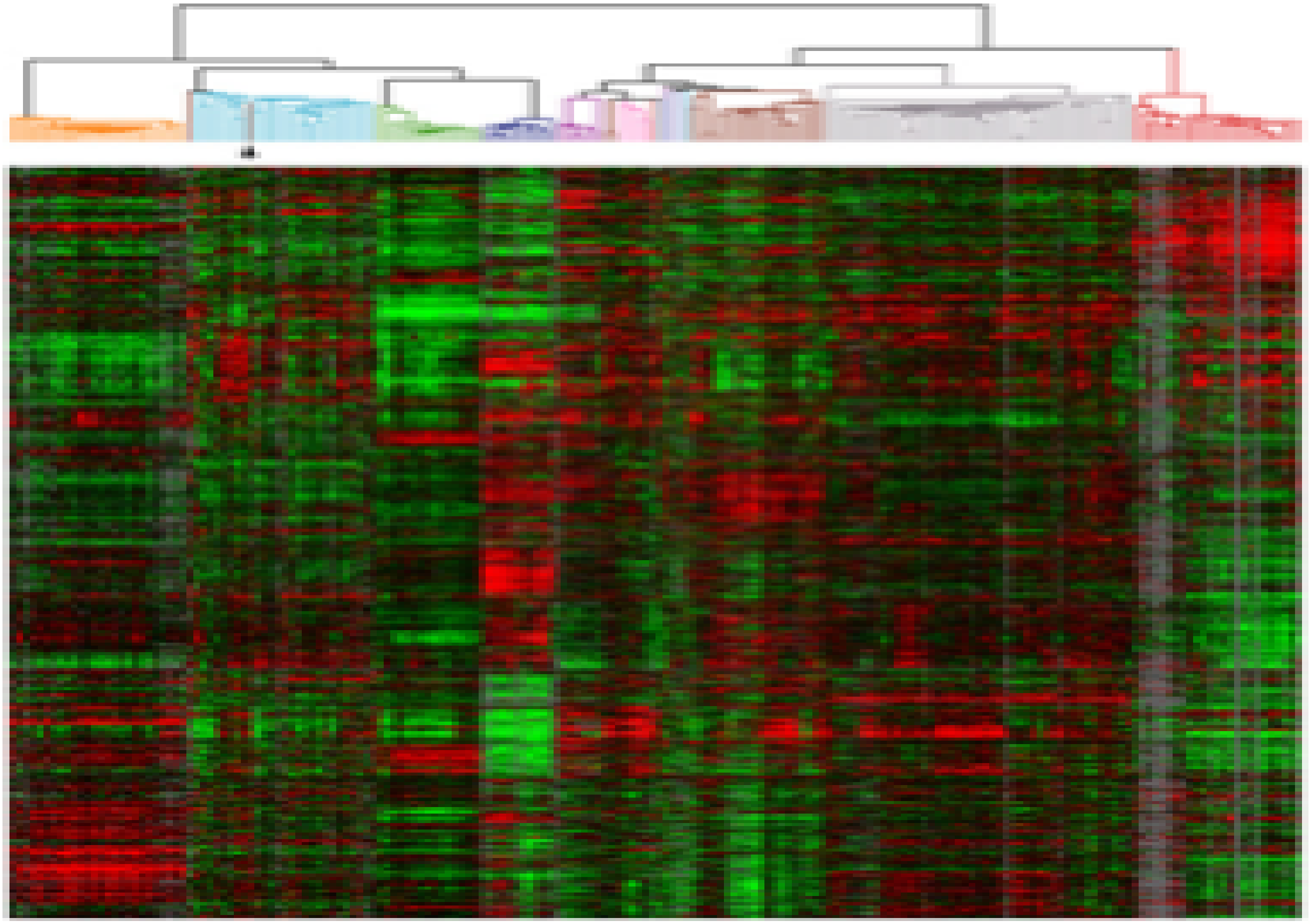
```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="13522" NEWID="8001">
<DATE>20-MAR-1987 16:54:10.55</DATE>
<TOPICS><D>earn</D></TOPICS>
<PLACES><D>usa</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;F
&#22;&#22;&#1;f2479&#31;reute
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

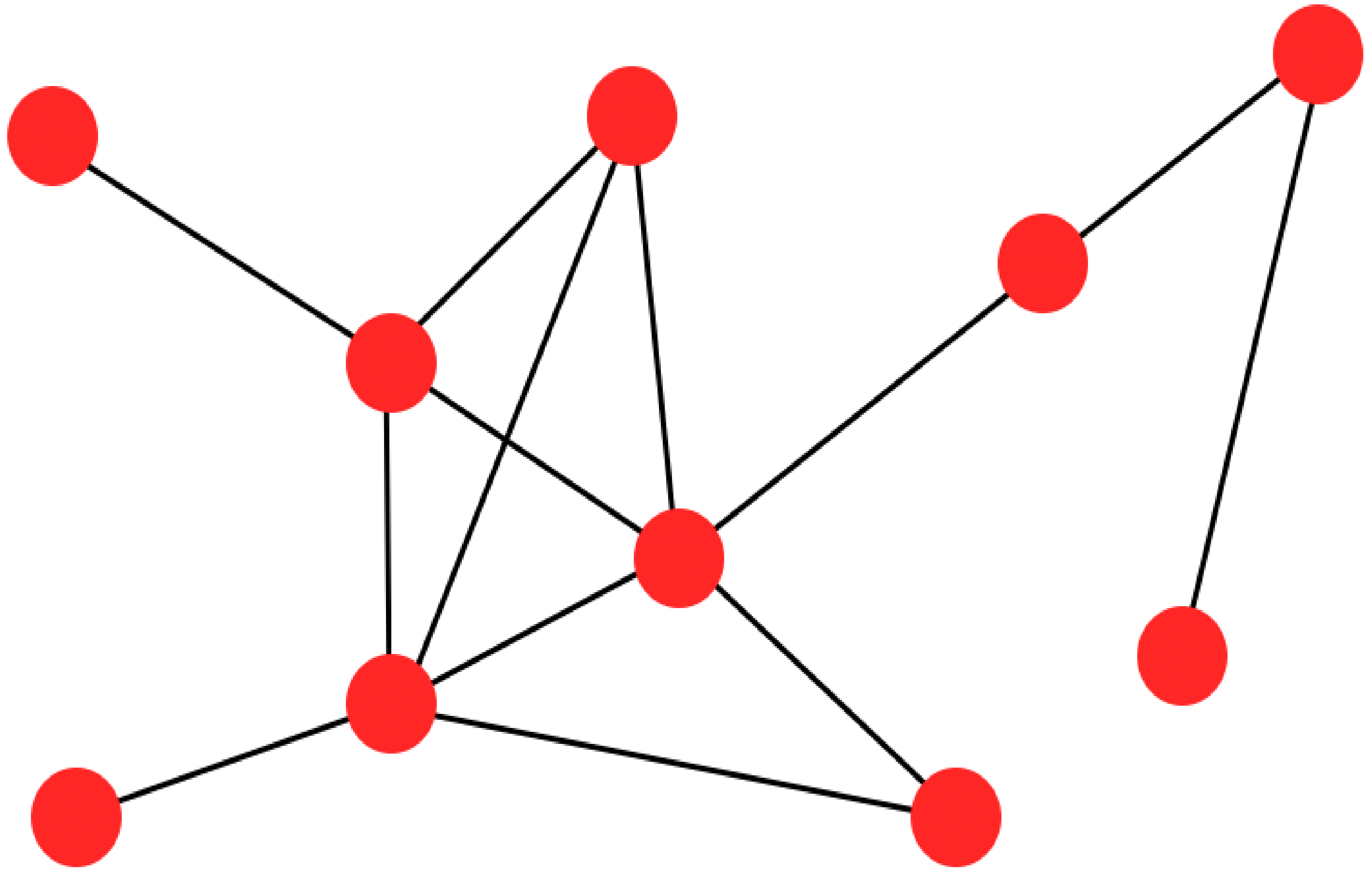
Microarray Data



Biological Sequences

```
>0_d1ctj__ 1.3.1.1.1 Cytochrome c6 (synonym: cytochrome
c553) [(Monoraphidium braunii)]
EADLALGKAVFDGNCAACHAGGGNNVIPDHTLQKAAIEQFLDGGFNIEAIVYQIENGKG
AMPAWDGRLDE
DEIAGVAAYVYDQAAGNKW
>0_d1dvh__ 1.3.1.1.2 Cytochrome c6 (synonym: cytochrome
c553) [Desulfovibrio vulgaris, strain miyazaki f]
[illegible]
ALADYMSKL
>0_d1c53__ 1.3.1.1.2 Cytochrome c6 (synonym: cytochrome
c553) [Desulfovibrio vulgaris, strain miyazaki f]
ADGAALYKSCVGCHGADGSKQAMGVGHAVKGQKADELFKKLKGYADGSYGGEKKAVMTN
LVKRYSDEEMK
AMADYMSKL
```

Graphs



Missing Variables

Incomplete Data

- Measurement devices may fail (e.g. dead pixels on camera)
- 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) . . .

What to do with data

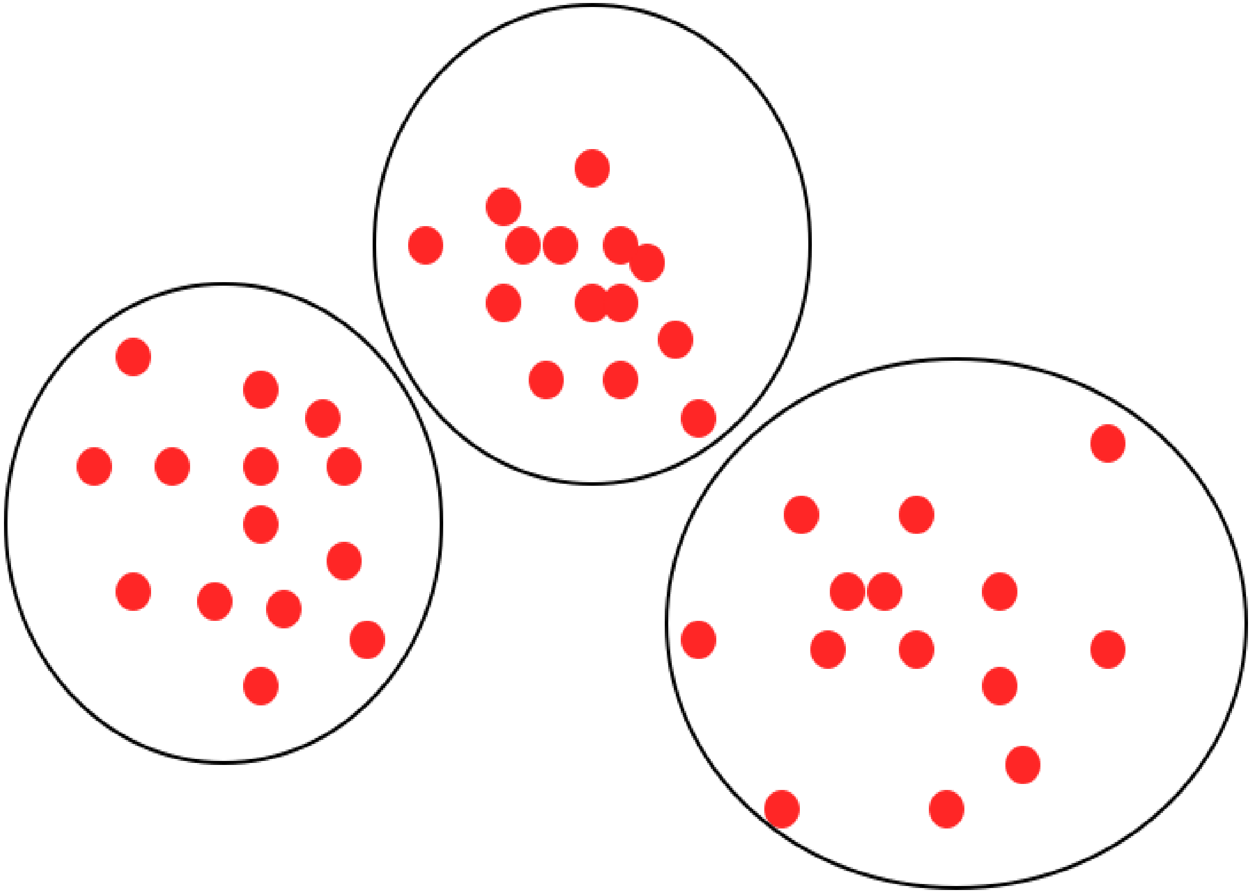
Unsupervised Learning

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

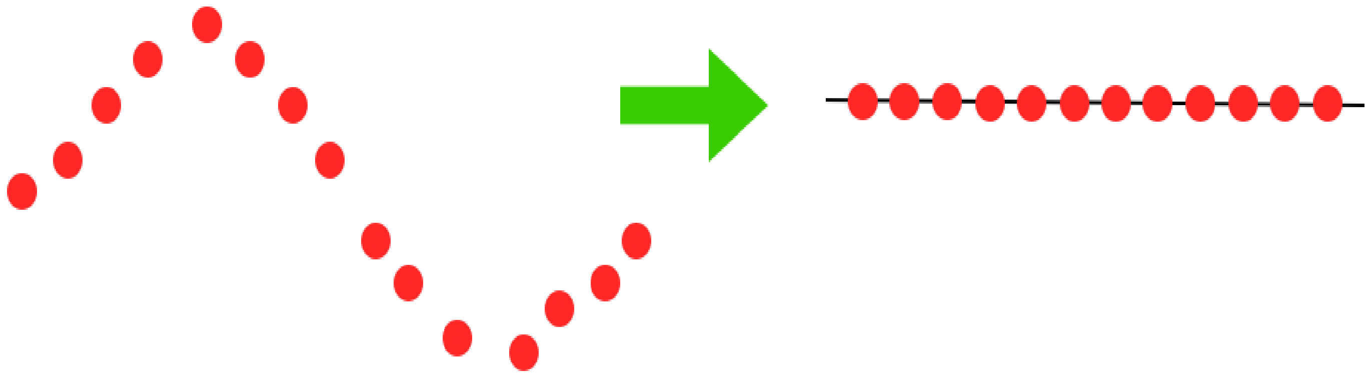
Supervised Learning

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

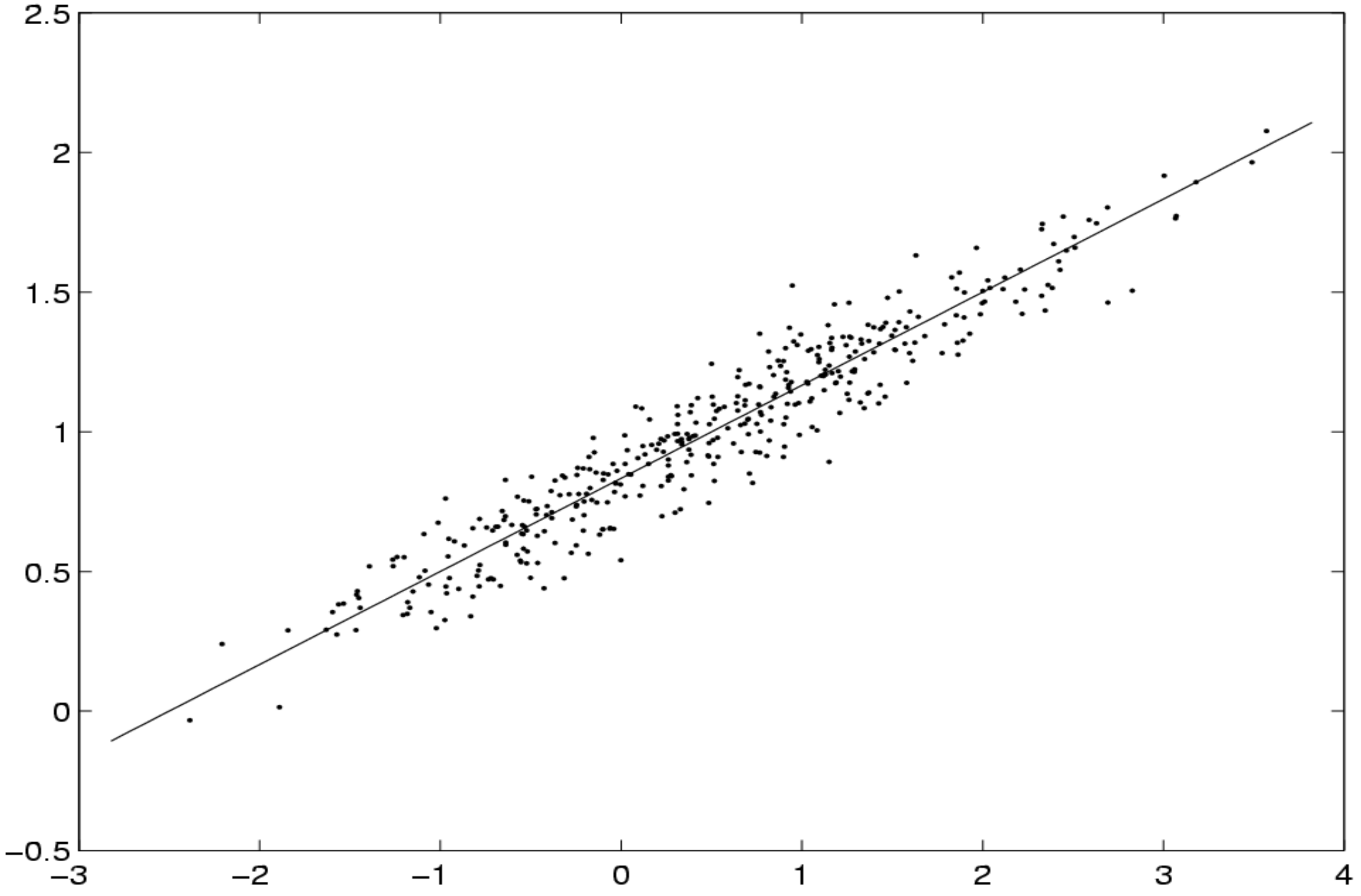
Clustering



Linear Subspace



Principal Components

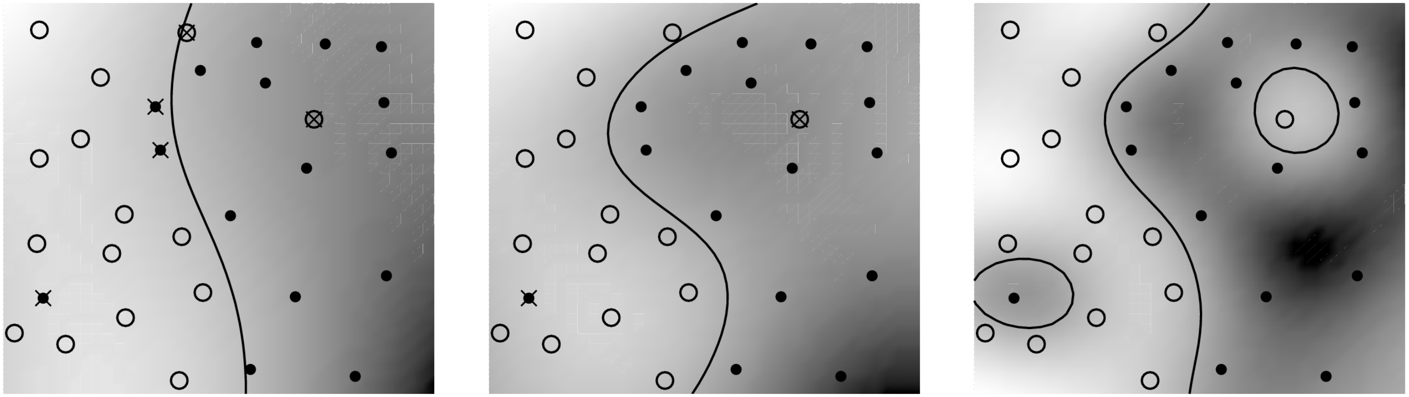


Classification

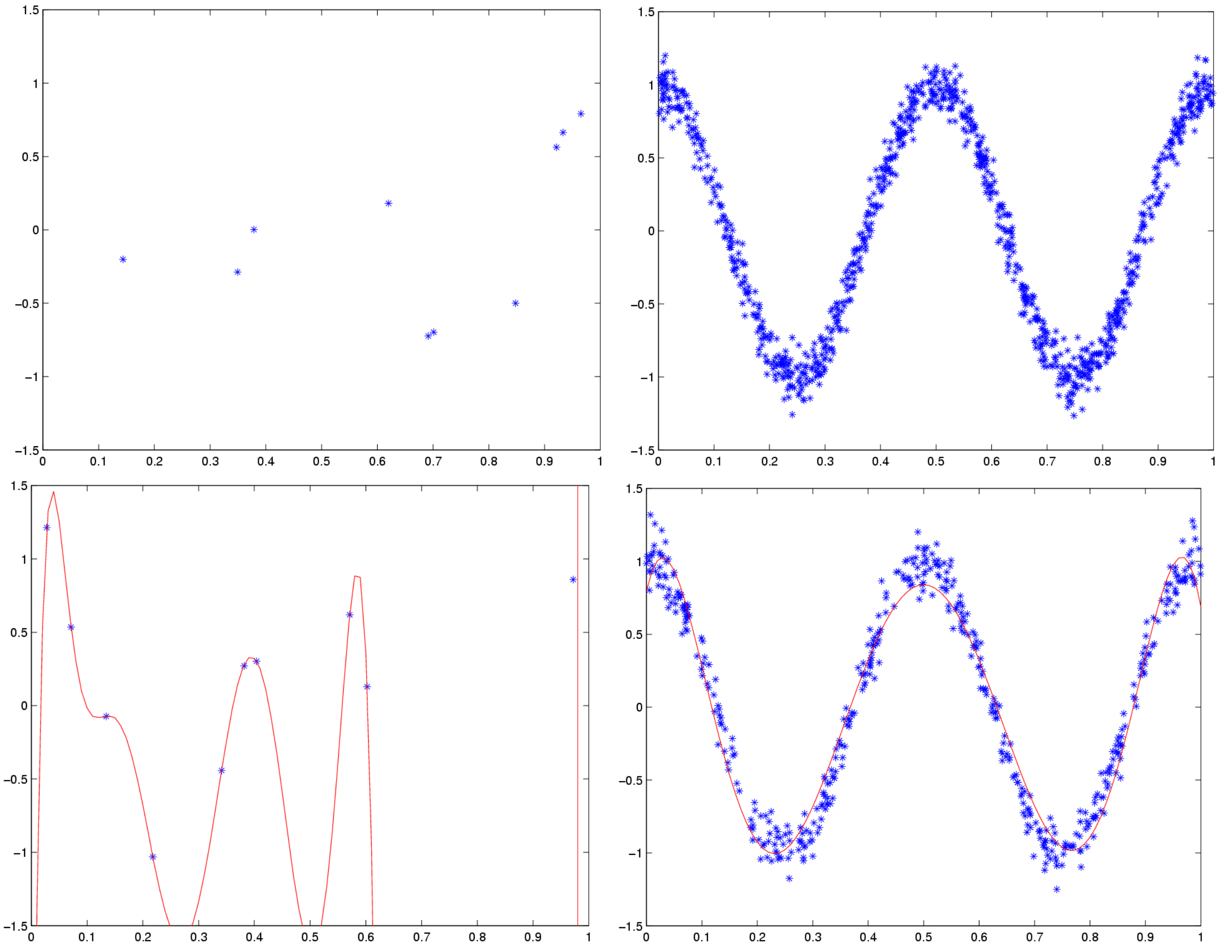
Data

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

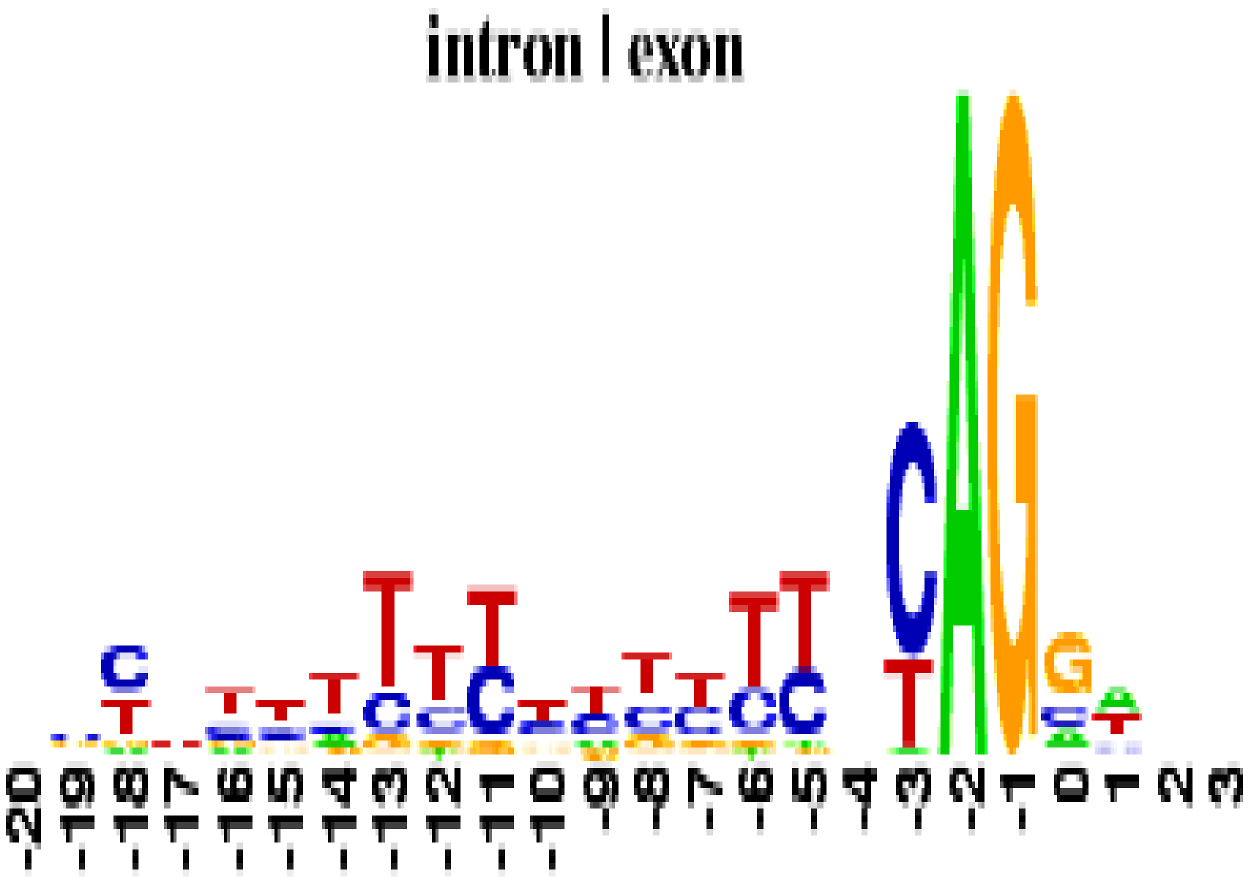
Task

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



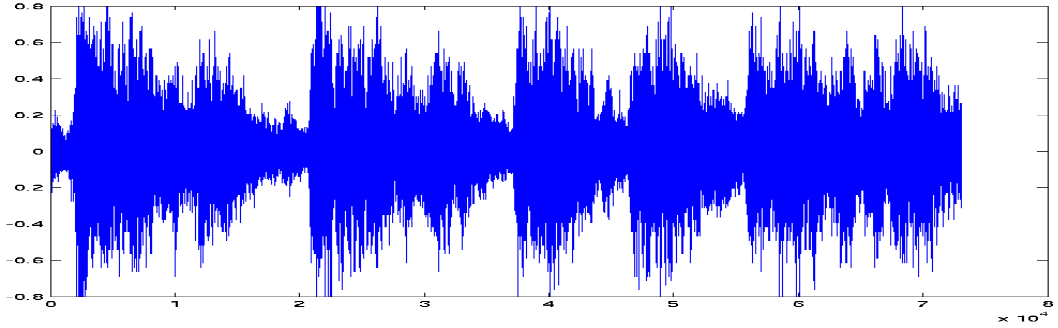
Annotating Audio

Goal

- Possible meaning of an audio sequence
- Give confidence measure

Example (from Australian Prime Minister's speech)

- a stray alien
- Australian



Novelty Detection

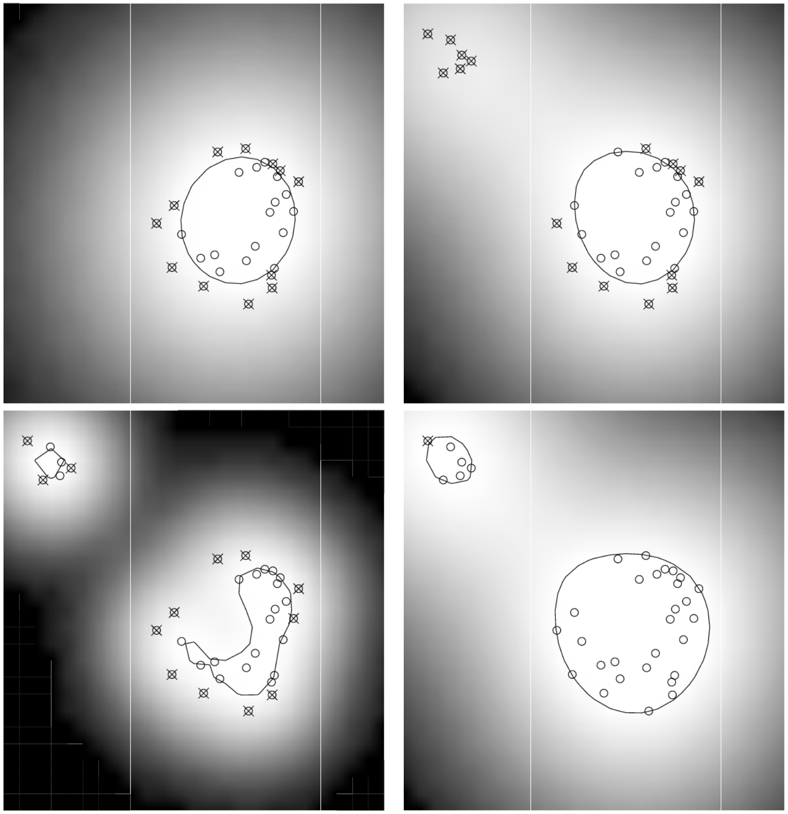
Data

Observations (x_i) 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.



What Machine Learning is not

Logic

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

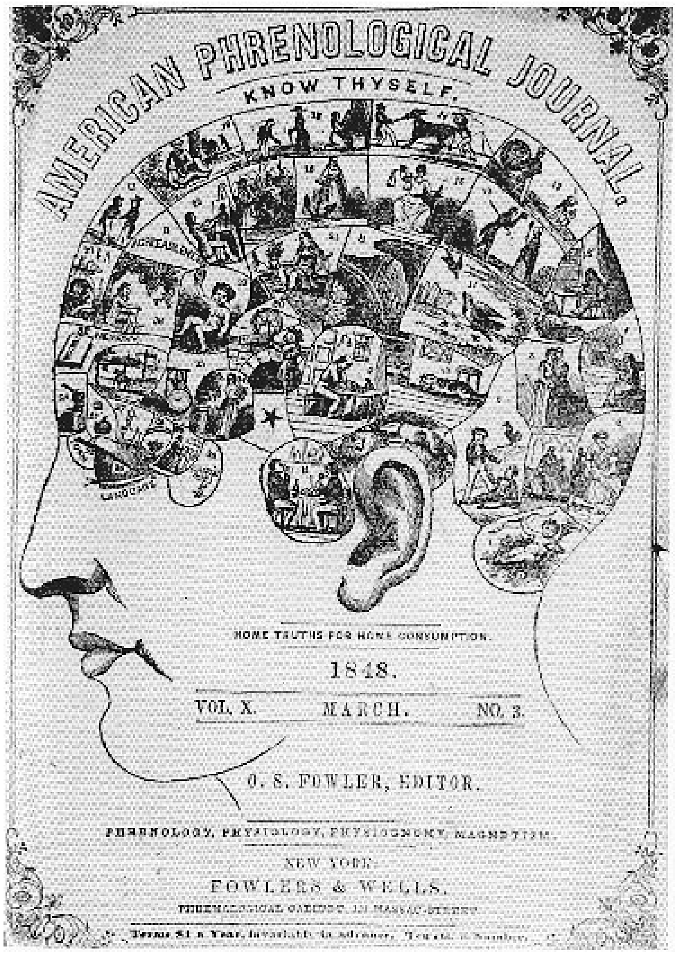
Artificial Intelligence

- 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)

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.

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 unbiased).

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.

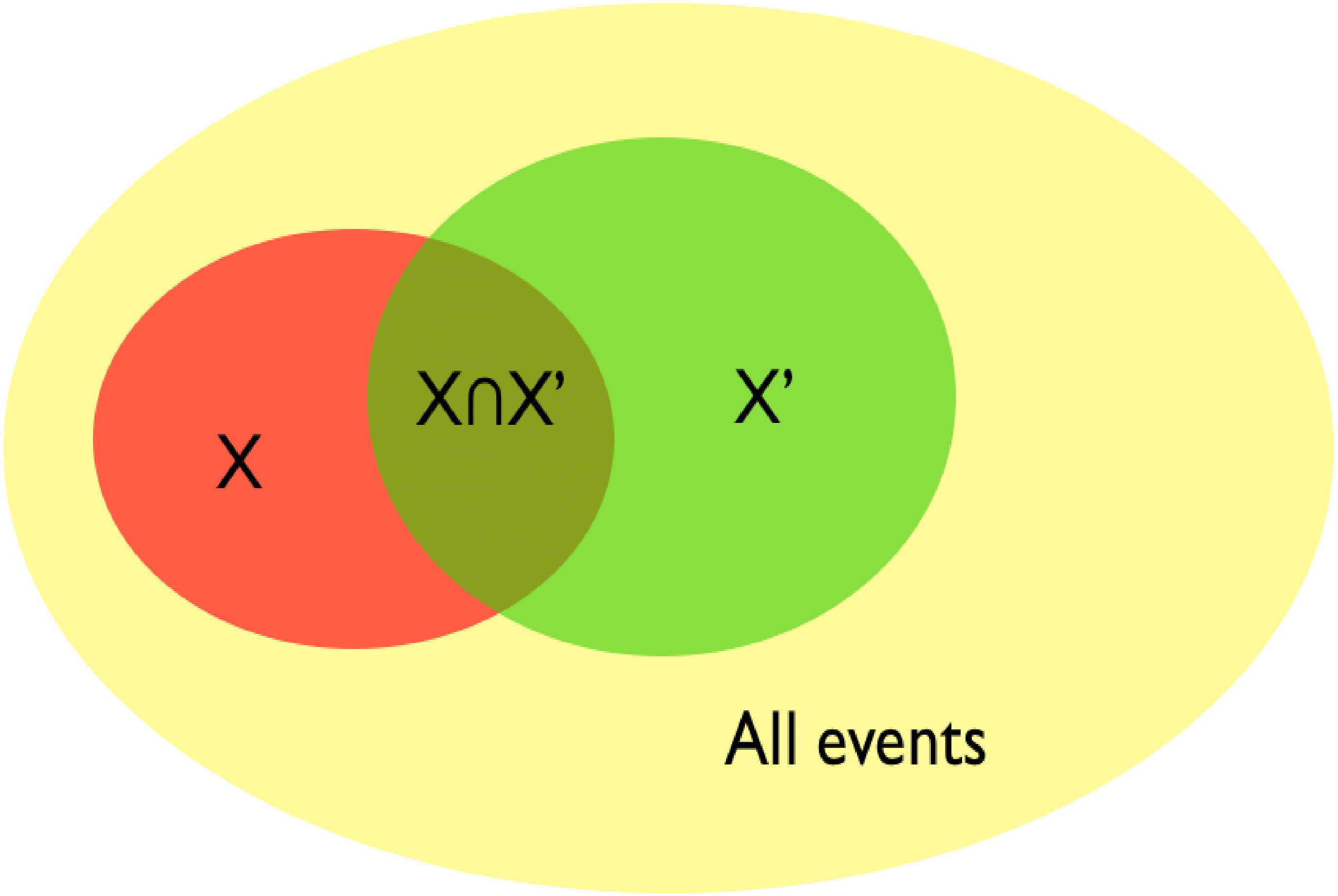
Basic Axioms

- $\Pr(X) \in [0, 1]$ for all $X \subseteq \mathfrak{X}$
- $\Pr(\mathfrak{X}) = 1$
- $\Pr(\cup_i X_i) = \sum_i \Pr(X_i)$ 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 $\mathcal{X}$ and $\mathcal{Y}$ are a probability measure on the product space of $\mathcal{X}$ and $\mathcal{Y}$. Consider the space of events $(x, x) \in \mathcal{X} \times \mathcal{Y}$.

Independence

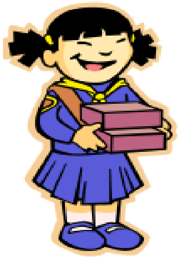
If x and y are independent, then for all $X \subset \mathcal{X}$ and $Y \subset \mathcal{Y}$

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

Independent Random Variables

Y
Outcome

X
Astrologist's
Prediction



0.25	0.25
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 $\Pr(Y, X) \leq \min(\Pr(X), \Pr(Y))$.

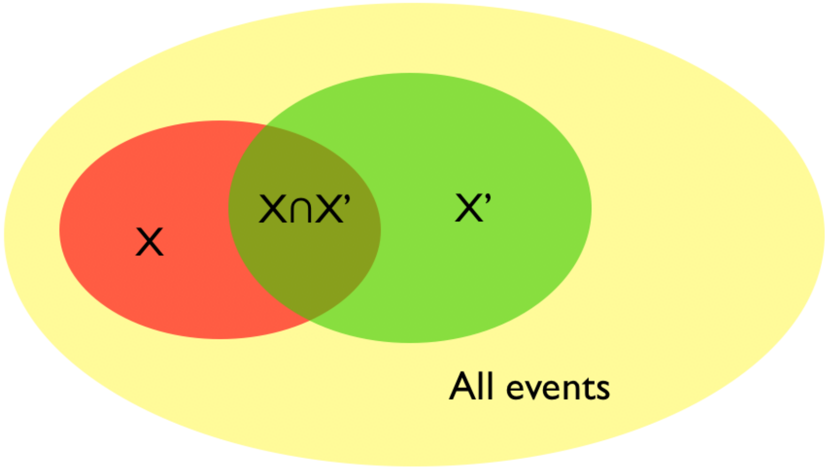
Bayes Rule

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

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.

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

- The AIDS test reports positive for all infections.

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

- 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

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

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 guilty, hence

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

- Eyewitness has equal confusion probability

$$p(Y = \text{eyewitness identifies}|X = \text{guilty}) = 0.9$$

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, 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.

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

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.

NATIONAL ICT AUSTRALIA

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 (cf. 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

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.

Maximum Likelihood for Multinomial Distribution:

We match the empirical probabilities via

$$\Pr_{\mathrm{emp}}(i) = \frac{\#\text{occurrences of } i}{\#\text{trials}}$$

In plain English this means that we take the number of occurrences of a particular event (say 7 times head) and divide this by the total number of trials (say 10 times). This yields 0.7.

Maximum Likelihood Proof

Goal

We want to estimate the parameter $\pi \in \mathbb{R}^n$ such that

$$\Pr(X|\pi) = \prod_{j=1}^{m} \Pr(X_j|\pi) = \prod_{i=1}^{n} \pi_i^{\#i}$$

is maximized while π is a probability (reparameterize $\pi_i = e^{\theta_i}$).

Constrained Optimization Problem

$$\text{minimize } \sum_{i=1}^{n} -\#i \cdot \theta_i \text{ subject to } \sum_{i=1}^{n} e^{\theta_i} = 1$$

Lagrange Function

$$L(\pi, \gamma) = \sum_{i=1}^{n} -\#i \cdot \theta_i + \gamma \left(1 - \sum_{i=1}^{n} e^{\theta_i} \right)$$

Maximum Likelihood Proof

First Order Optimality Conditions

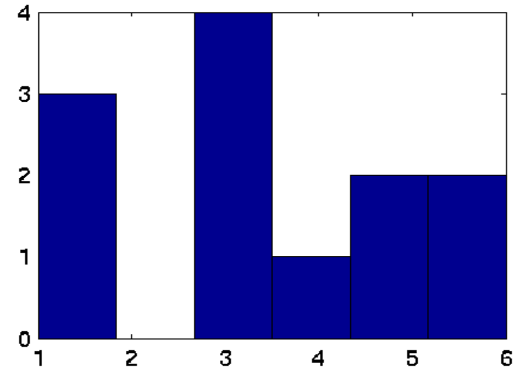
$$L(\pi, \alpha, \gamma) = \sum_{i=1}^{n} -\#i \cdot \theta_i + \gamma \left(\sum_{i=1}^{n} e^{\theta_i} - 1 \right)$$

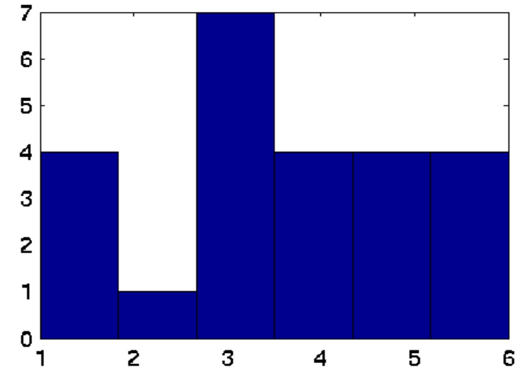
$$\partial_{\theta_i} = -\#i + \gamma e^{\theta_i} \text{ vanishes}$$

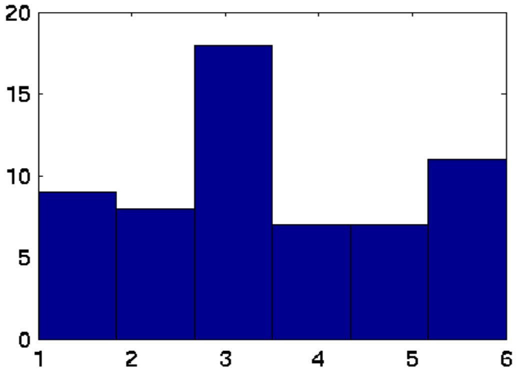
$$\Longrightarrow \pi_i = e^{\theta_i} = \frac{\#i}{\gamma}$$

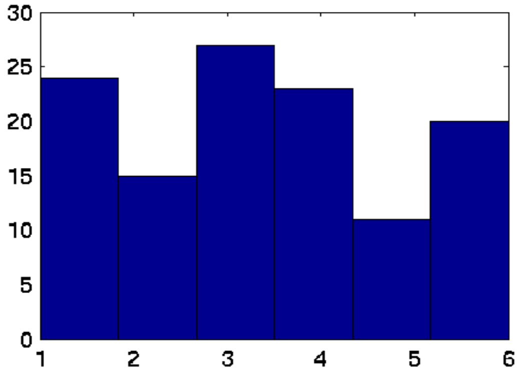
Finally, the sum constraint is satisfied if $\gamma = \sum_i \#i$.

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, **not uniformly over all i**.

Improved Approach

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

Summary

Data

What to do with data

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

Statistics and probability theory

- Probability of an event
- Dependence, independence, conditional probability
- Bayes rule, Hypothesis testing
- Maximum Likelihood Estimation
- Confidence bounds

An Introduction to Machine Learning with Kernels

Lecture 2

Alexander J. Smola
Alex.Smola@nicta.com.au

Statistical Machine Learning Program
National ICT Australia, Canberra

Day 1

Machine learning and probability theory

Introduction to pattern recognition, classification, regression, novelty detection, probability theory, Bayes rule, inference

Density estimation and Parzen windows

Kernels and density estimation, Silverman's rule, Watson Nadaraya estimator, crossvalidation

Perceptron and kernels

Hebb's rule, perceptron algorithm, convergence, feature maps, kernel trick, examples

Support Vector classification

Geometrical view, dual problem, convex optimization, kernels and SVM

L2 Density estimation

Density estimation

- empirical frequency, bin counting
- priors and Laplace rule

Parzen windows

- Smoothing out the estimates
- Examples

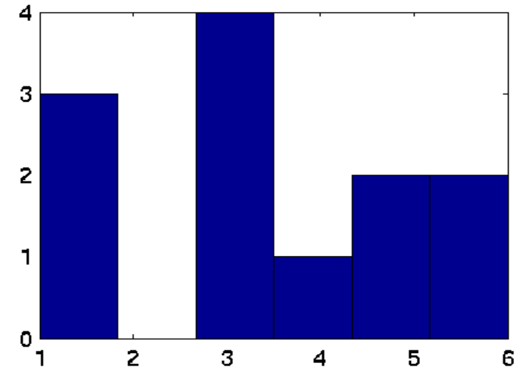
Adjusting parameters

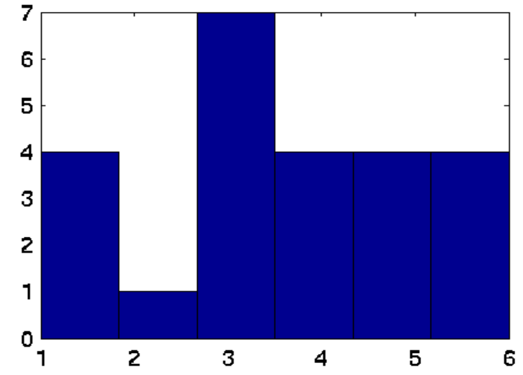
- Cross validation
- Silverman’s rule

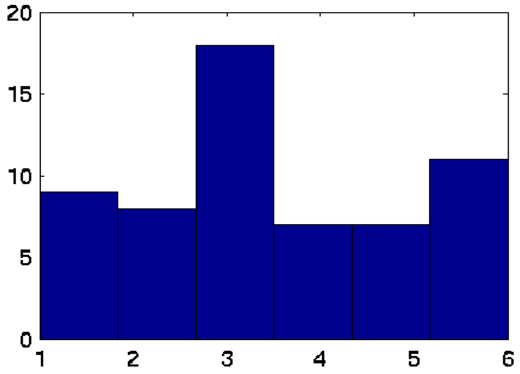
Classification and regression with Parzen windows

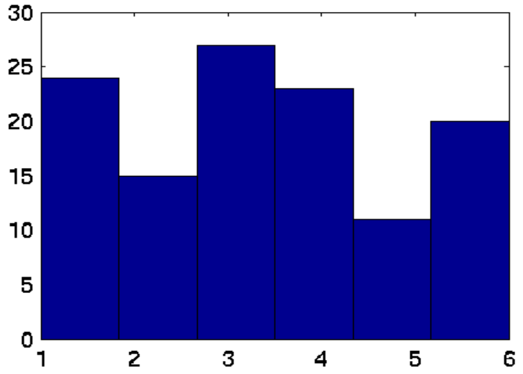
- Watson-Nadaraya estimator
- Nearest neighbor classifier

Tossing a dice (again)









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.

Priors

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

Practical Example

If we assume that the dice is even, then we can add m_0 observations to each event $1 \leq i \leq 6$. This yields

$$\pi_i = \frac{\#\text{occurrences of } i + u_i - 1}{\#\text{trials} + \sum_j (u_j - 1)}.$$

For $m_0 = 1$ this is the famous **Laplace Rule**.

Example: Dice

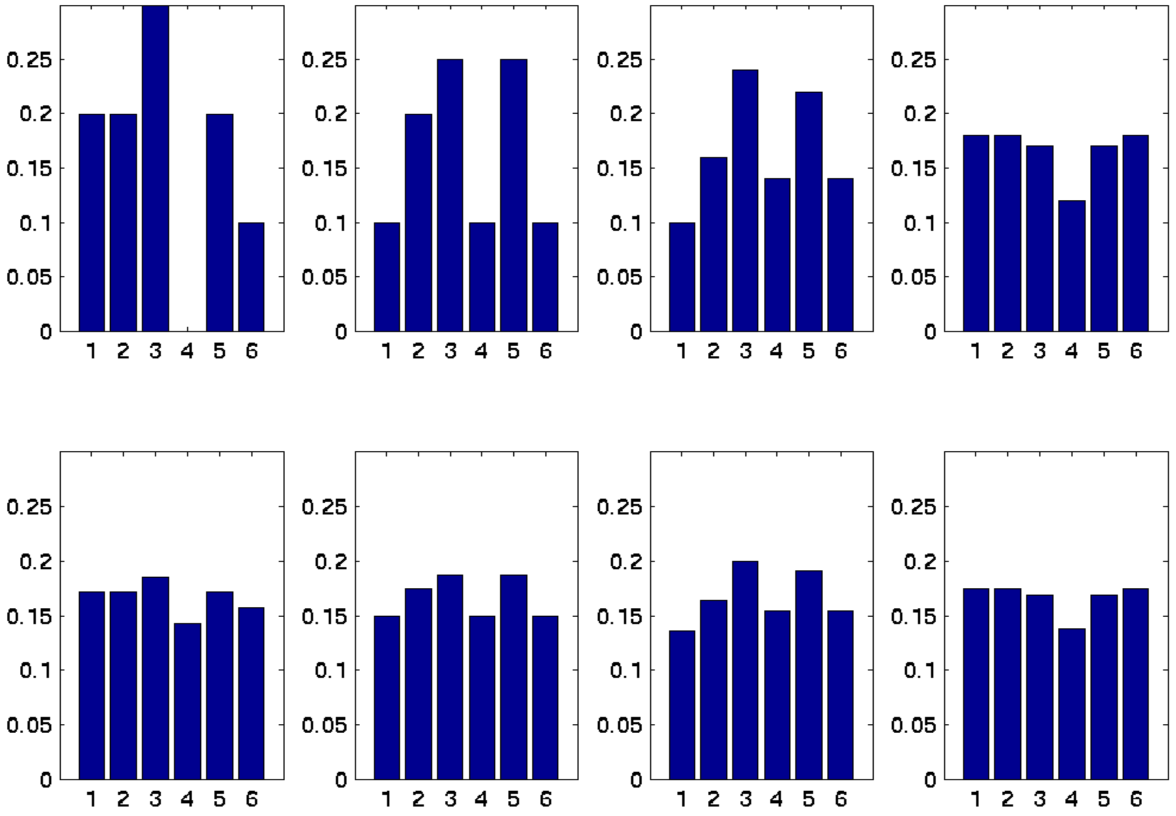
20 tosses of a dice

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

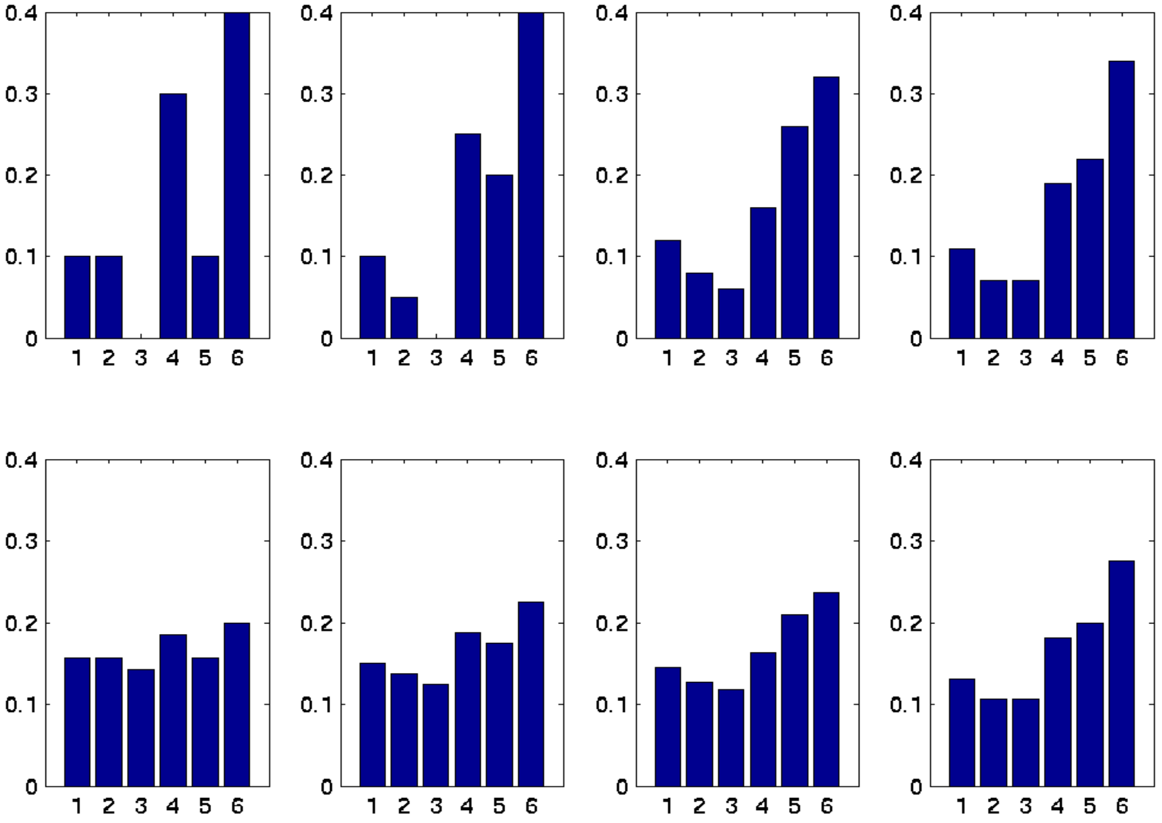
Consequences

- 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



Density Estimation

Data

Continuous valued random variables.

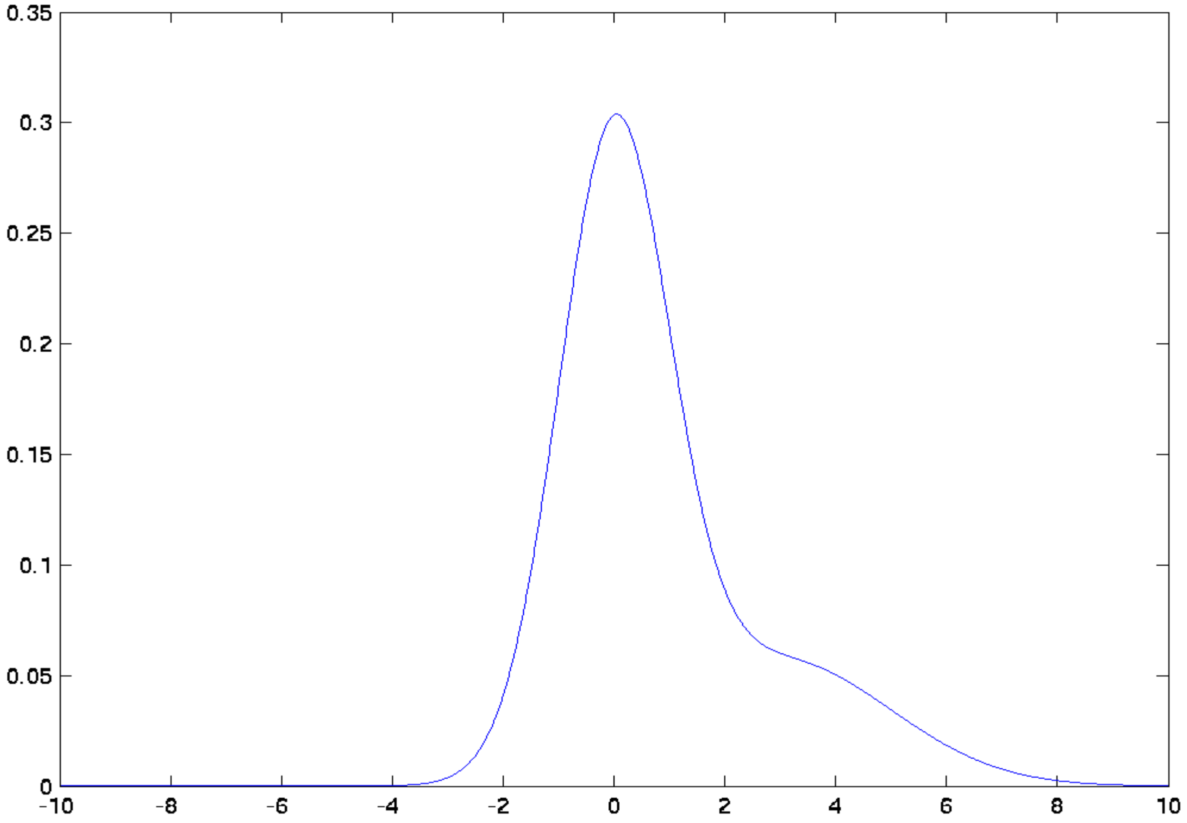
Naive Solution

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

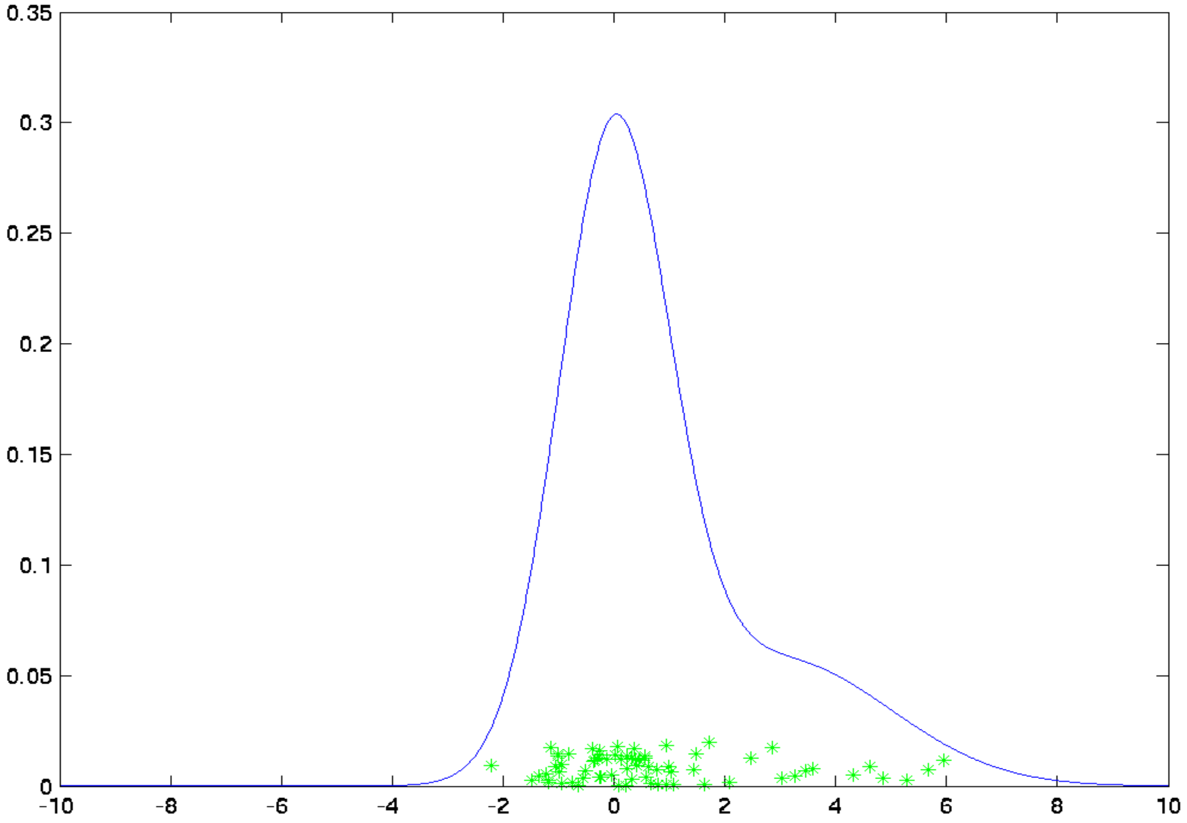
Problems

- 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 $\mathbb{R}^{10}$ $\ldots$

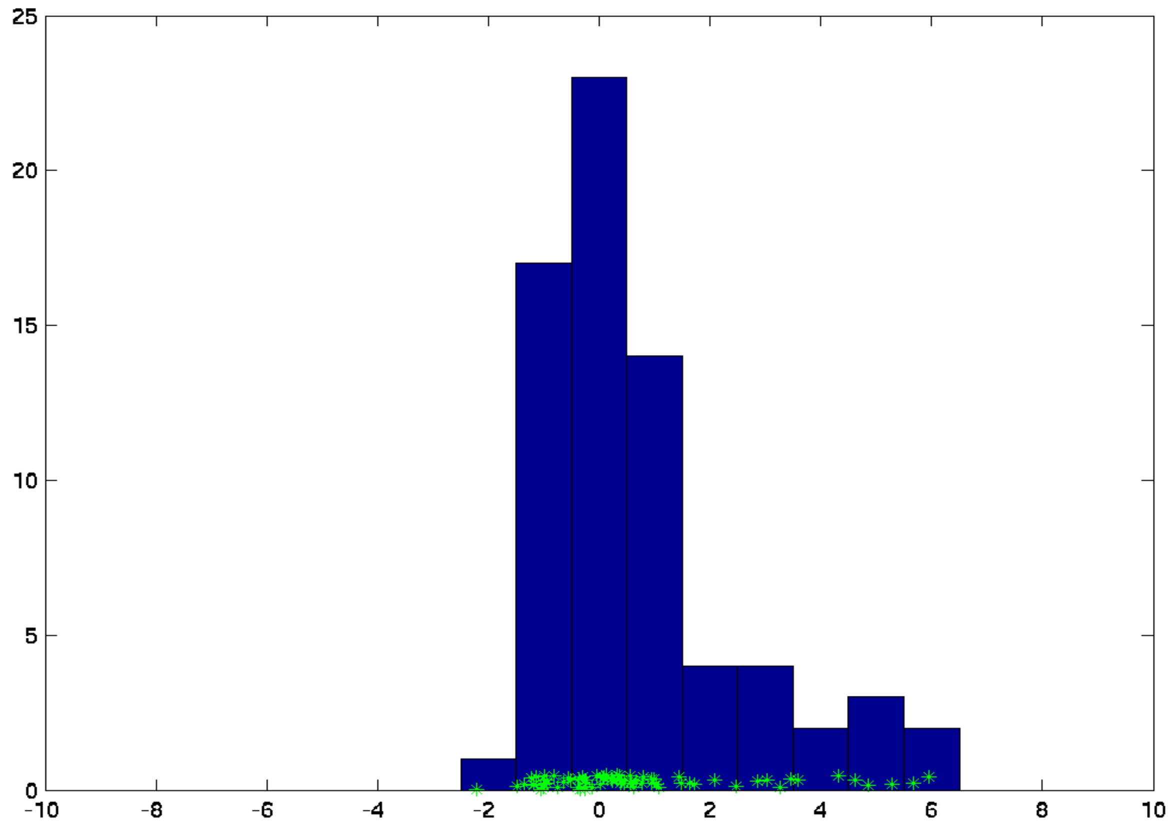
Mixture Density



Sampling from $p(x)$



Bin counting



Parzen Windows

Naive approach

Use the empirical density

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

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 $k(x, x')$.

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

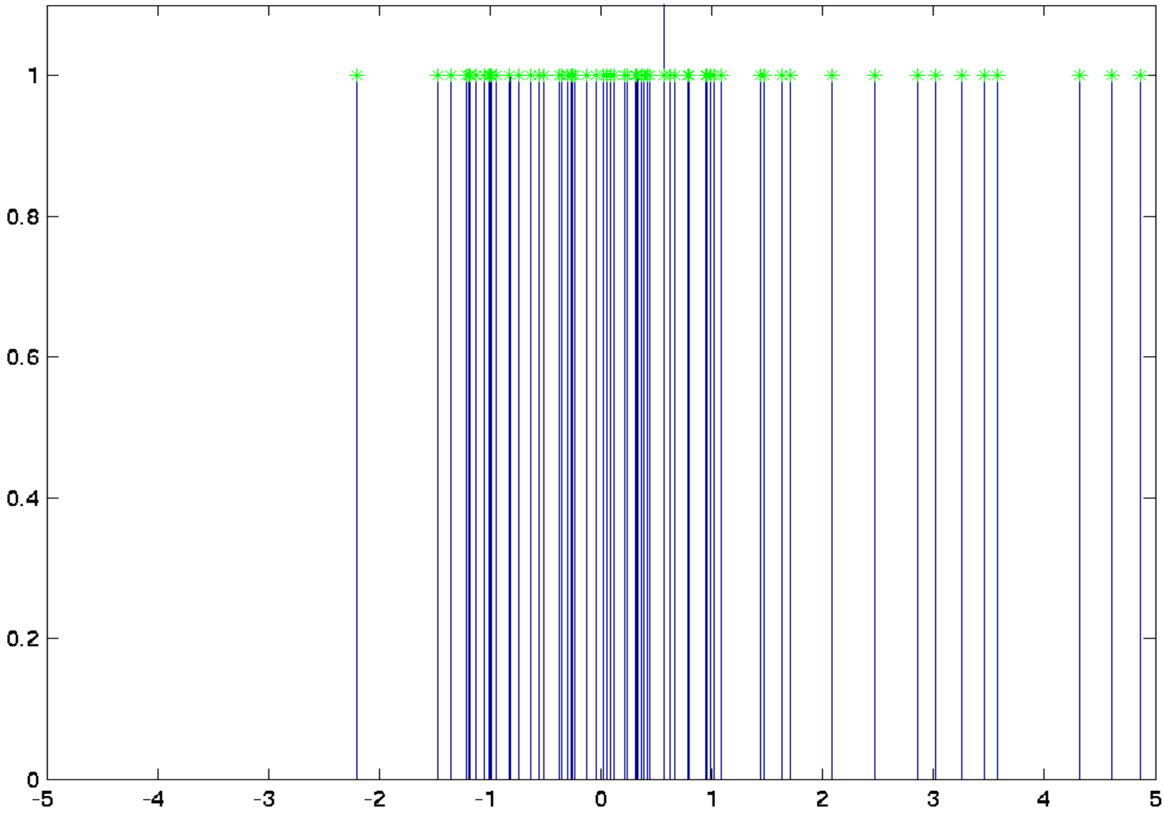
Adjusting the kernel width

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

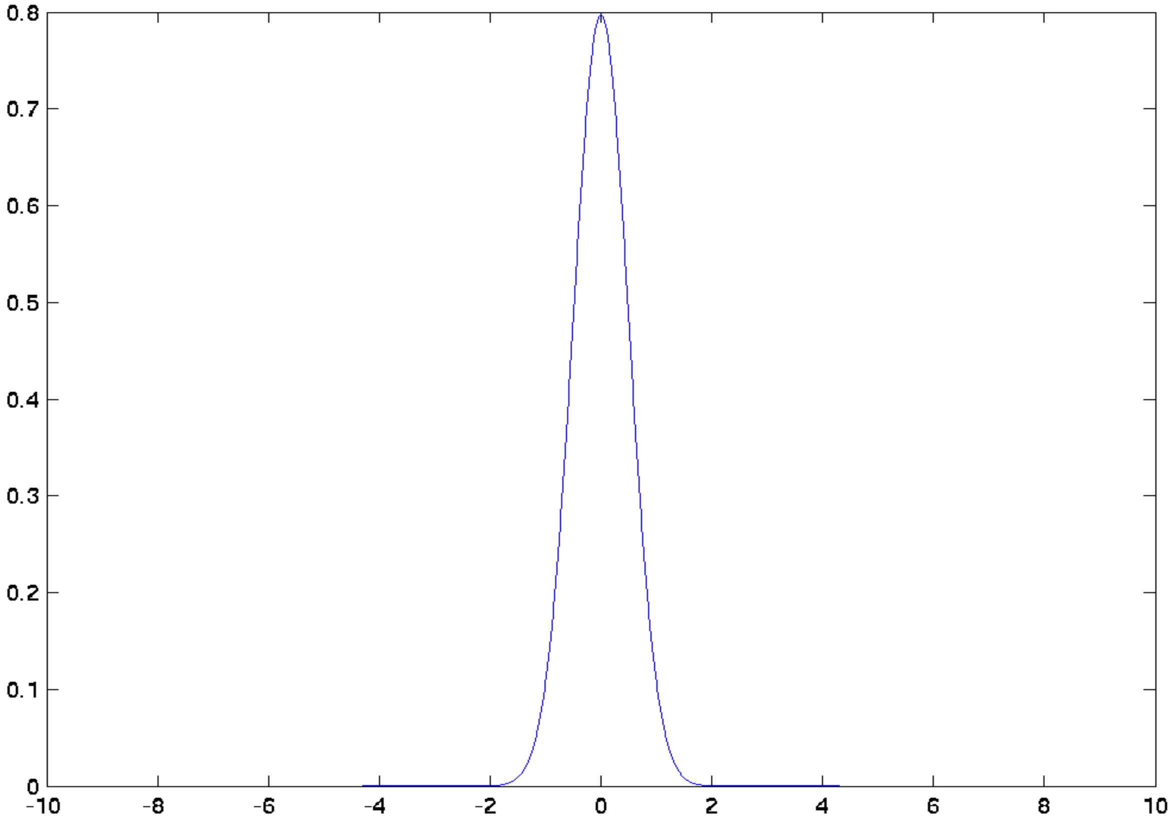
$$k_r(x, x') := r^n k(rx, rx')$$

Here n is the dimensionality of x.

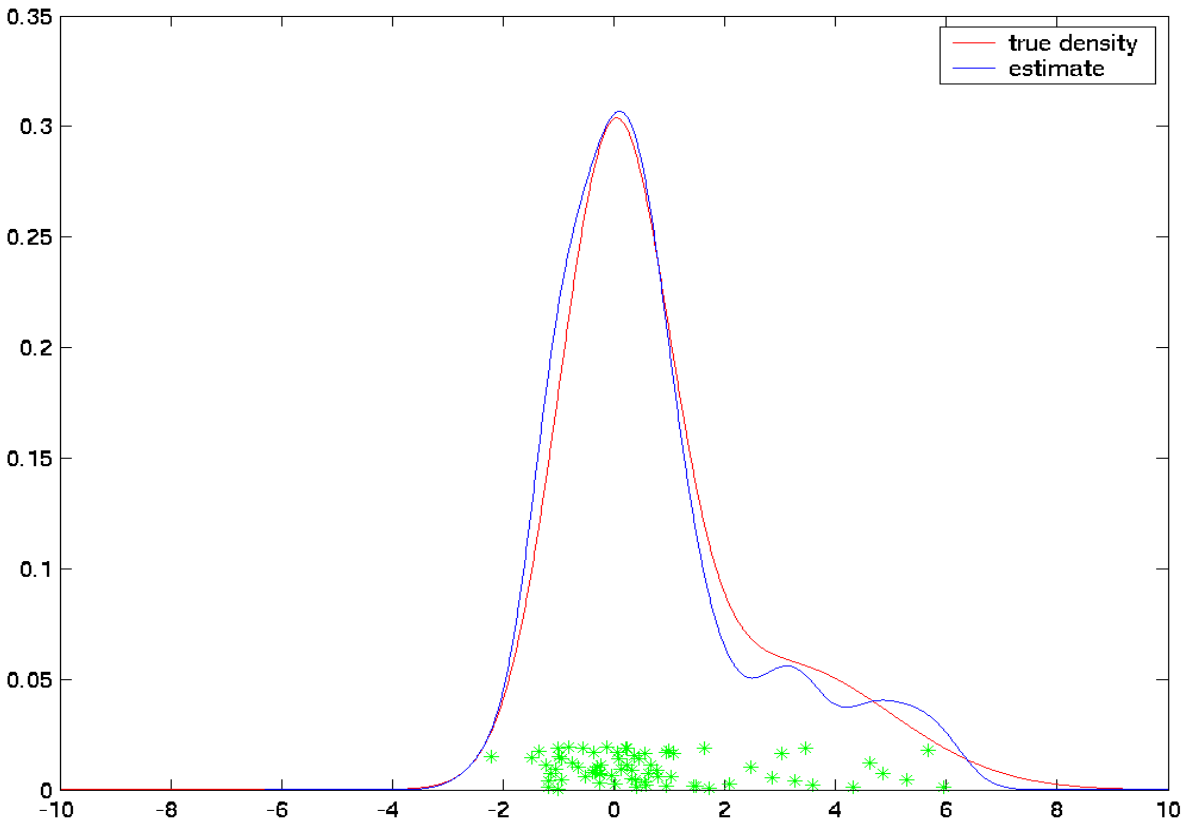
Discrete Density Estimate



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

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

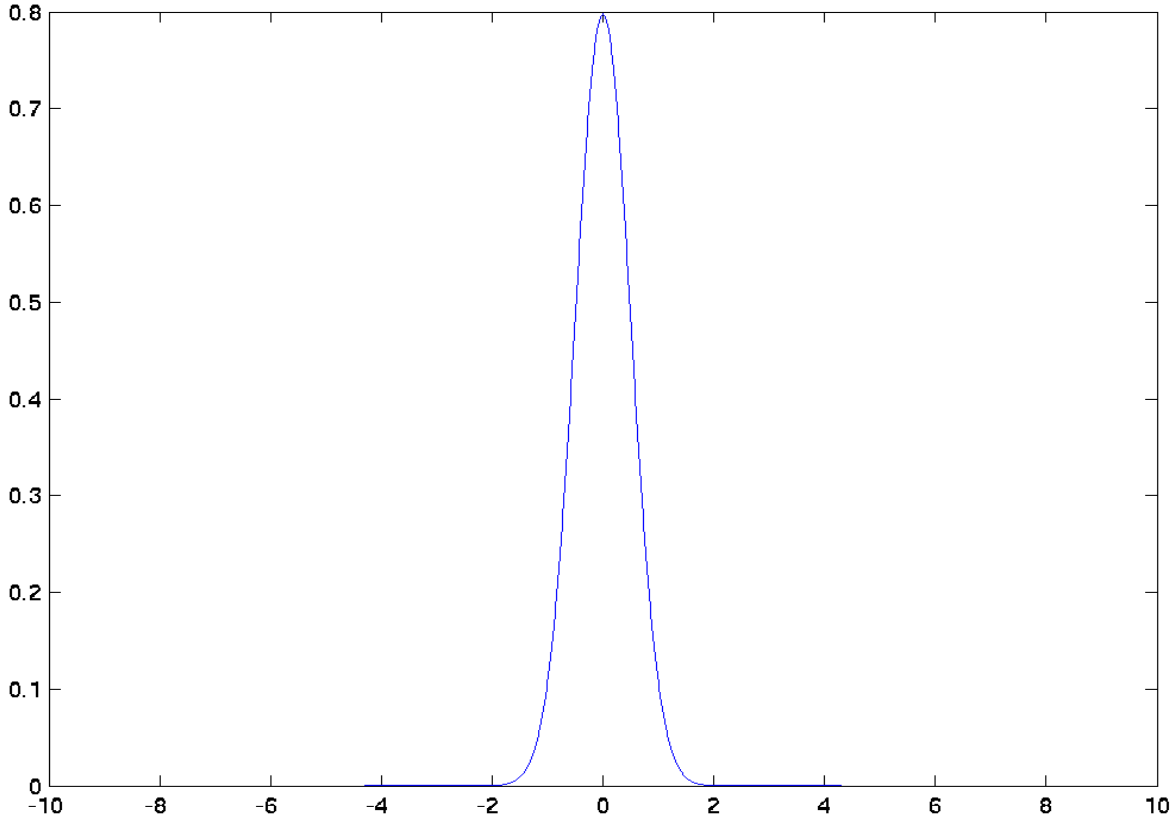
Indicator Kernel

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

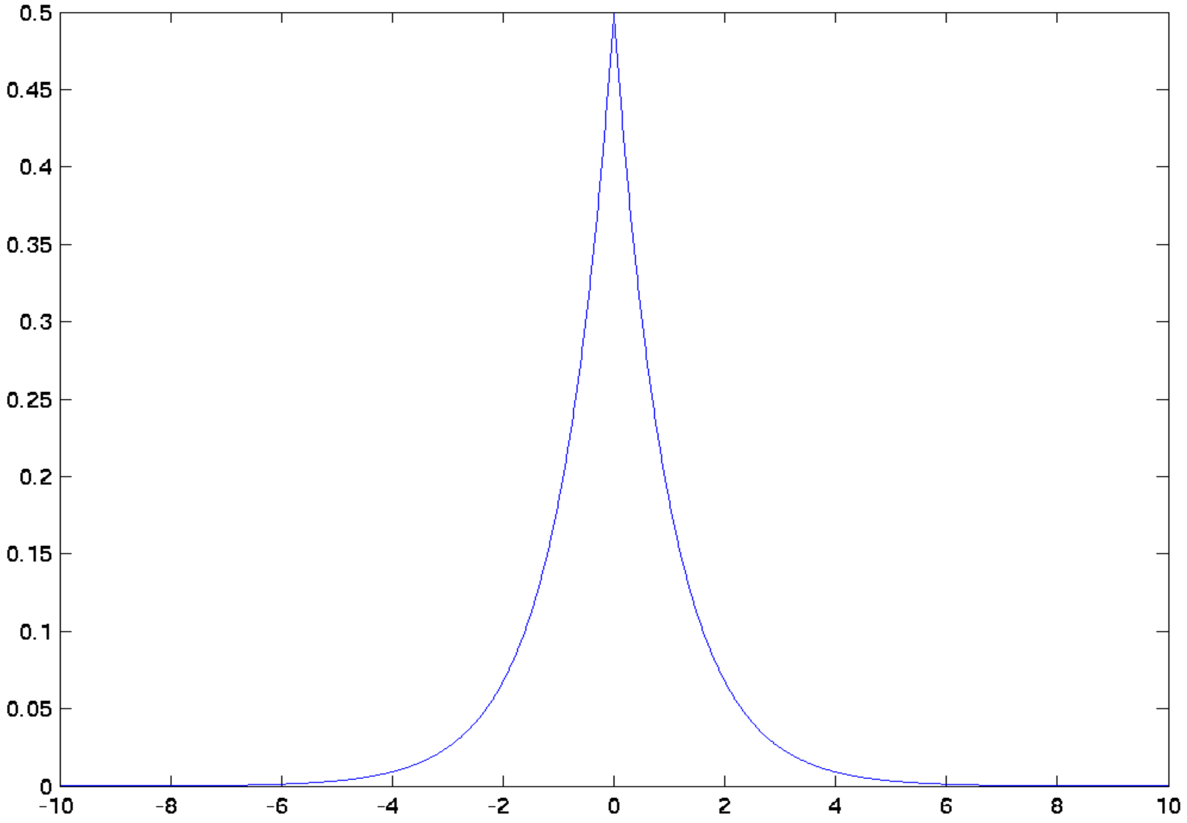
Important Issue

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

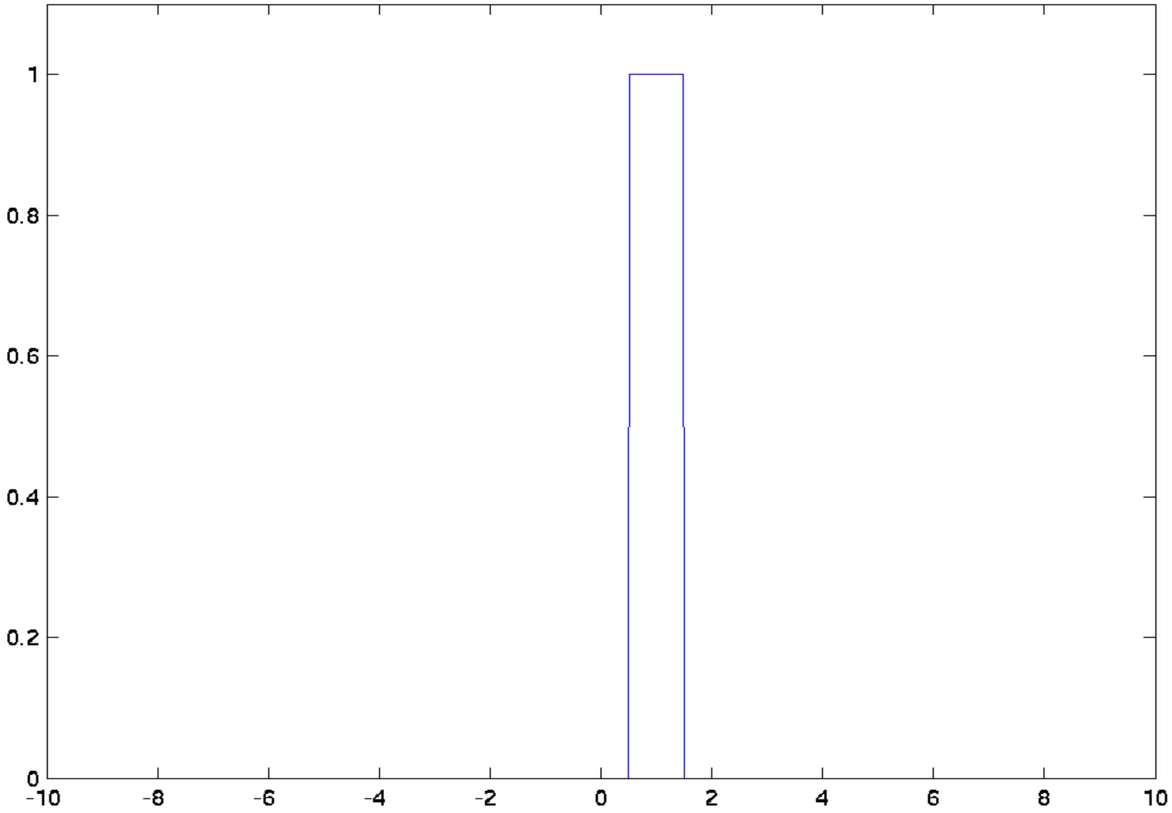
Gaussian Kernel



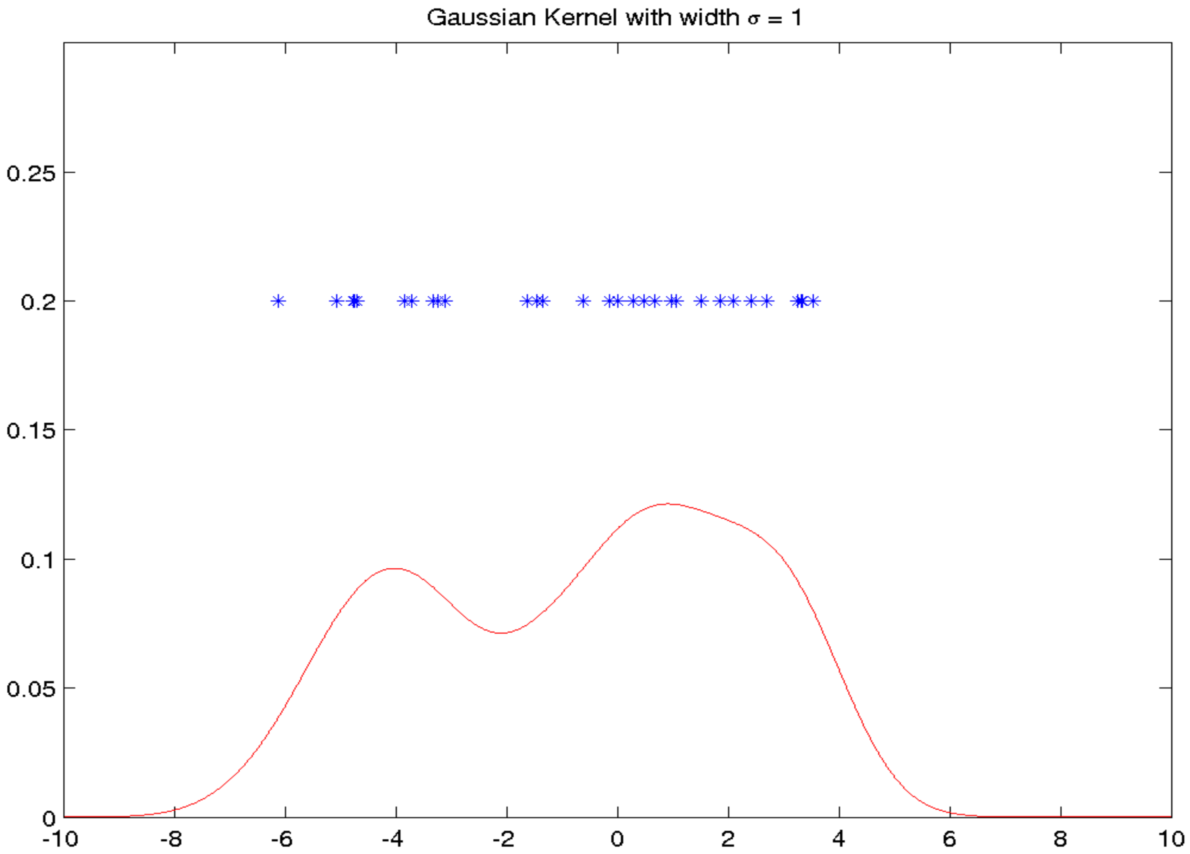
Laplacian Kernel



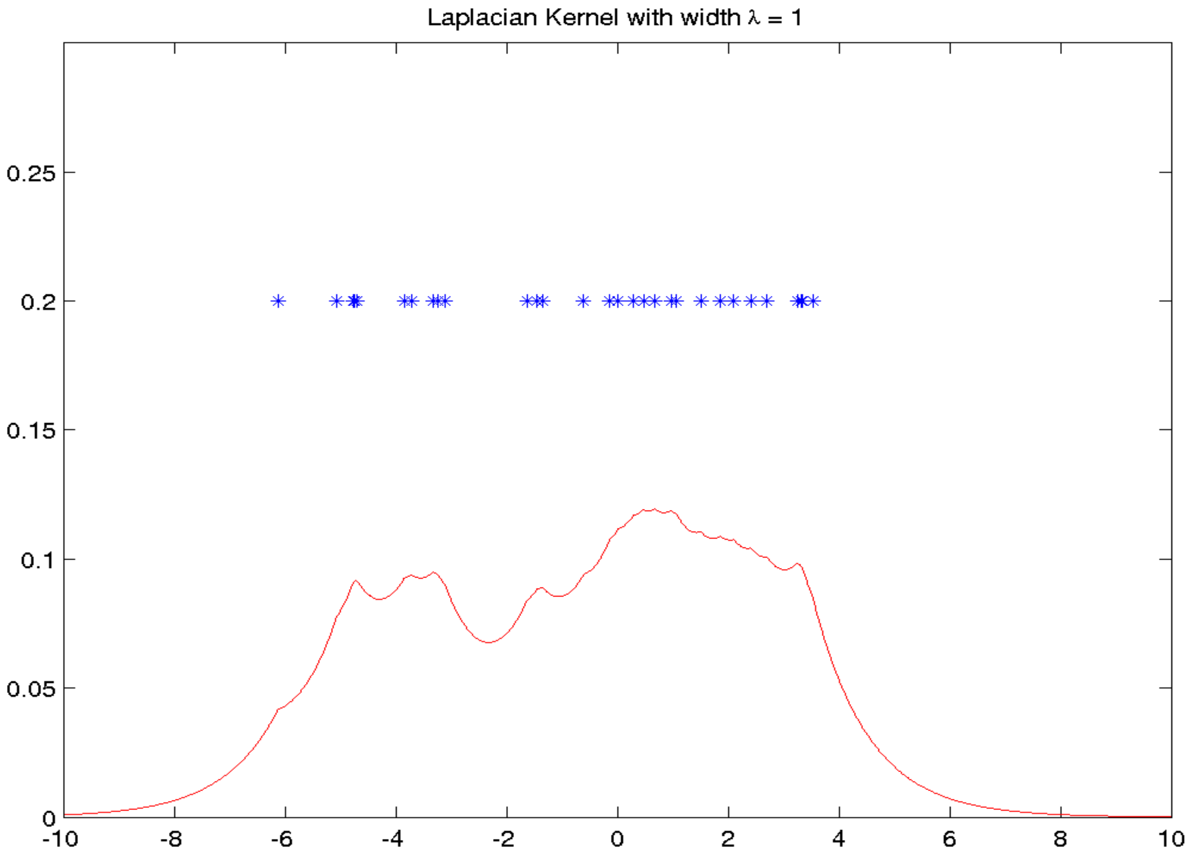
Indicator Kernel



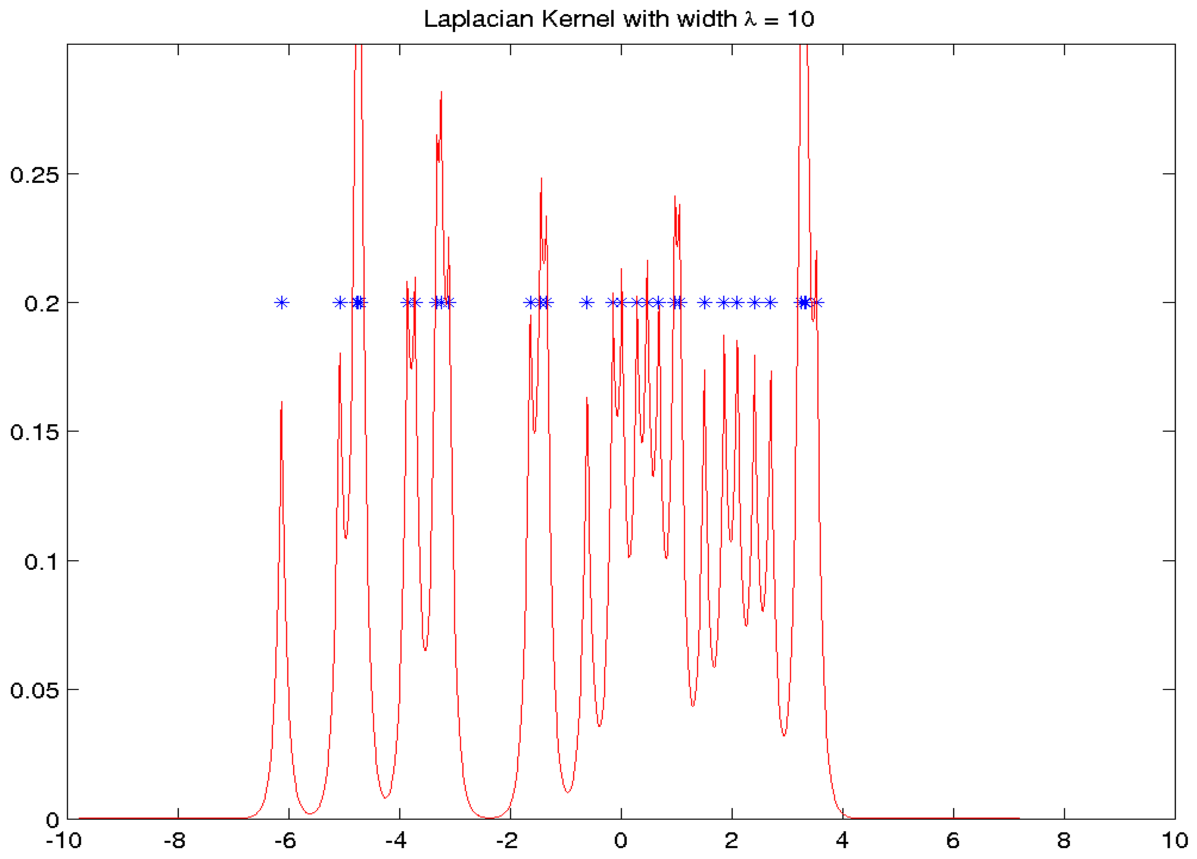
Gaussian Kernel



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

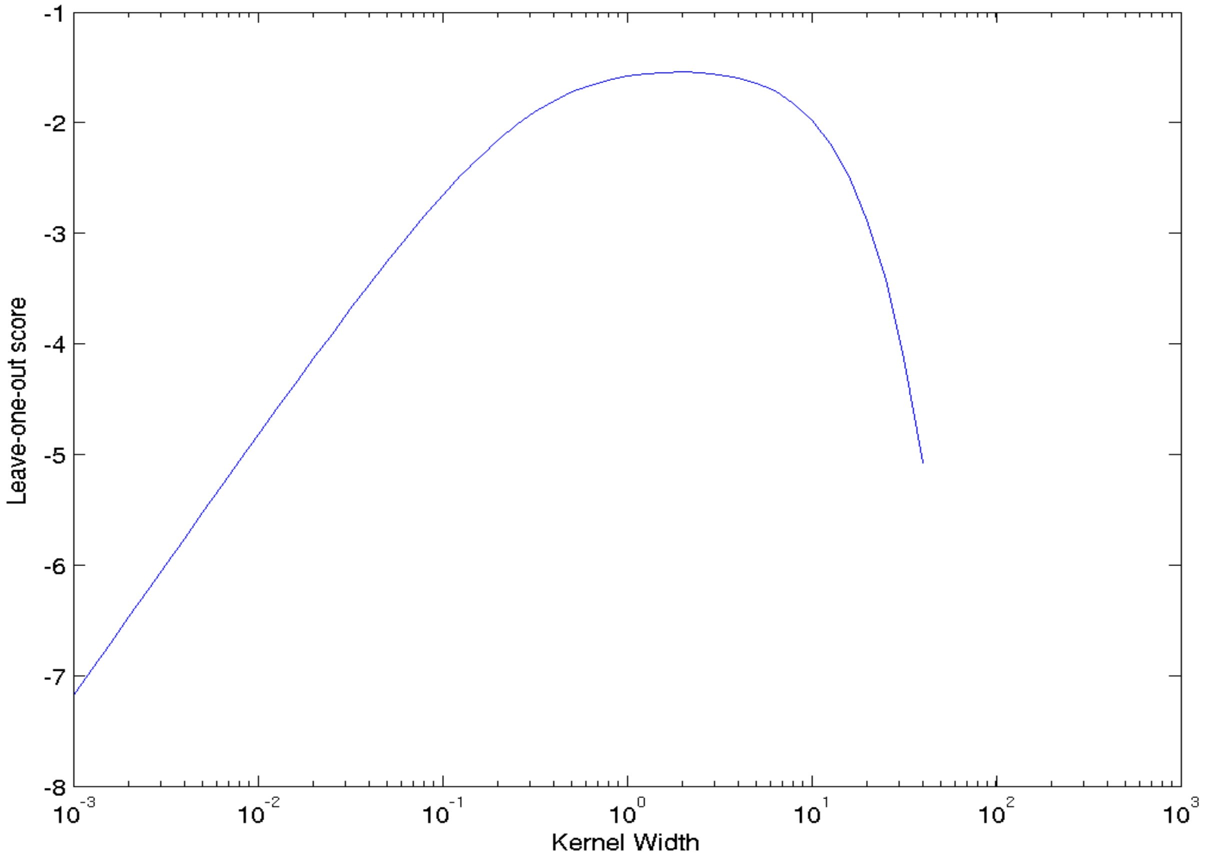
$$\frac{1}{n}\sum_i^n \frac{1}{|X_i|}\log p(X_i|\theta(X|\backslash X_i))$$

Pick the parameter which maximizes the above estimate.

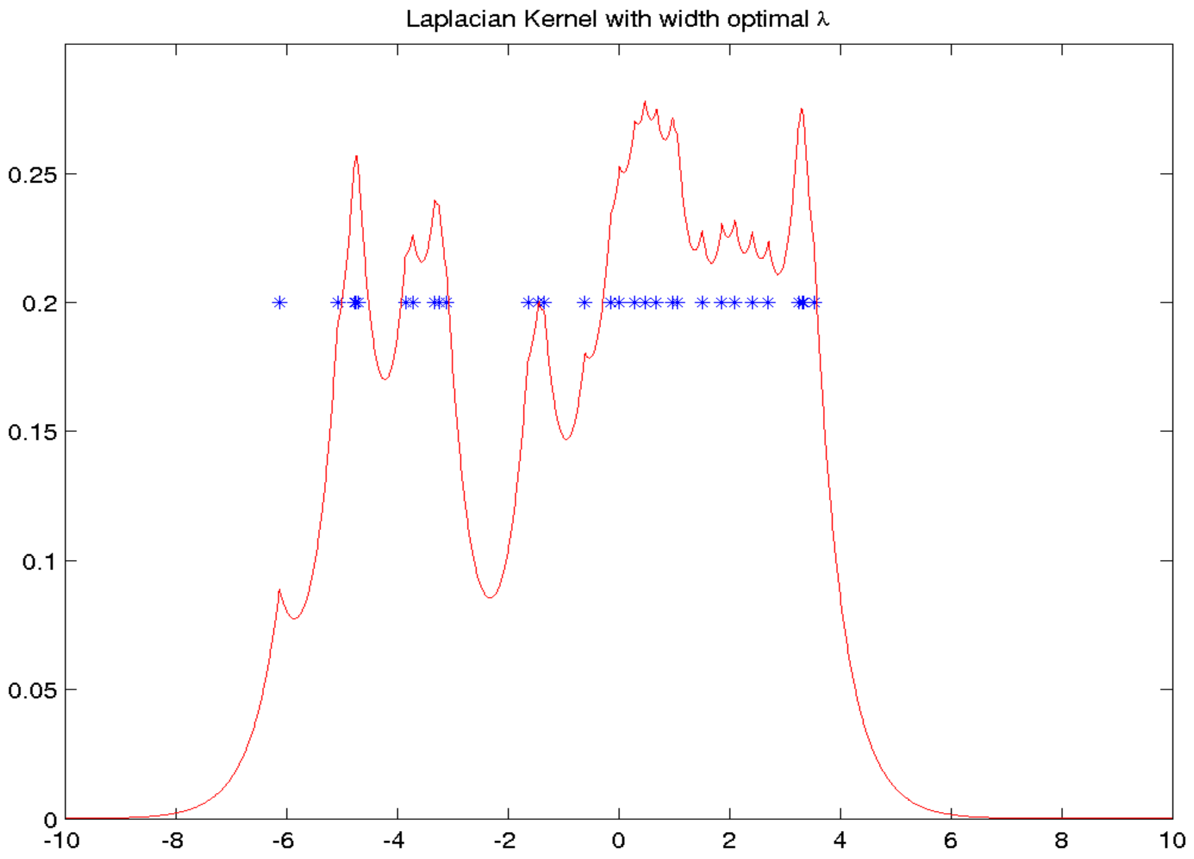
Special Case: Leave-one-out Crossvalidation

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

Cross Validation



Best Fit ($\lambda = 1.9$)



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.

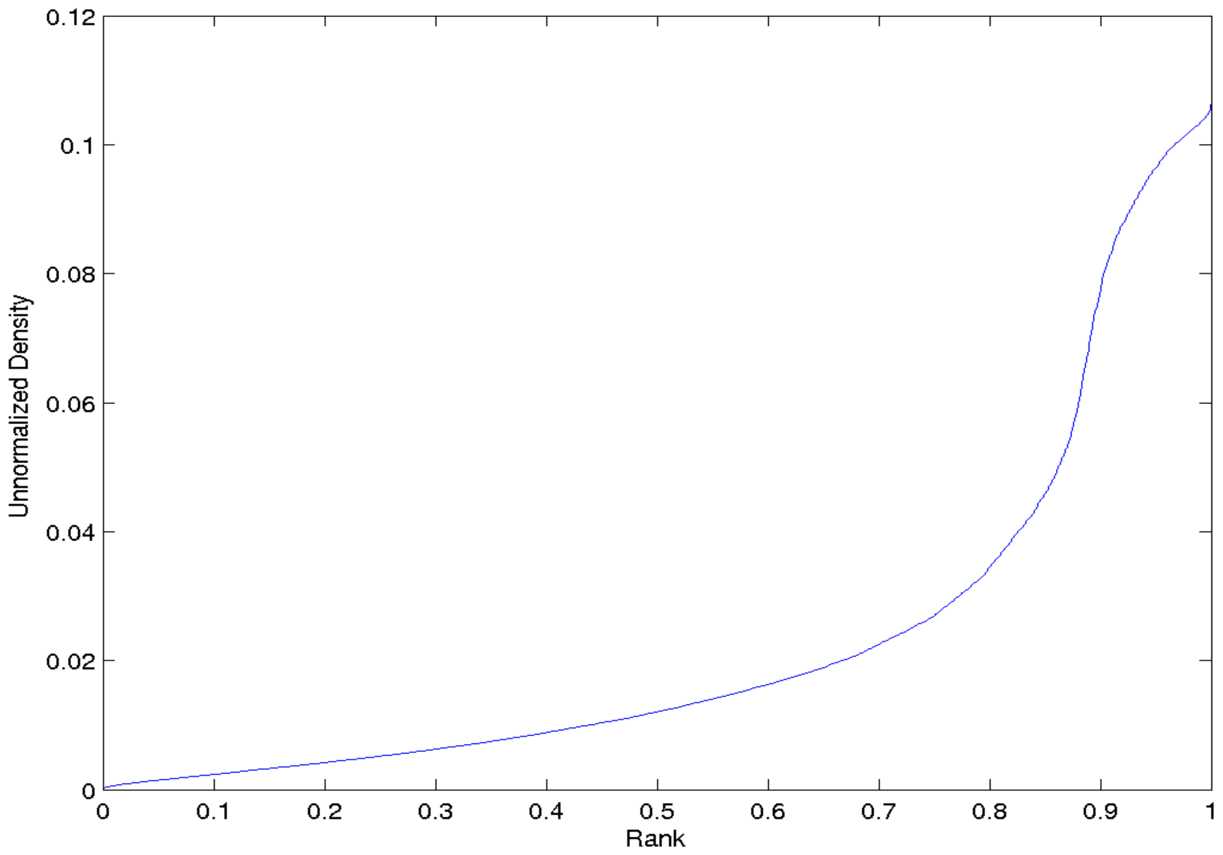
Fraud Detection

Credit Cards, Telephone Bills, Medical Records

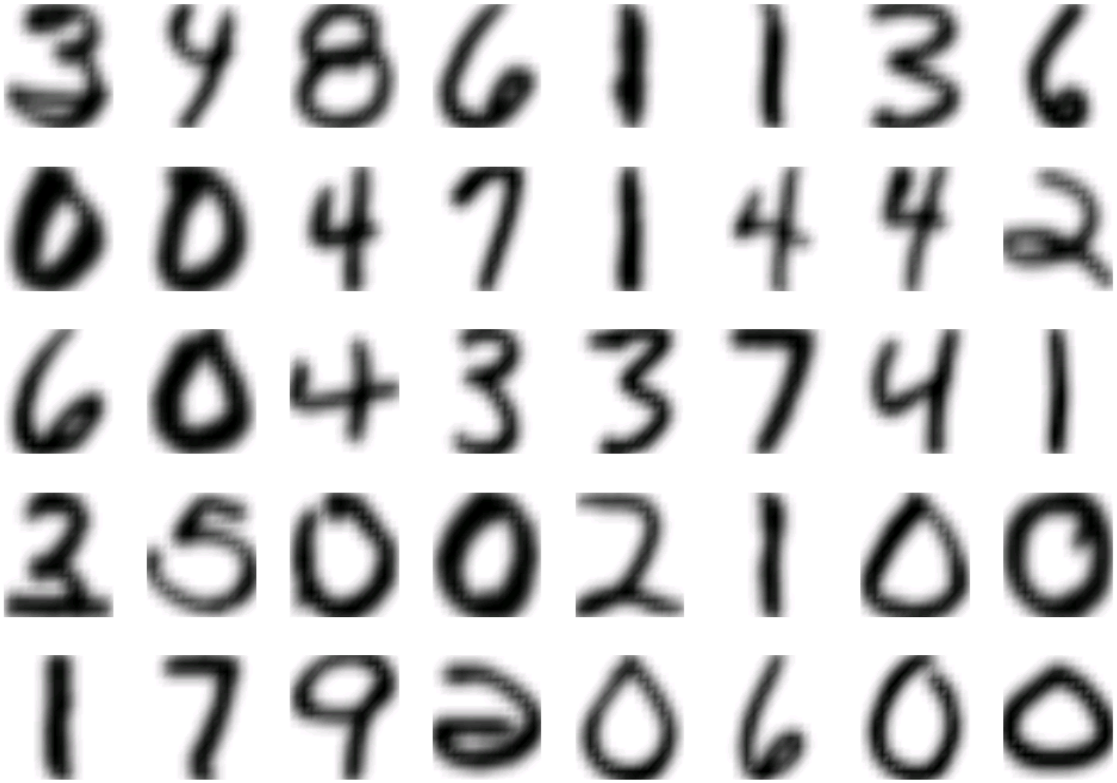
Self calibrating alarm devices

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

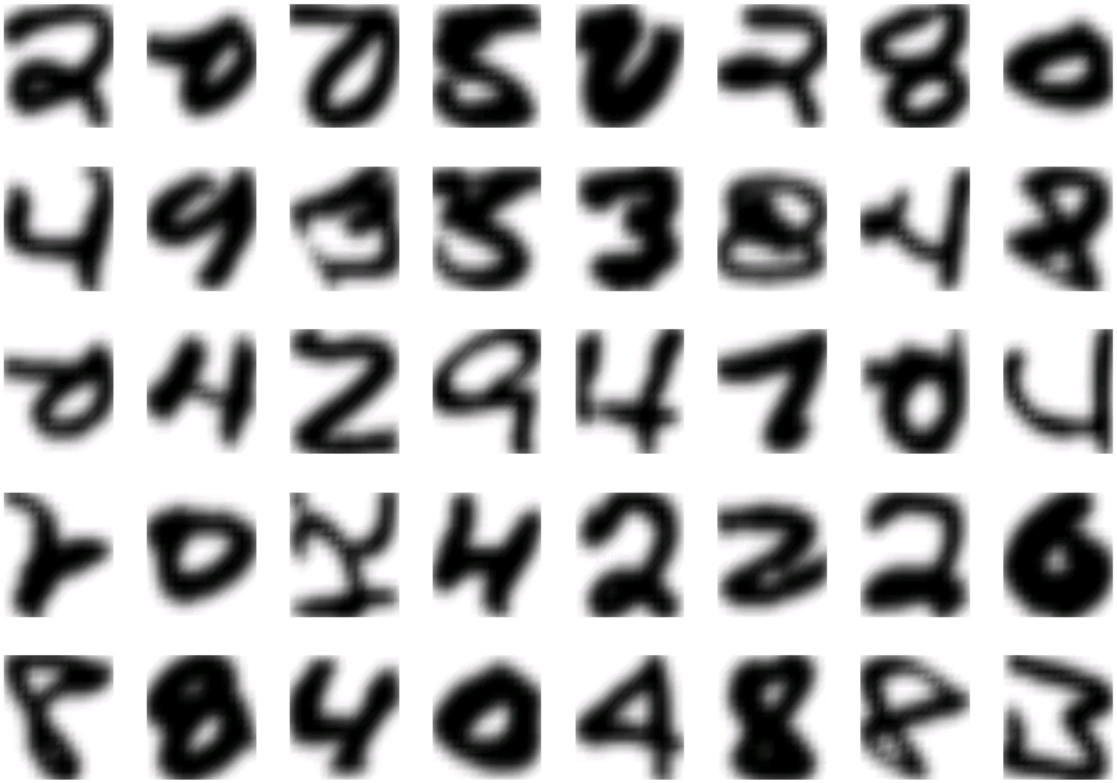
Order Statistic of Densities



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

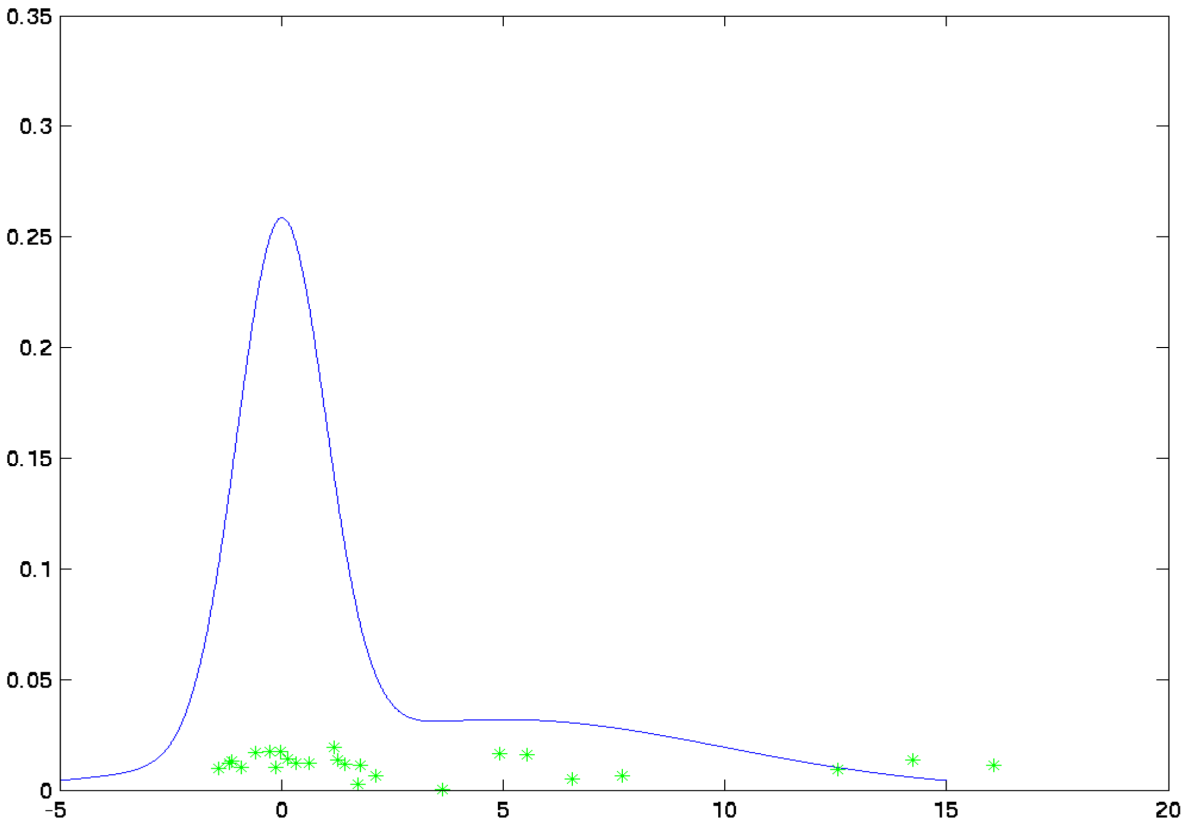
$$h_i = \frac{r}{k} \sum_{x_j \in \mathrm{NN}(x_i, k)} \|x_j - x_i\|$$

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.

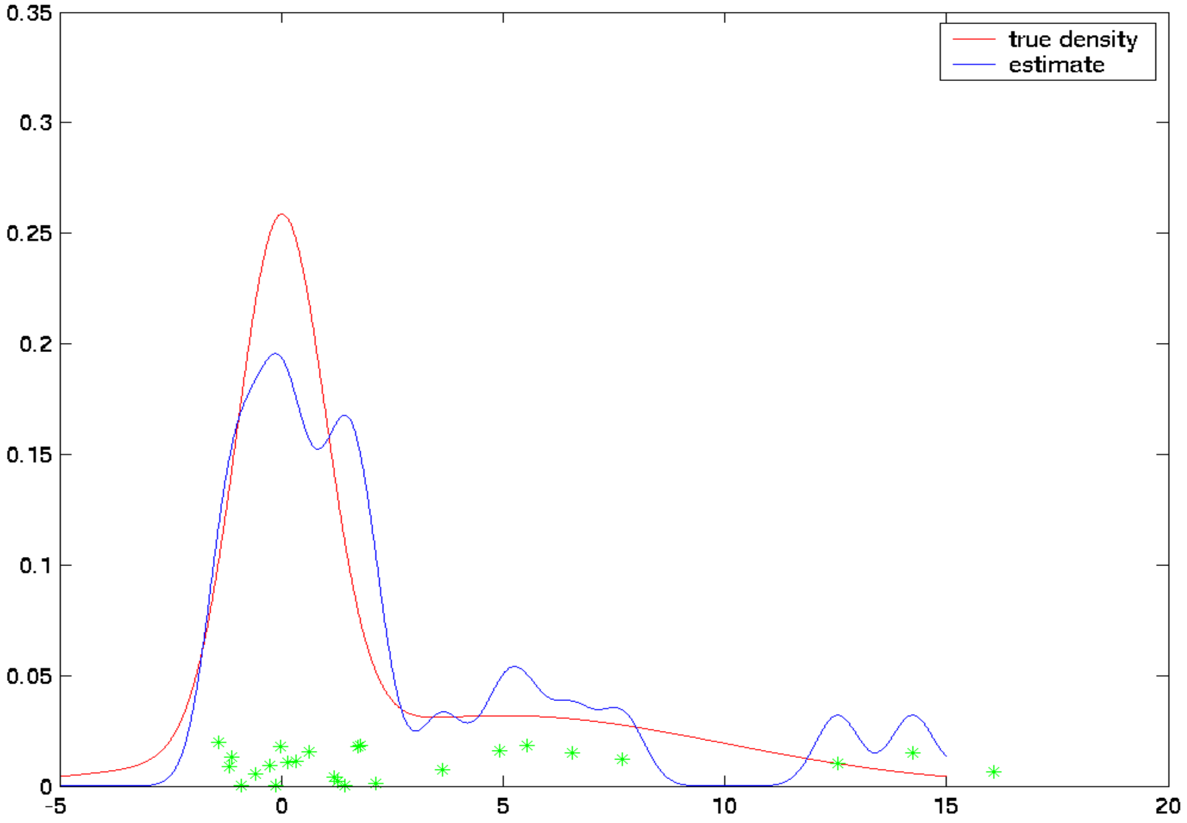
Result

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

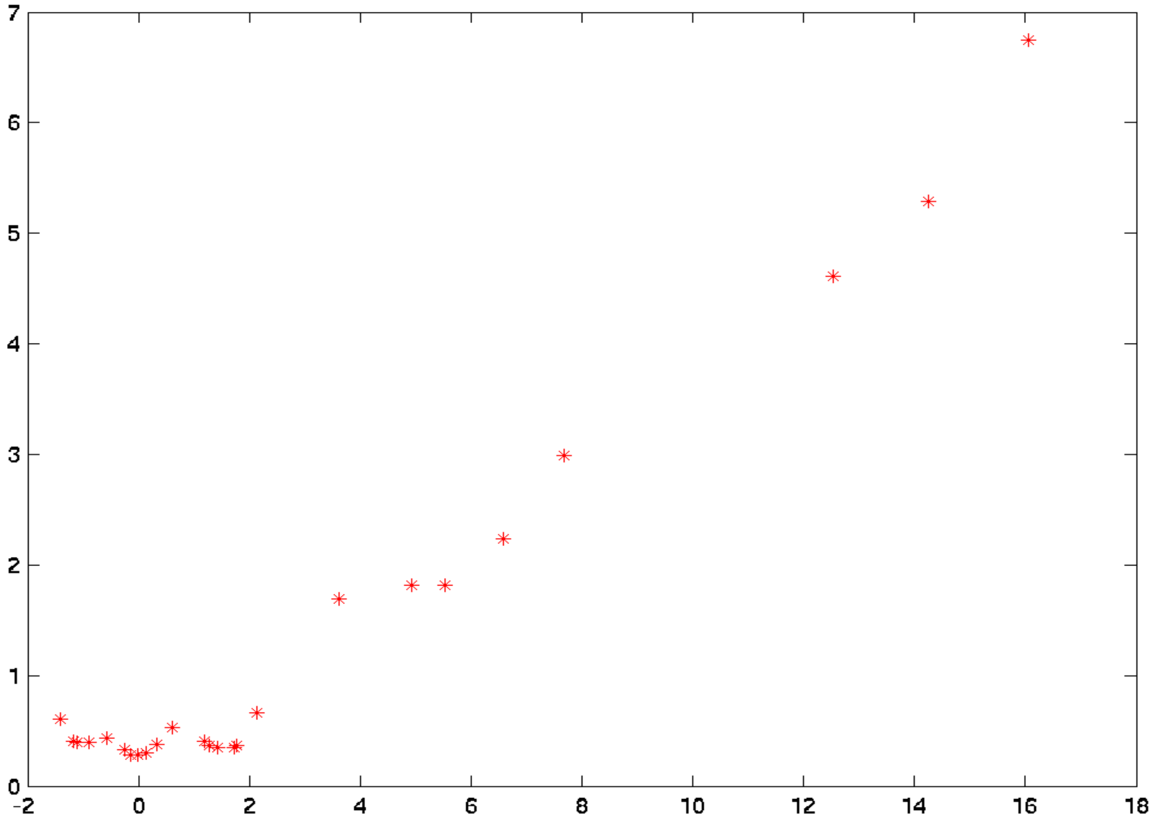
Sampling from $p(x)$



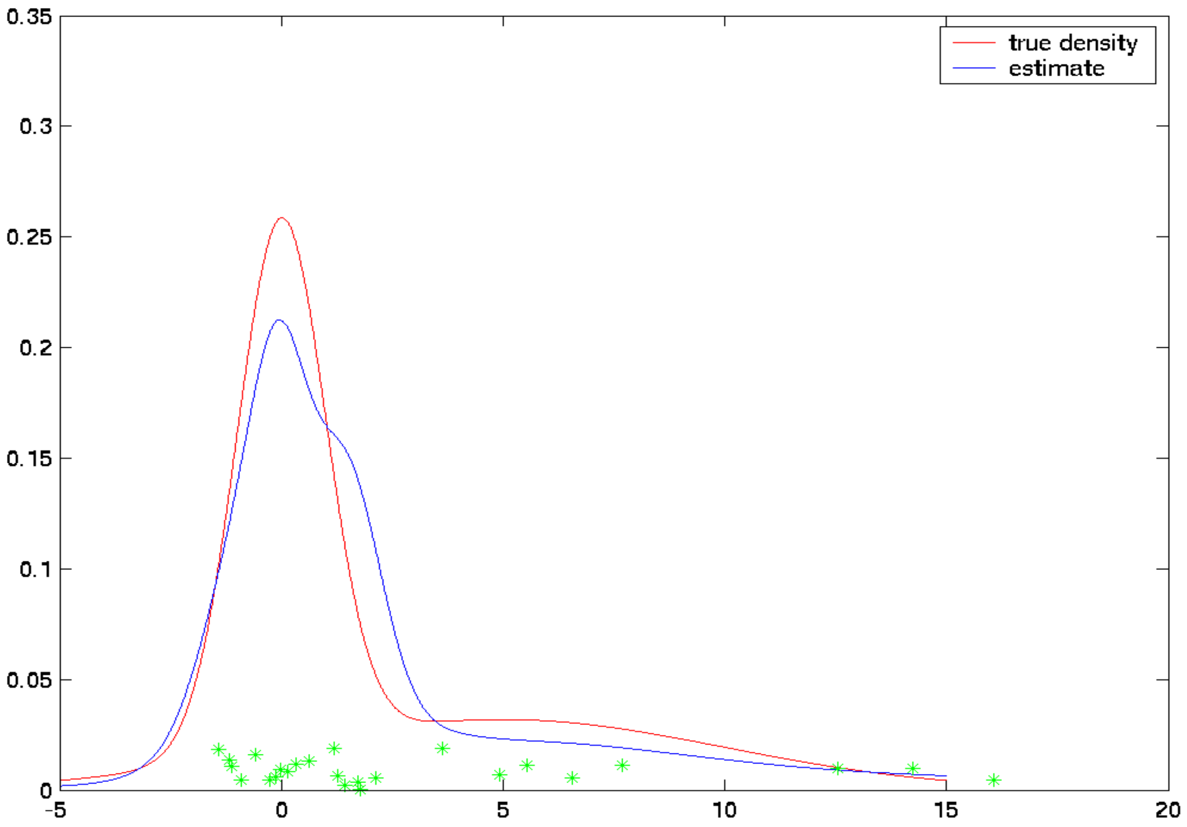
Uneven Scales



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

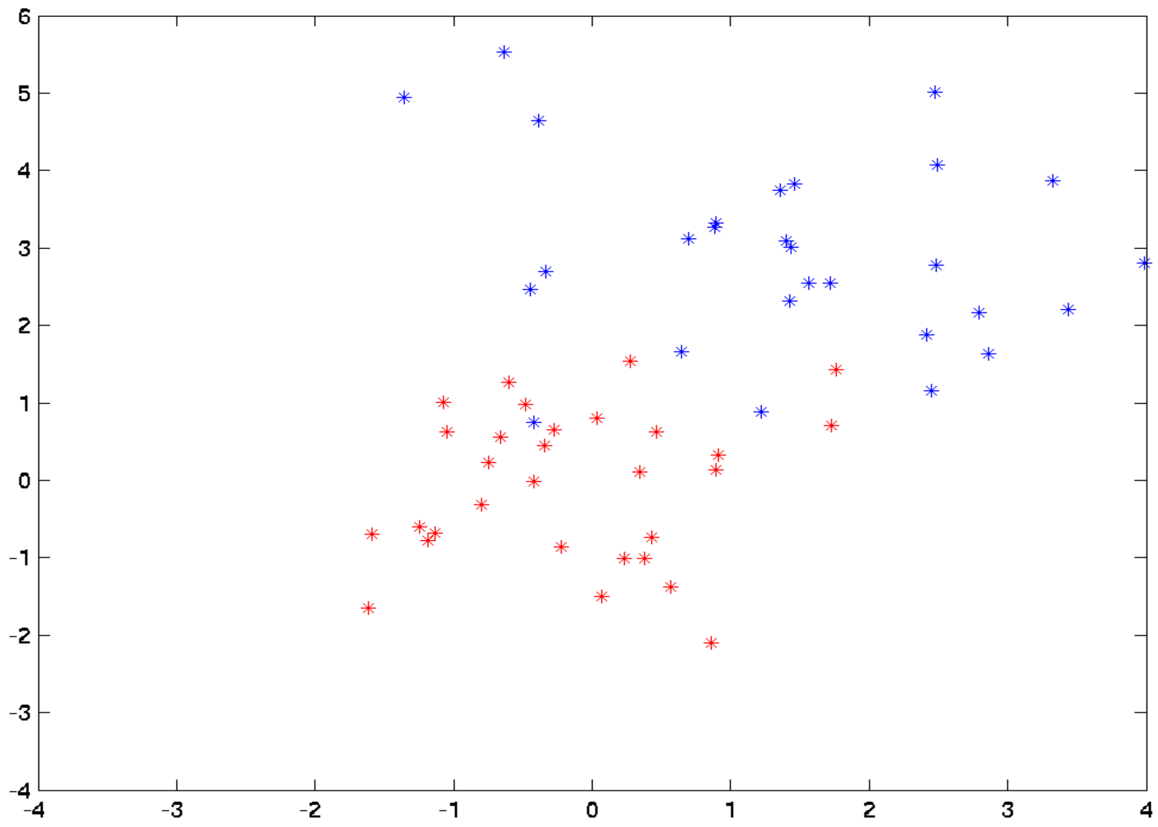
$$\Pr(y=1|x) = \frac{P(y=1,x)}{P(x)} = \frac{m^{-1}\sum_{y_i=1} k(x_i,x)}{m^{-1}\sum_i k(x_i,x)}$$

Bayes optimal decision

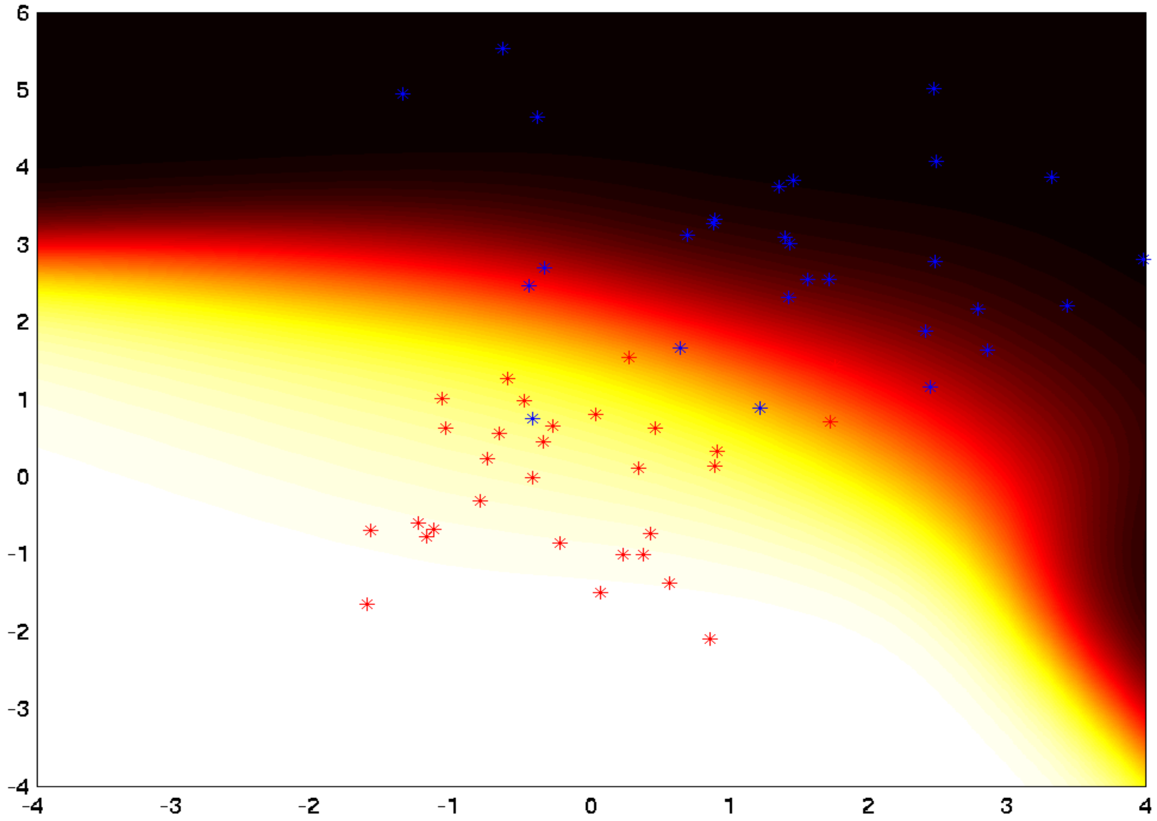
We want to classify $y = 1$ for $\Pr(y=1|x) > 0.5$. This is equivalent to checking the sign of

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

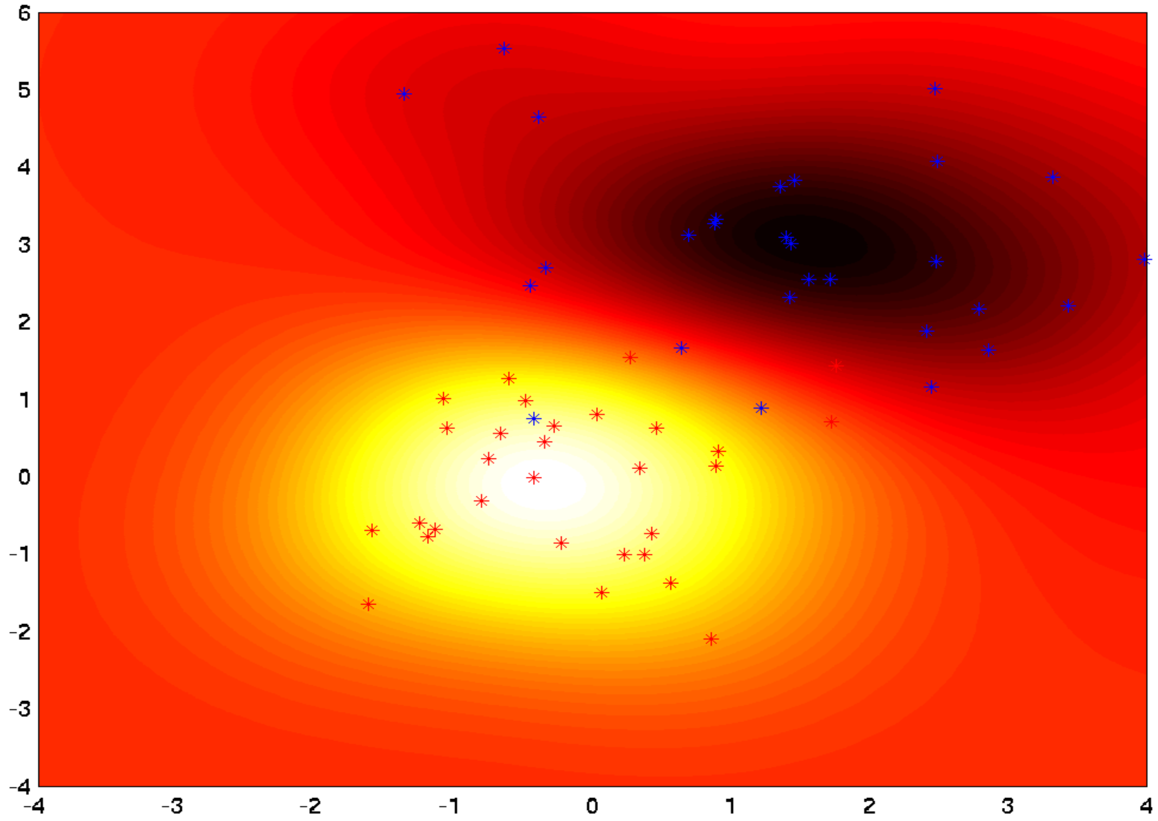
Training Data



Watson Nadaraya Classifier



Difference in Signs



Watson Nadaraya Regression

Decision Boundary

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

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

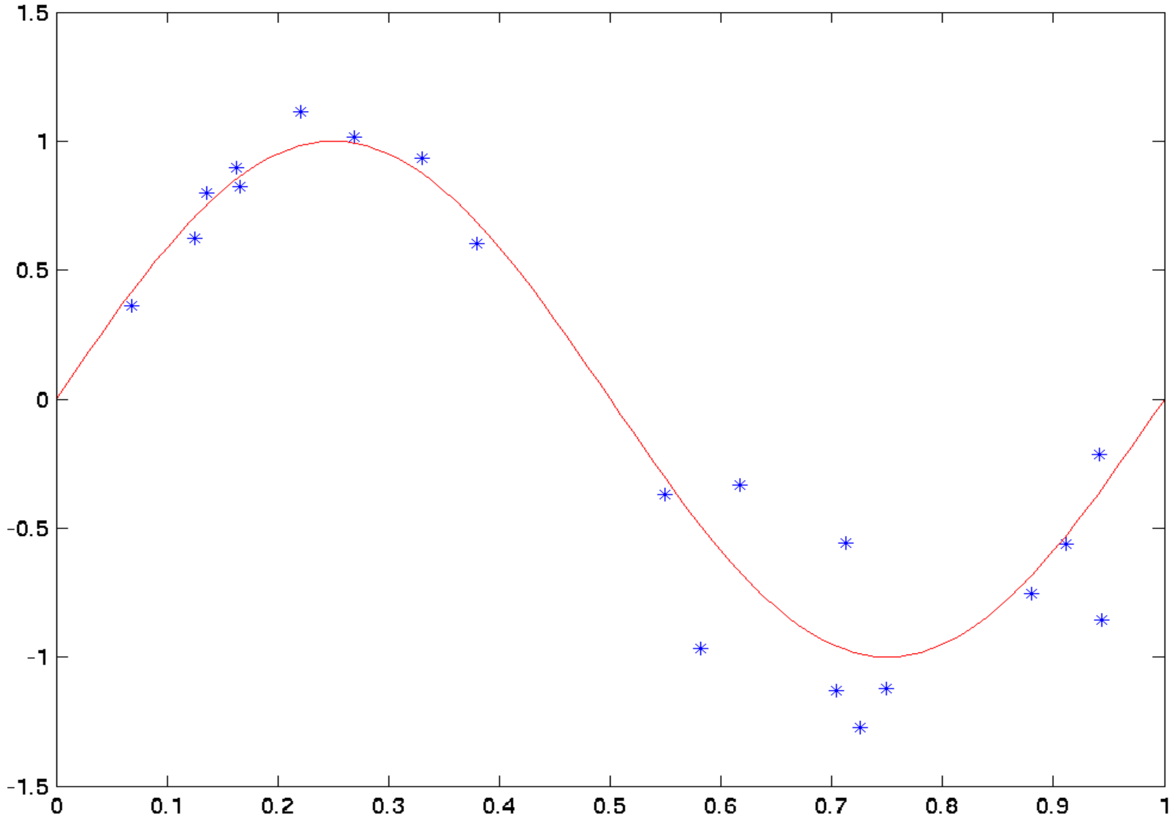
Extension to Regression

- Use the same equation for regression. This means that

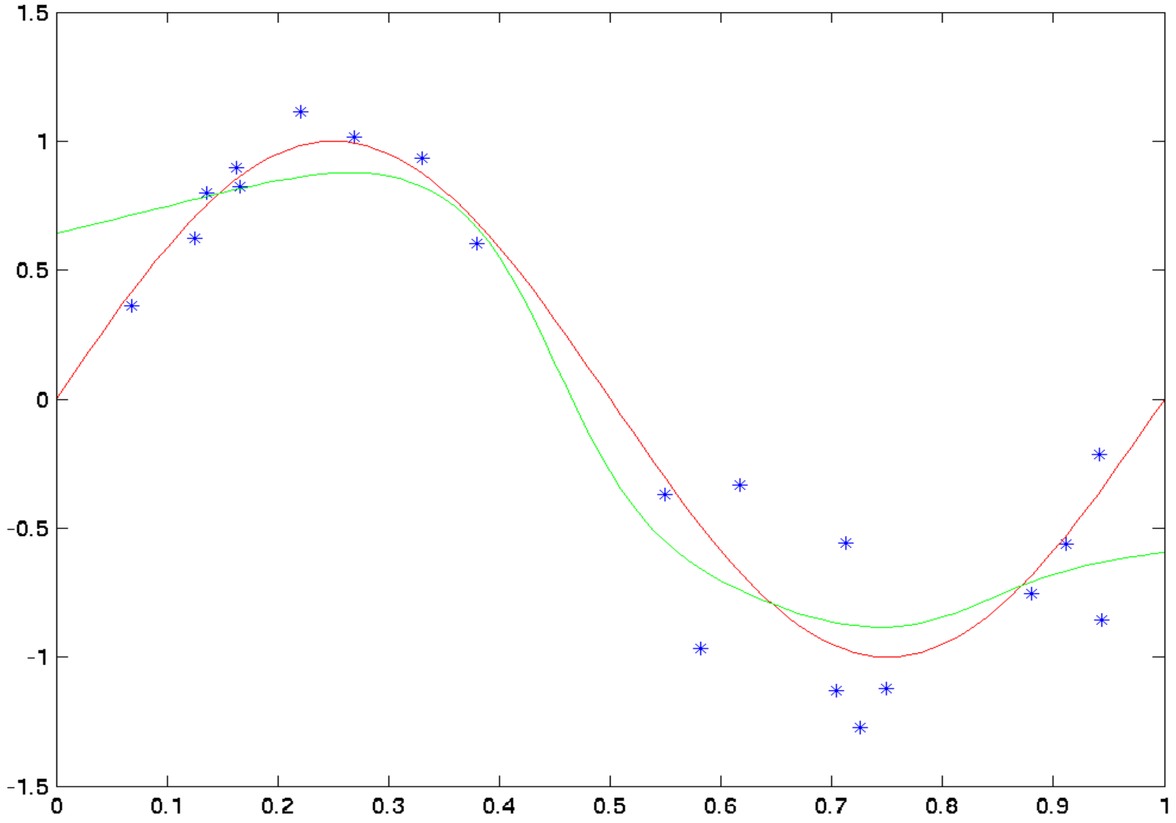
$$f(x) = \frac{\sum_i y_i k(x_i, x)}{\sum_i k(x_i, x)}$$

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

Regression Problem



Watson Nadaraya Regression



Nearest Neighbor Classifier

Extension of Silverman's trick

Use the density estimator for classification and regression.

Simplification

Rather than computing a *weighted* combination of labels to estimate the label, use an *unweighted* combination over the nearest neighbors.

Result

k-nearest neighbor classifier. Often used as baseline to compare a new algorithm.

Nice Properties

Given enough data, k-nearest neighbors converges to the best estimator possible (it is consistent).

Practical Implementation

Nearest Neighbor Rule

- Need distance measure between data
- Given data x, find nearest point x_i
- Classify according to the label y_i

k-Nearest Neighbor Rule

- Find k nearest neighbors of x
- Decide class of x according to majority of labels y_i.
- Hence prefer odd k.

Neighborhood Search Algorithms

- Brute force search (OK if data not too large)
- Random projection tricks (fast but difficult)
- Neighborhood trees (very fast, implementation tricky)

Baseline

Use k-NN as reference before fancy algorithms.

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 with Kernels

Lecture 3

Alexander J. Smola
Alex.Smola@nicta.com.au

Statistical Machine Learning Program
National ICT Australia, Canberra

Day 1

Machine learning and probability theory

Introduction to pattern recognition, classification, regression, novelty detection, probability theory, Bayes rule, inference

Density estimation and Parzen windows

Kernels and density estimation, Silverman's rule, Watson Nadaraya estimator, crossvalidation

Perceptron and kernels

Hebb's rule, perceptron algorithm, convergence, feature maps, kernel trick, examples

Support Vector classification

Geometrical view, dual problem, convex optimization, kernels and SVM

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

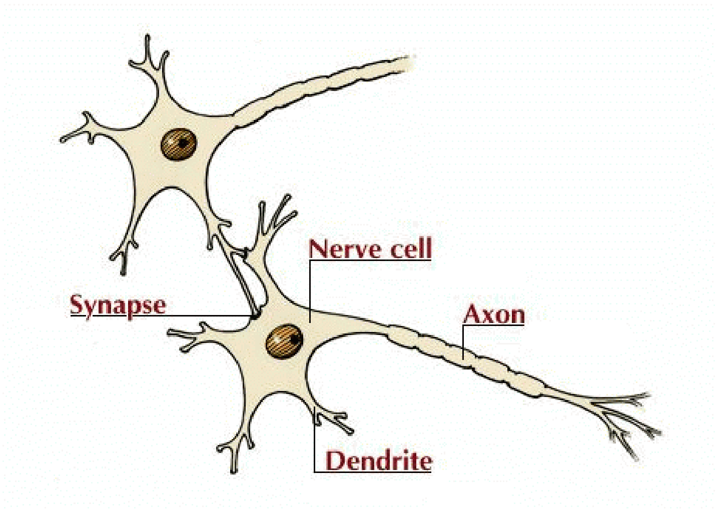
Basic Idea

- Good behavior should be rewarded, bad behavior punished (or not rewarded). This improves the fitness of the system.
- Example: hitting a sabertooth tiger over the head 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").

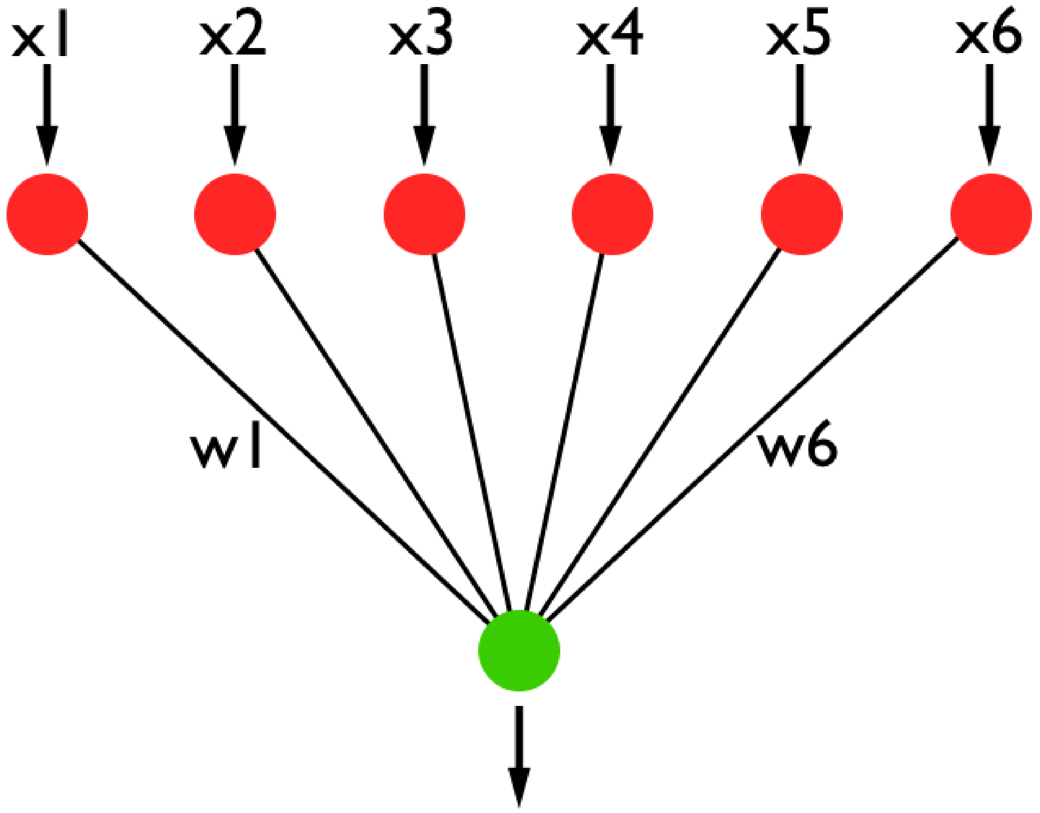
Synapse

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 + ... + 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).

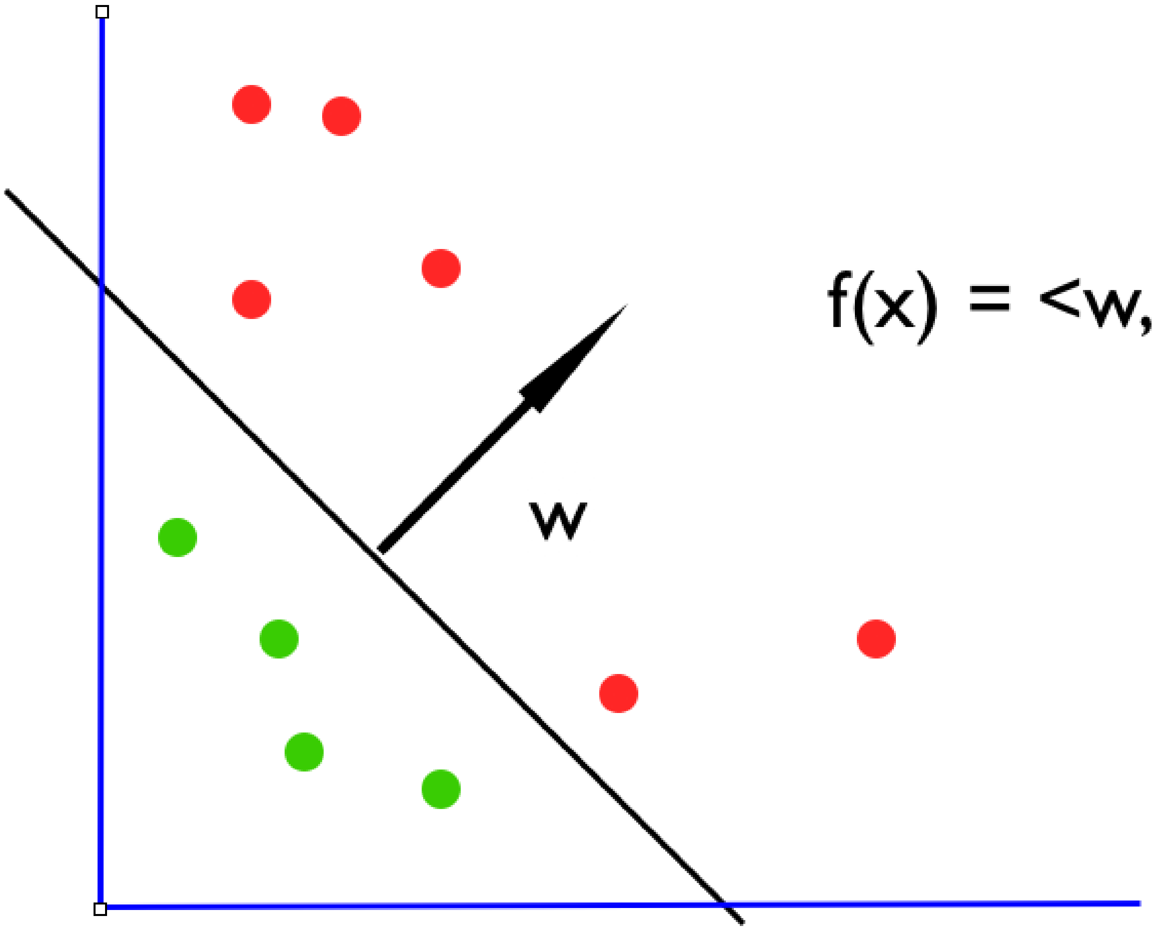
Applications

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

Learning

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

Linear Separation



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

Perceptron Algorithm

```
argument:  $X := \{x_1, \ldots, x_m\} \subset \mathcal{X}$ (data)
           $Y := \{y_1, \ldots, y_m\} \subset \{\pm 1\}$ (labels)
function $(w, b) = \mathrm{Perceptron}(X, Y, \eta)$
    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

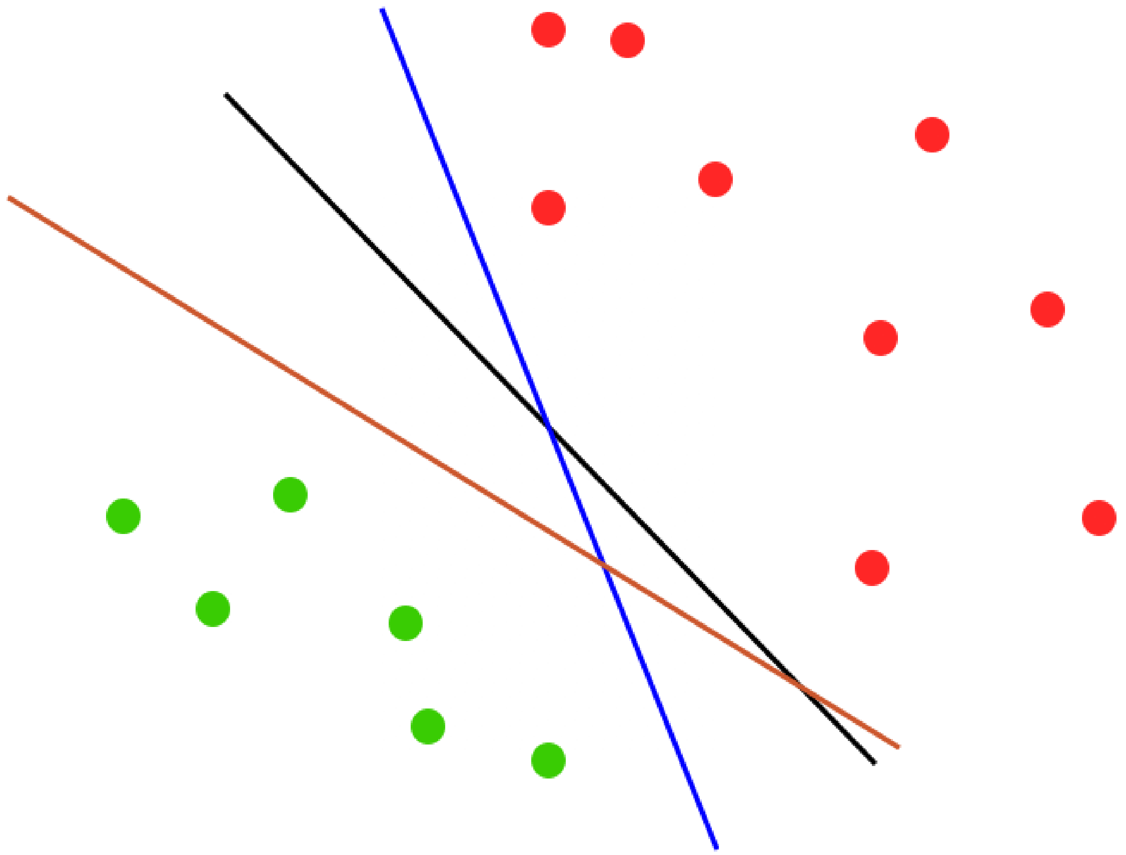
$$y_i \left(\langle w^*, x_i\rangle + b^*\right) \geq \rho \text{ for all } 1 \leq i \leq m.$$

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 $R = \max_i \|x_i\|$.

Solutions of the Perceptron



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).

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

For error in observation (x_i, y_i) we get

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

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

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

Solving for j proves the theorem.

What does it mean?

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 $\mathfrak{X}$.

Sample Expansion

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

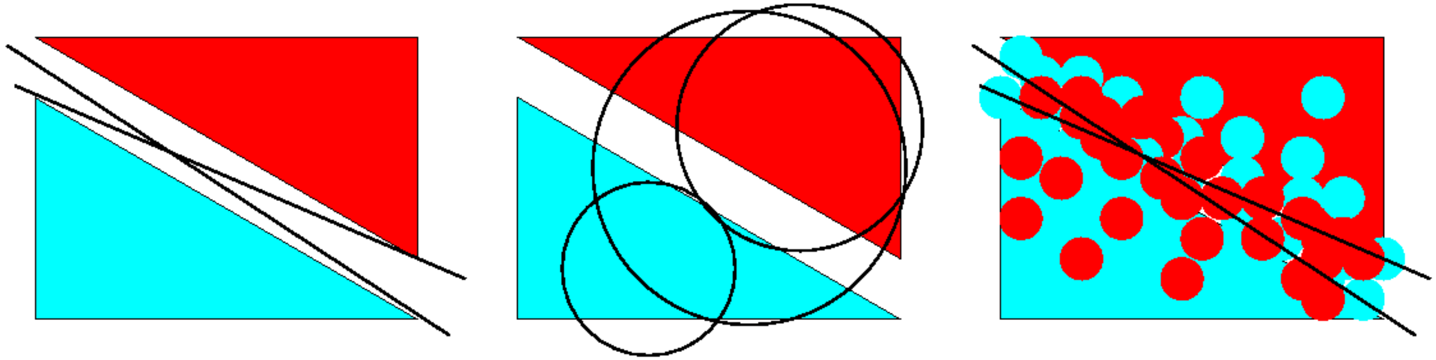
Realizable and Non-realizable Concepts

Realizable Concept

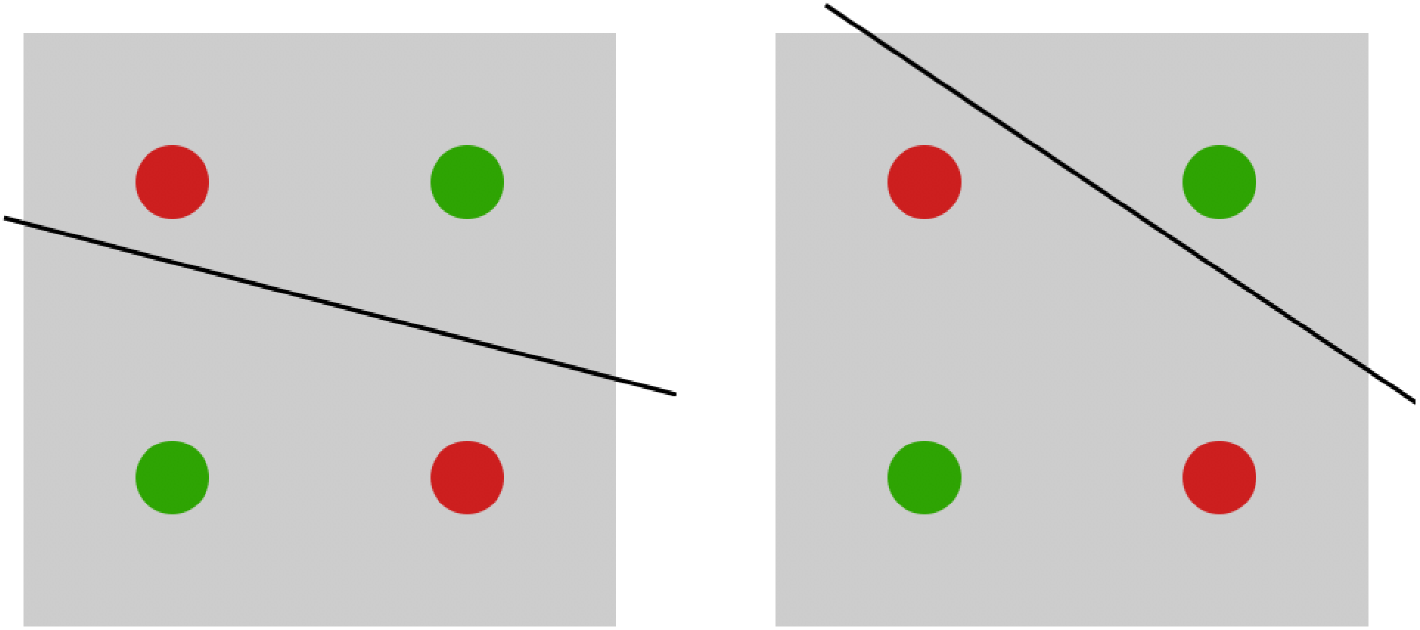
Here some w^*, b^* exists such that y is generated by $y = \operatorname{sgn}(\langle w^*, x\rangle + b)$. In general realizable means that the exact functional dependency is included in the class of admissible hypotheses.

Unrealizable Concept

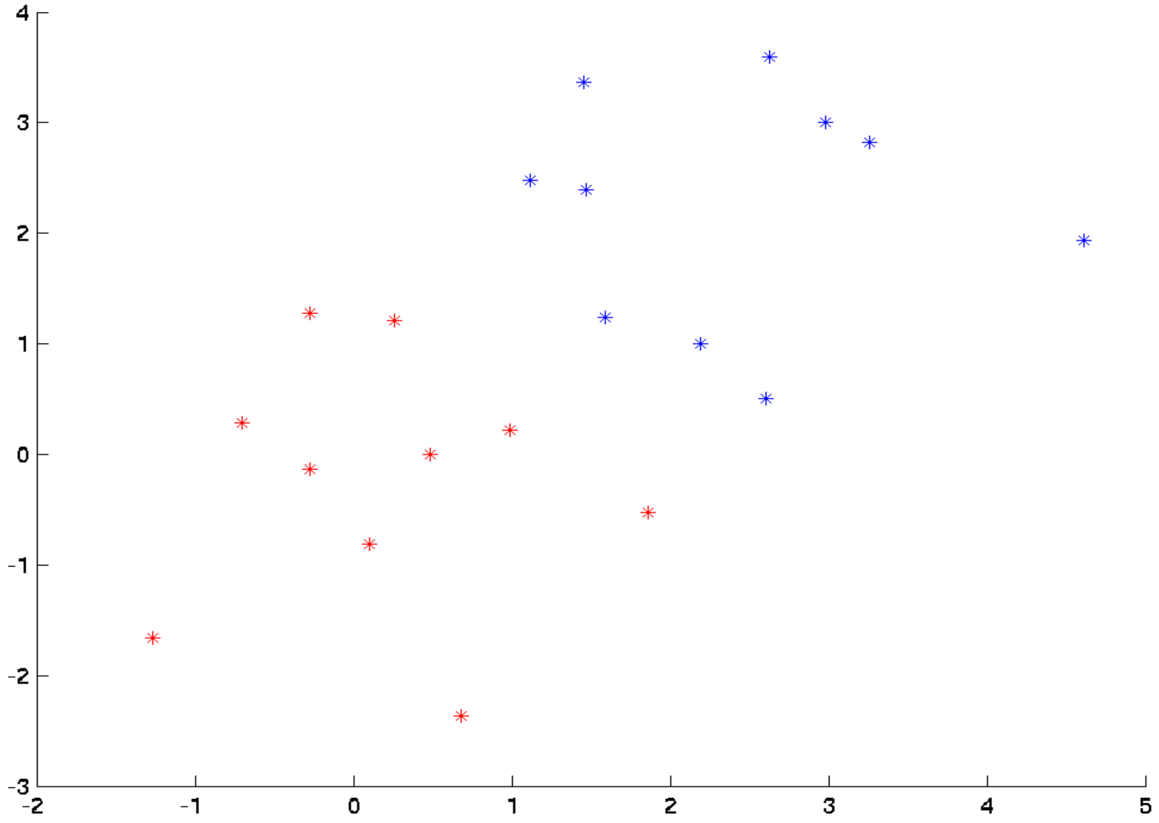
In this case, the exact concept does not exist or it is not included in the function class.



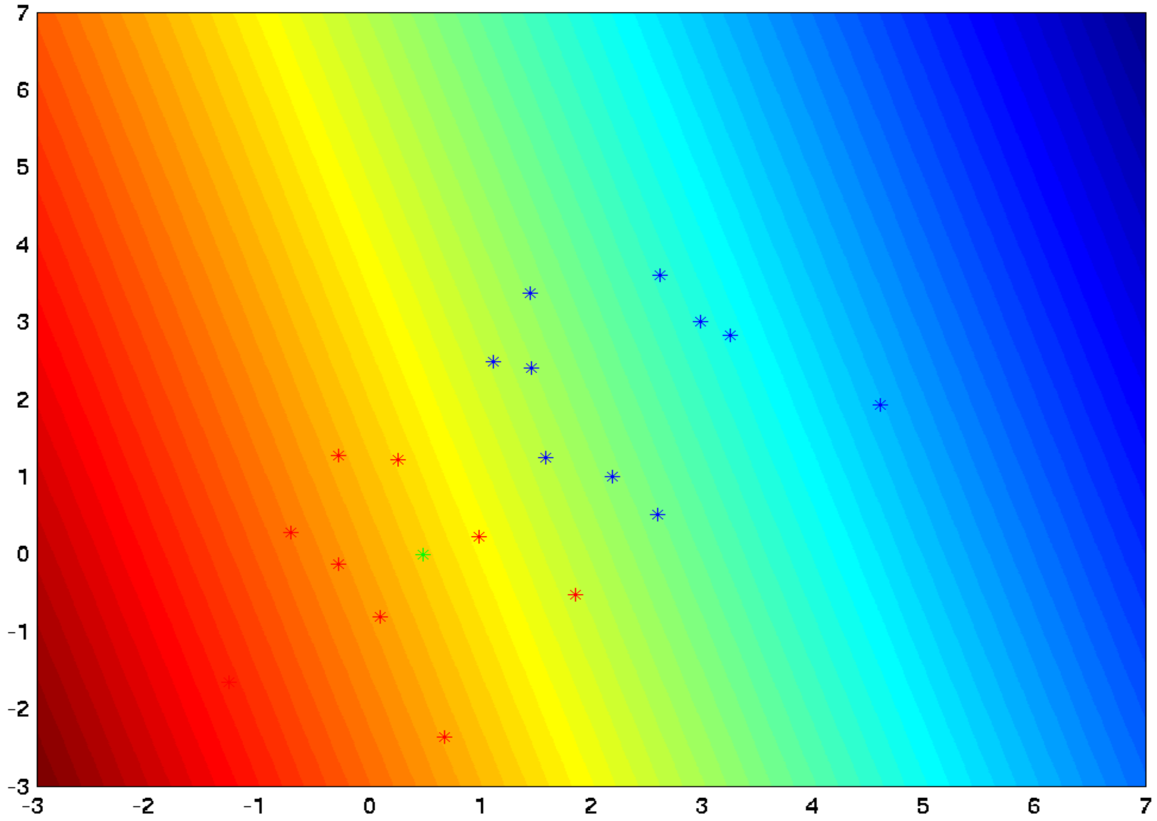
The XOR Problem



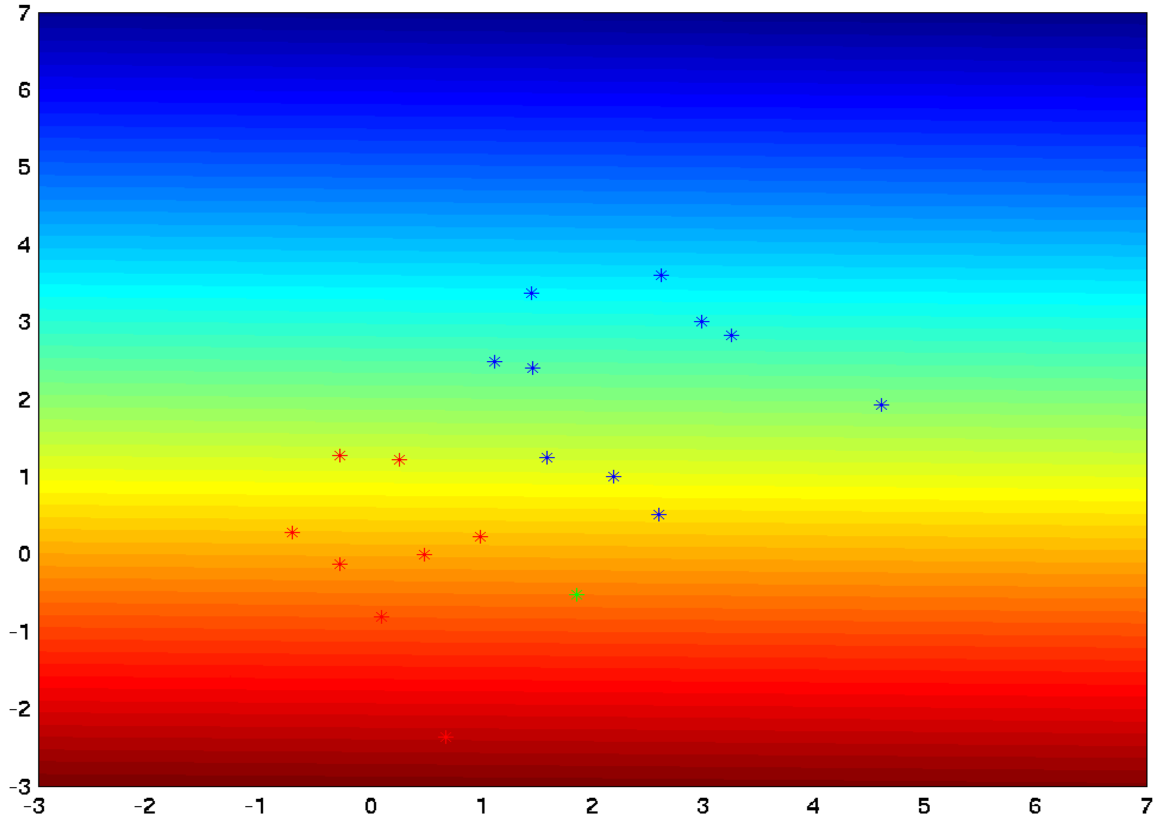
Training data



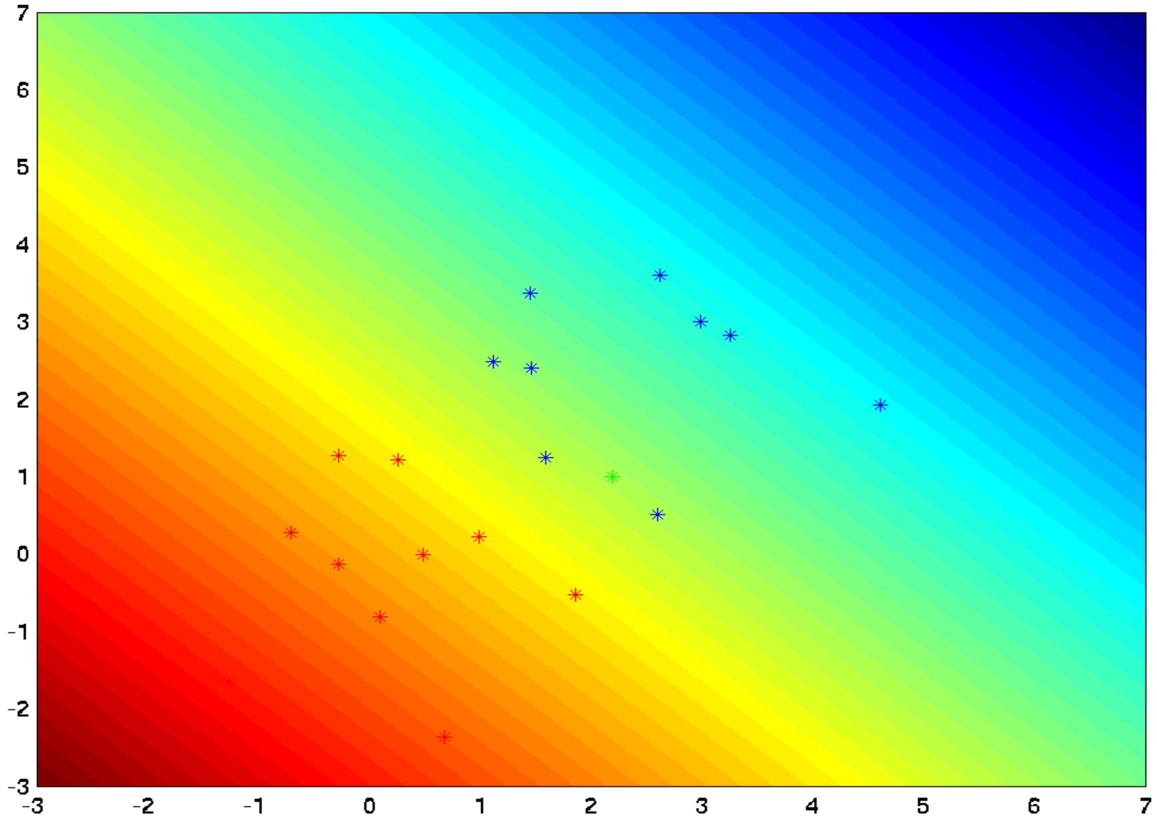
Perceptron algorithm (i=7)



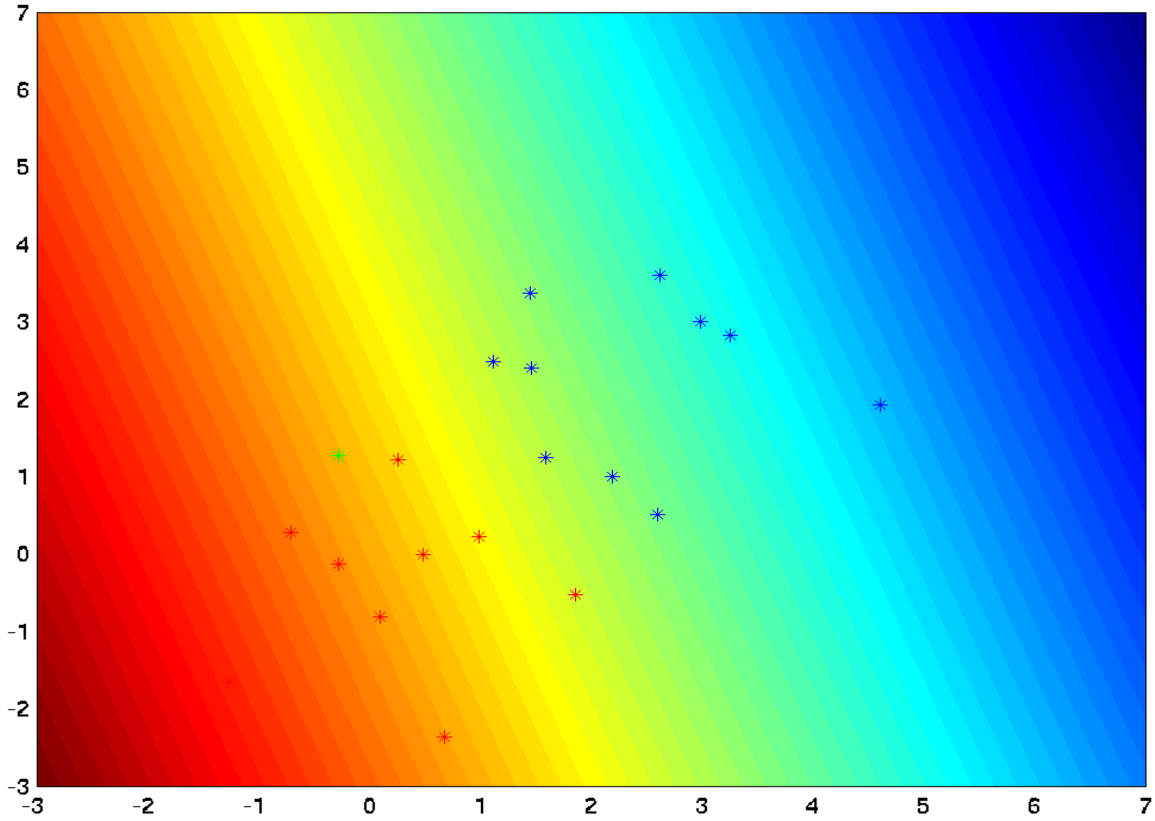
Perceptron algorithm (i=16)



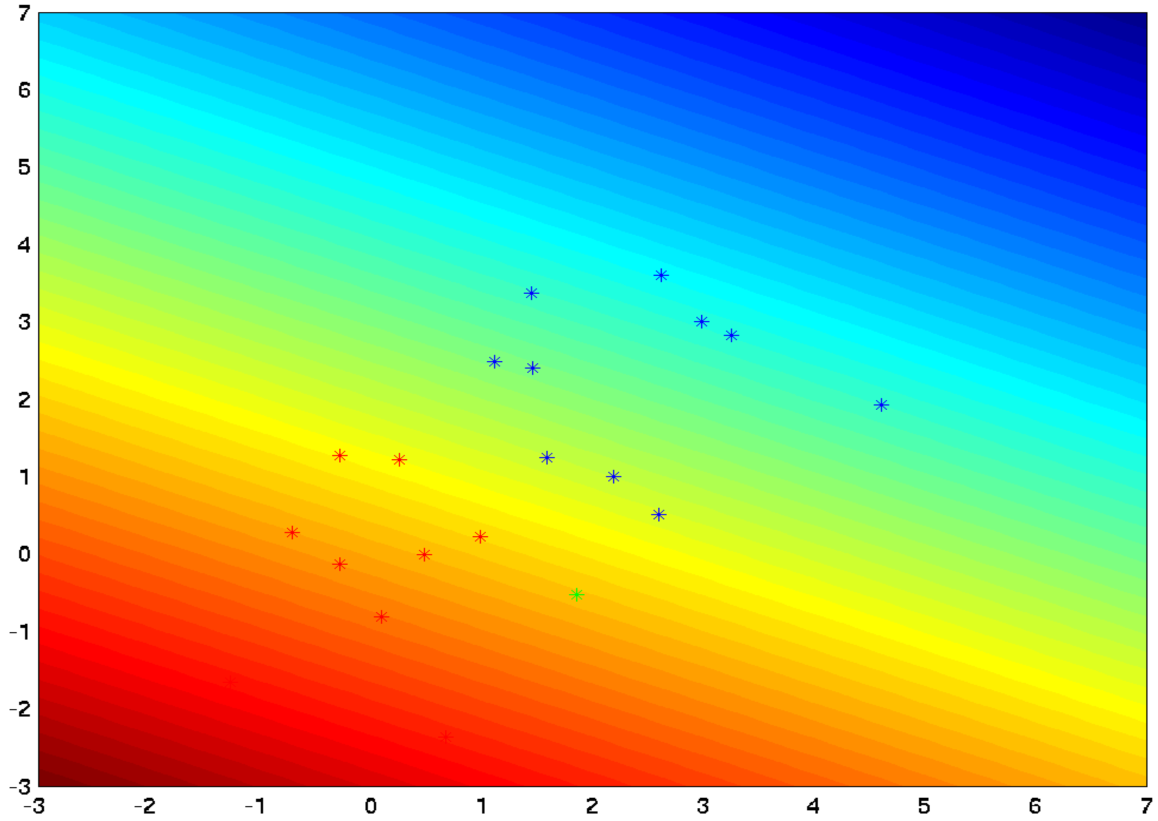
Perceptron algorithm (i=2)



Perceptron algorithm (i=4)

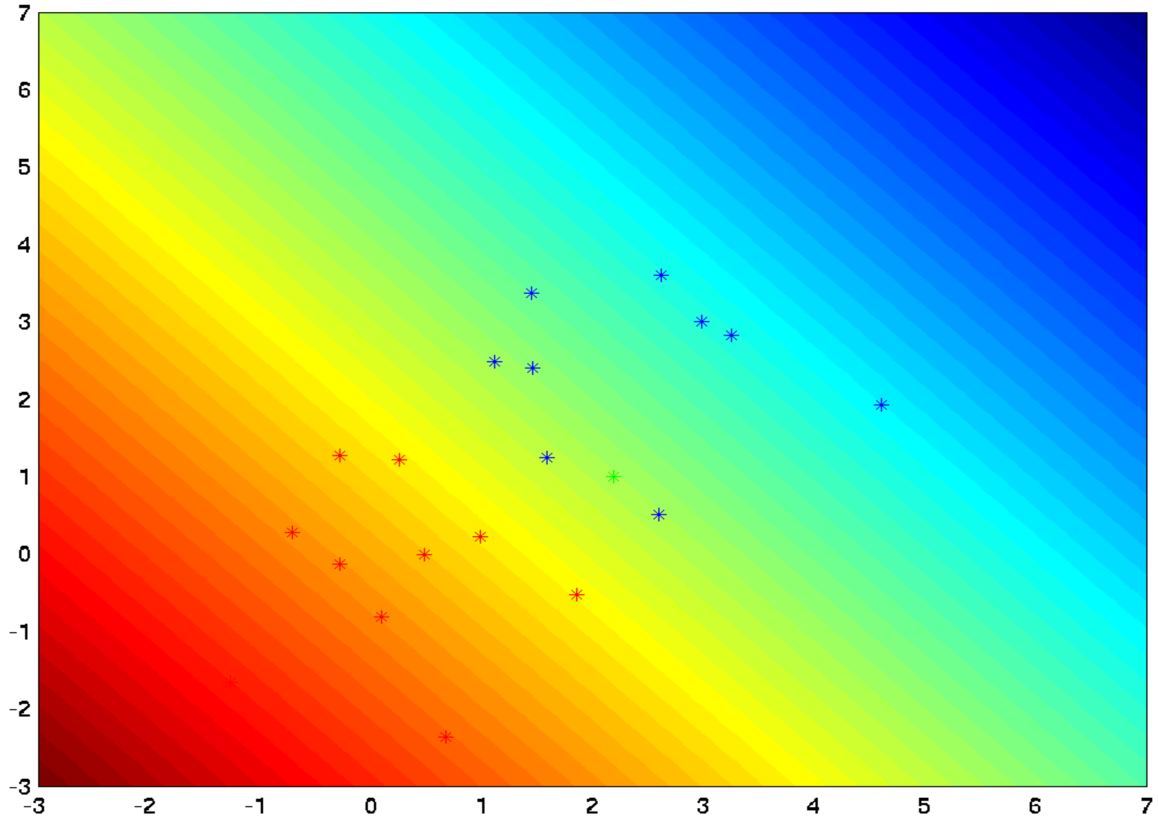


Perceptron algorithm (i=16)

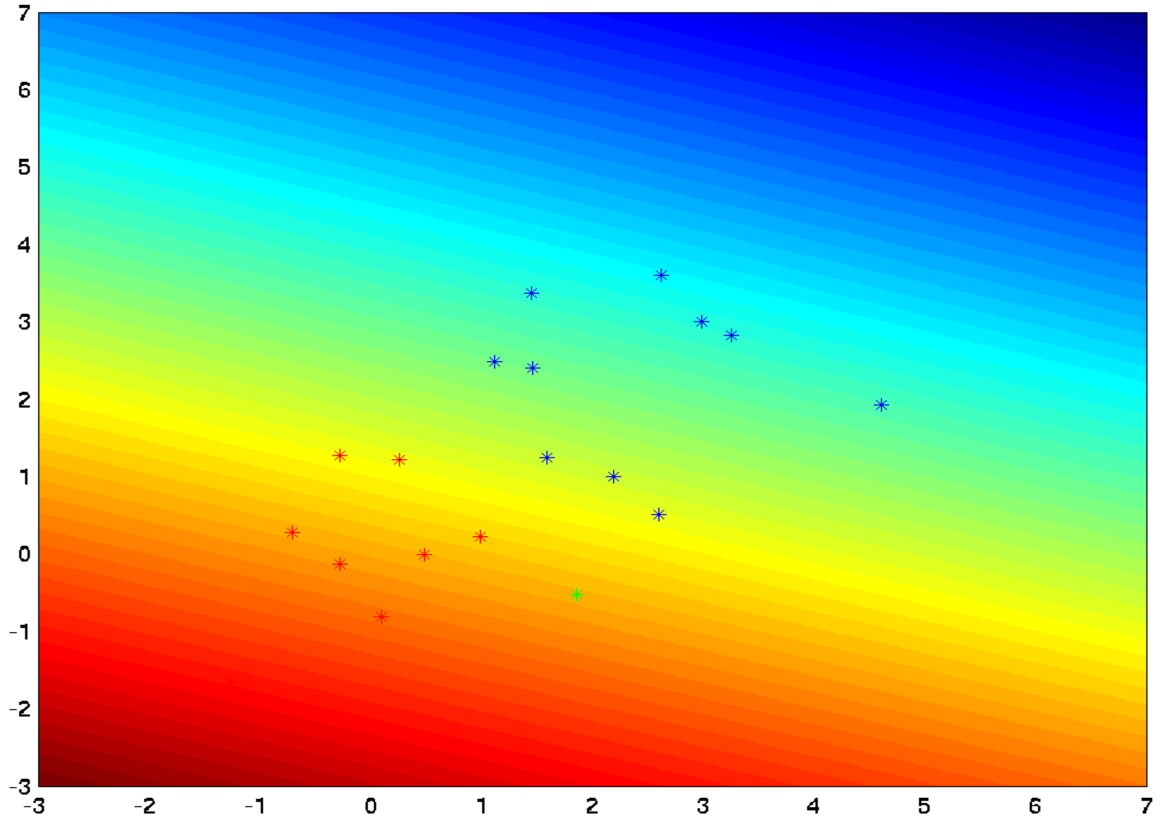


NATIONAL ICT AUSTRALIA

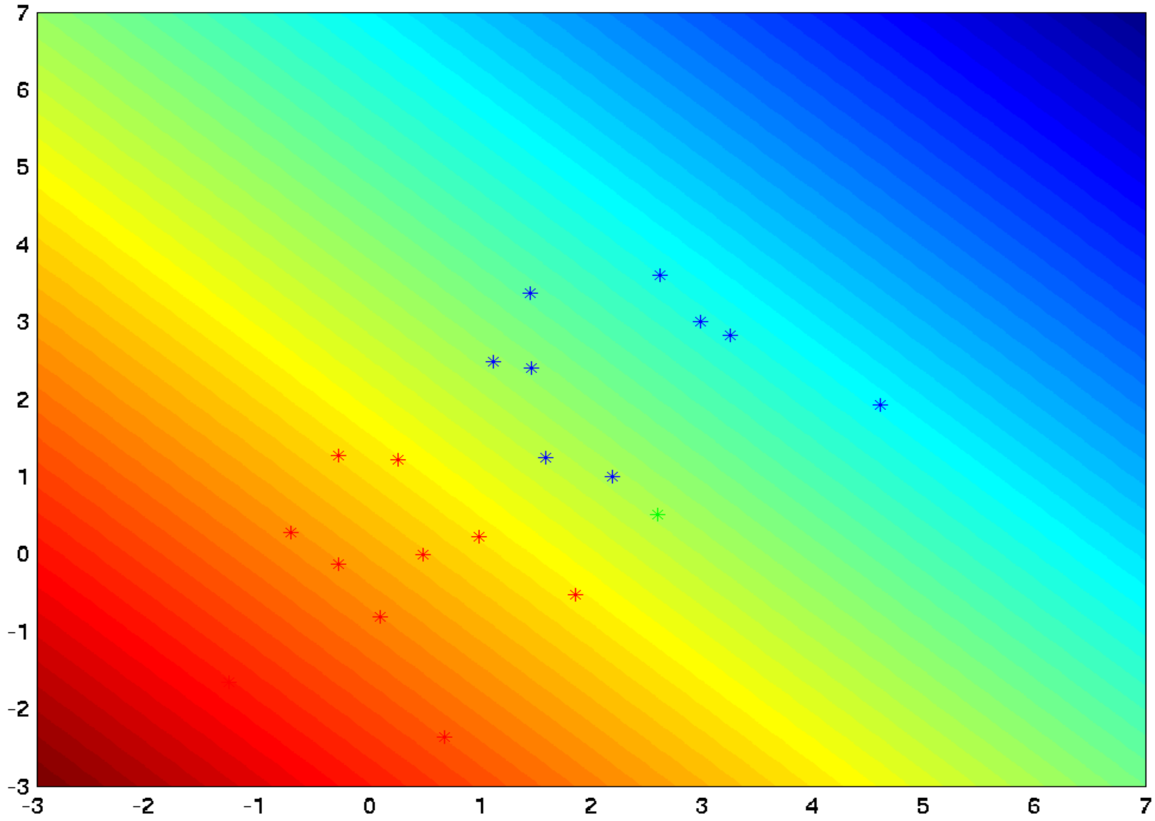
Perceptron algorithm (i=2)



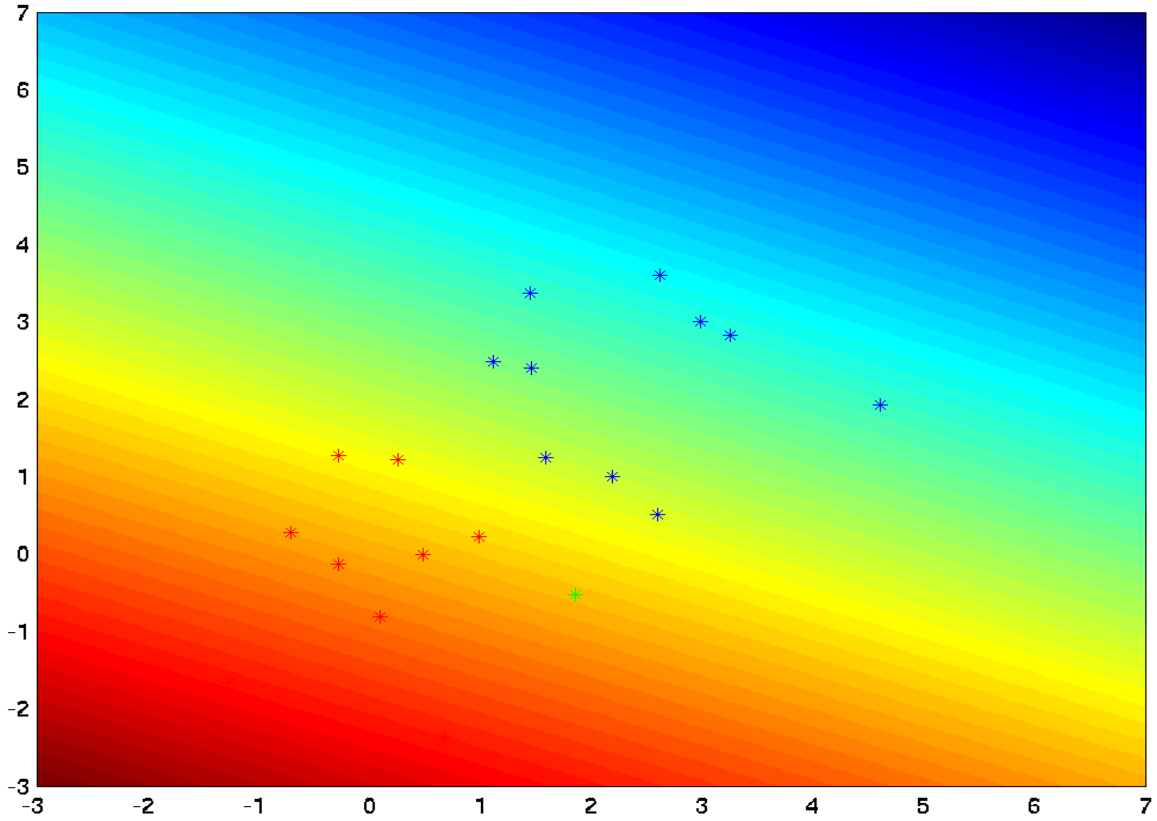
Perceptron algorithm (i=16)



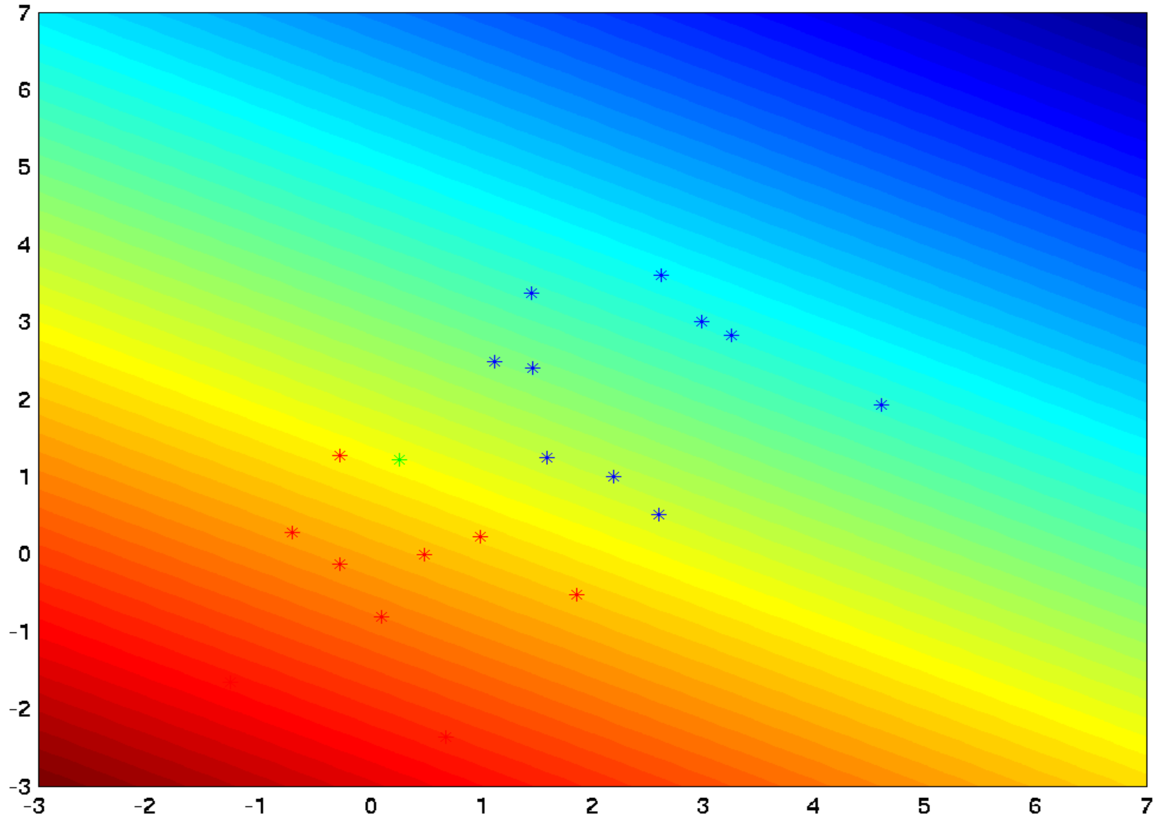
Perceptron algorithm (i=12)



Perceptron algorithm (i=16)



Perceptron algorithm (i=20)



Stochastic Gradient Descent, 1

Linear Function

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

Objective Function

$$\begin{aligned} R[f] :=& \frac{1}{m} \sum_{i=1}^{m} \max(0, -y_i f(x_i)) \\ =& \sum_{i=1}^{m} \max\left(0, -y_i \left(\langle w, x_i \rangle + b\right)\right) \end{aligned}$$

Stochastic Gradient

We use each term in the sum as a stochastic approximation of the overall objective function:

$$\begin{aligned} w &\longleftarrow w - \eta \partial_w \left(0, -y_i \left(\langle w, x_i \rangle + b\right)\right) \\ b &\longleftarrow b - \eta \partial_b \left(0, -y_i \left(\langle w, x_i \rangle + b\right)\right) \end{aligned}$$

Stochastic Gradient Descent, 2

Details

$$\partial_w \max\left(0, -y_i\left(\langle w, x_i\rangle + b\right)\right) = \begin{cases} -y_i x_i & \text{for } f(x_i) < 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\partial_b \max\left(0, -y_i\left(\langle w, x_i\rangle + b\right)\right) = \begin{cases} -y_i & \text{for } f(x_i) < 0 \\ 0 & \text{otherwise} \end{cases}$$

Overall Strategy

- Have complicated function consisting of lots of terms
- Want to minimize this monster
- Solve it performing descent into one direction at a time
- Randomly pick directions and converge
- **Often need to adjust learning rate η**

Nonlinearity via Preprocessing

Problem

Linear functions are often too simple to provide good estimators.

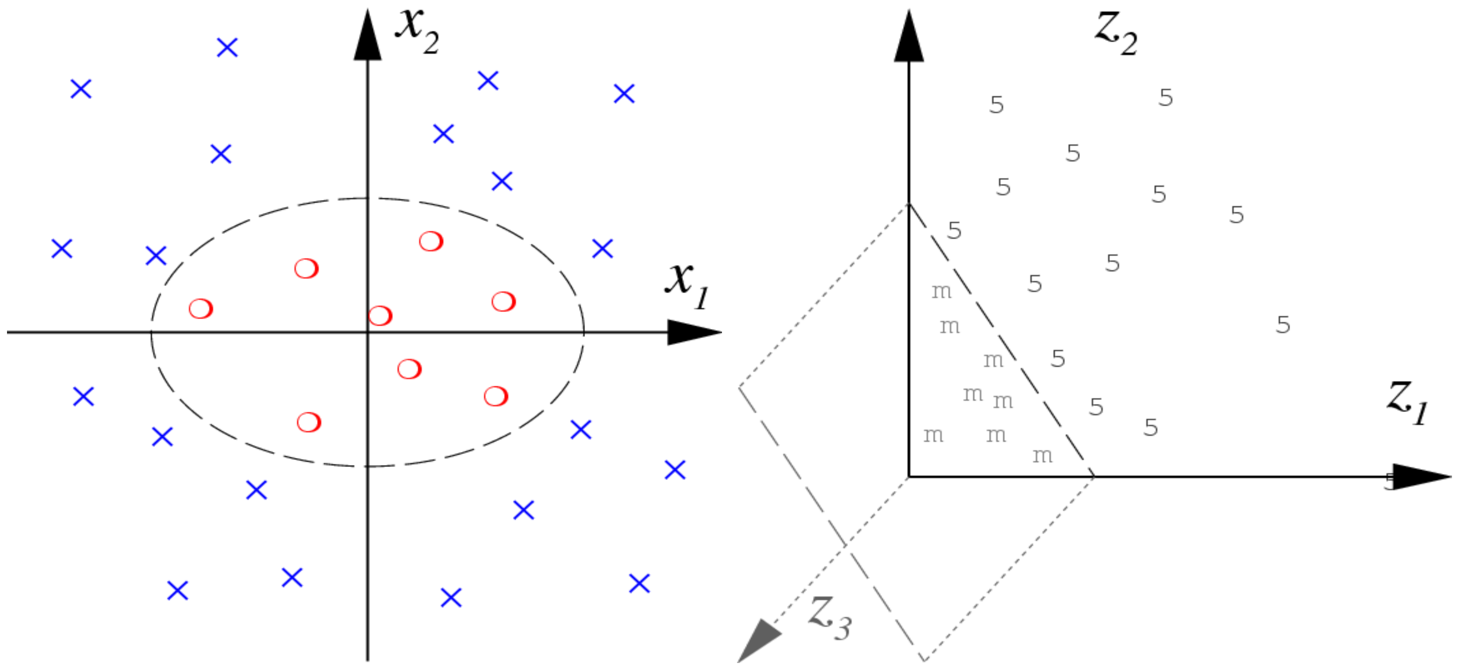
Idea

- Map to a higher dimensional feature space via $\Phi : x \longrightarrow \Phi(x)$ 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



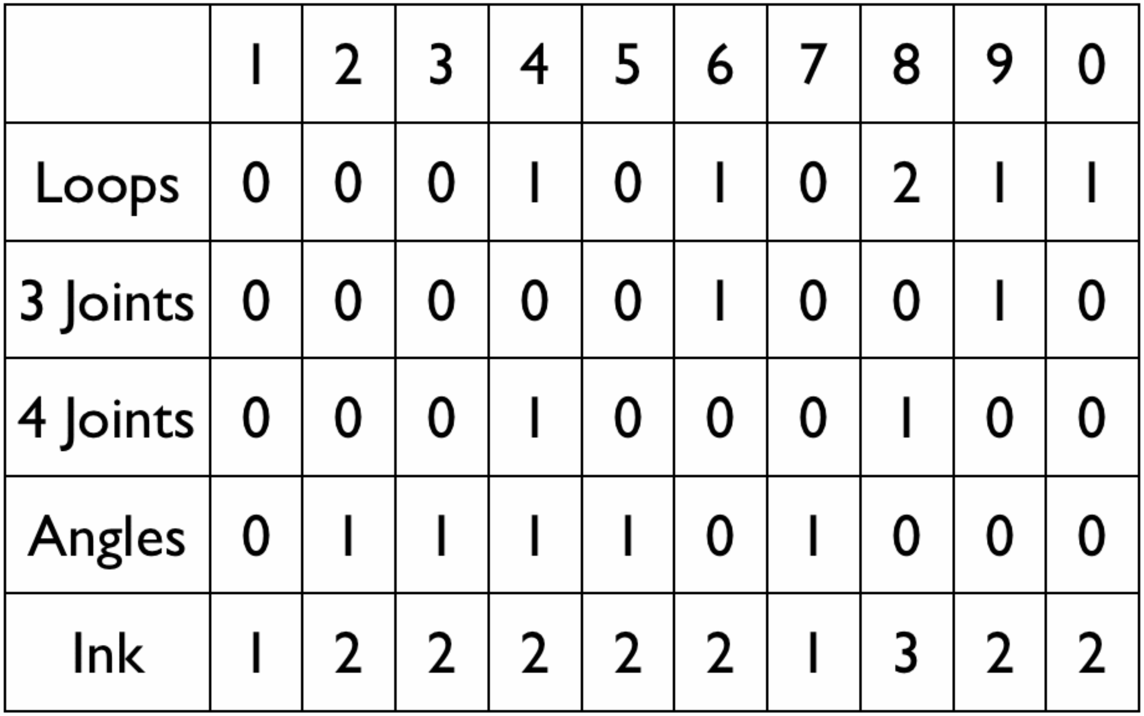
Features

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

$w = \sum_j y_j \Phi(x_j)$ 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 $\mathbb{R}^2$

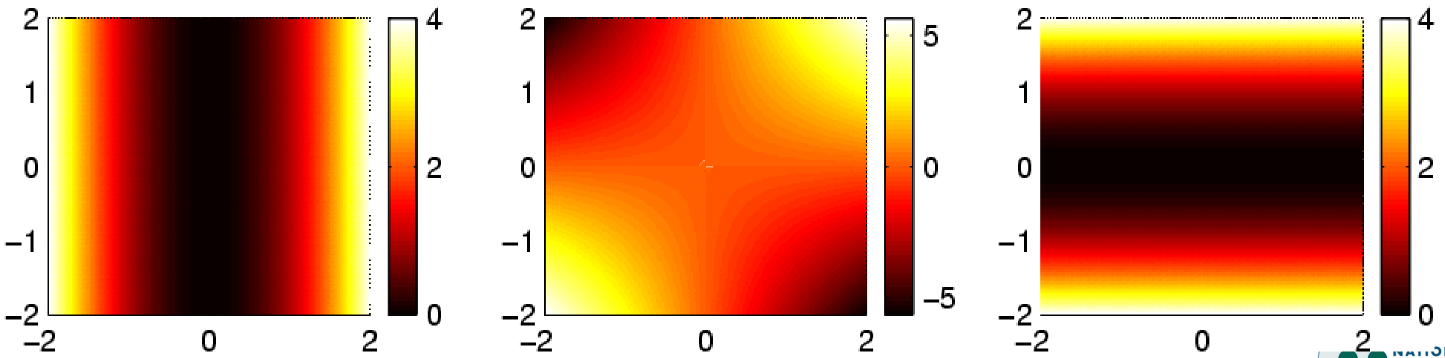
$$\Phi(x) := \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

Dot Product

$$\begin{aligned}\langle \Phi(x), \Phi(x')\rangle &= \left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right), \left({x'_1}^2, \sqrt{2}x'_1x'_2, {x'_2}^2\right)\right\rangle \\ &= \langle x, x'\rangle^2.\end{aligned}$$

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

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

- 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

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

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

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

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 the continuous version of a vector-matrix-vector multiplication. For positive semidefinite matrices we have

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

Properties of the Kernel

Distance in Feature Space

Distance between points in feature space via

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

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:

$$\sum_{i,j}^m \alpha_i\alpha_j K_{ij} = \sum_{i,j}^m \alpha_i\alpha_j \langle\Phi(x_i),\Phi(x_j)\rangle$$

$$= \left\langle \sum_i^m \alpha_i\Phi(x_i), \sum_j^m \alpha_j\Phi(x_j)\right\rangle = \left\|\sum_{i=1}^m \alpha_i\Phi(x_i)\right\|^2$$

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

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

as eigensystem. Hence k is not a kernel.

Some Good Kernels

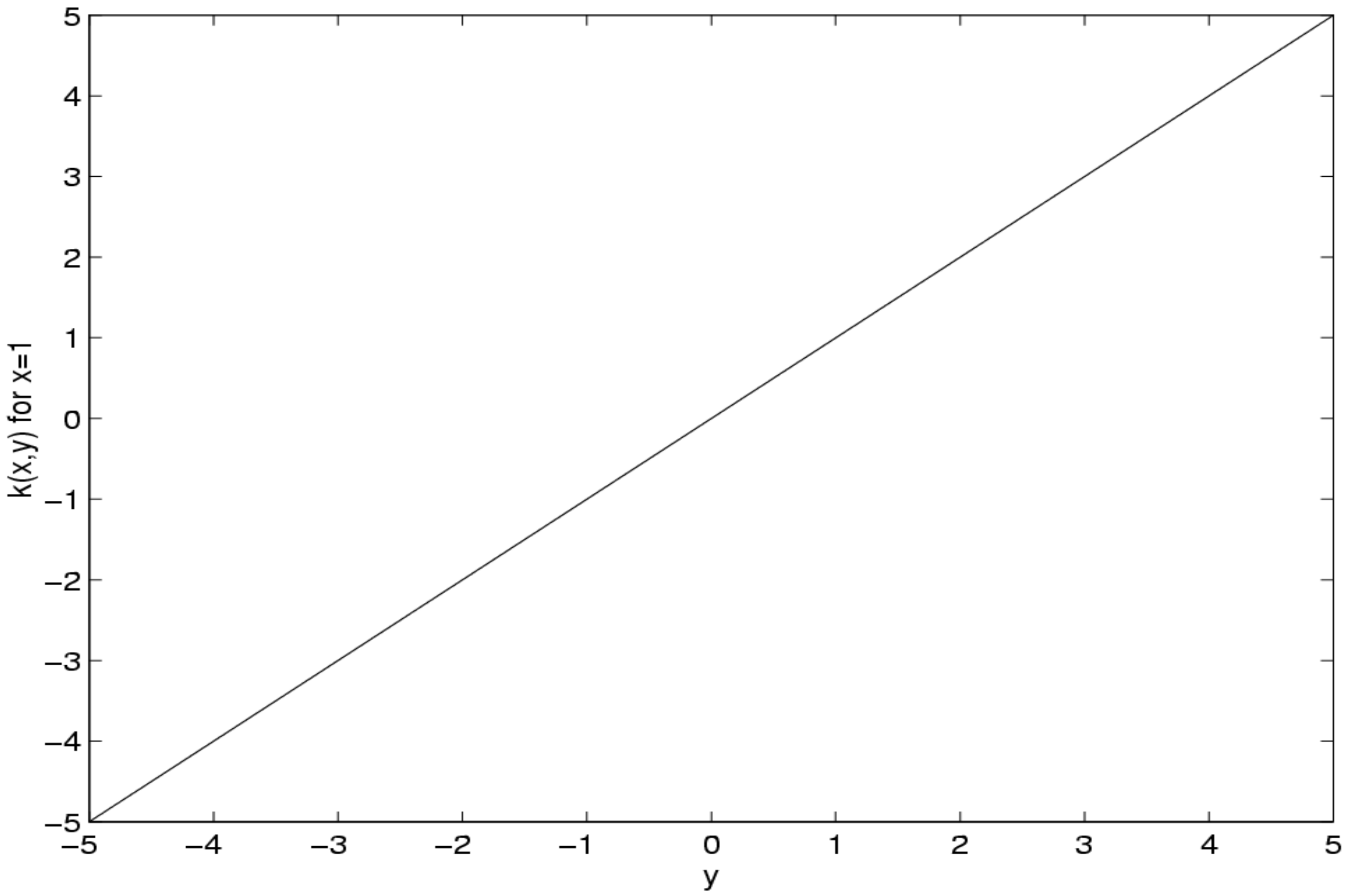
Examples of kernels $k(x, x')$

Linear	$\langle x, x' \rangle$
Laplacian RBF	$\exp\left(-\lambda \|x - x'\|\right)$
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	$\mathbf{E}_c[p(x\|c)p(x'\|c)]$

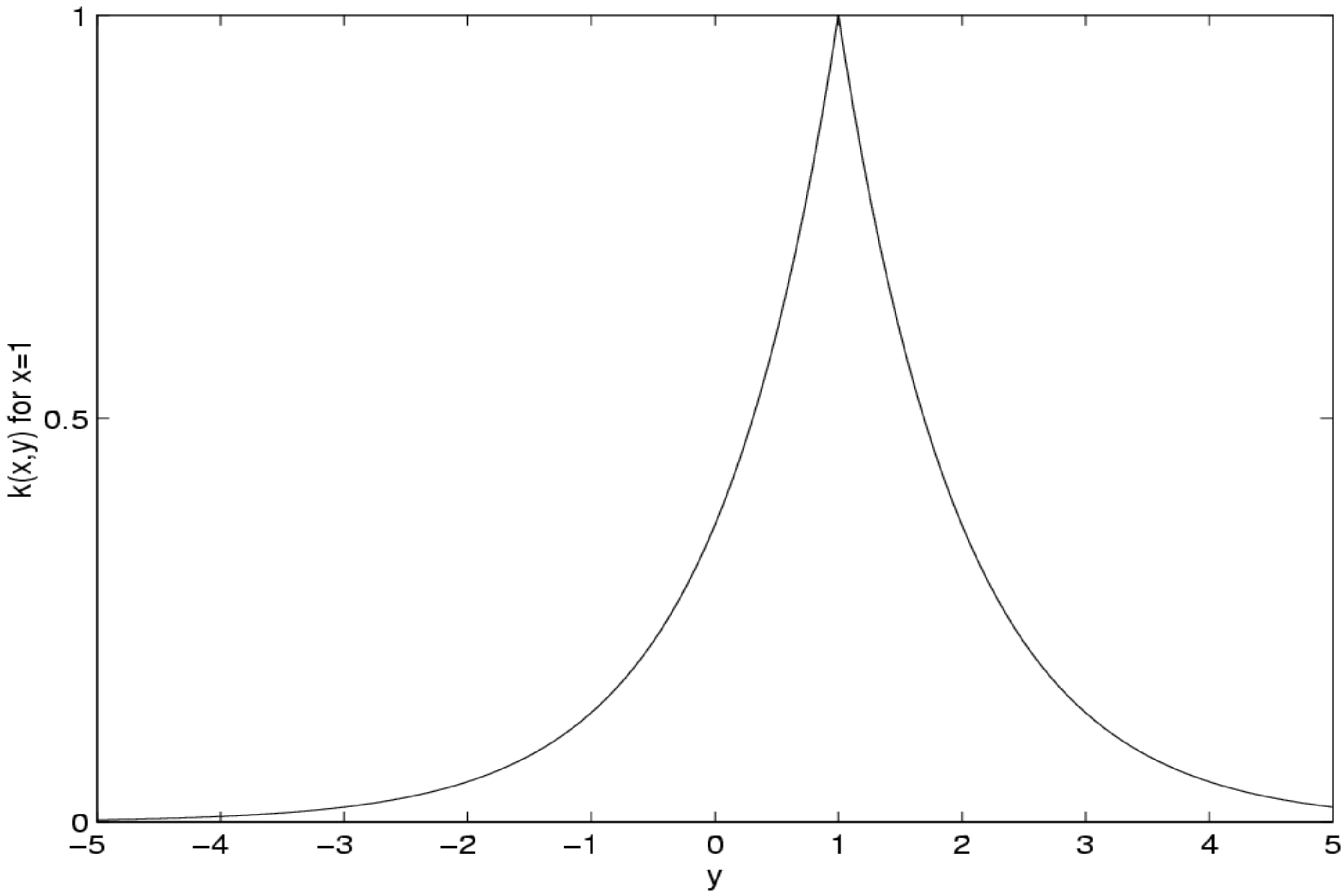
Simple trick for checking Mercer's condition

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

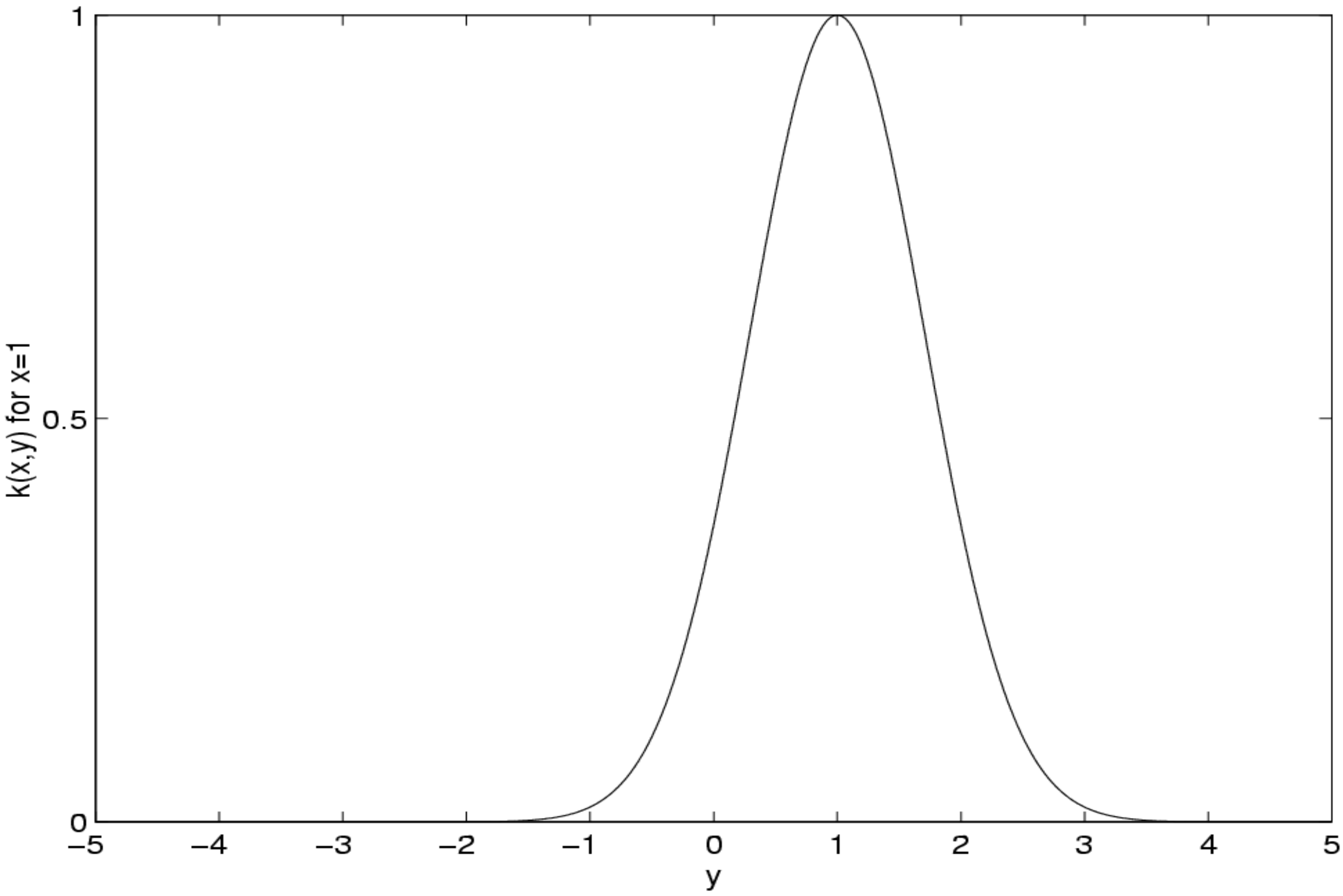
Linear Kernel



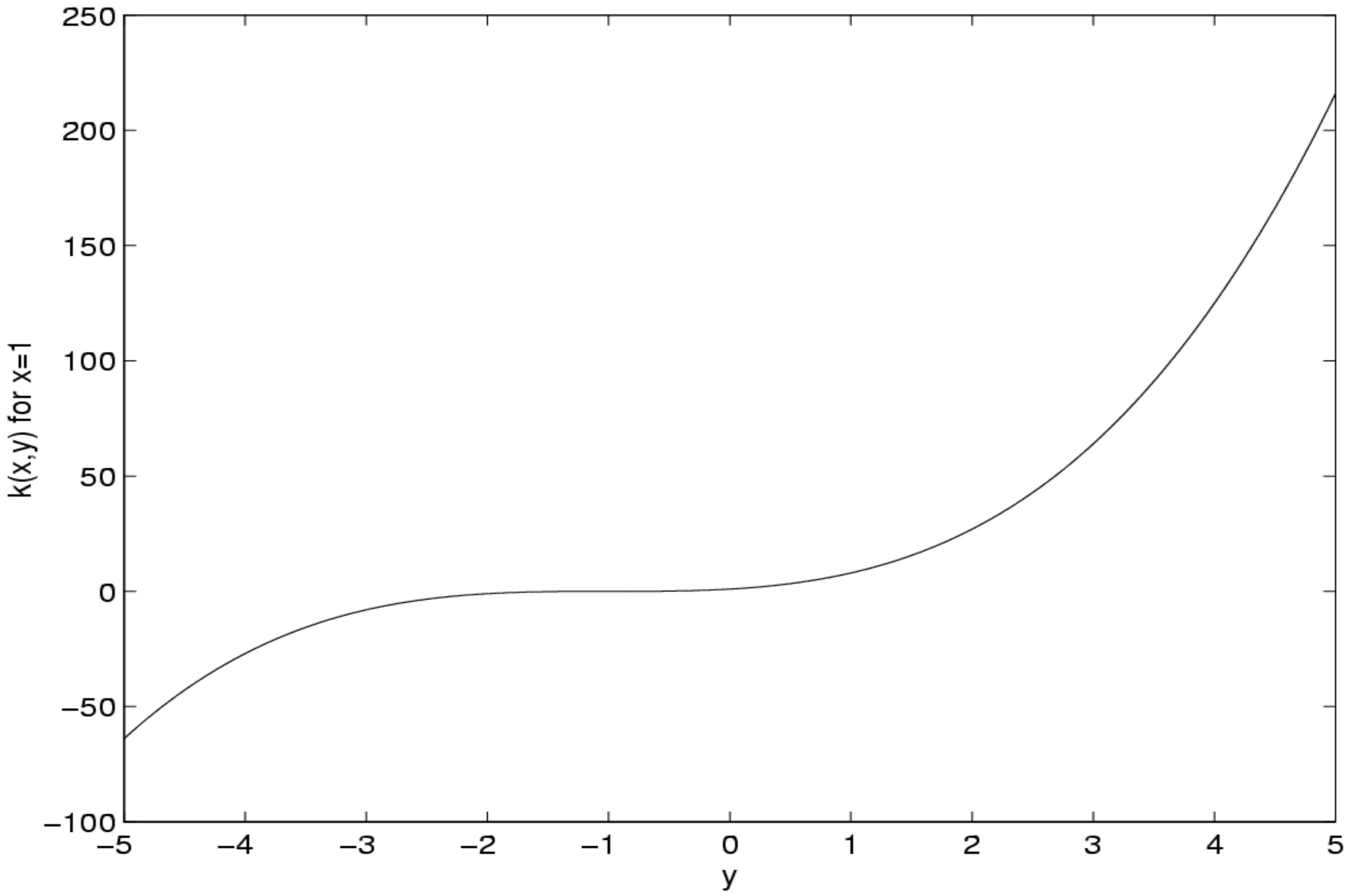
Laplacian Kernel



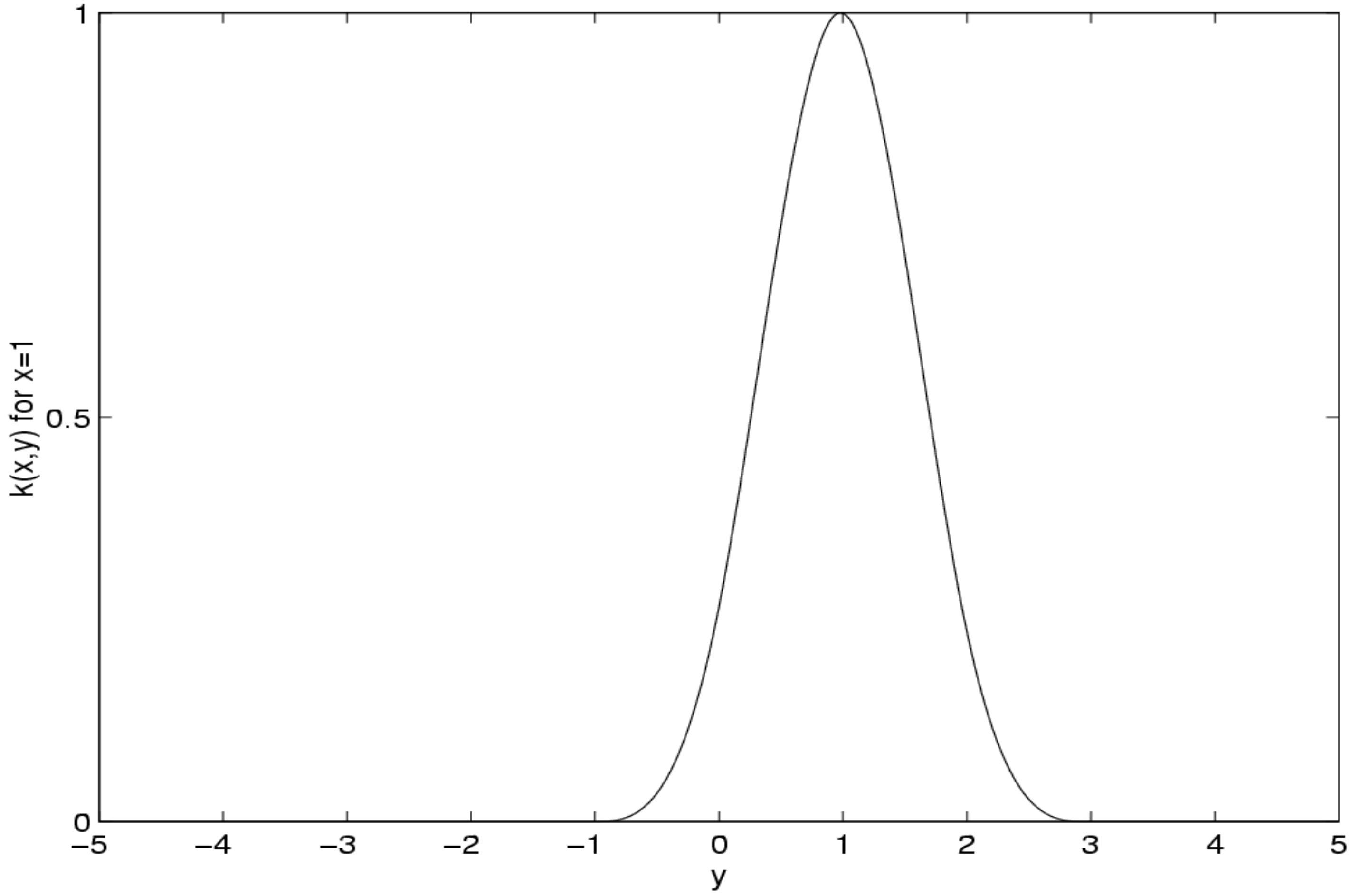
Gaussian Kernel



Polynomial (Order 3)



B_3-Spline Kernel



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 with Kernels

Lecture 4

Alexander J. Smola
Alex.Smola@nicta.com.au

Statistical Machine Learning Program
National ICT Australia, Canberra

Day 1

Machine learning and probability theory

Introduction to pattern recognition, classification, regression, novelty detection, probability theory, Bayes rule, inference

Density estimation and Parzen windows

Kernels and density estimation, Silverman's rule, Watson Nadaraya estimator, crossvalidation

Perceptron and kernels

Hebb's rule, perceptron algorithm, convergence, feature maps, kernel trick, examples

Support Vector classification

Geometrical view, dual problem, convex optimization, kernels and SVM

L4 Support Vector Classification

Support Vector Machine

- Problem definition
- Geometrical picture
- Optimization problem

Optimization Problem

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

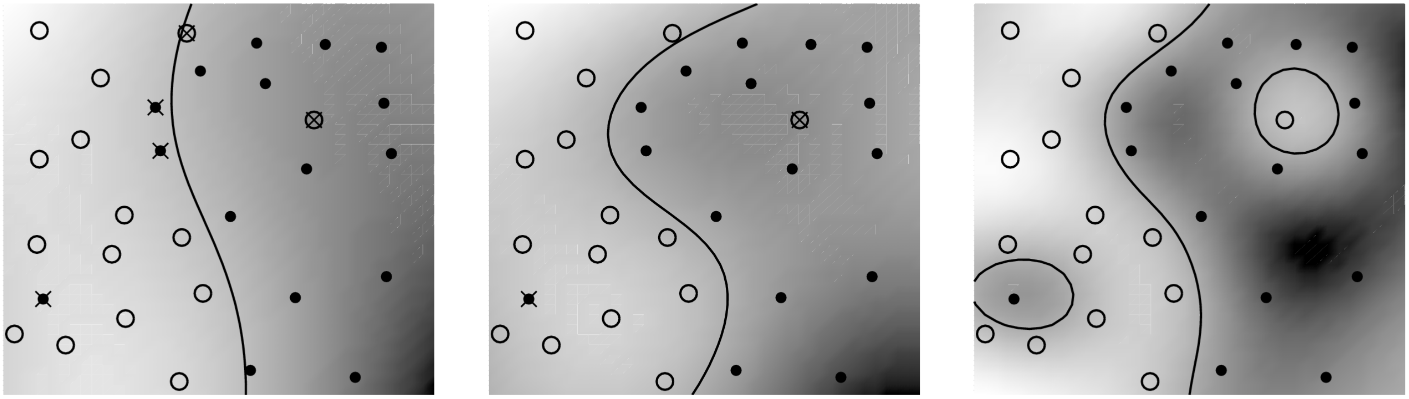
Classification

Data

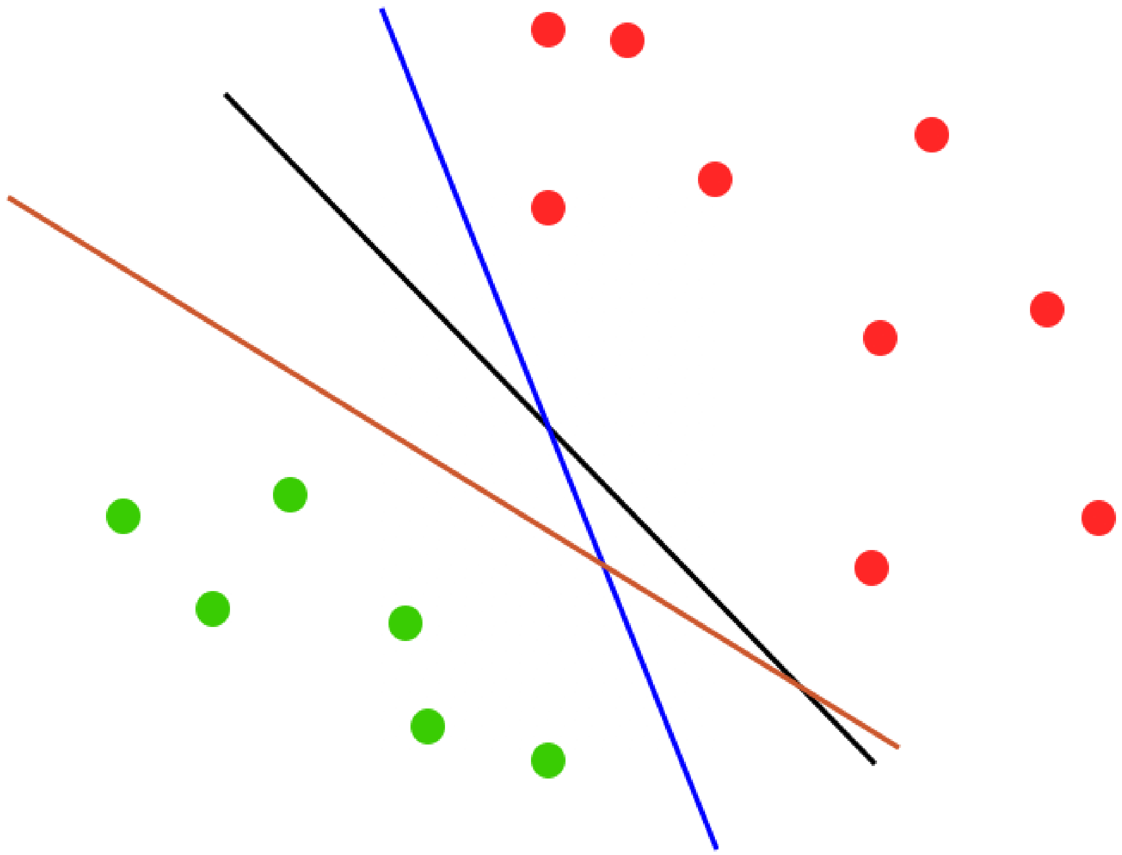
Pairs of observations (x_i, y_i) generated from some distribution $\mathrm{P}(x, y)$, e.g., (blood status, cancer), (credit transaction, fraud), (profile of jet engine, defect)

Task

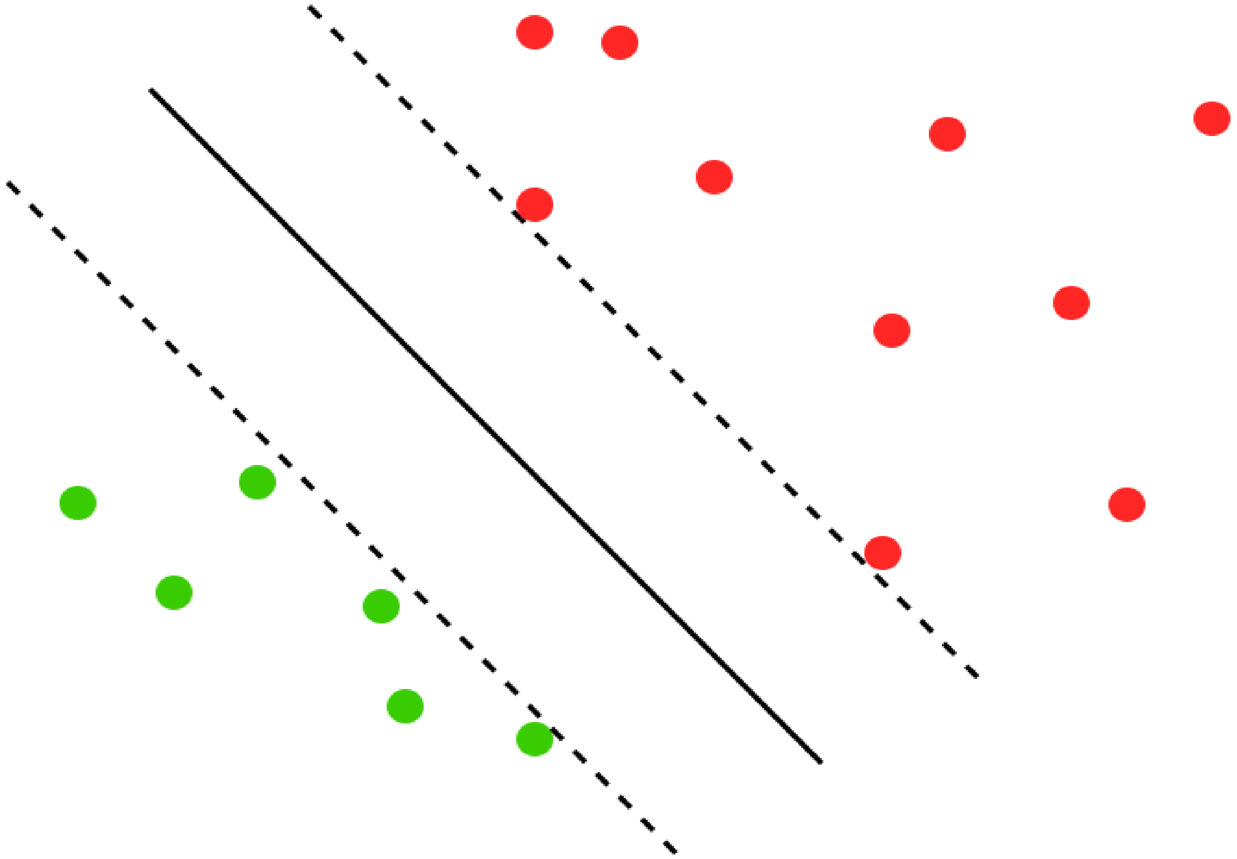
- Estimate y given x at a new location.
- Modification: find a function $f(x)$ that does the task.



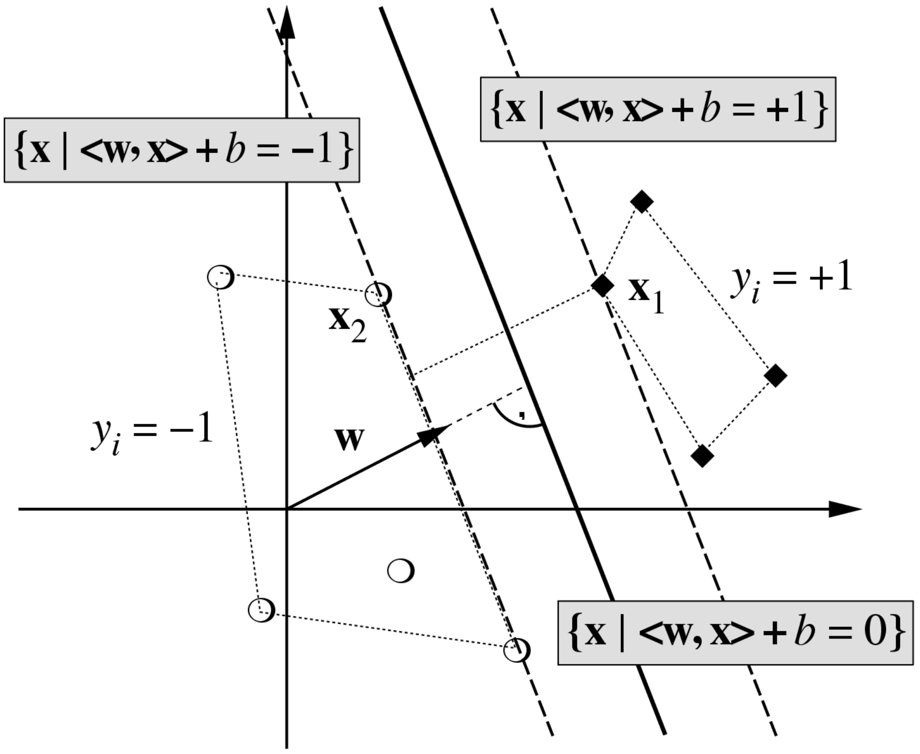
So Many Solutions



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 $\|w\|$.
- Equivalently minimize $\|w\|^2$.

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

We have $\frac{1}{2}\|w\|^2$.

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

Partial derivatives of L with respect to w and b need to vanish.

Solving the Equations

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 $\alpha_i \geq 0$.

Solving the Equations

Dual Optimization Problem

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

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

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

Practical Modification

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

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

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

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

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

Kernels

Nonlinearity via Feature Maps

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

Equivalent optimization problem

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

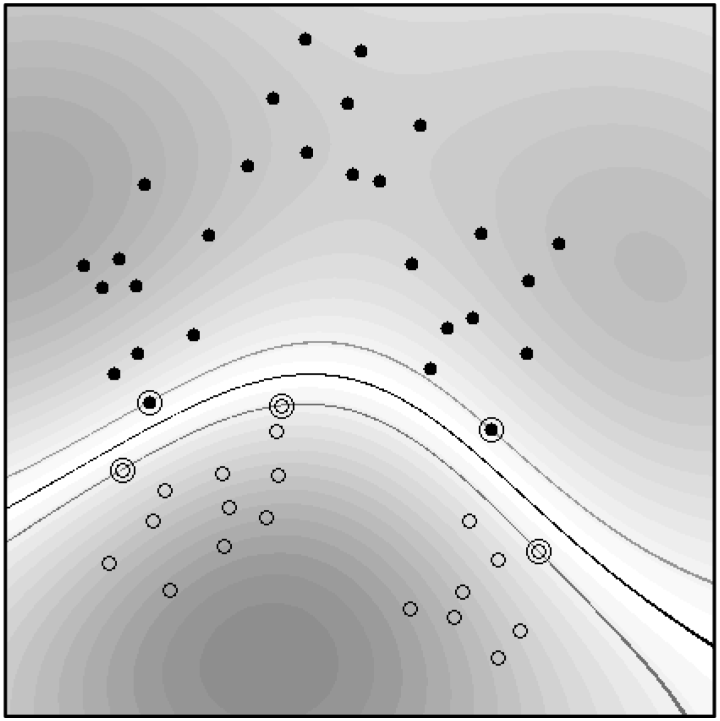
$$\text{subject to } \sum_{i=1}^{m} \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0 \text{ for all } 1 \leq i \leq m$$

Decision Function

From $w = \sum_{i=1}^{m} \alpha_i y_i \Phi(x_i)$ we conclude

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

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)

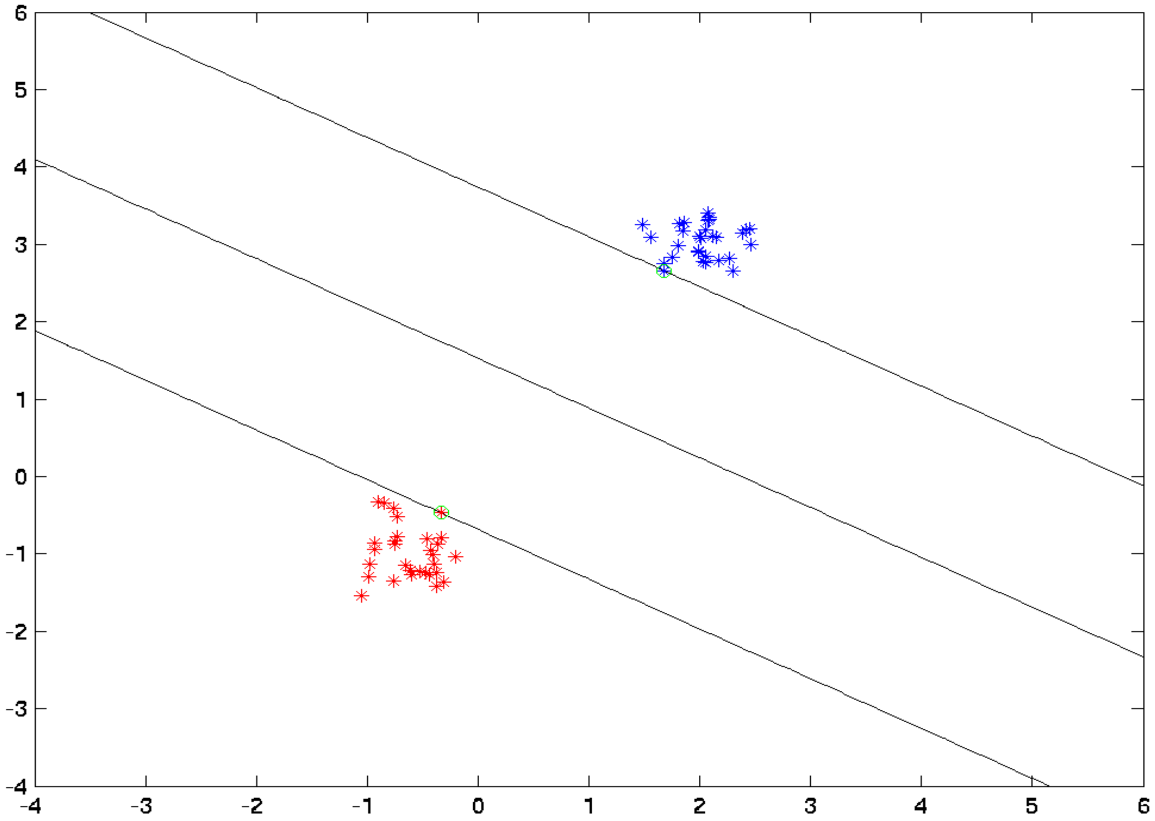
Recall: Hard Margin Problem

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

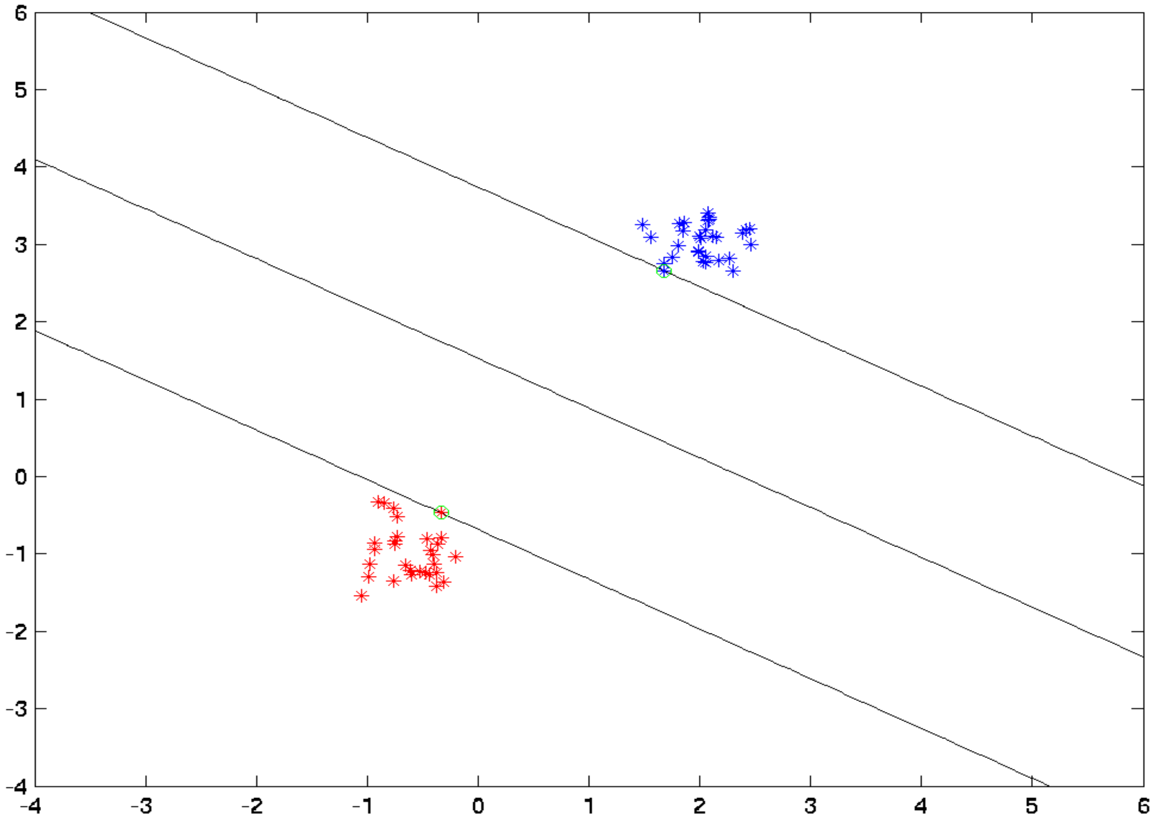
Softening the Constraints

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

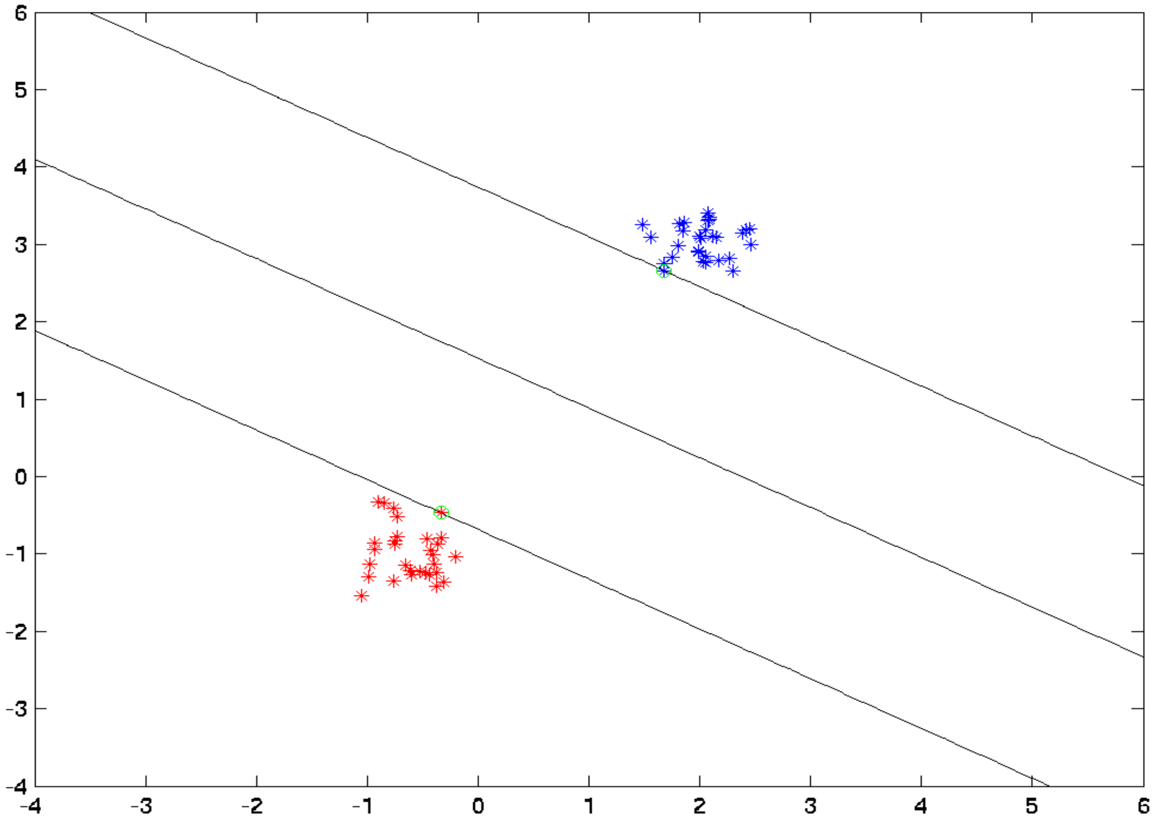
Linear SVM $C = 1$



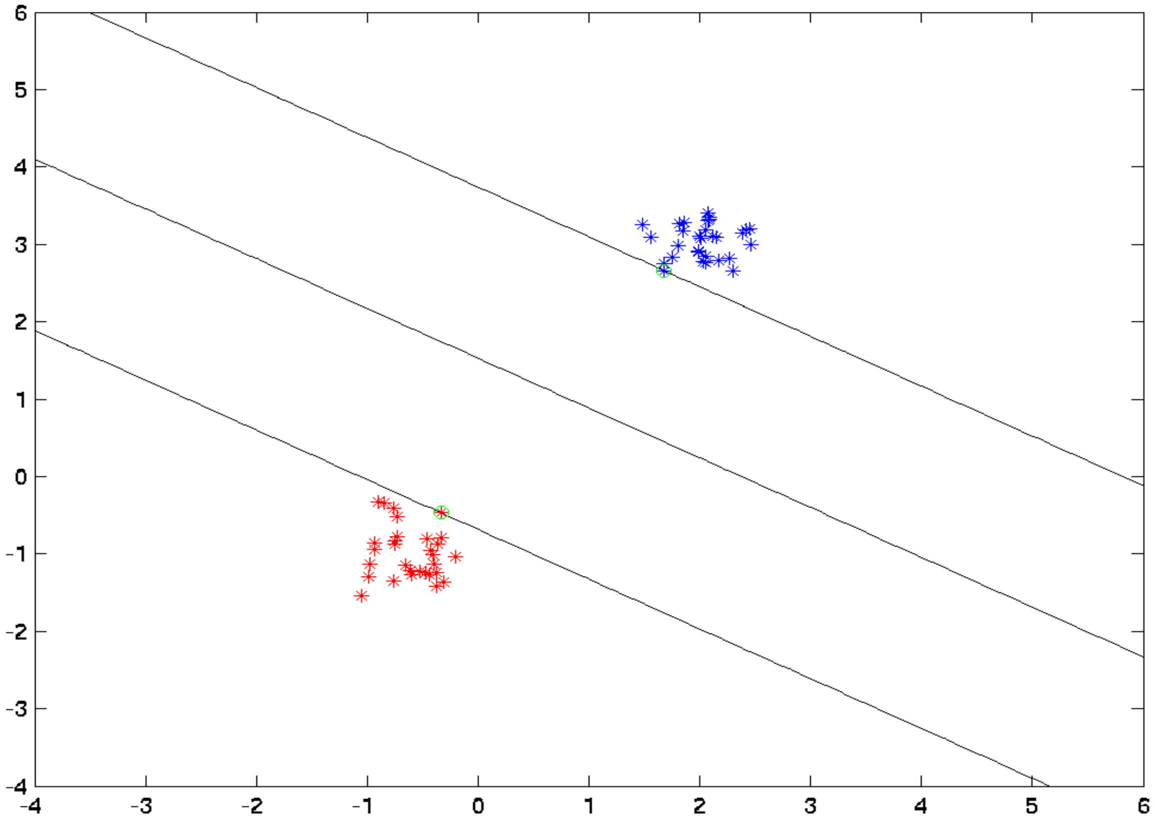
Linear SVM $C = 2$



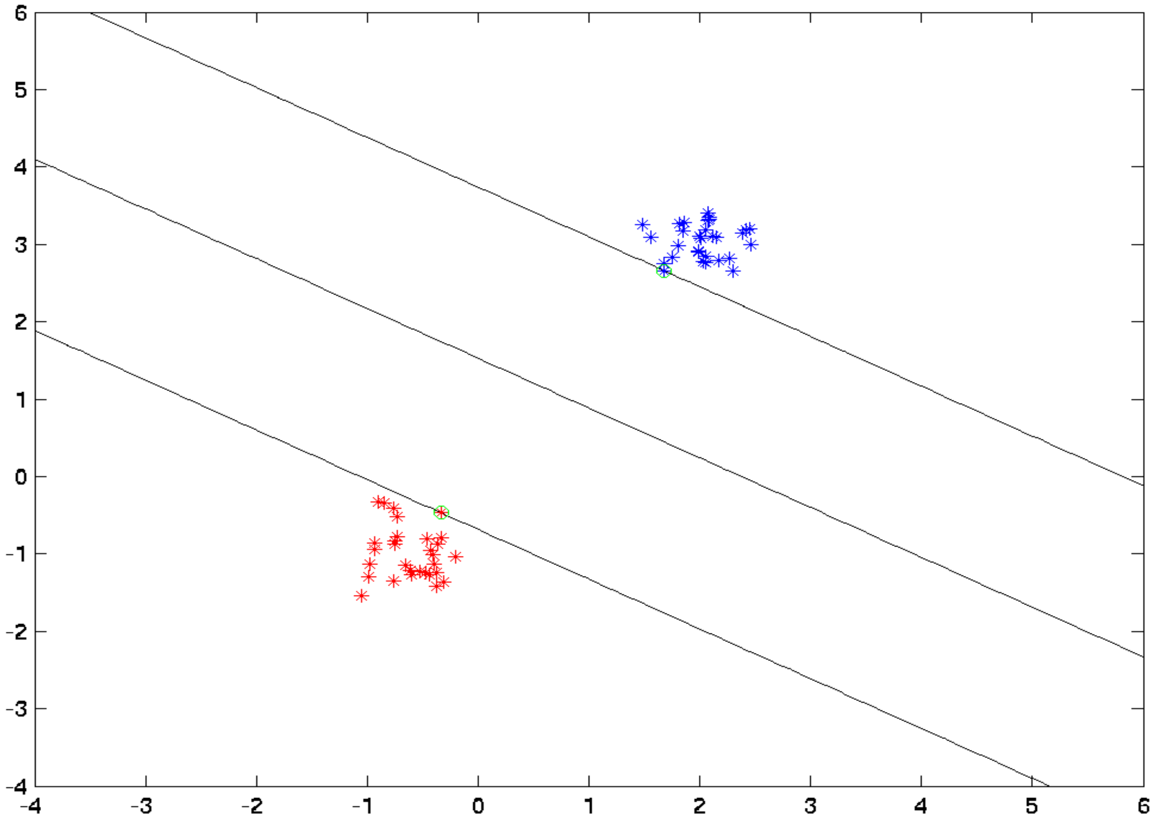
Linear SVM $C = 5$



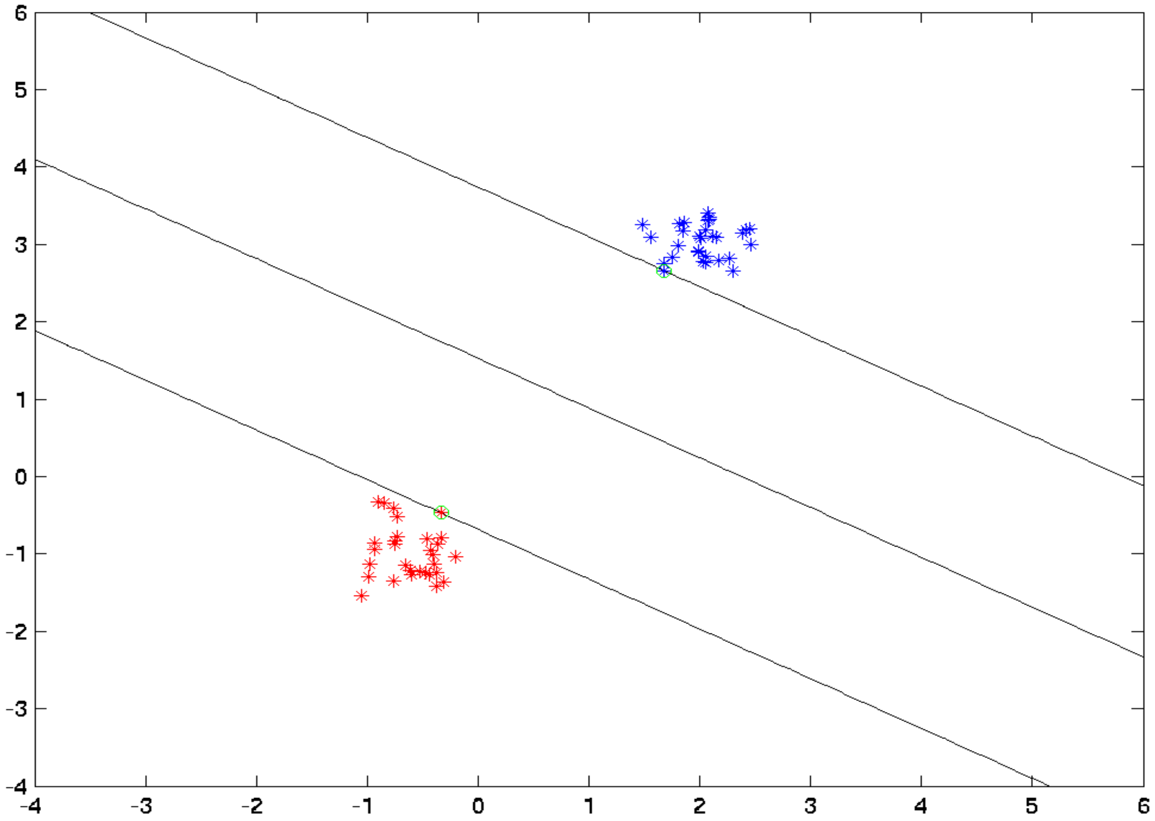
Linear SVM $C = 10$



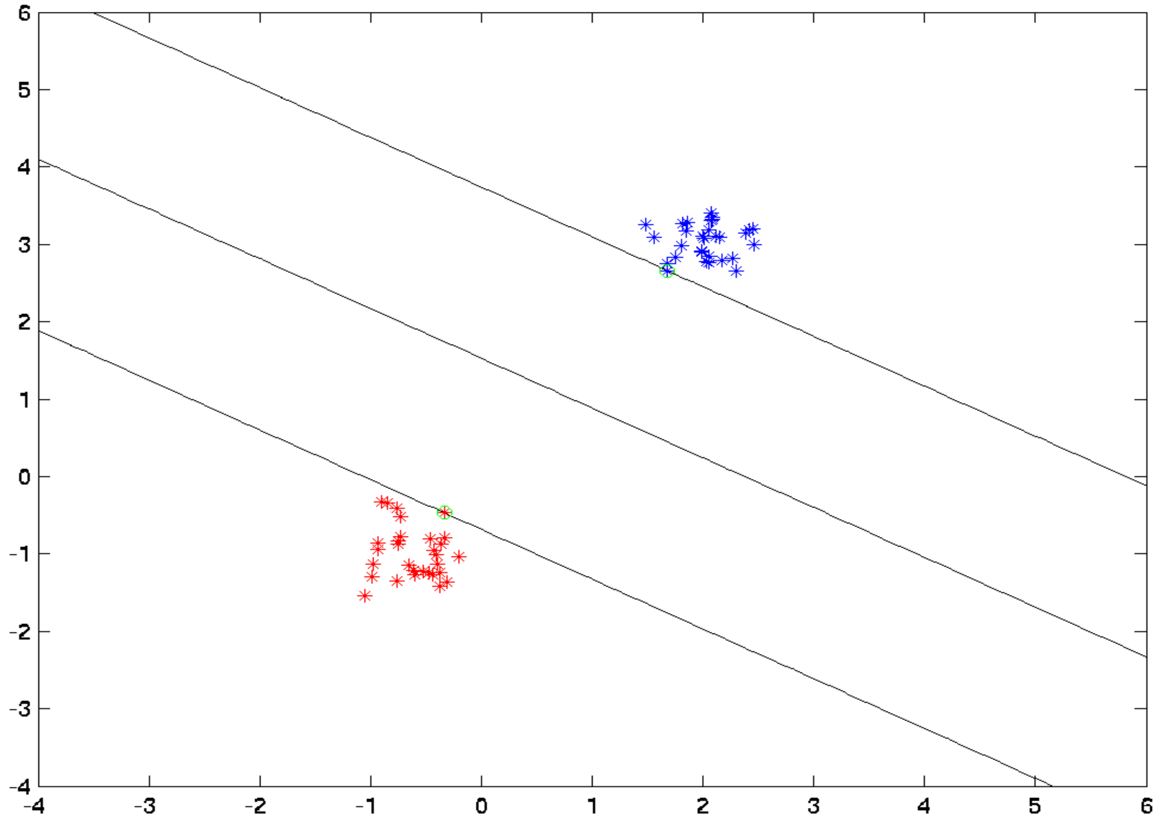
Linear SVM $C = 20$



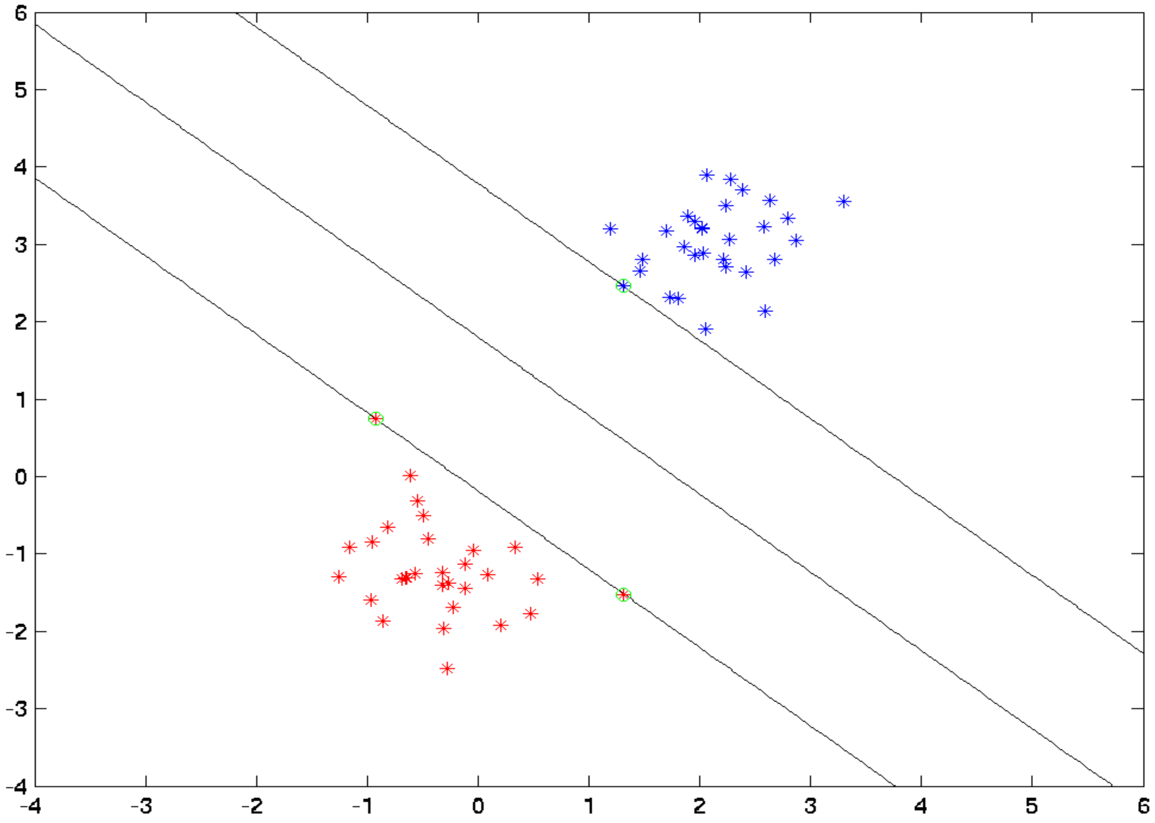
Linear SVM $C = 50$



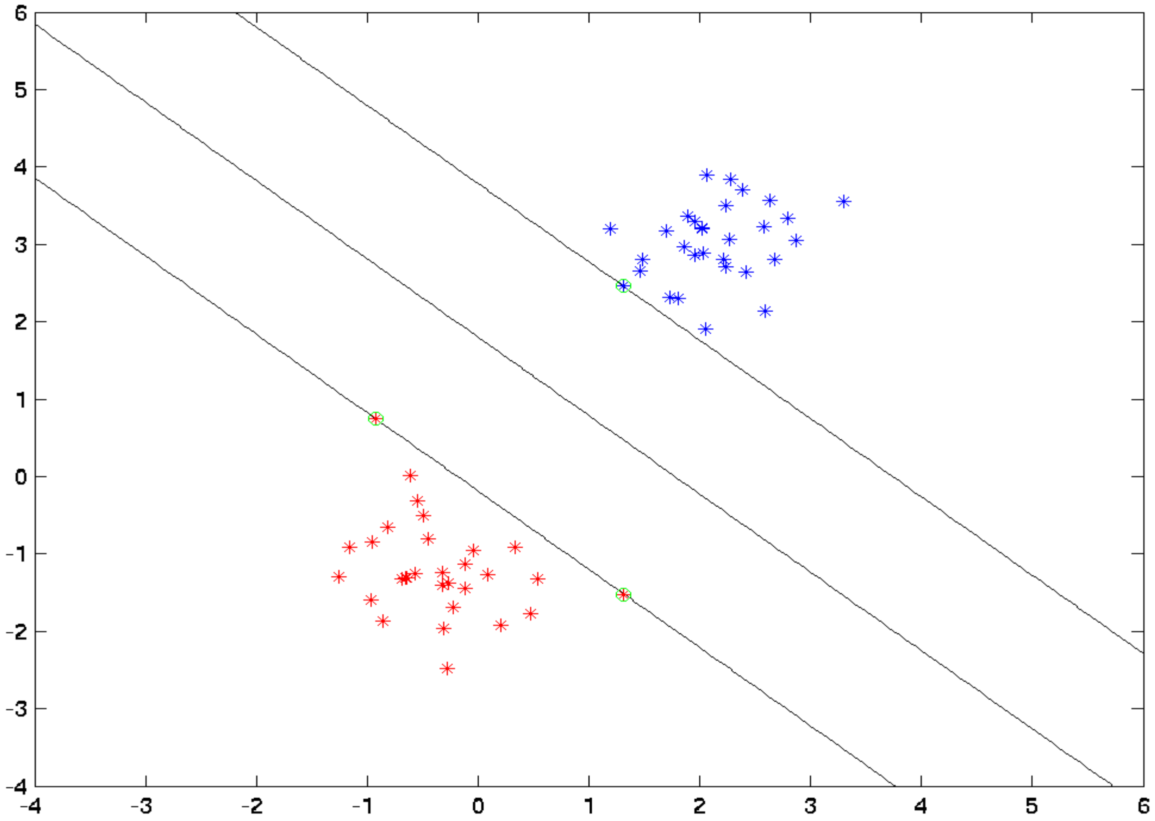
Linear SVM $C = 100$



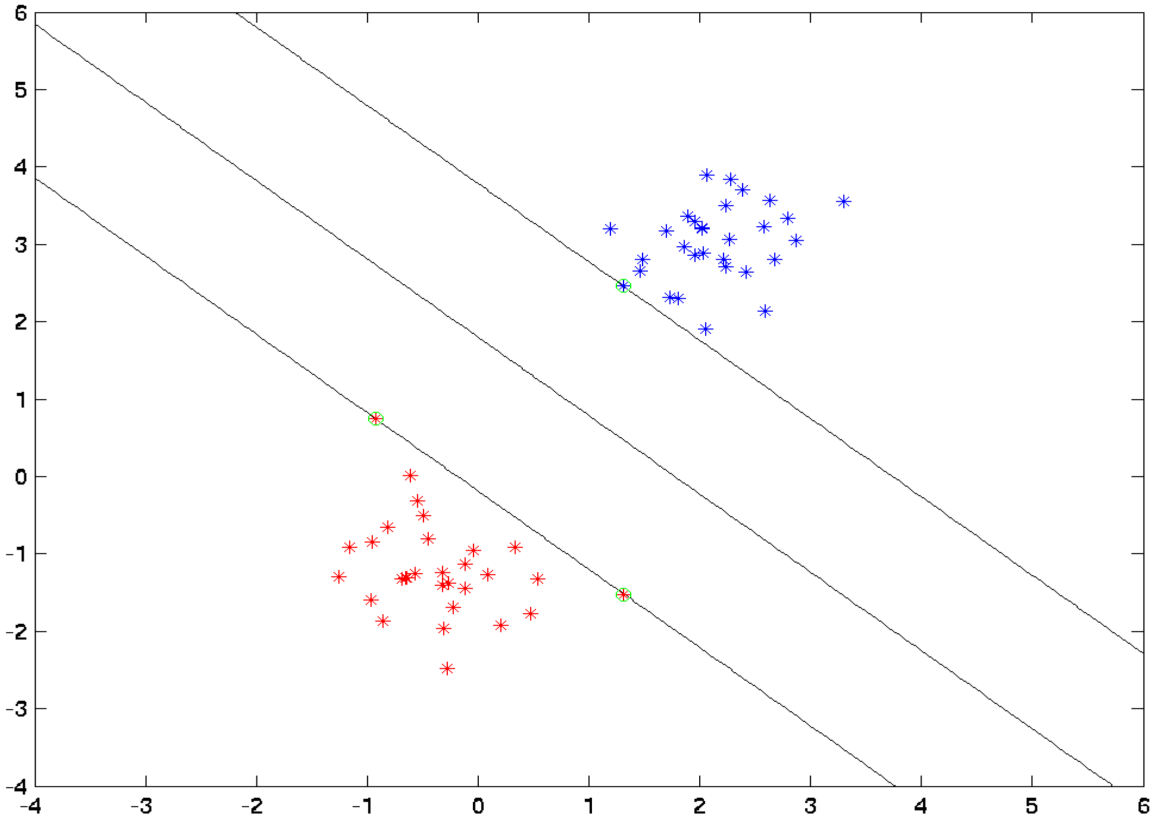
Linear SVM $C = 1$



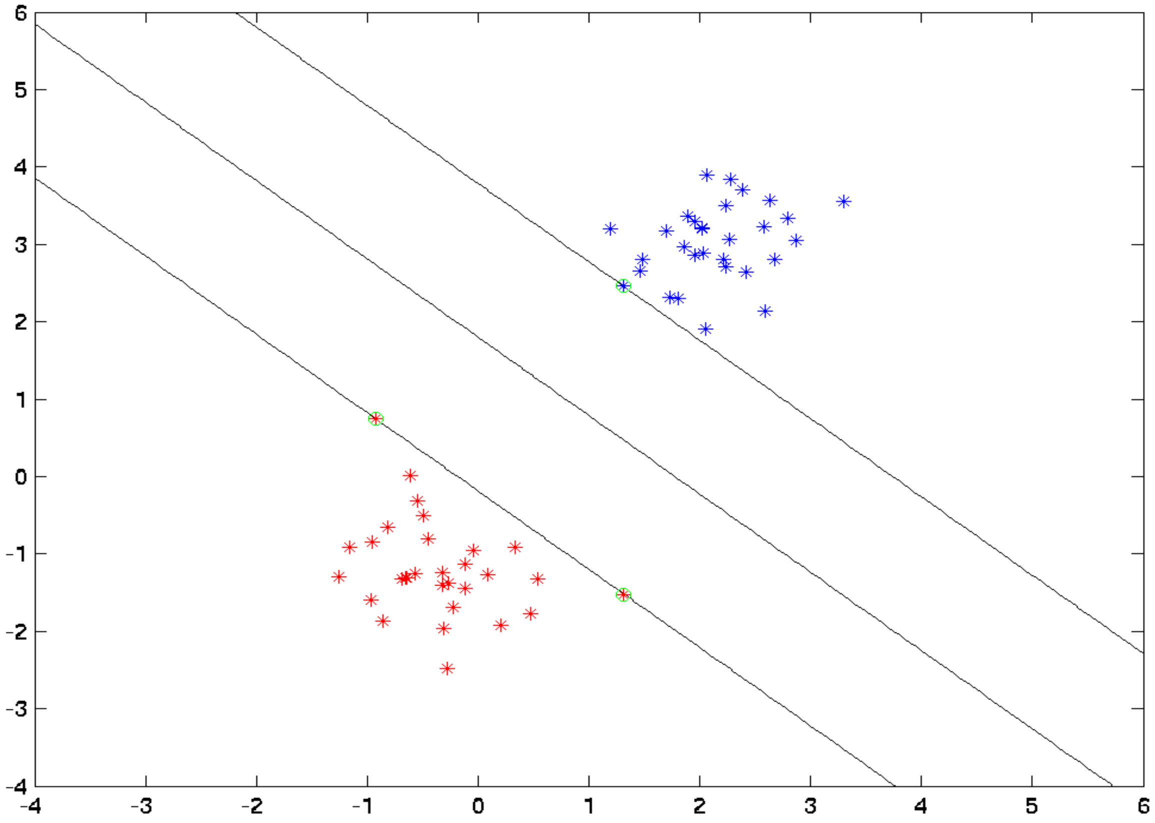
Linear SVM $C = 2$



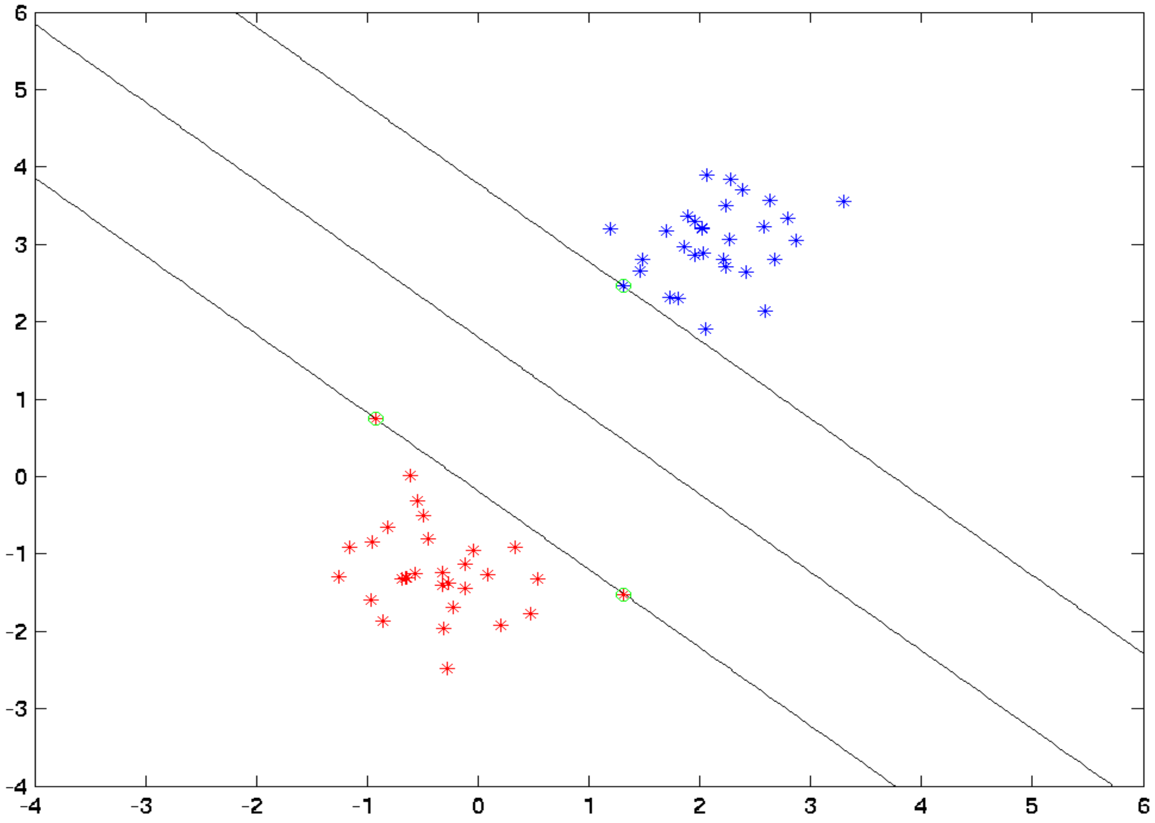
Linear SVM $C = 5$



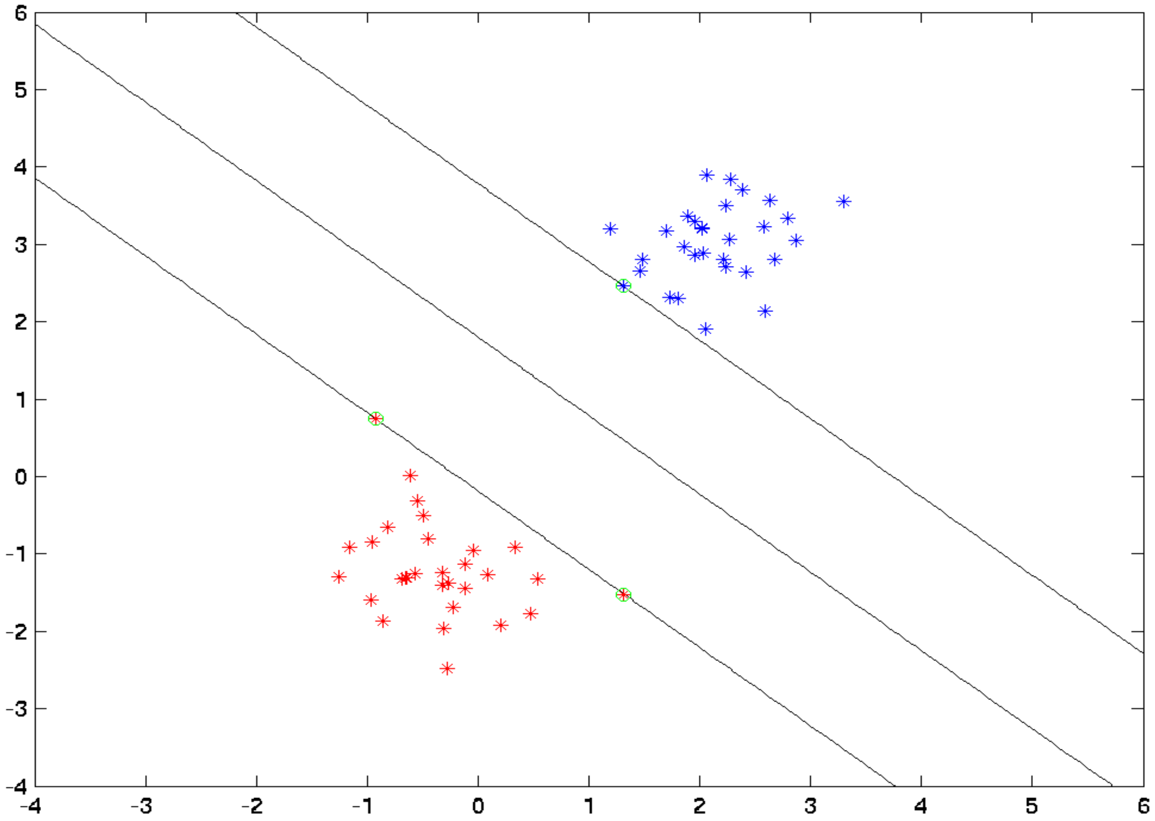
Linear SVM $C = 10$



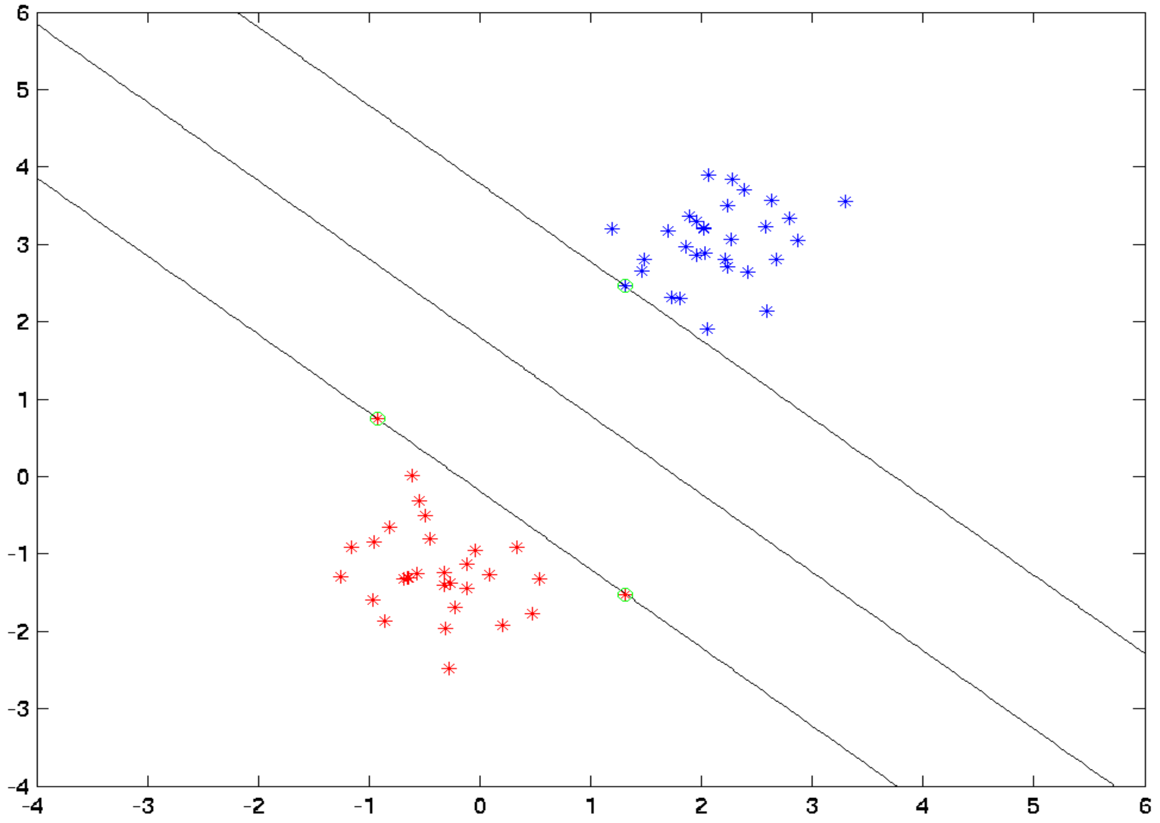
Linear SVM $C = 20$



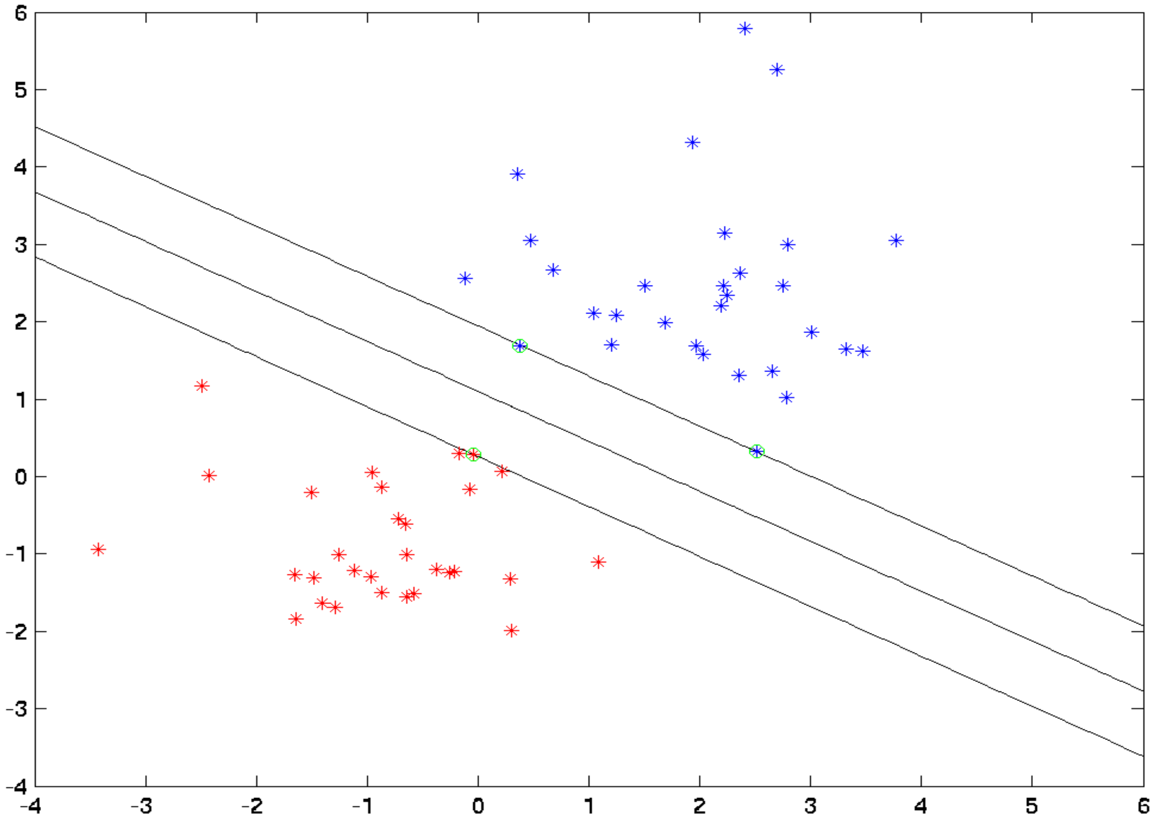
Linear SVM $C = 50$



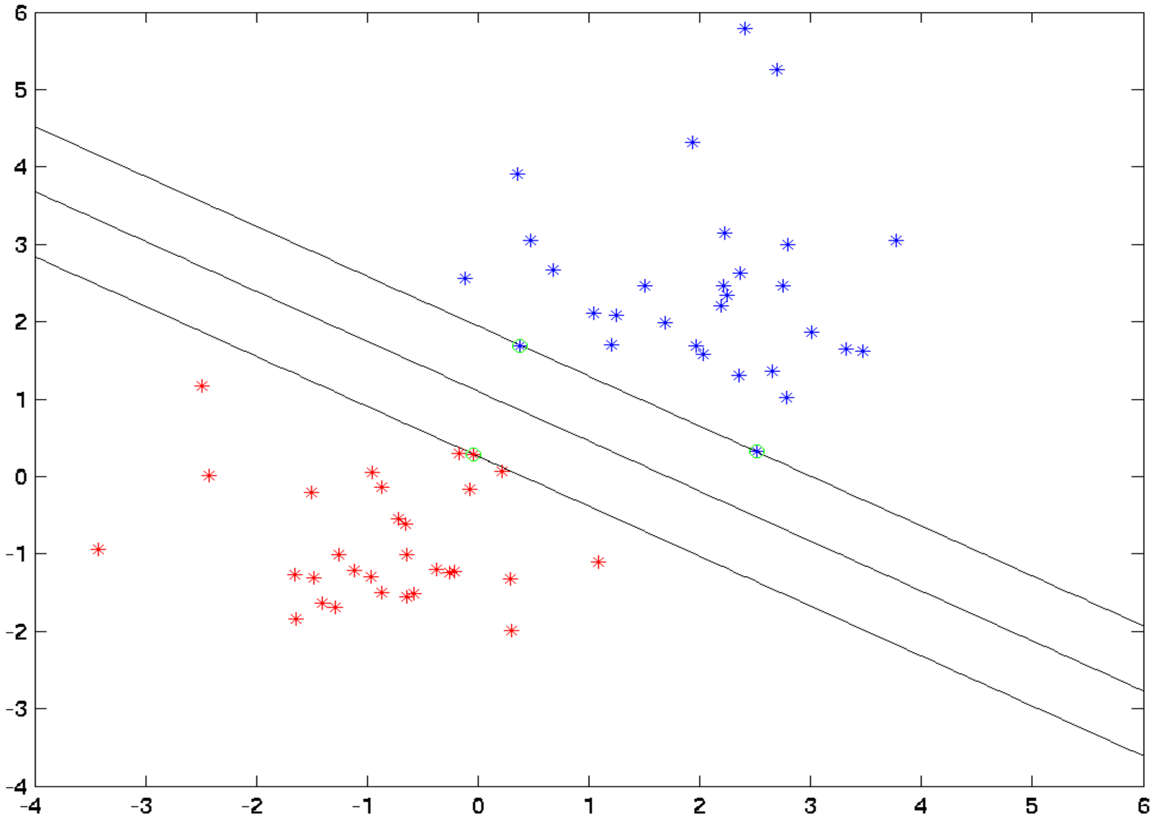
Linear SVM $C = 100$



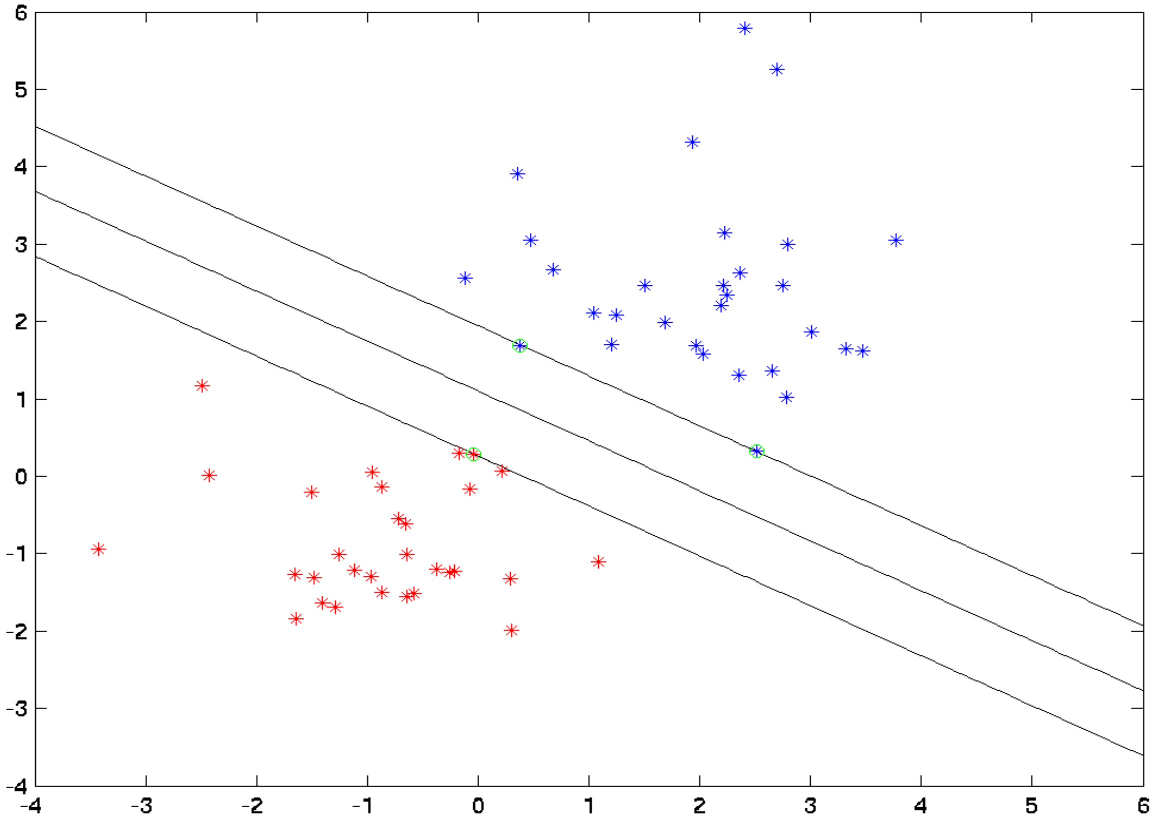
Linear SVM $C = 1$



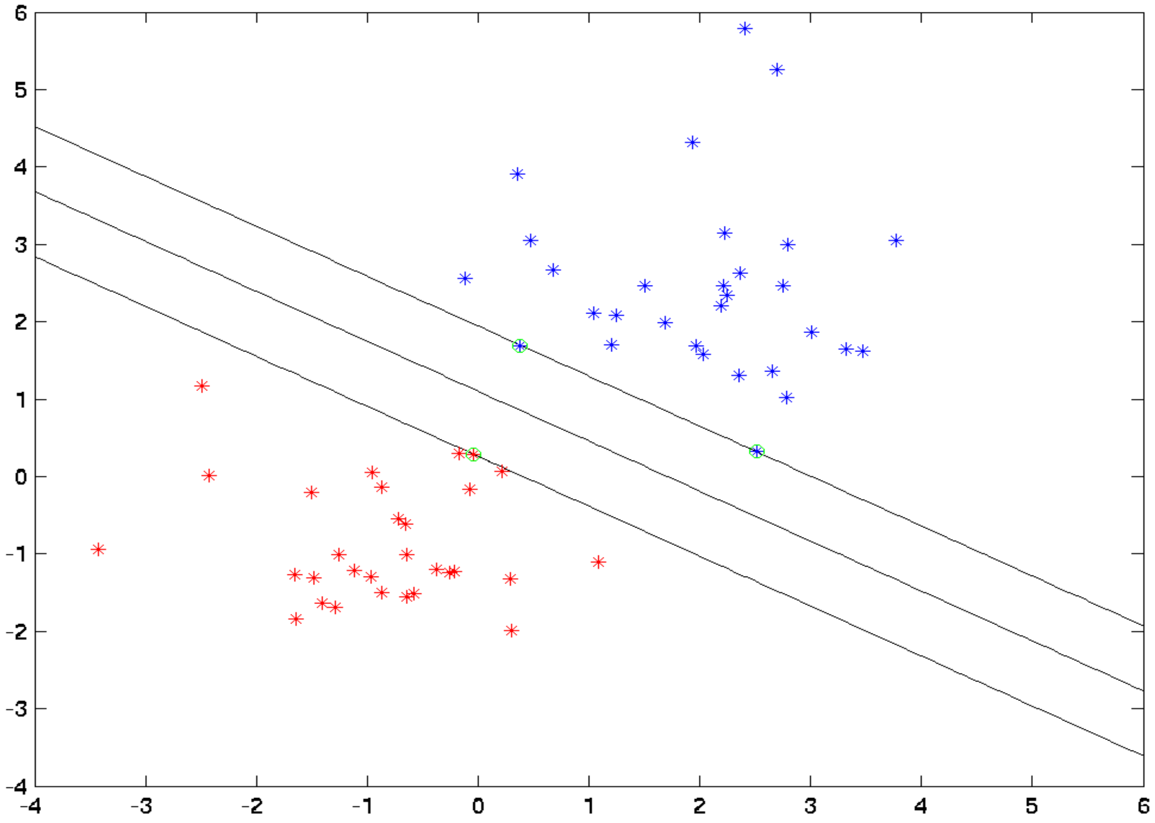
Linear SVM $C = 2$



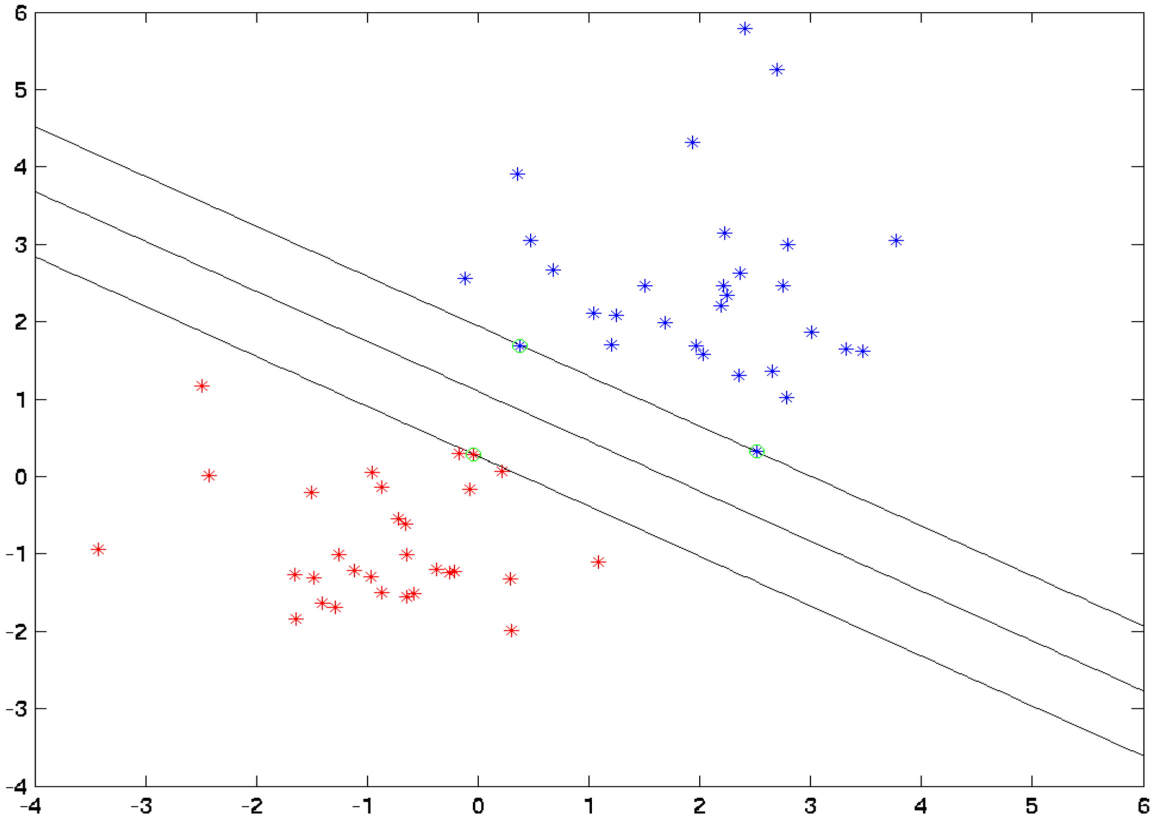
Linear SVM $C = 5$



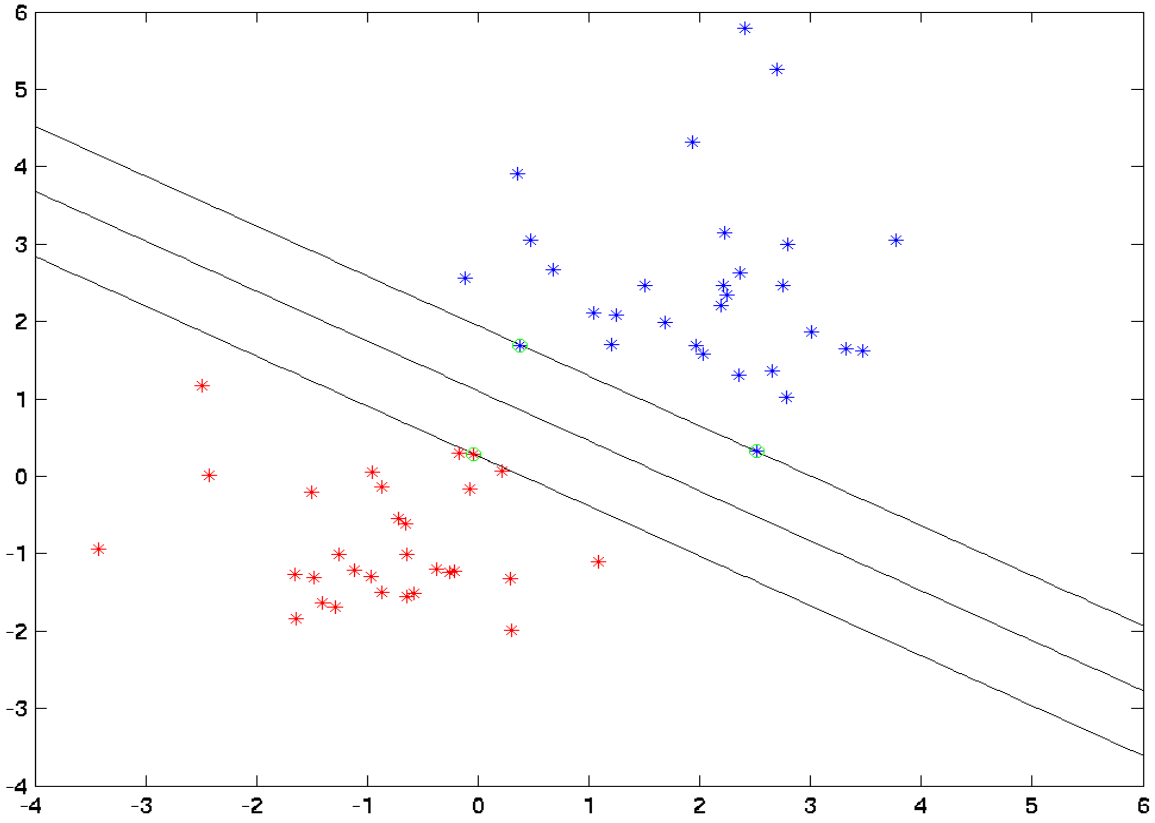
Linear SVM $C = 10$



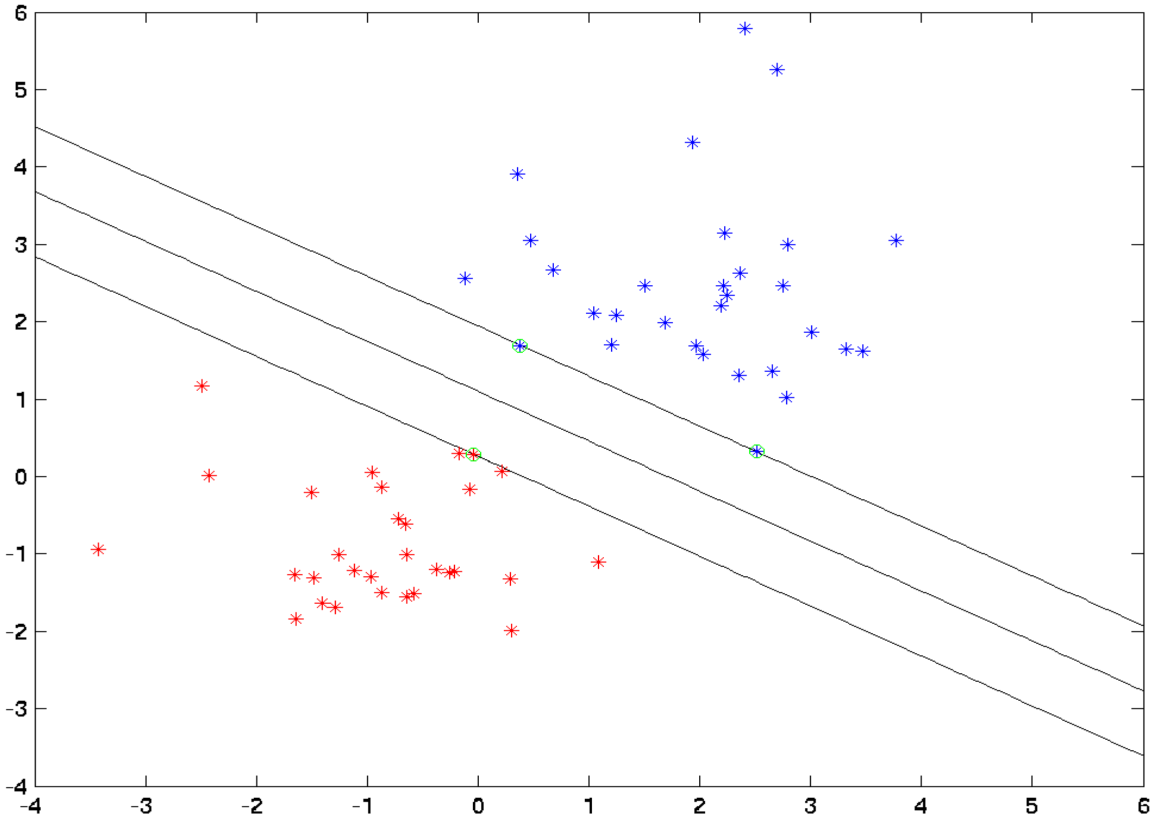
Linear SVM $C = 20$



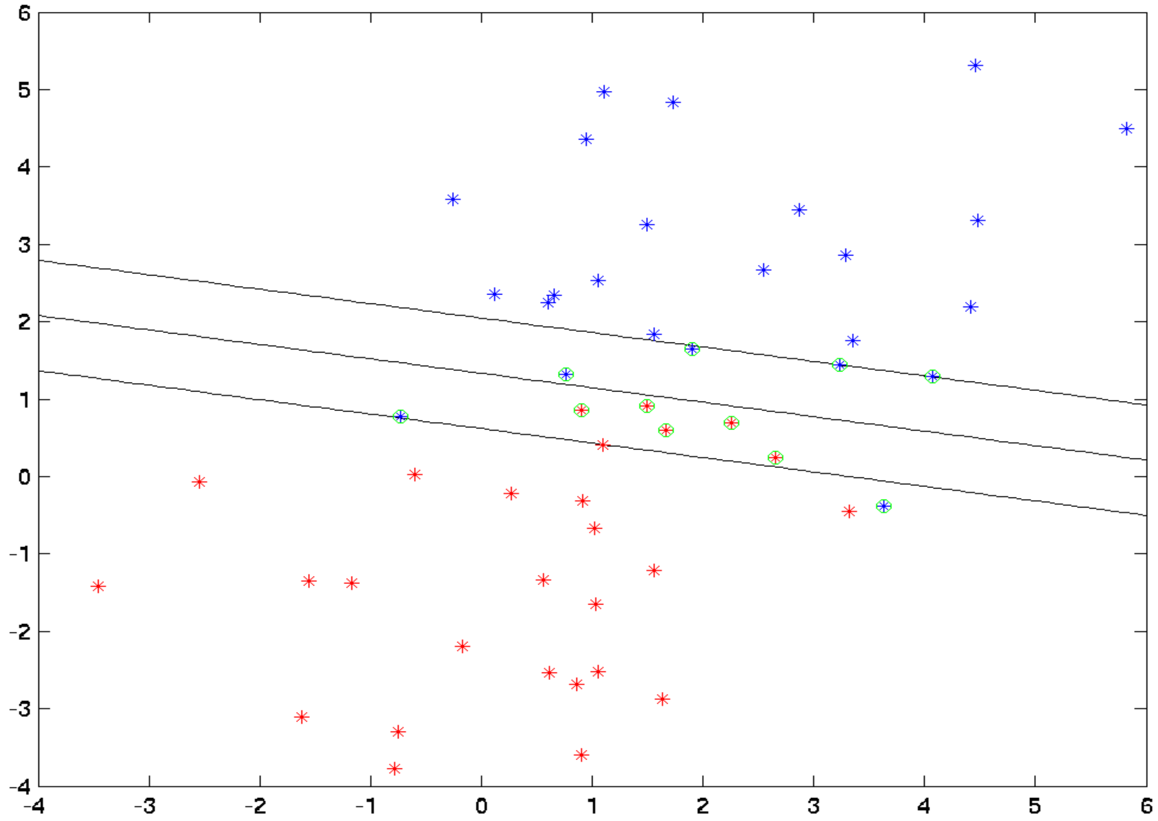
Linear SVM $C = 50$



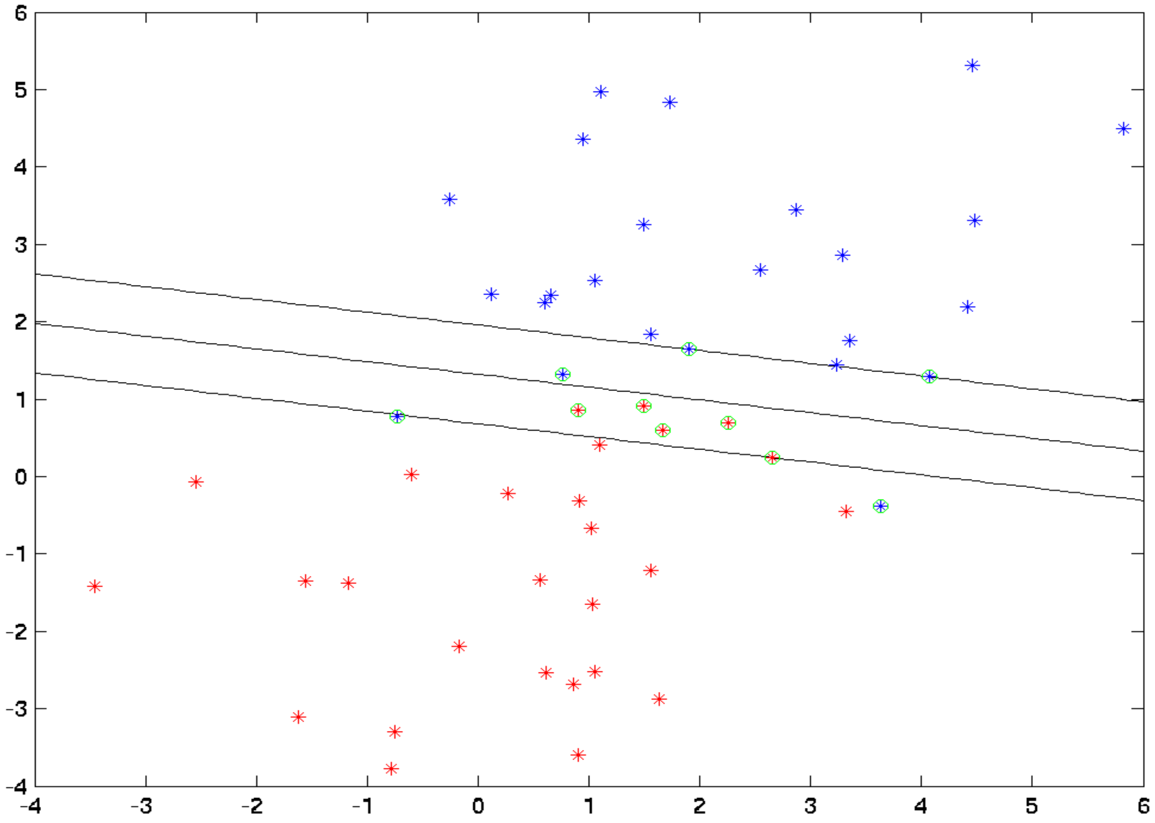
Linear SVM $C = 100$



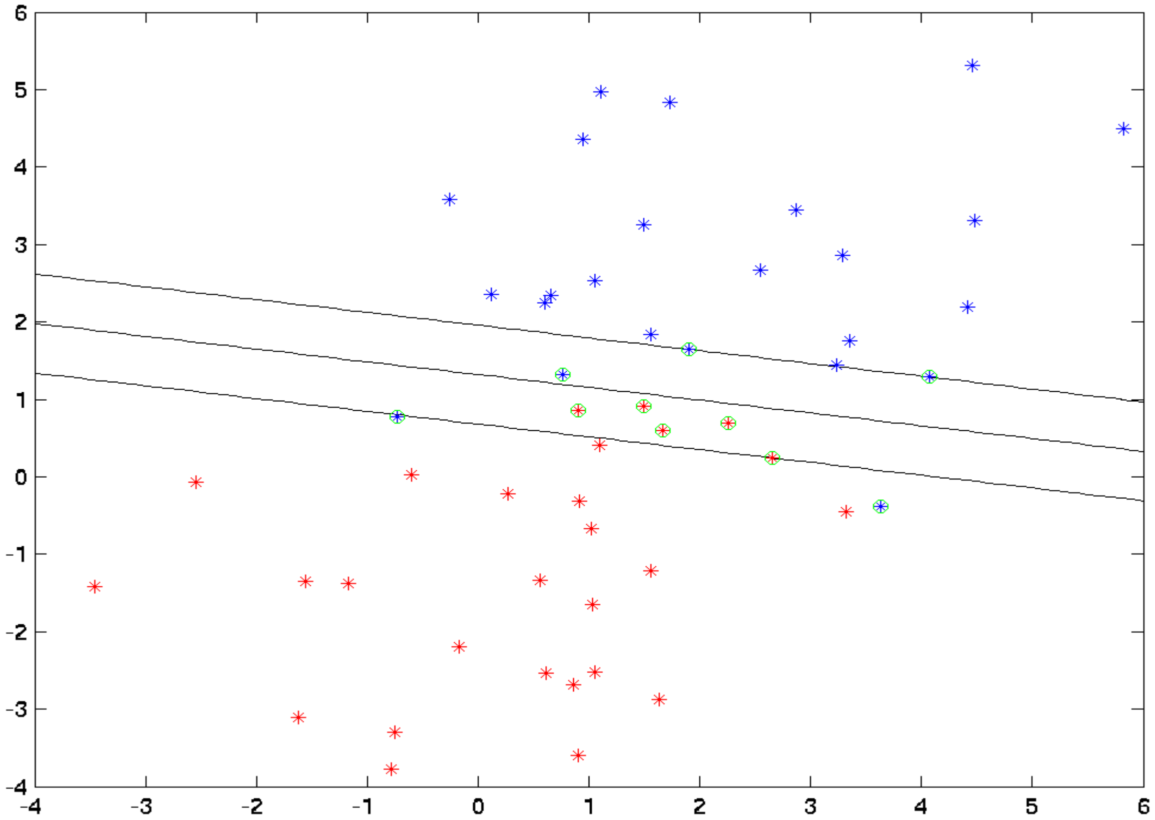
Linear SVM $C = 1$



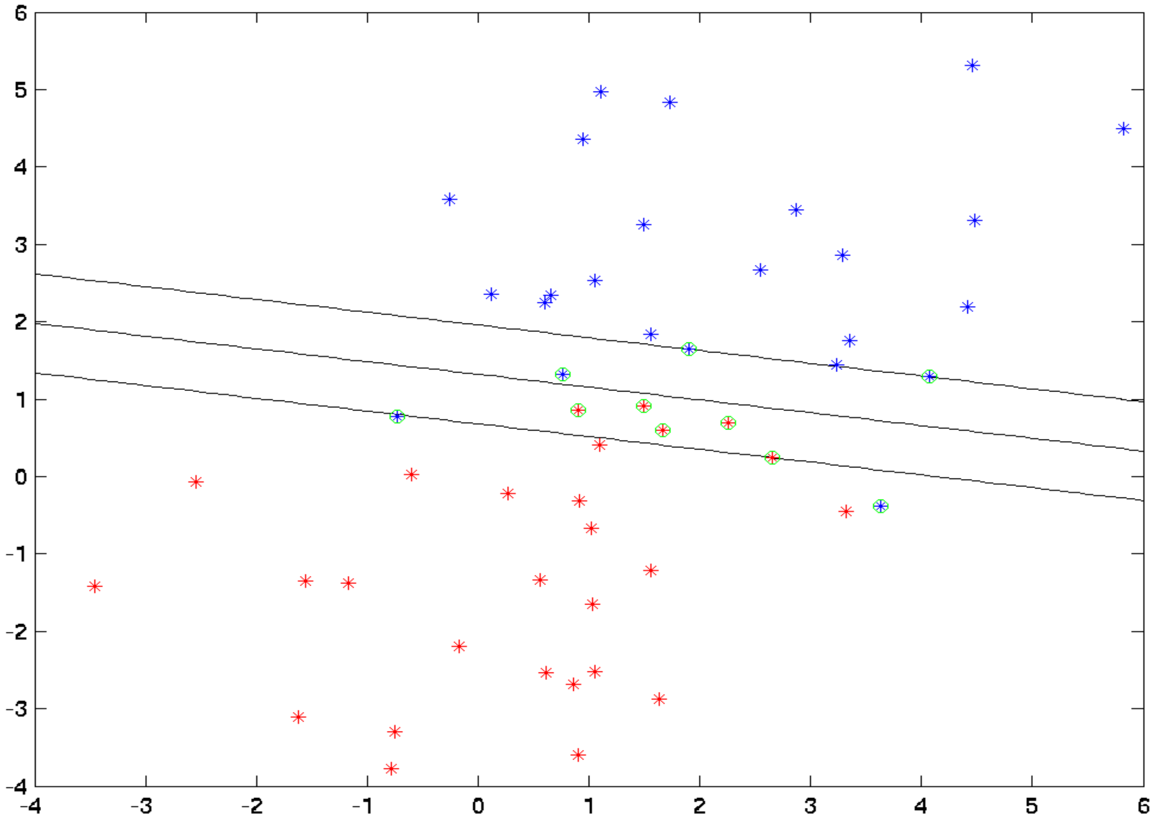
Linear SVM $C = 2$



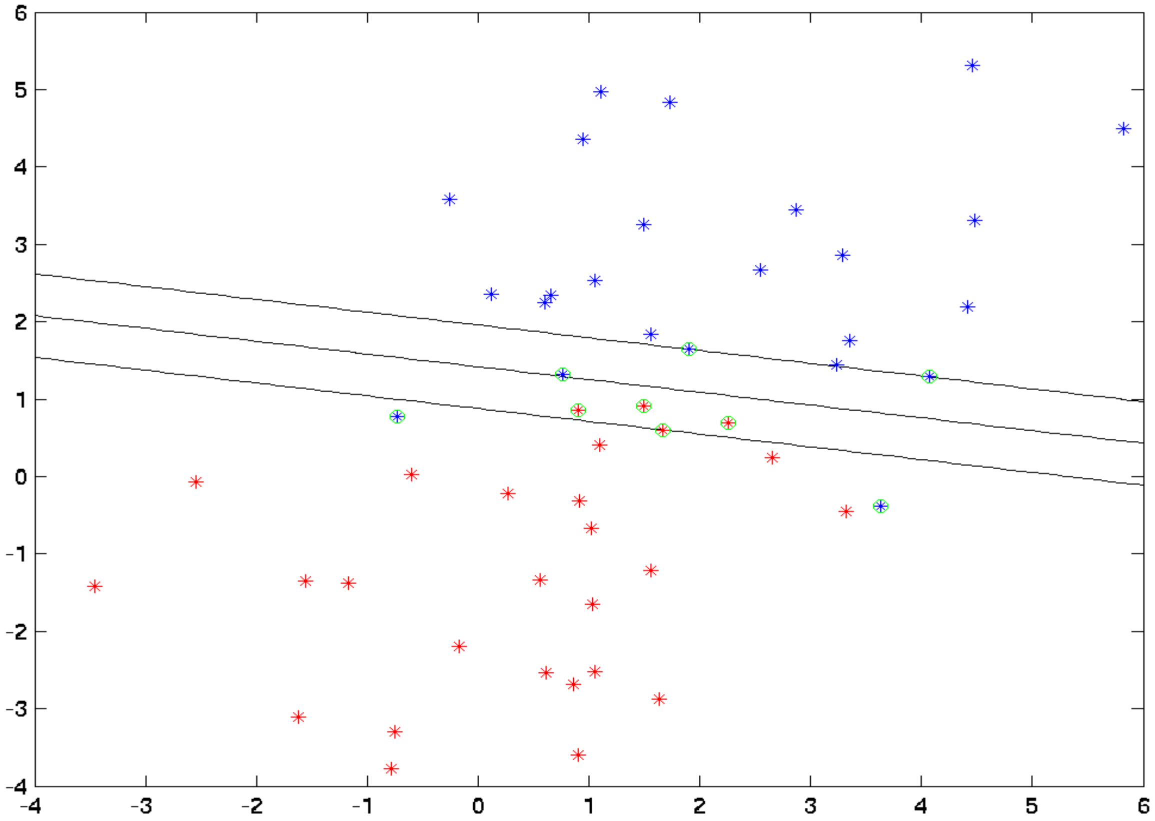
Linear SVM $C = 5$



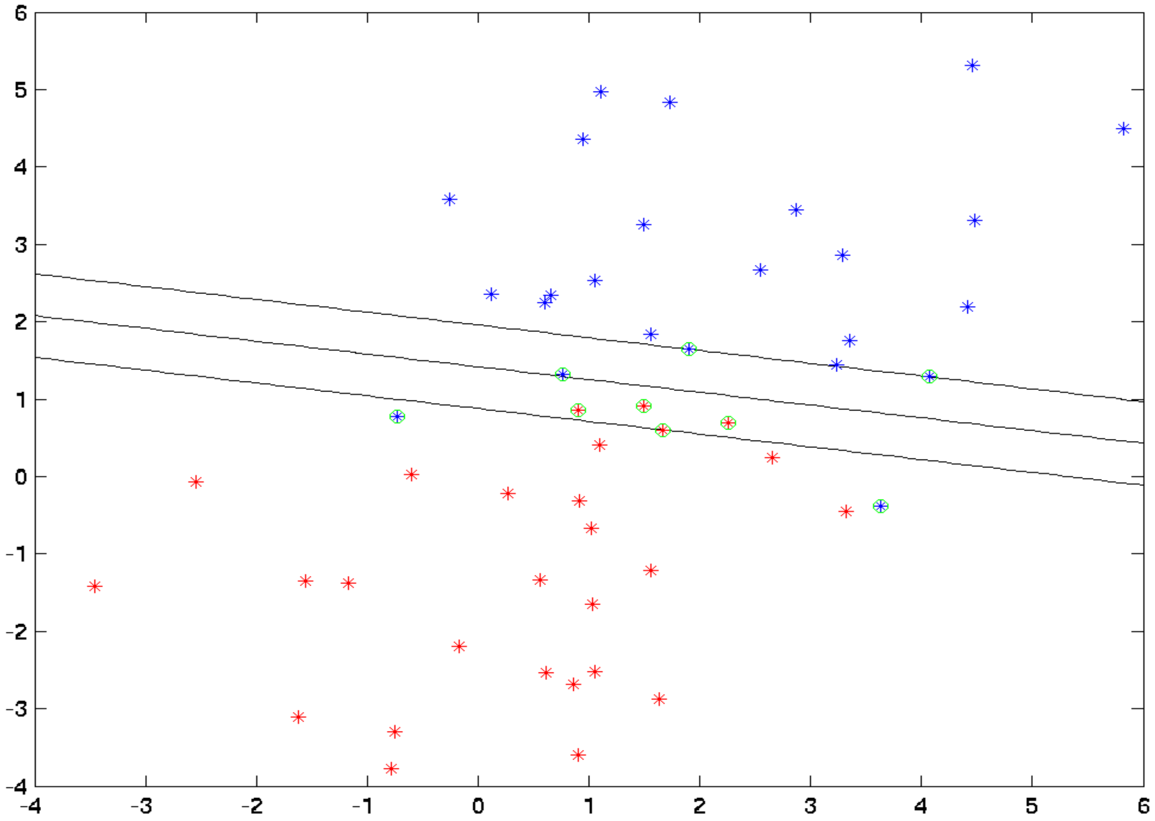
Linear SVM $C = 10$



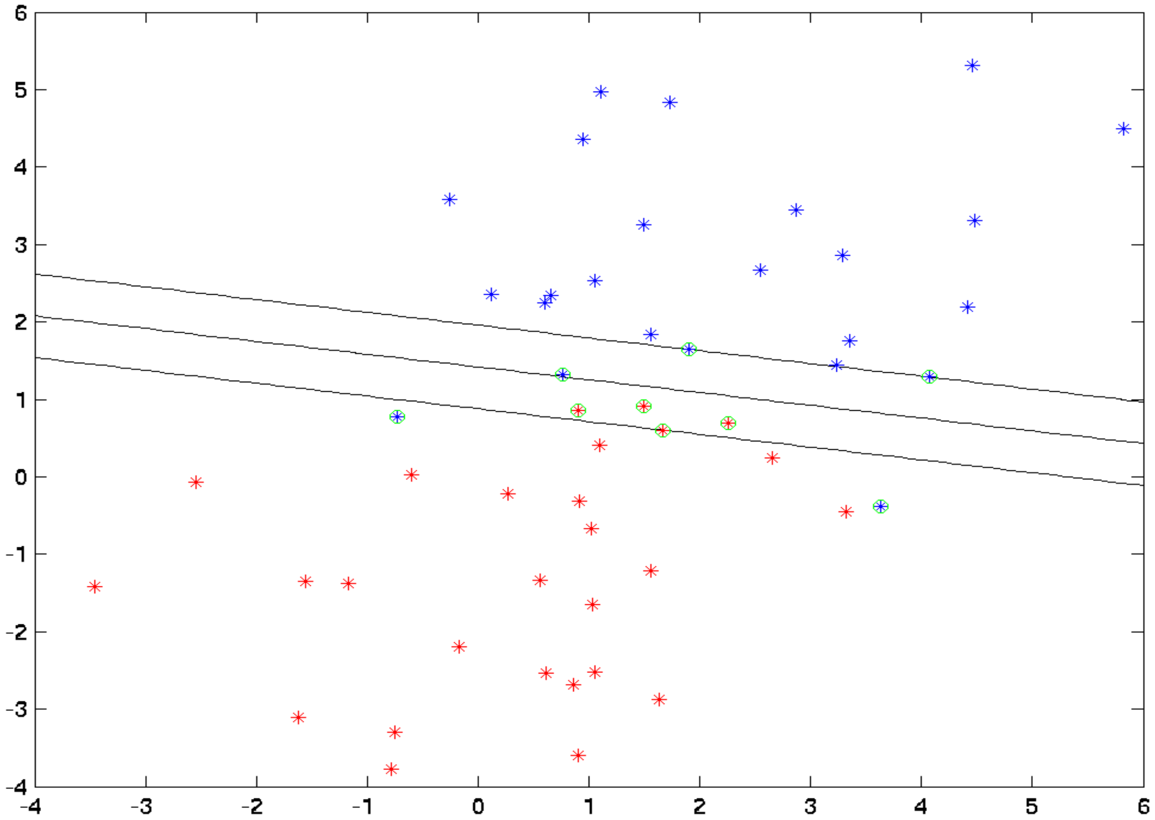
Linear SVM $C = 20$



Linear SVM $C = 50$



Linear SVM $C = 100$



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

Lagrange Function and Constraints

Lagrange Function

We have m more constraints, namely those on the ξ_i, for which we will use η_i as Lagrange multipliers.

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

Saddle Point Conditions

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

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

$$C - \alpha_i - \eta_i = 0 \iff \alpha_i \in [0, C]$$

Dual Optimization Problem

Optimization Problem

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

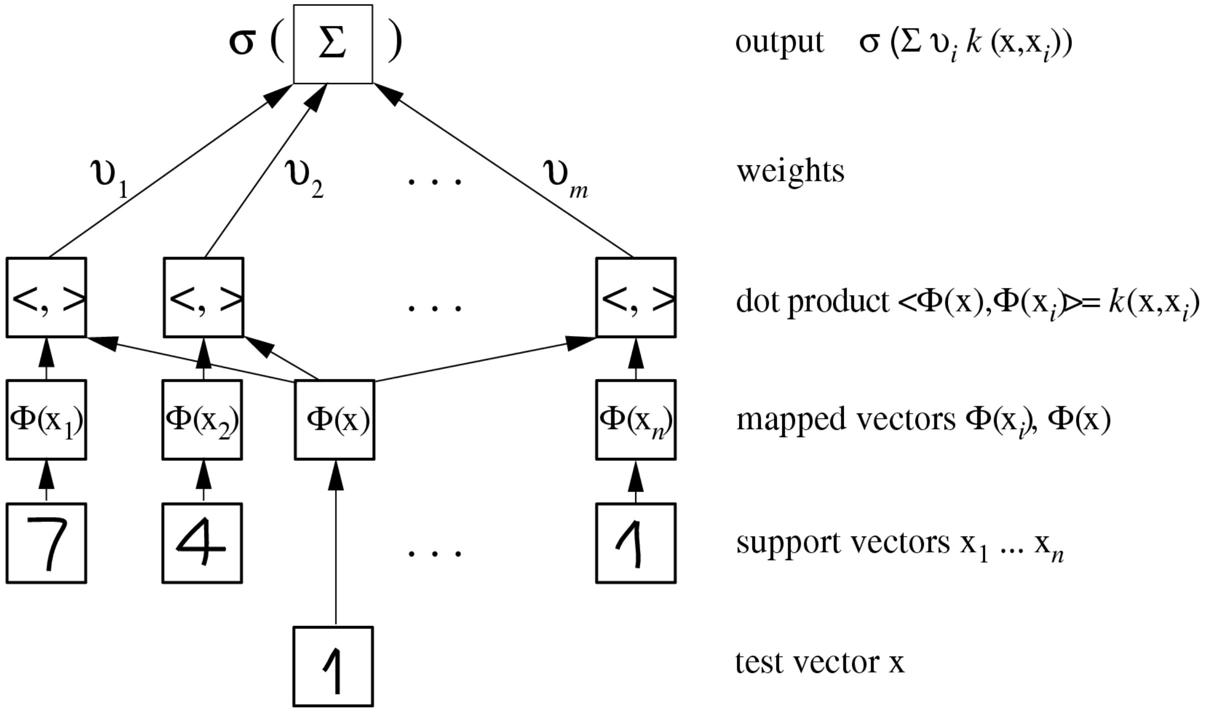
$$\text{subject to } \sum_{i=1}^{m}\alpha_i y_i = 0 \text{ and } C \geq \alpha_i \geq 0 \text{ for all } 1 \leq i \leq m$$

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).

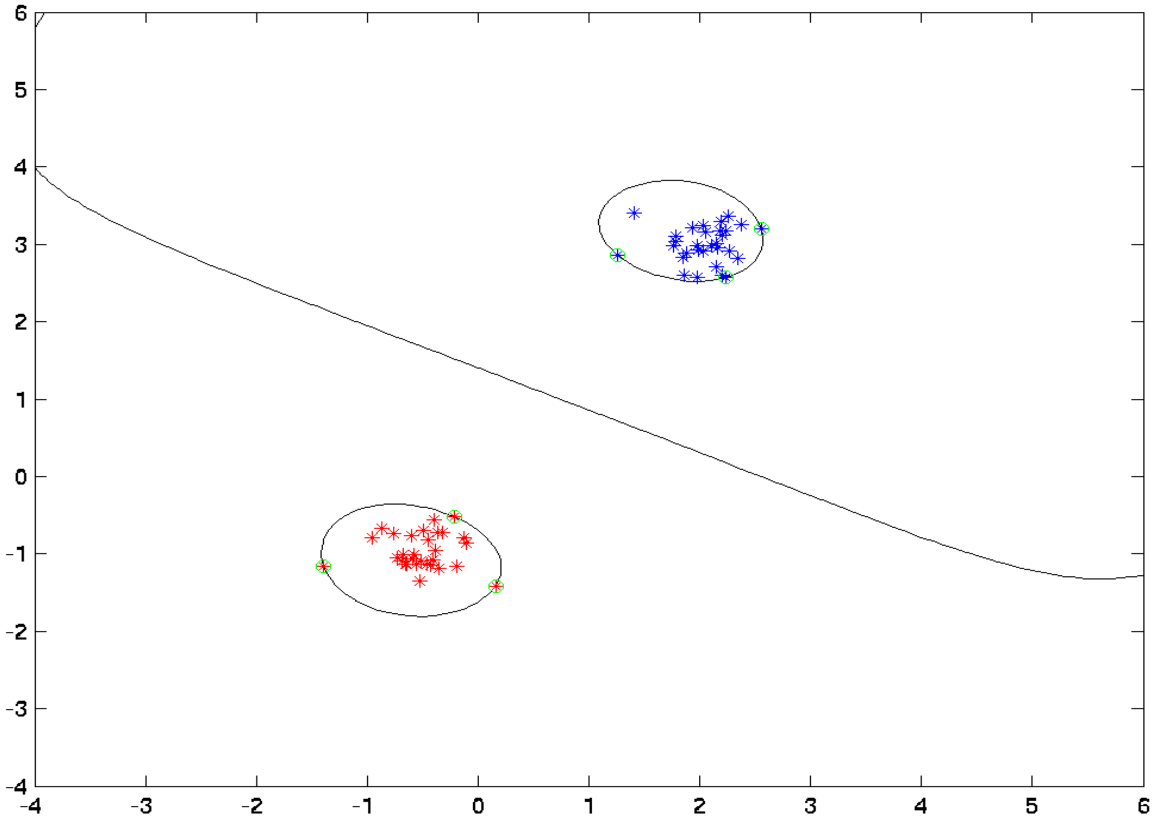
NATIONAL ICT AUSTRALIA

SV Classification Machine

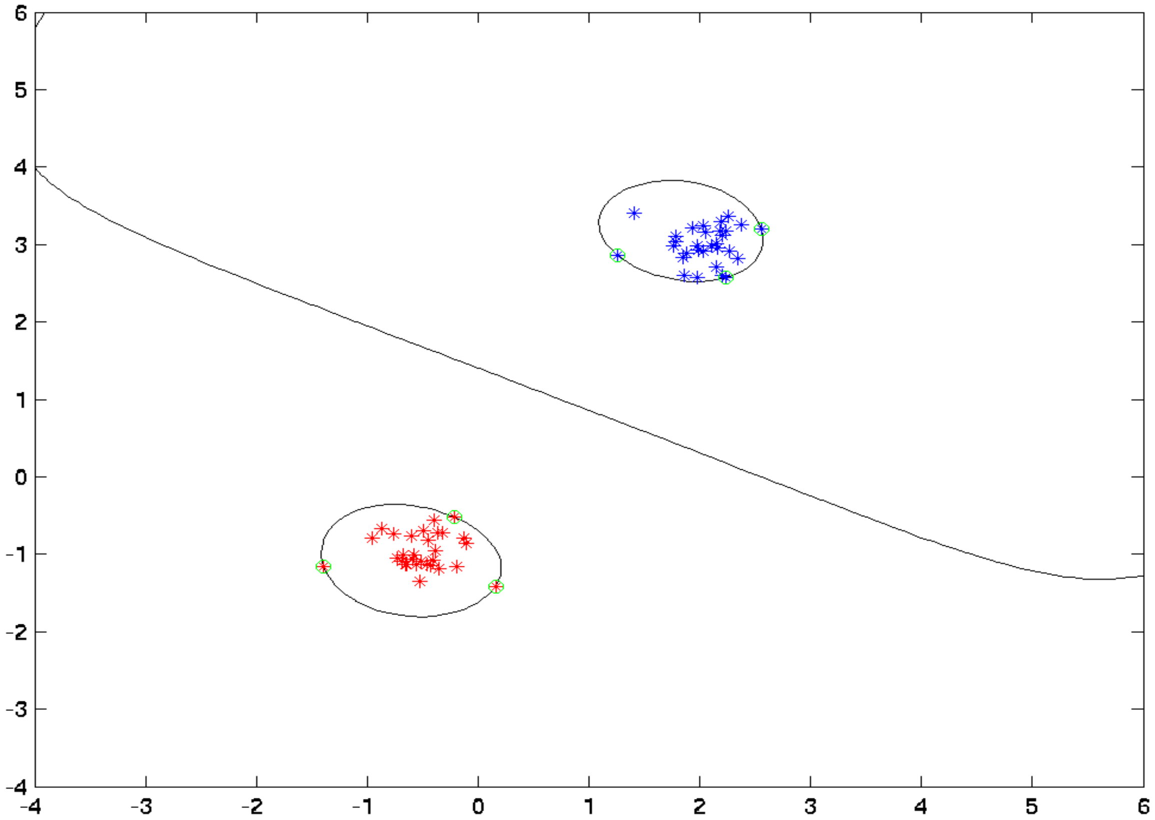


$\sigma\ (\ \Sigma\ )$
output $\sigma\,(\Sigma\, \upsilon_i\, k\,(x,x_i))$
υ_1
υ_2
. . .
υ_m
weights
<, >
dot product $<\Phi(x),\Phi(x_i)> = k(x,x_i)$
$\Phi(x_1)$
$\Phi(x_2)$
$\Phi(x)$
$\Phi(x_n)$
mapped vectors $\Phi(x_i)$, $\Phi(x)$
7
4
1
support vectors x_1 ... x_n
1
test vector x

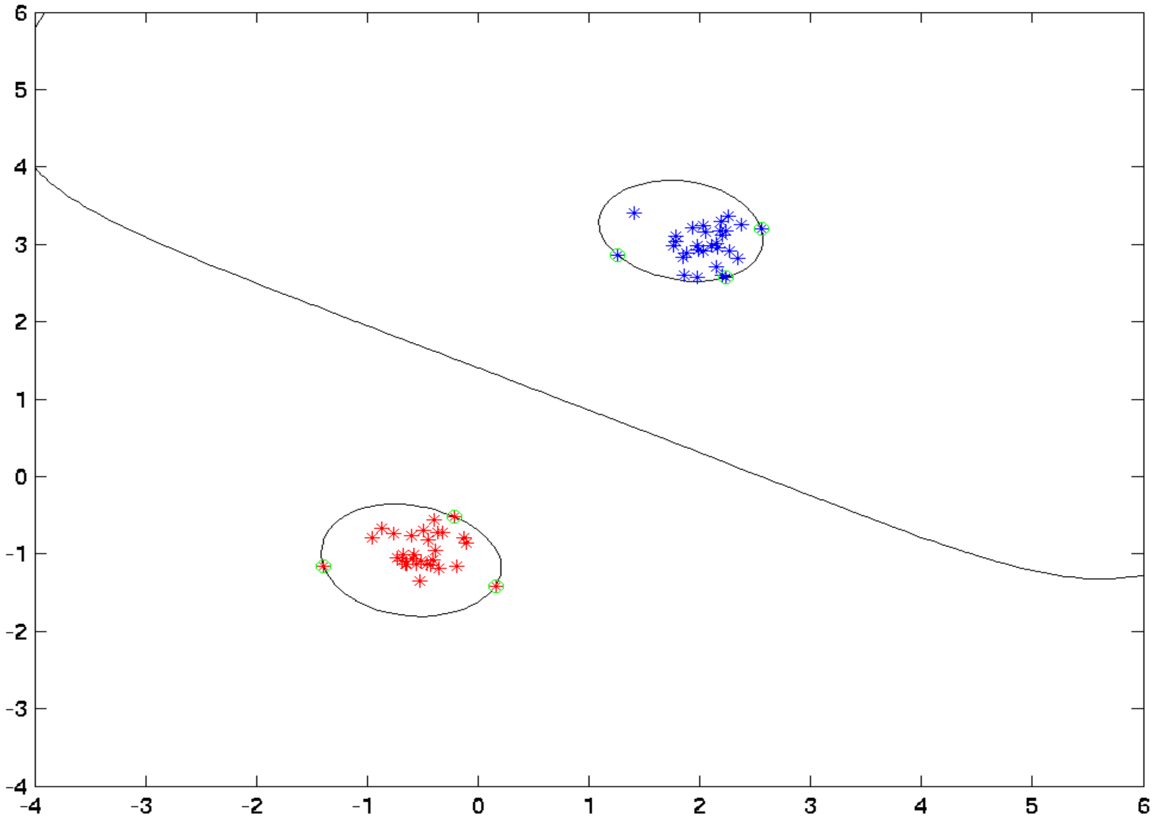
Gaussian RBF with $C = 1$



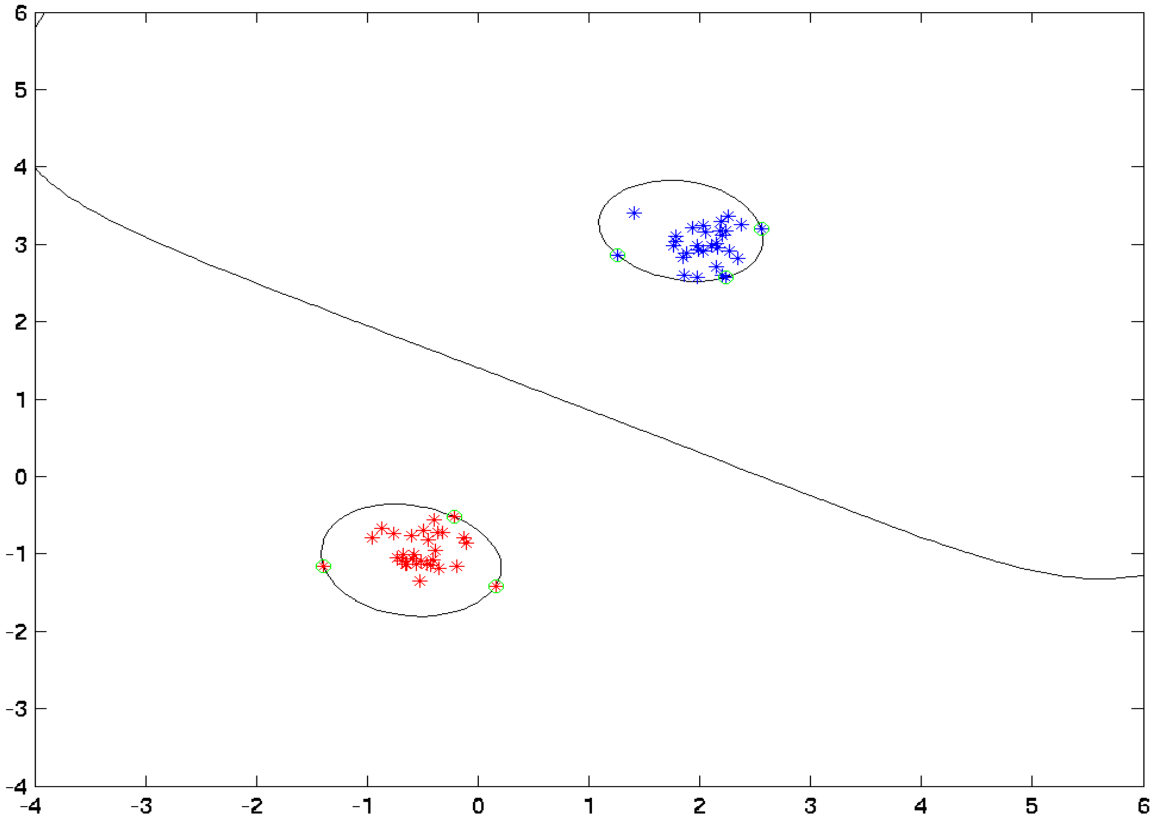
Gaussian RBF with $C = 2$



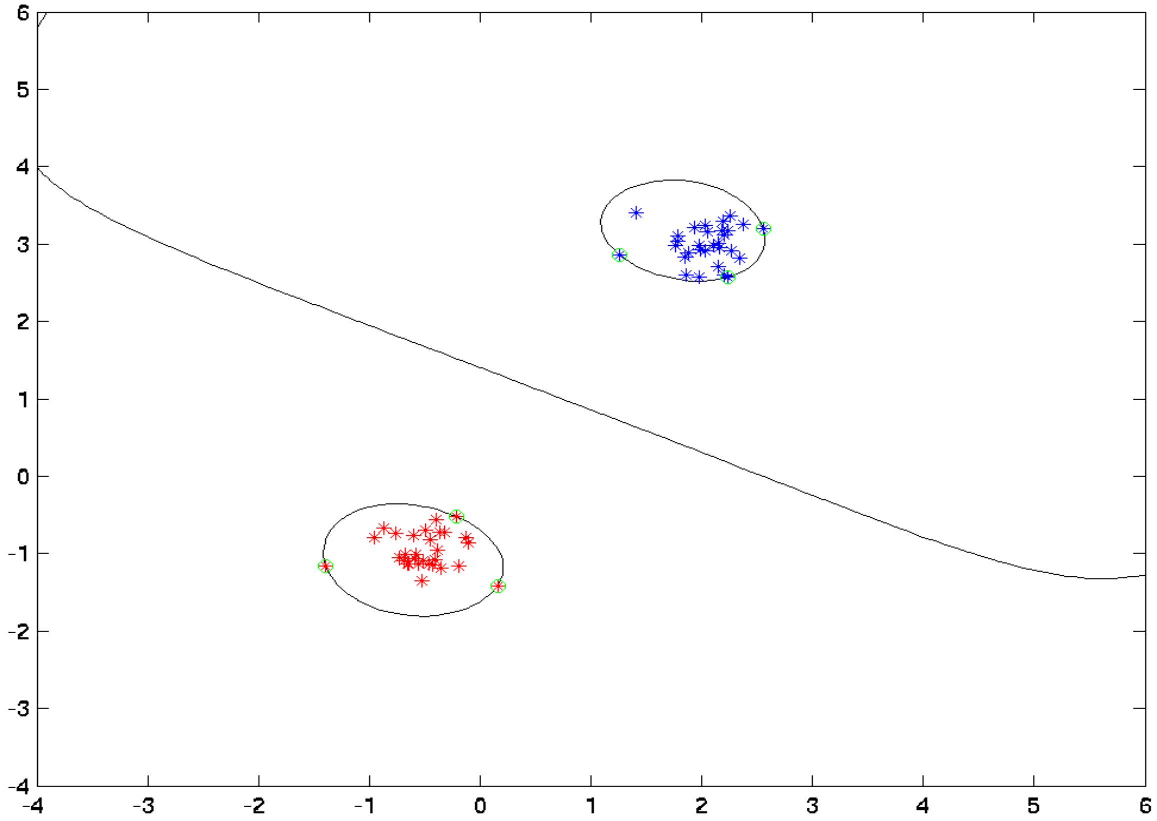
Gaussian RBF with $C = 5$



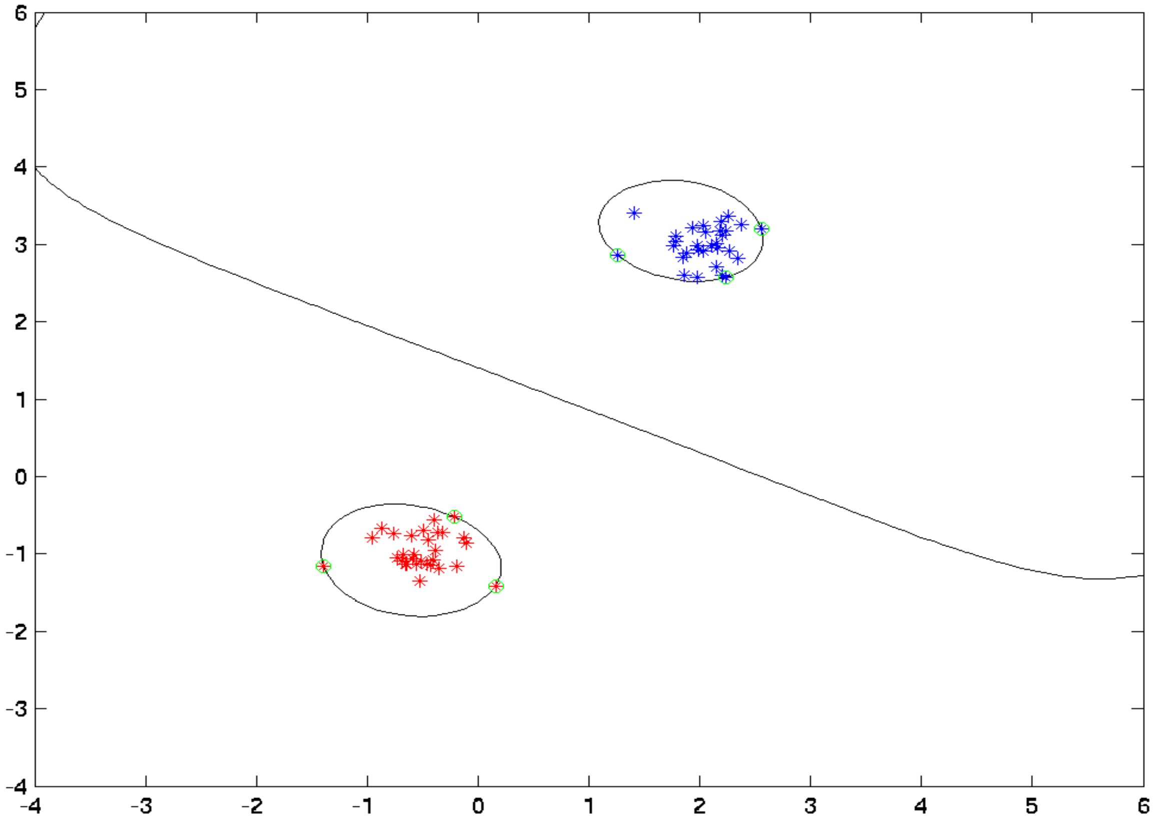
Gaussian RBF with $C = 10$



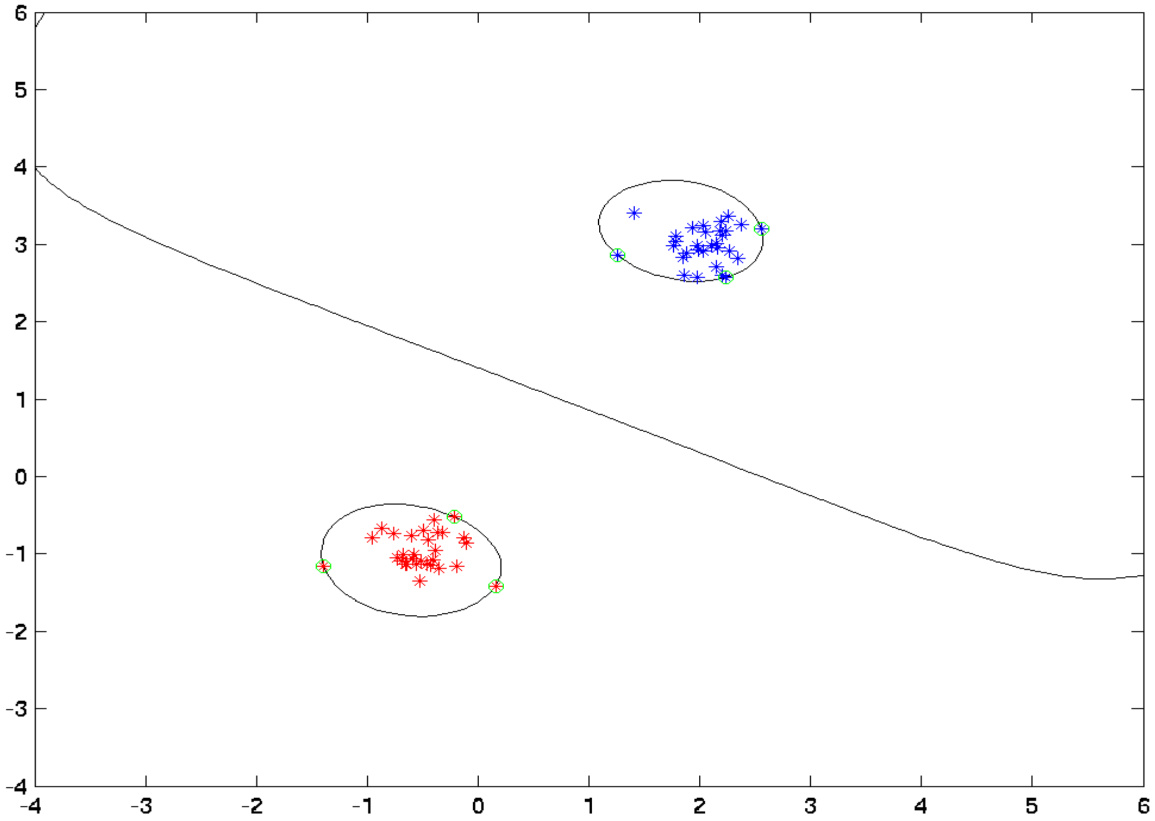
Gaussian RBF with $C = 20$



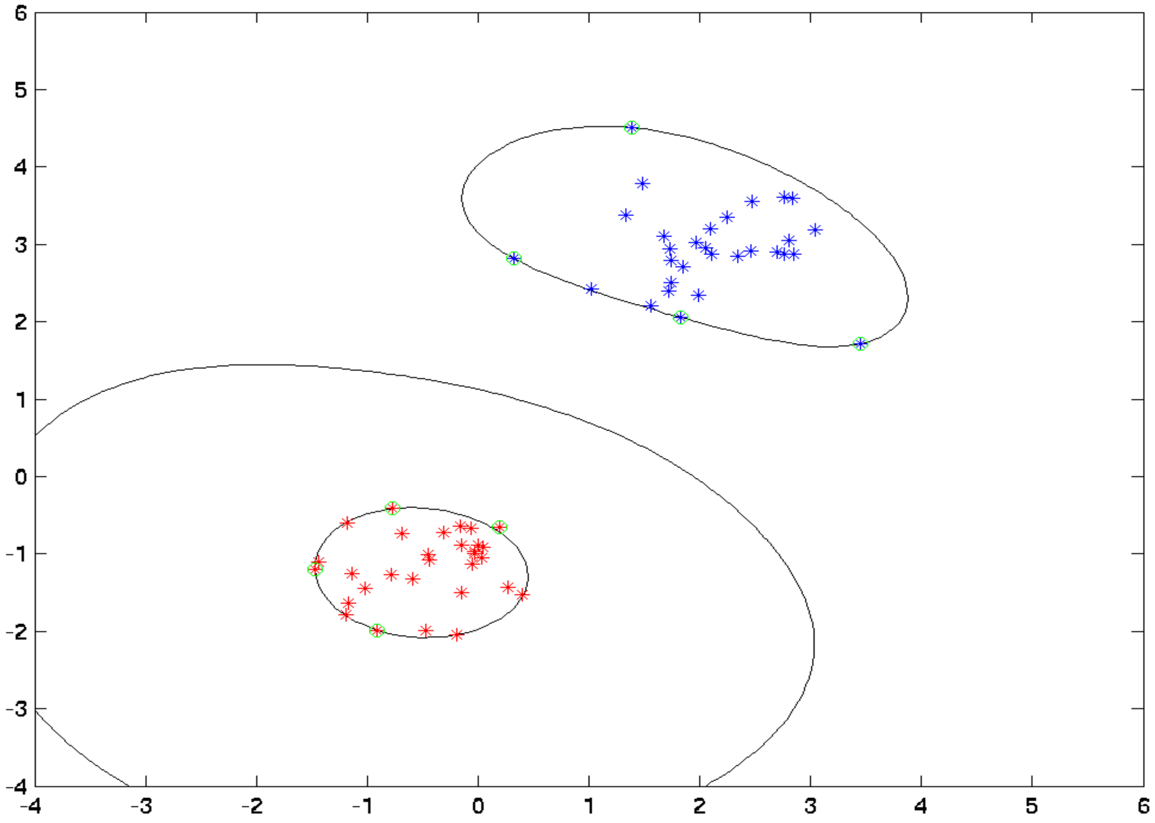
Gaussian RBF with $C = 50$



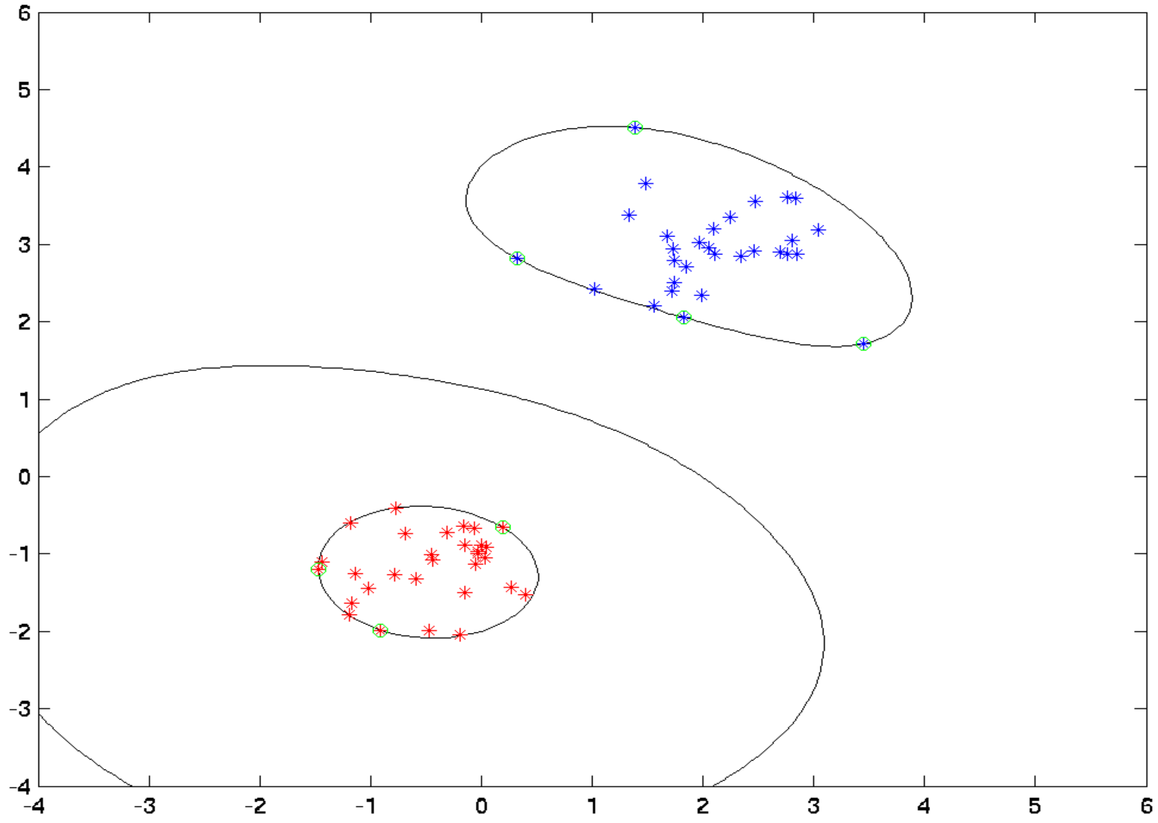
Gaussian RBF with $C = 100$



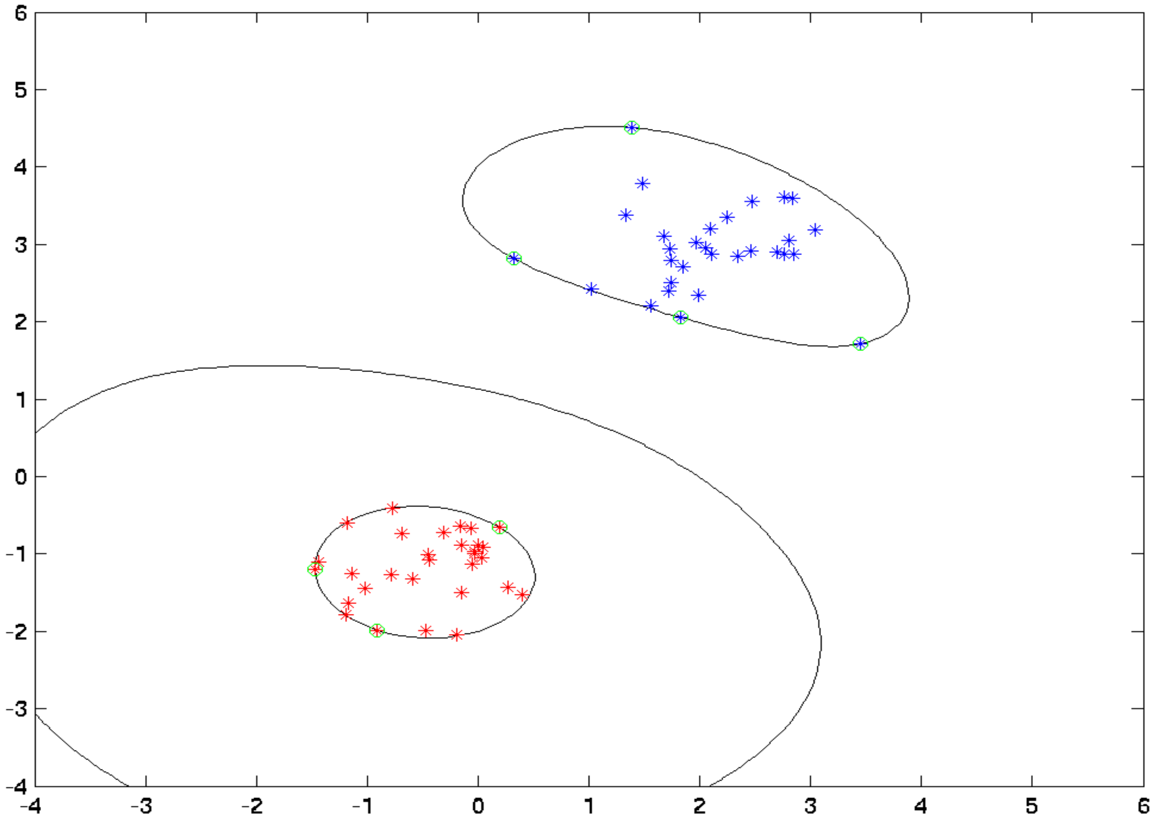
Gaussian RBF with $C = 1$



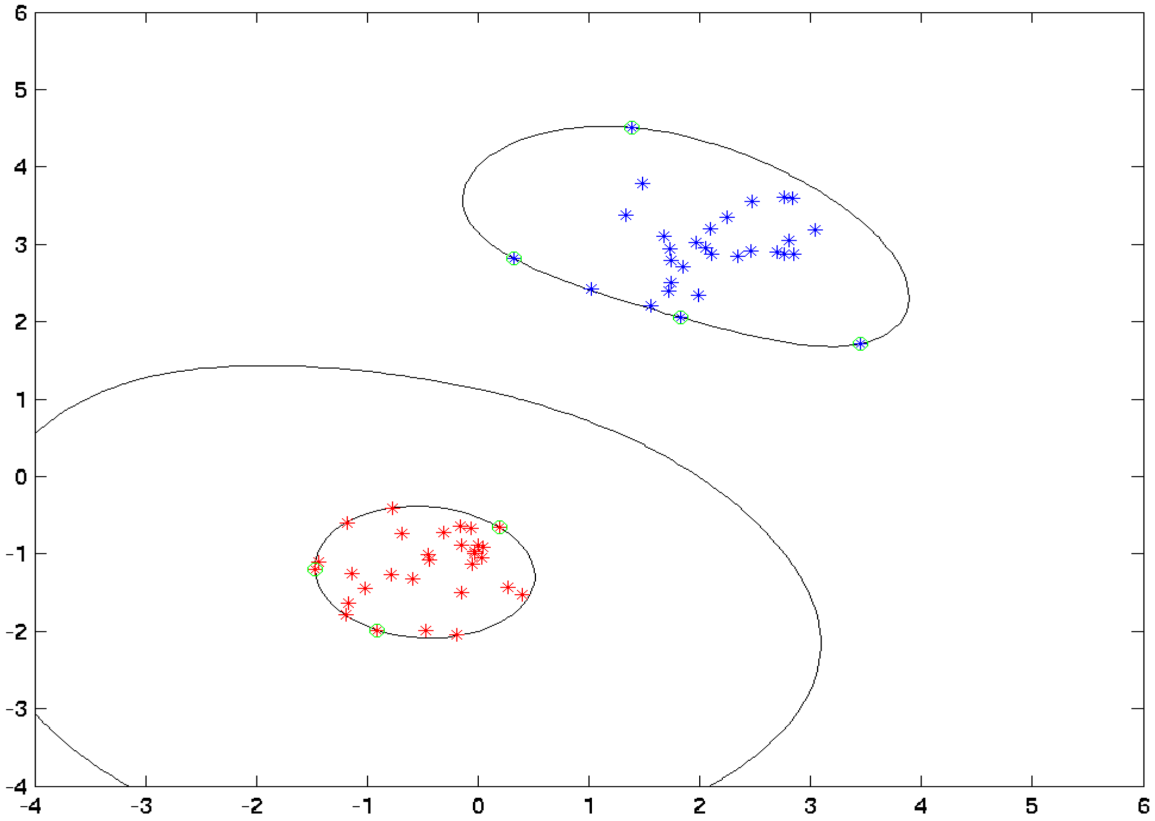
Gaussian RBF with $C = 2$



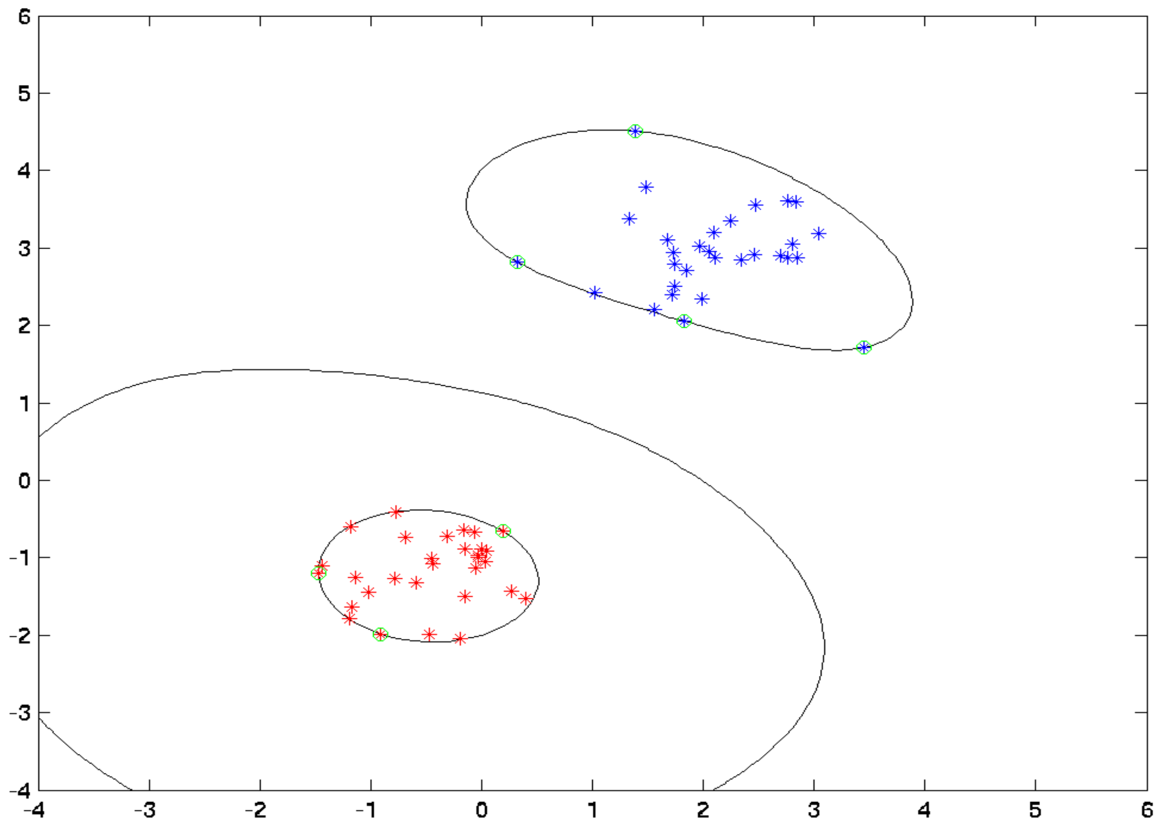
Gaussian RBF with $C = 5$



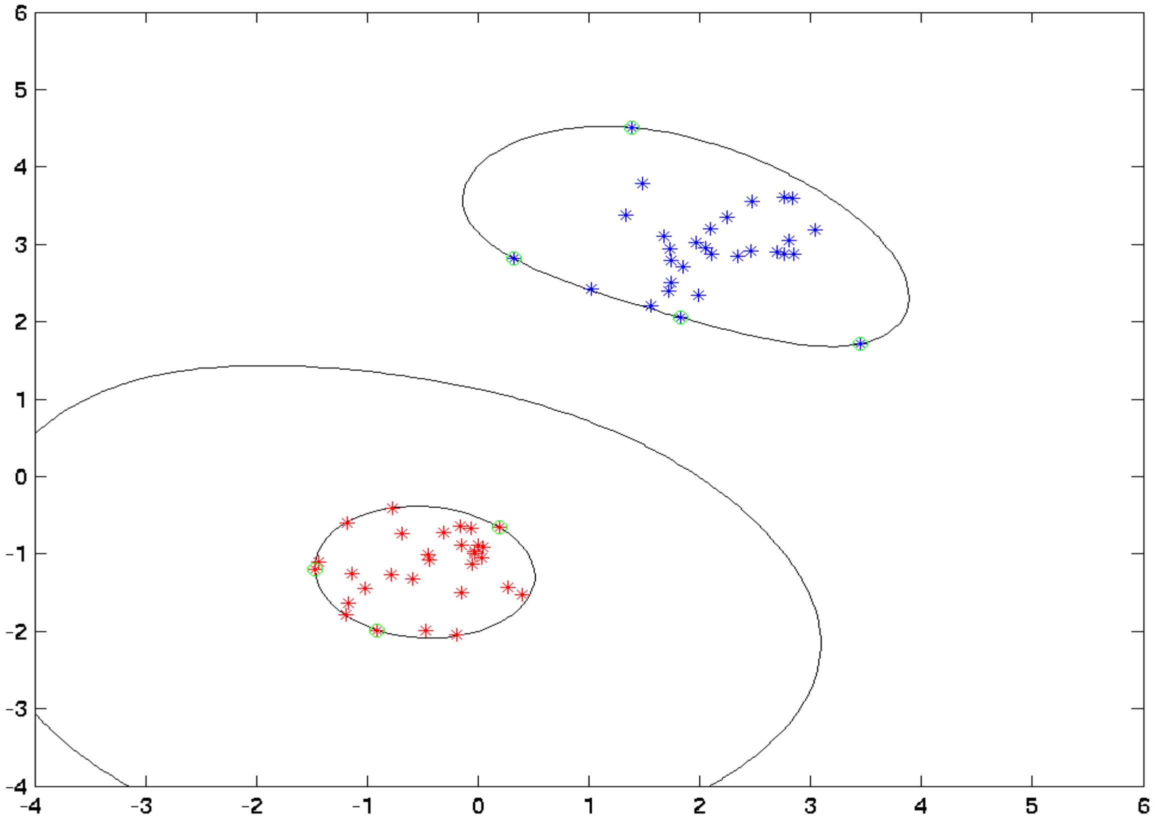
Gaussian RBF with $C = 10$



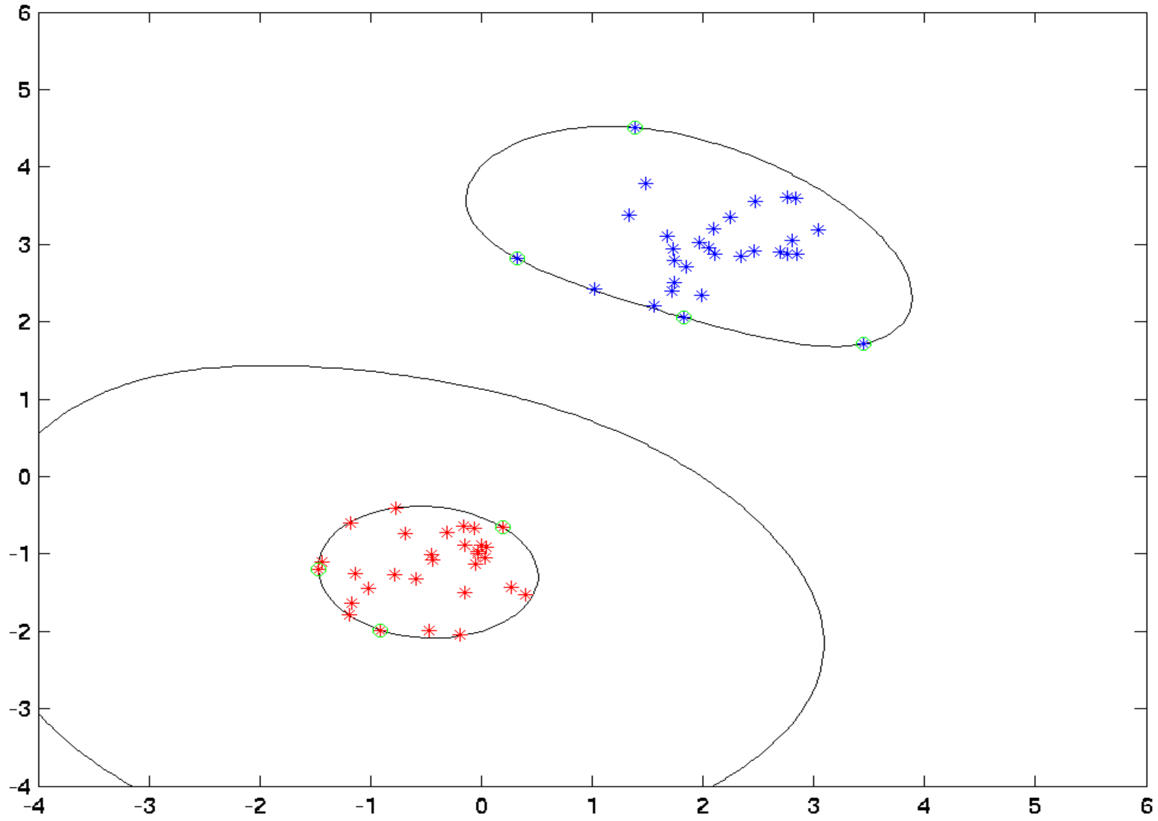
Gaussian RBF with $C = 20$



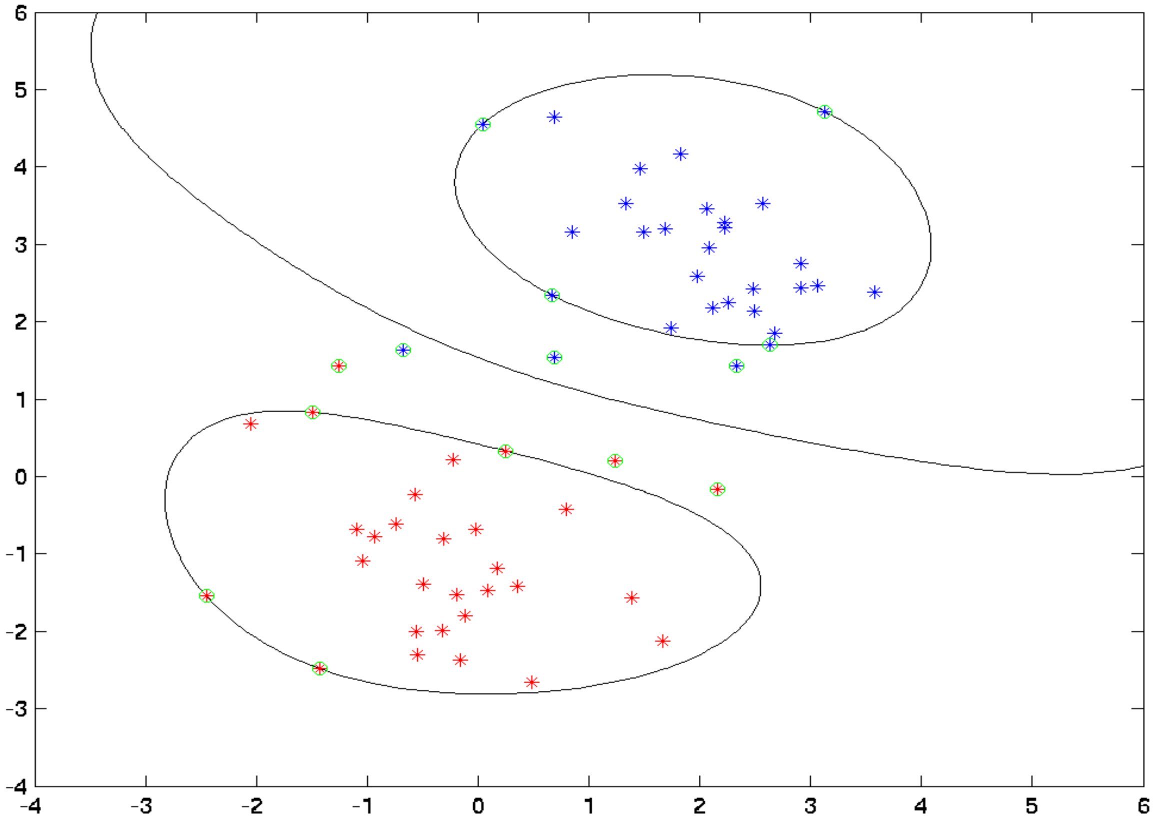
Gaussian RBF with $C = 50$



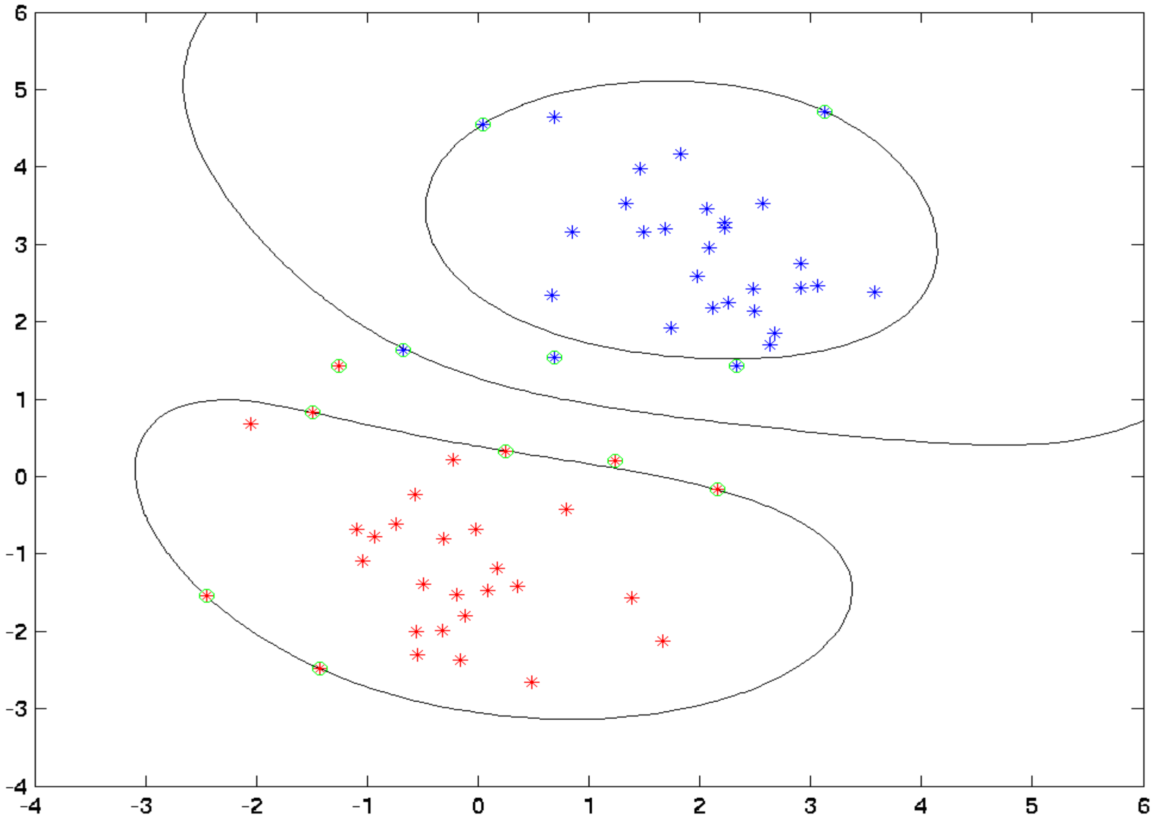
Gaussian RBF with $C = 100$



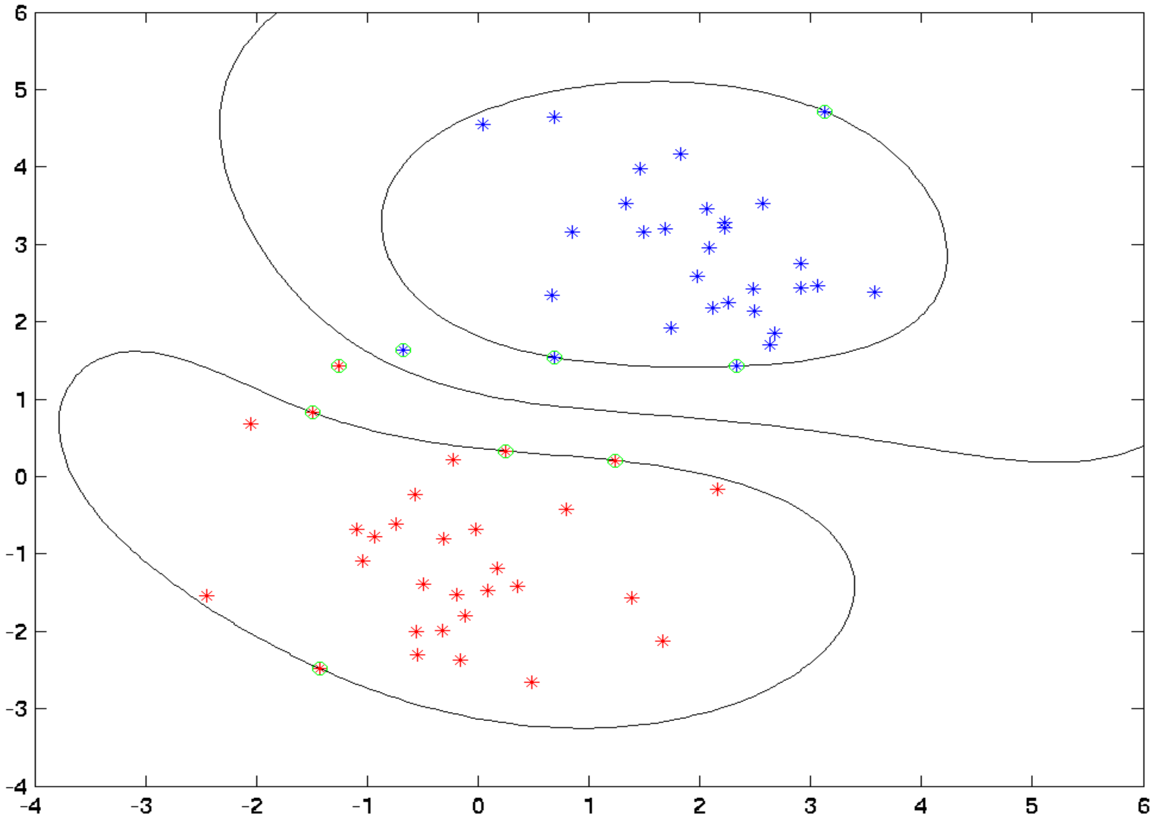
Gaussian RBF with $C = 1$



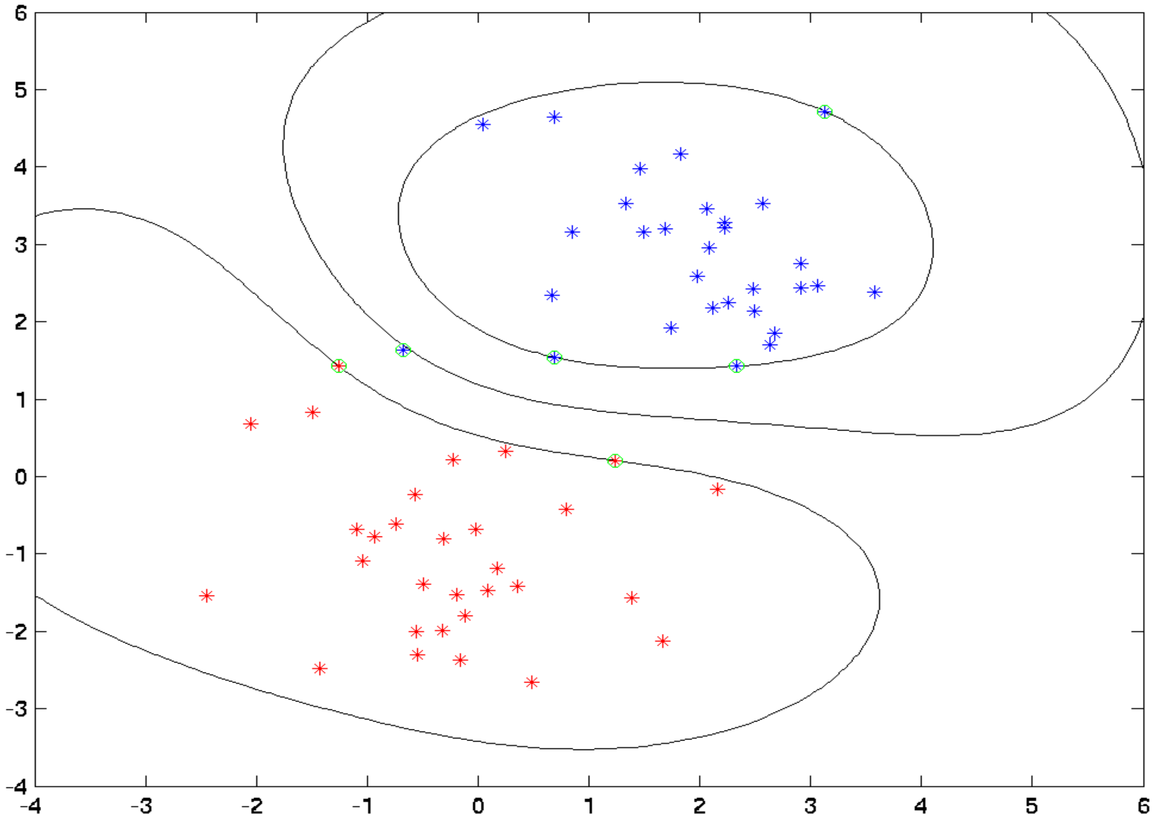
Gaussian RBF with $C = 2$



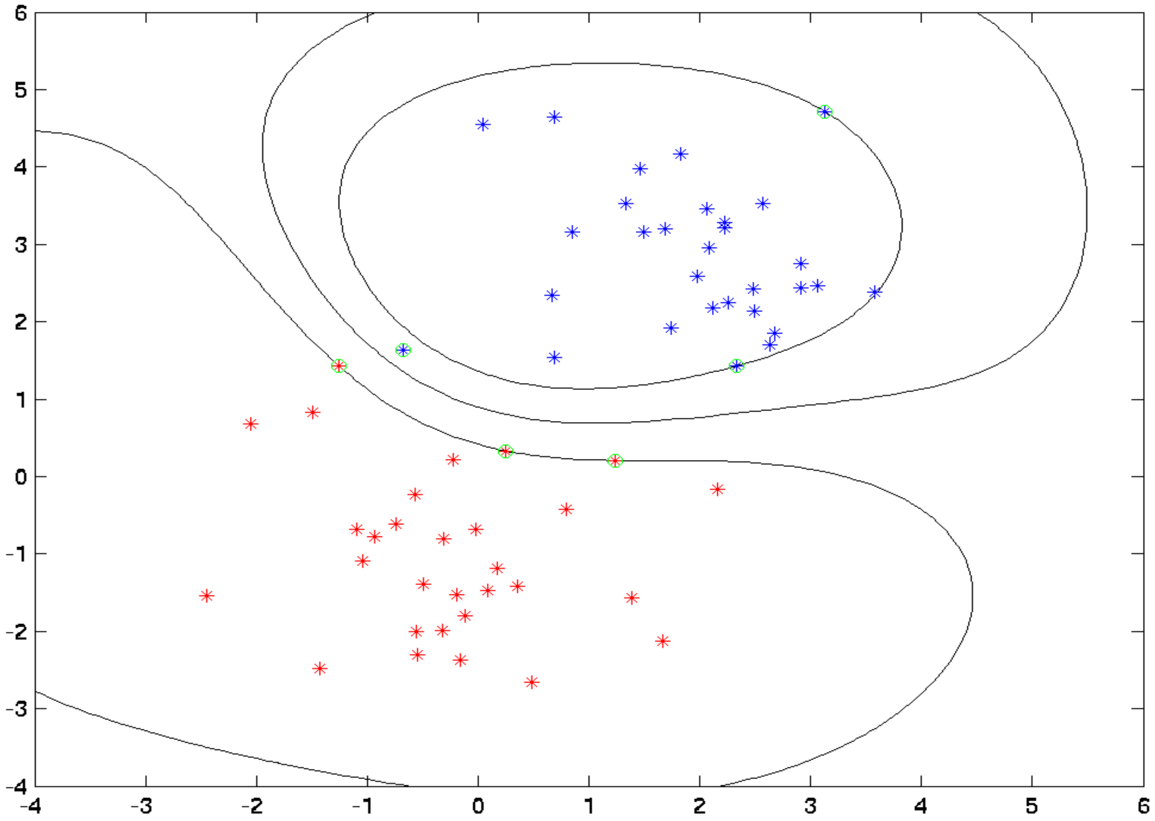
Gaussian RBF with $C = 5$



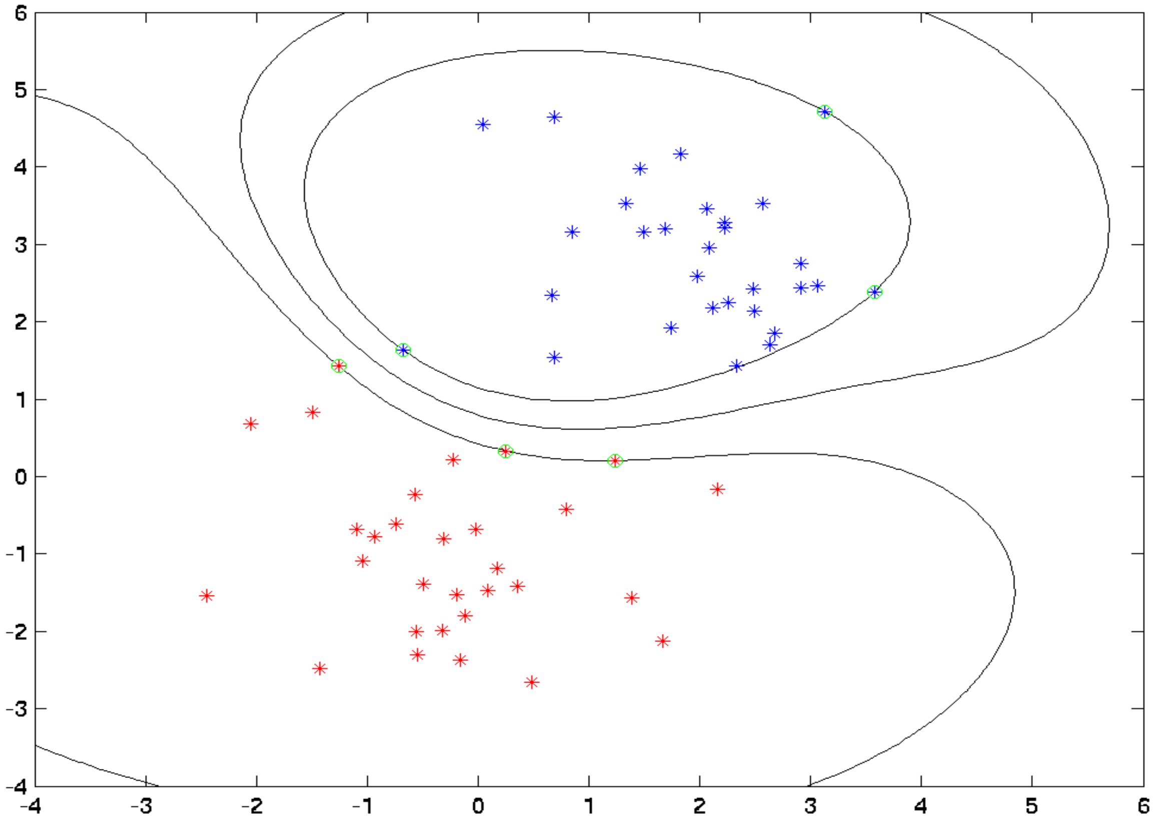
Gaussian RBF with $C = 10$



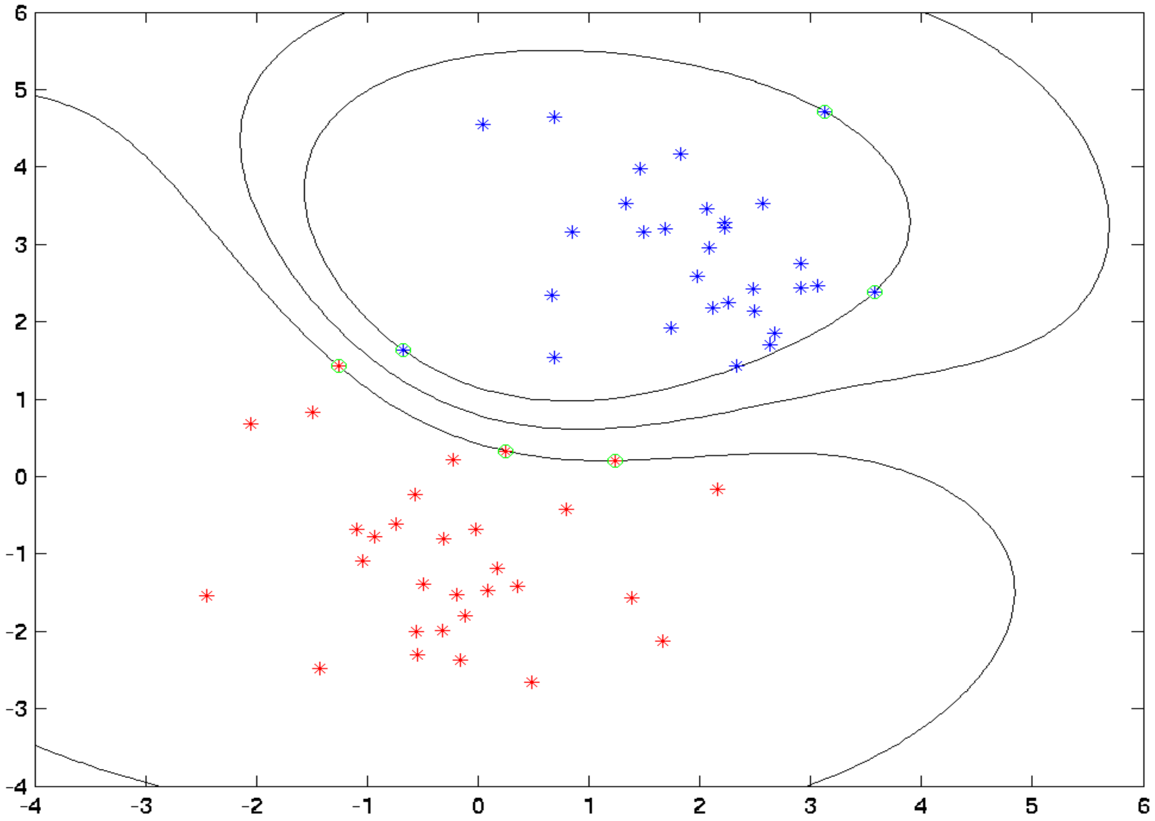
Gaussian RBF with $C = 20$



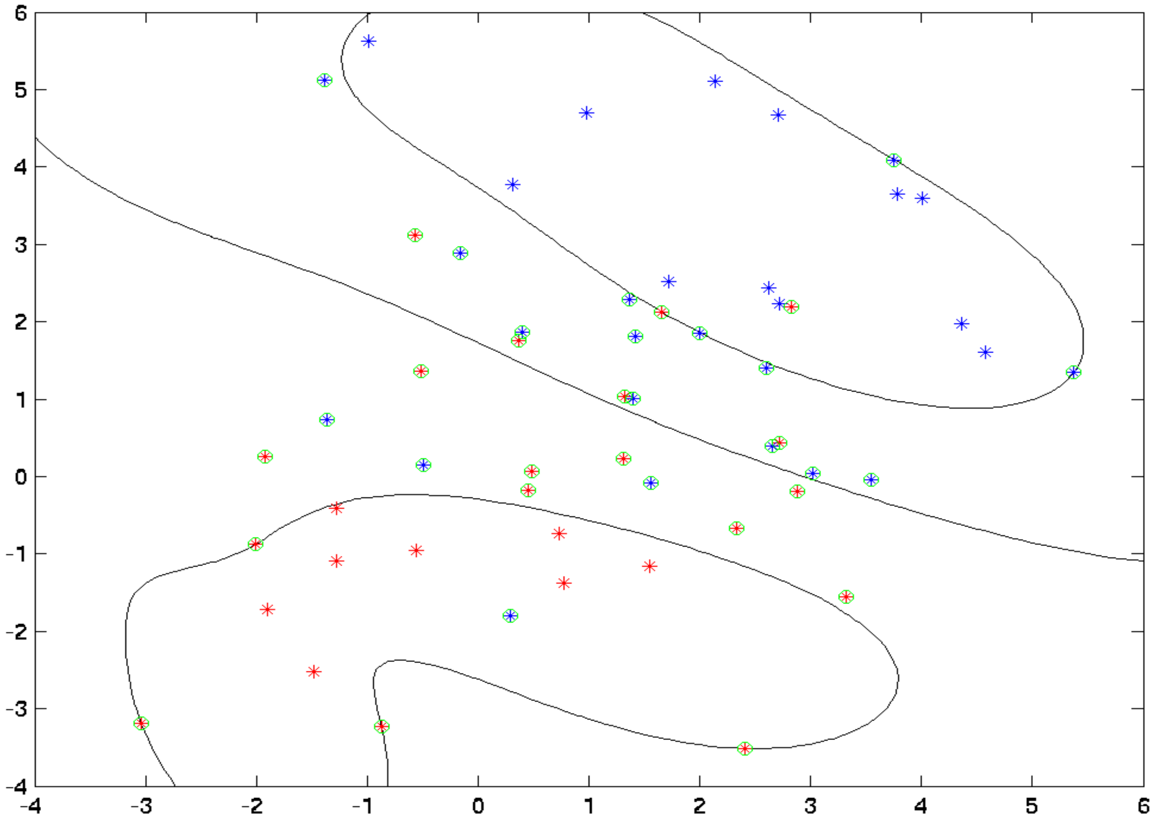
Gaussian RBF with $C = 50$



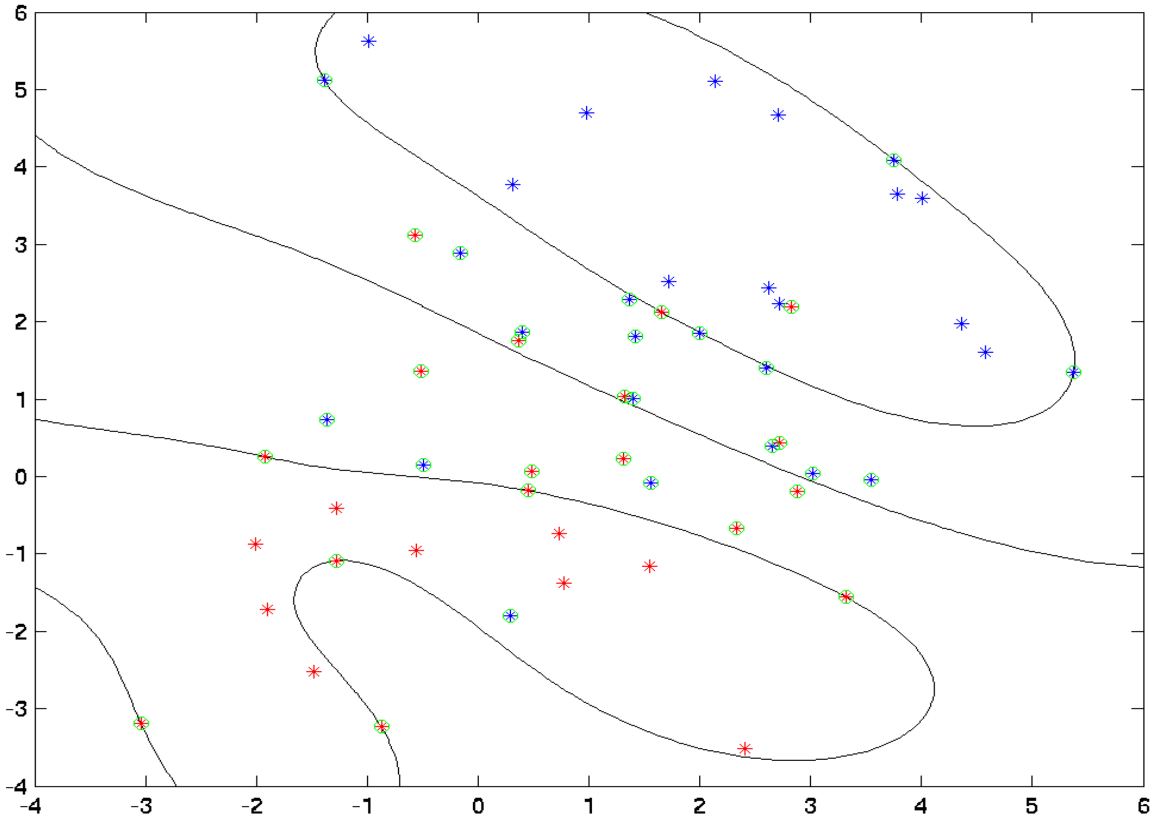
Gaussian RBF with $C = 100$



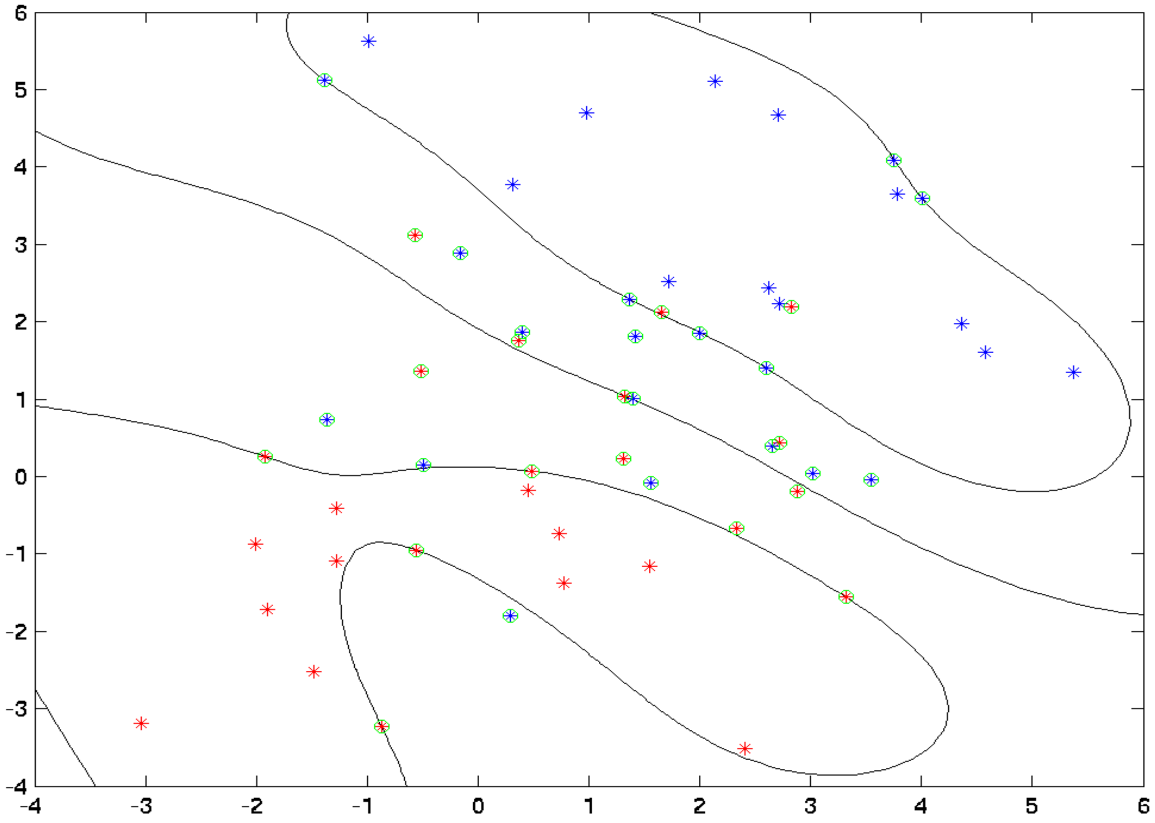
Gaussian RBF with $C = 1$



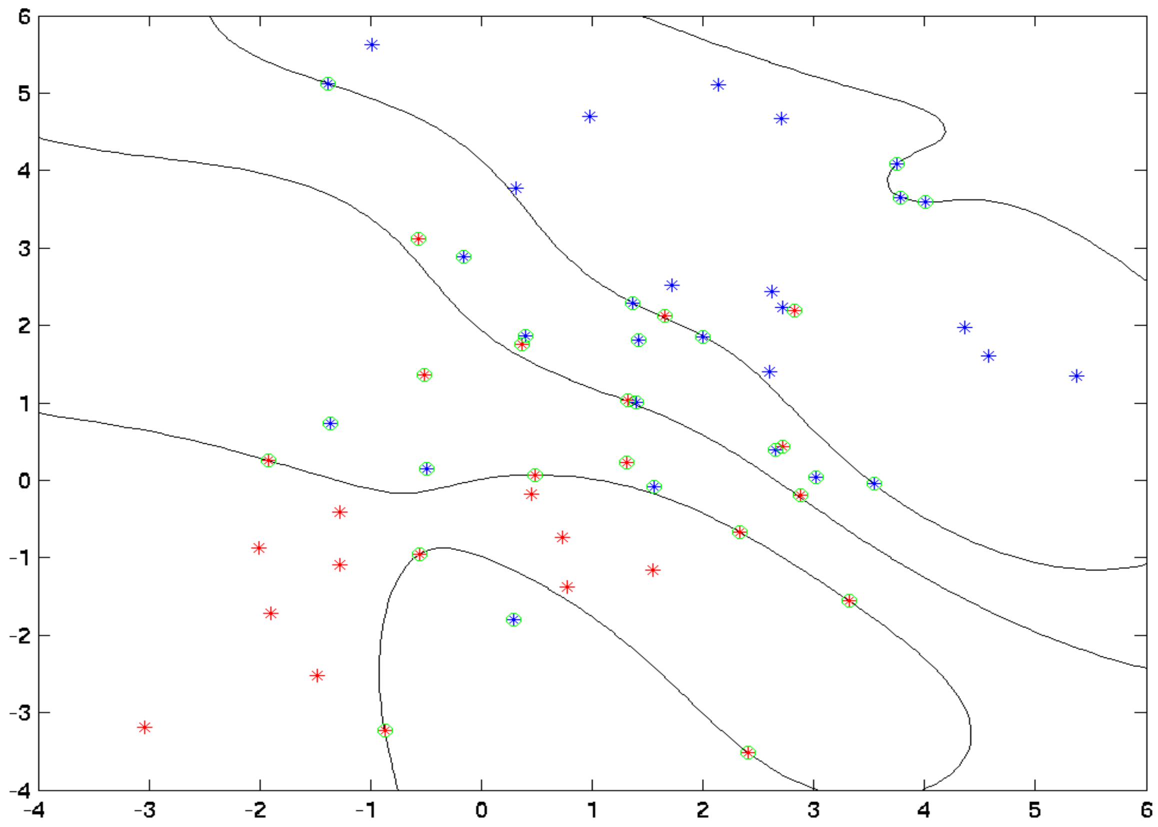
Gaussian RBF with $C = 2$



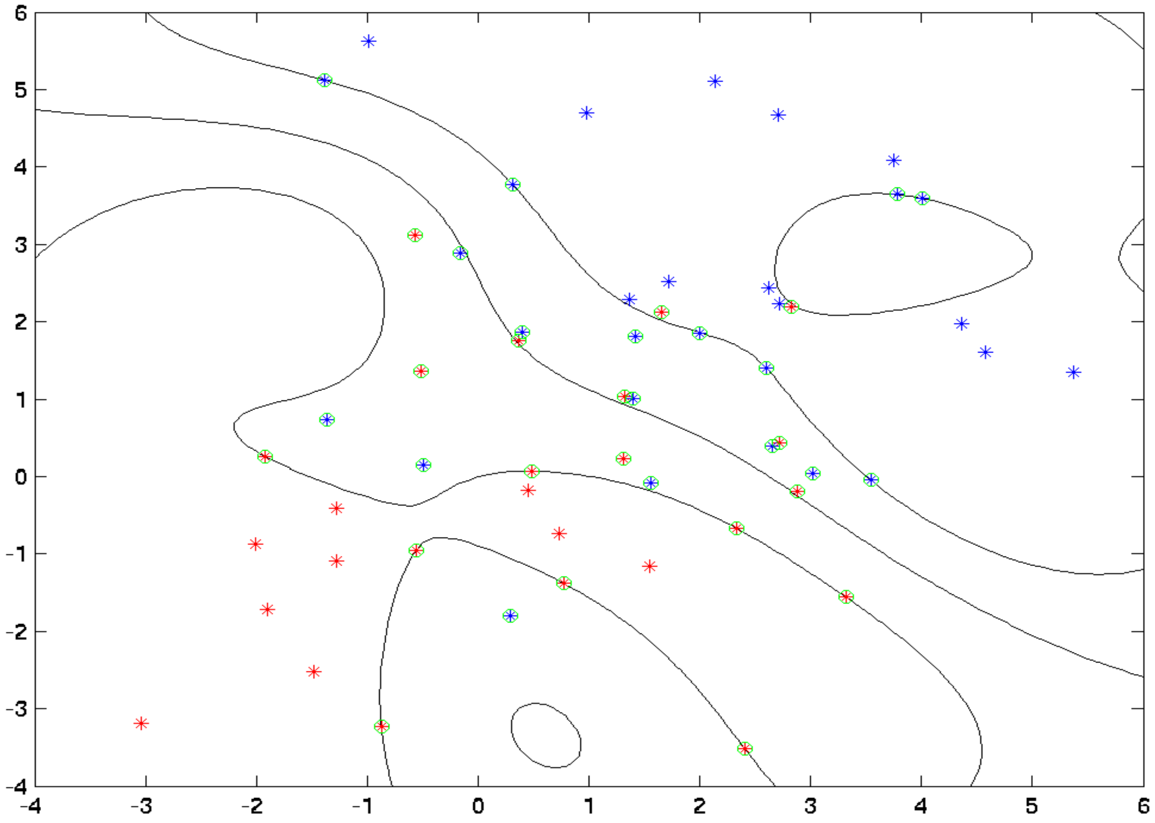
Gaussian RBF with $C = 5$



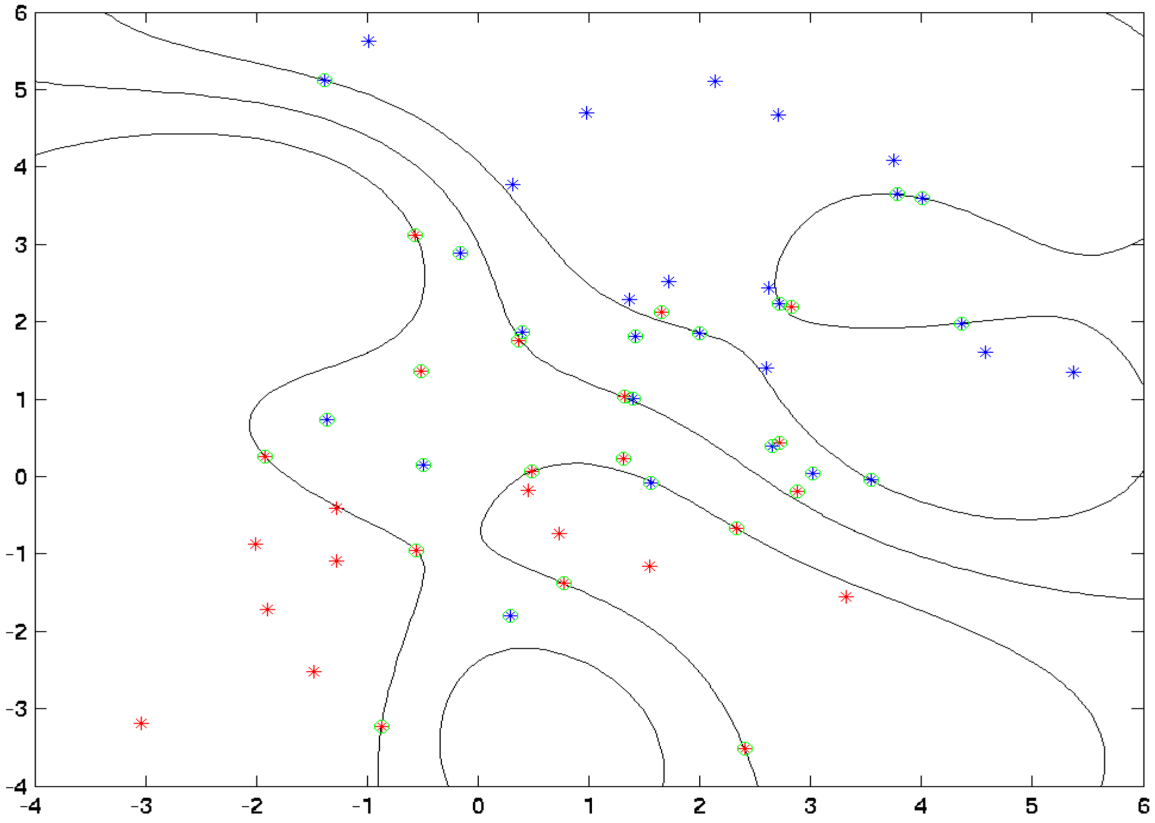
Gaussian RBF with $C = 10$



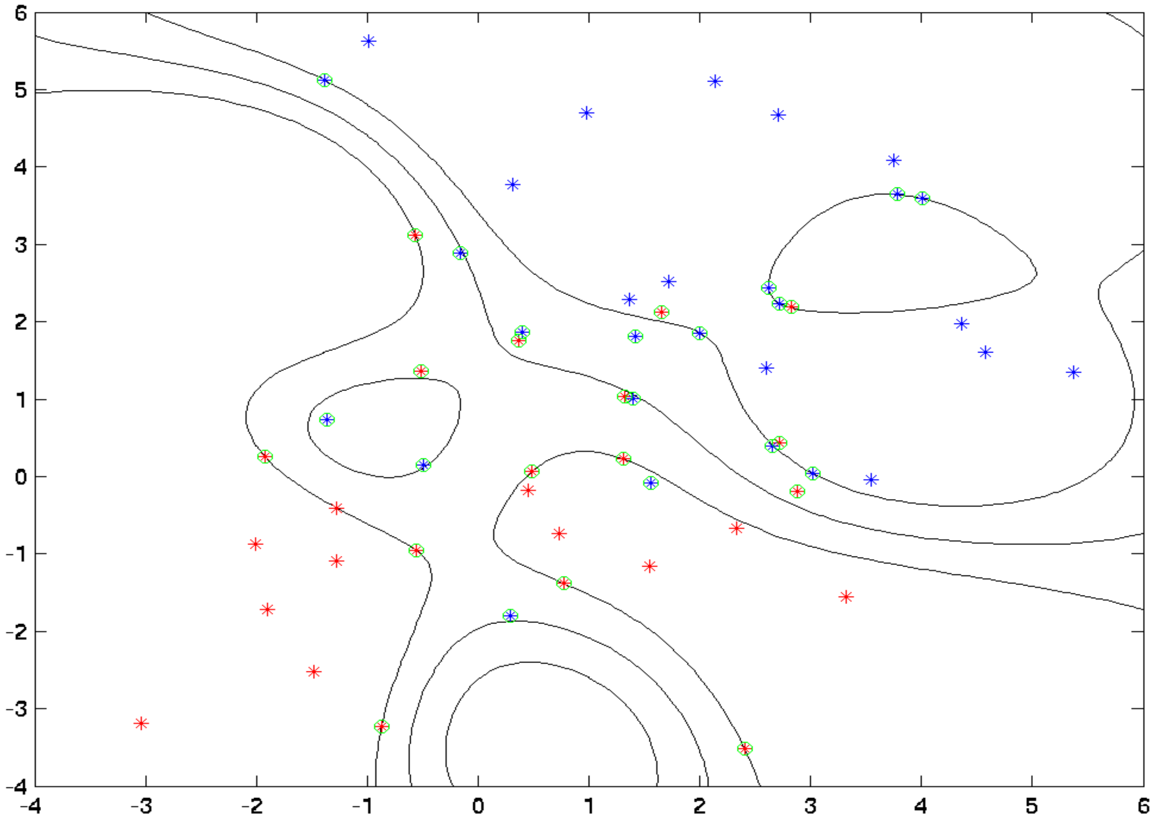
Gaussian RBF with $C = 20$



Gaussian RBF with $C = 50$



Gaussian RBF with $C = 100$



Insights

Changing C

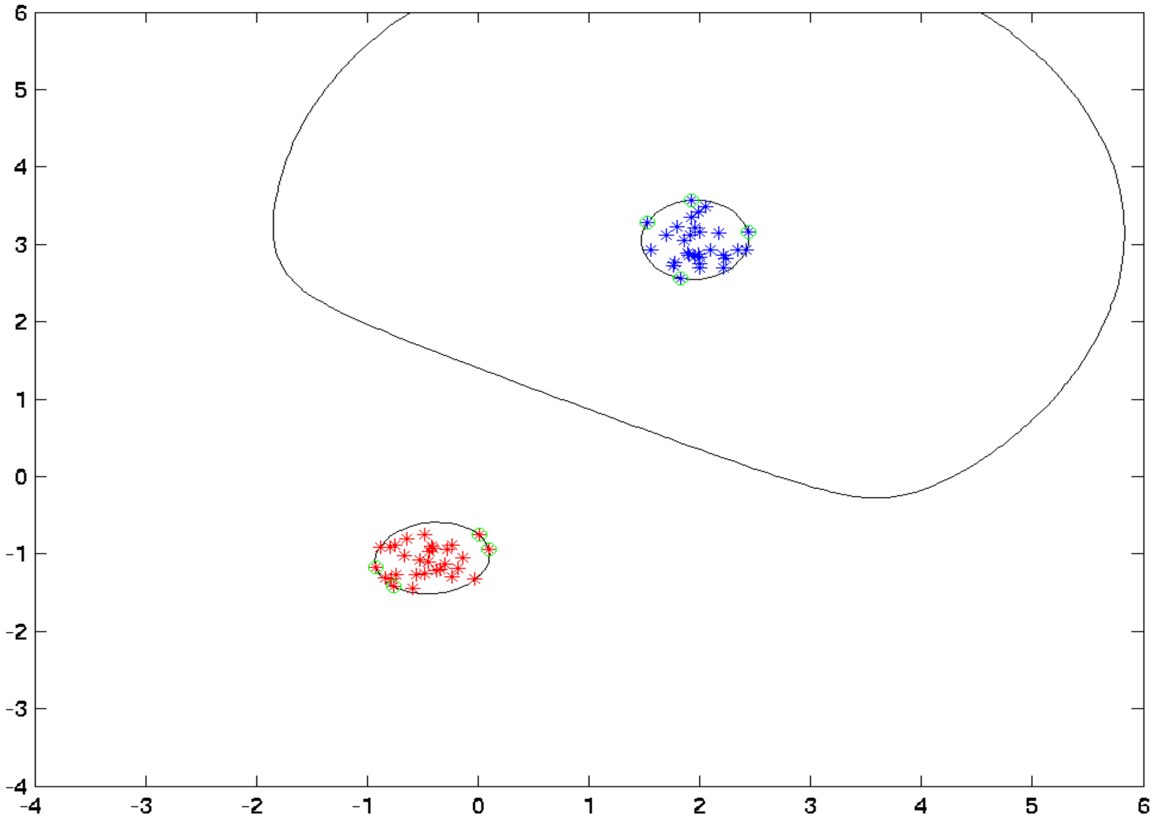
- For clean data C doesn't matter much.
- For noisy data, large C leads to more complicated margin (SVM tries to do a good job at separating, even though it isn't possible)
- Overfitting for large C

Noisy data

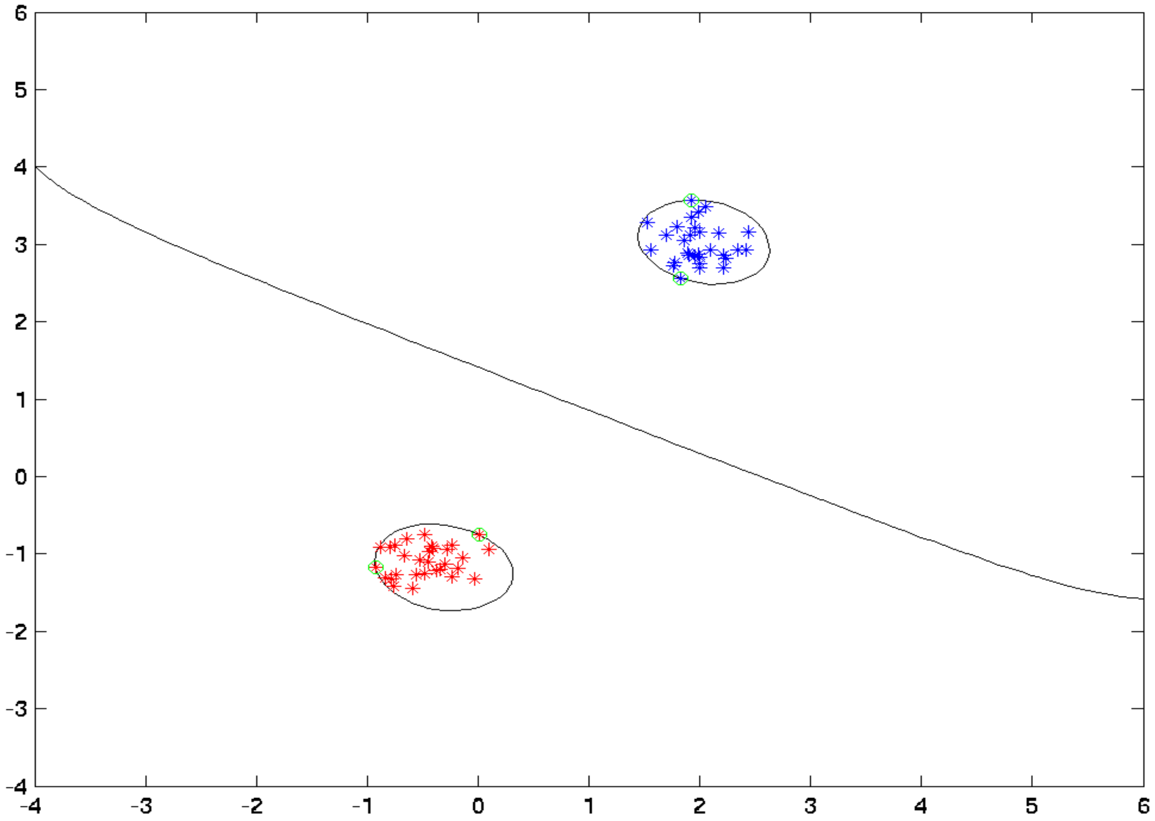
- Clean data has few support vectors
- Noisy data leads to data in the margins
- More support vectors for noisy data

NATIONAL ICT AUSTRALIA

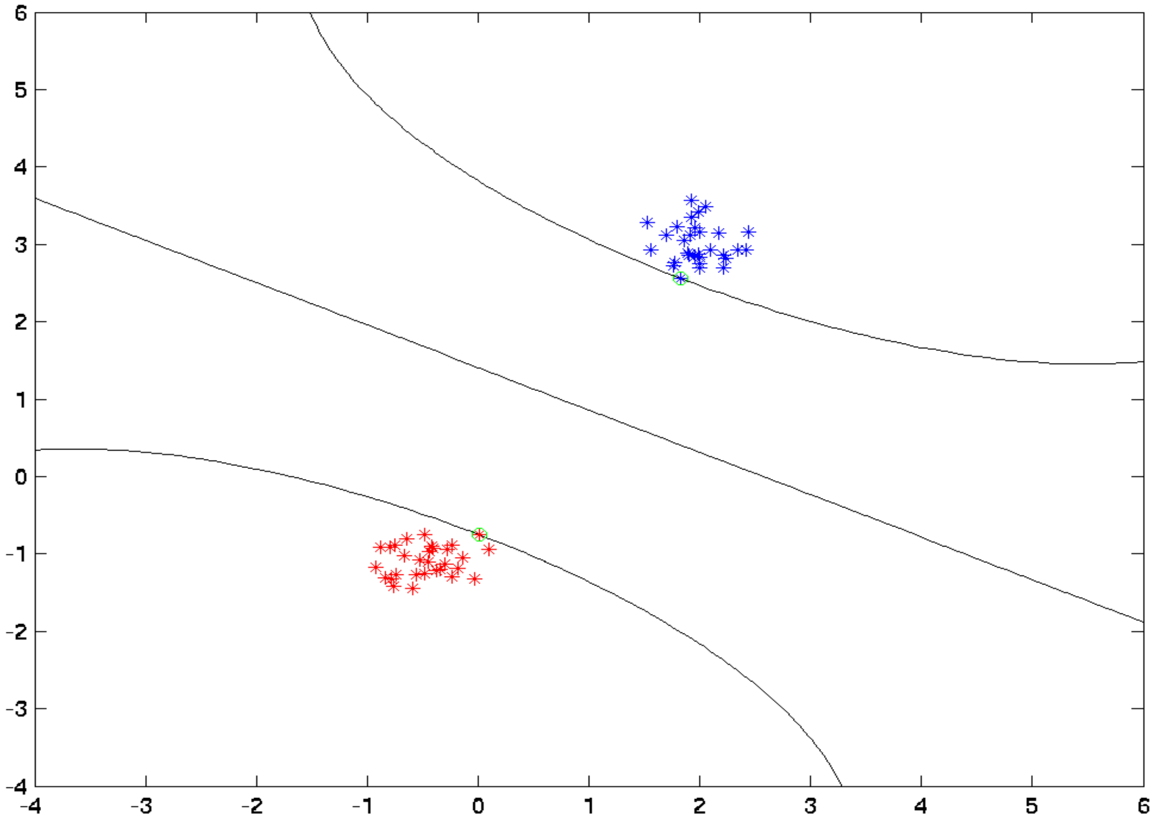
Gaussian RBF with $\sigma = 1$



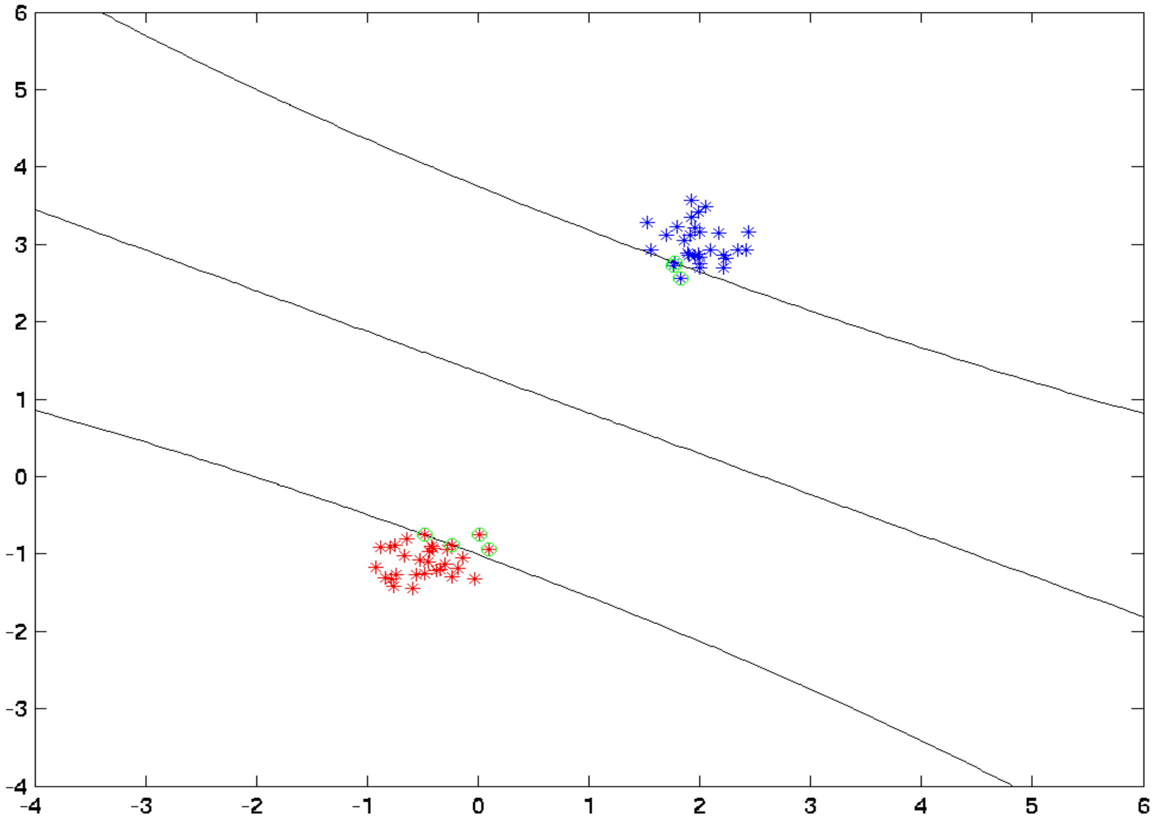
Gaussian RBF with $\sigma = 2$



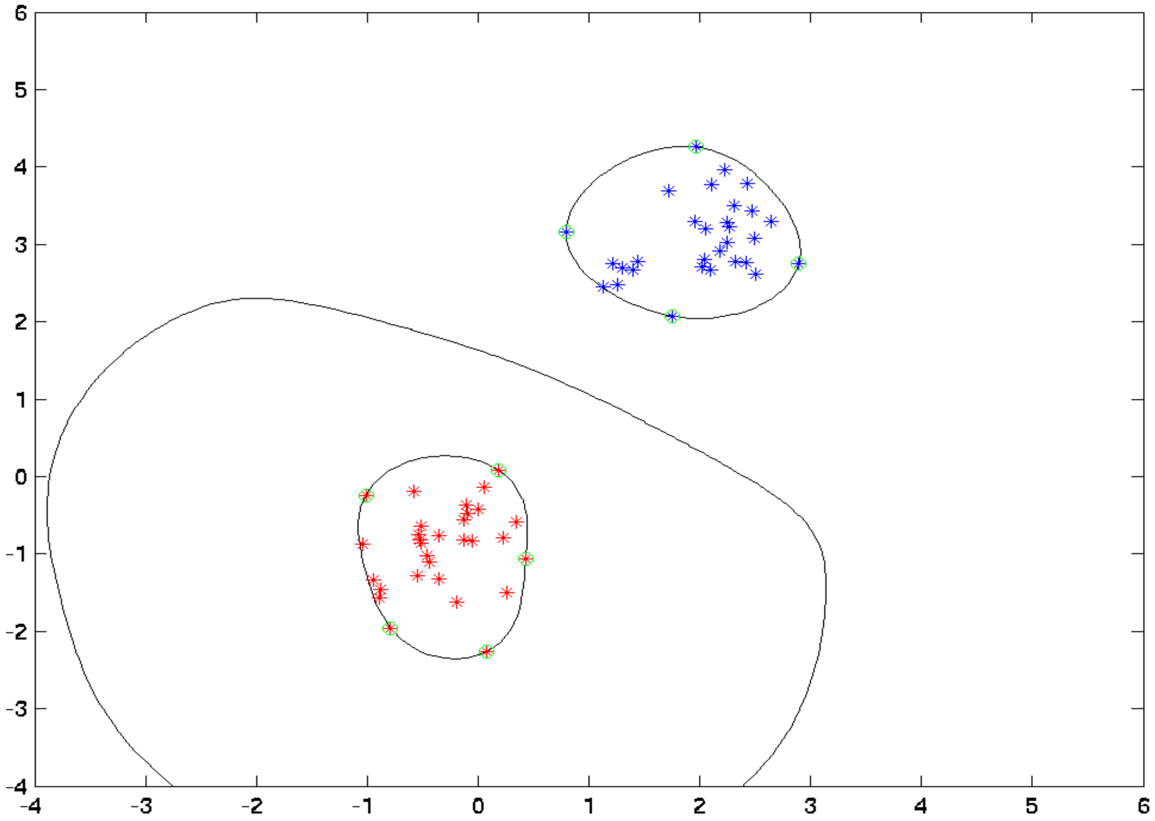
Gaussian RBF with $\sigma = 5$



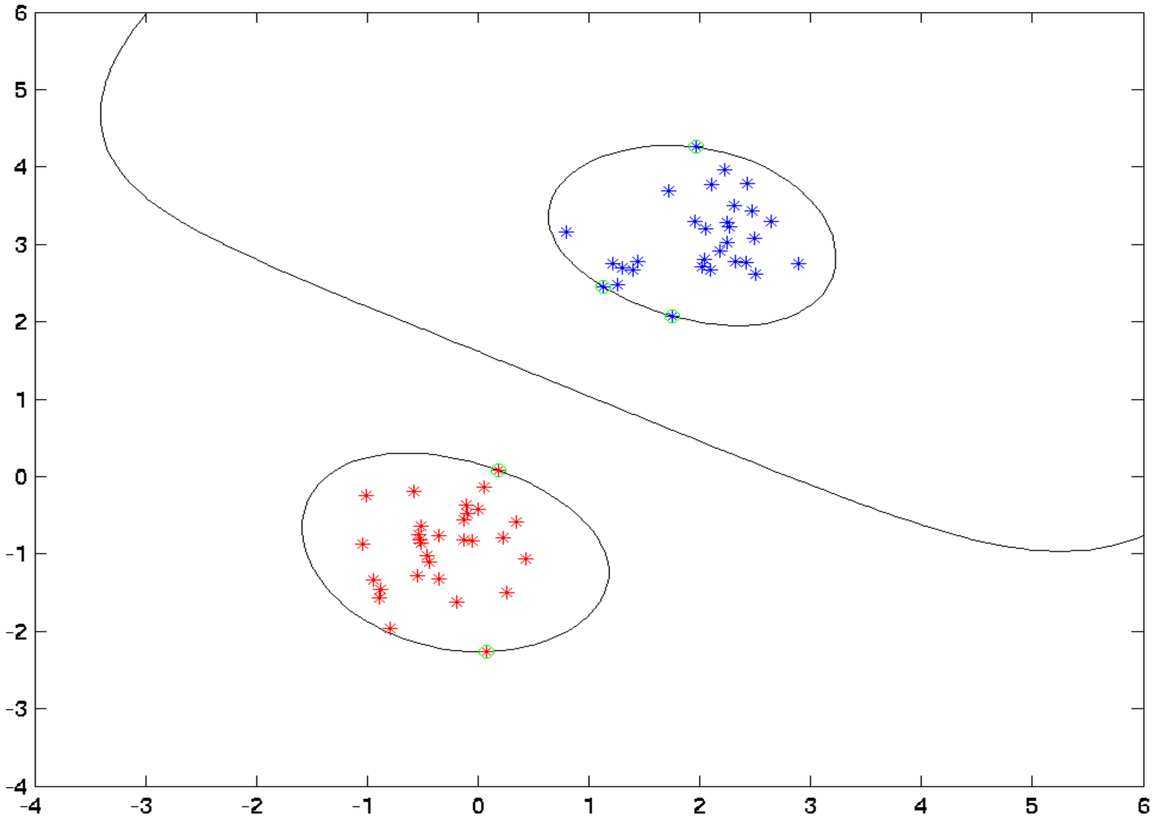
Gaussian RBF with $\sigma = 10$



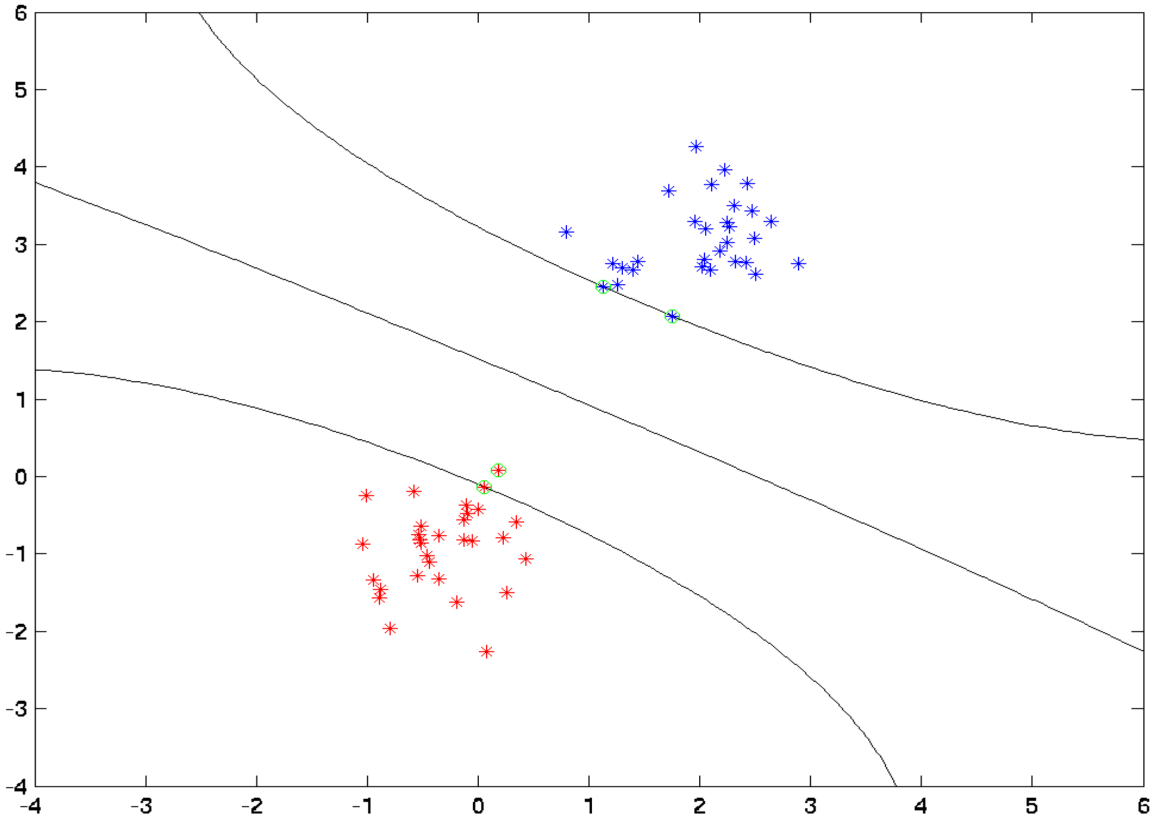
Gaussian RBF with $\sigma = 1$



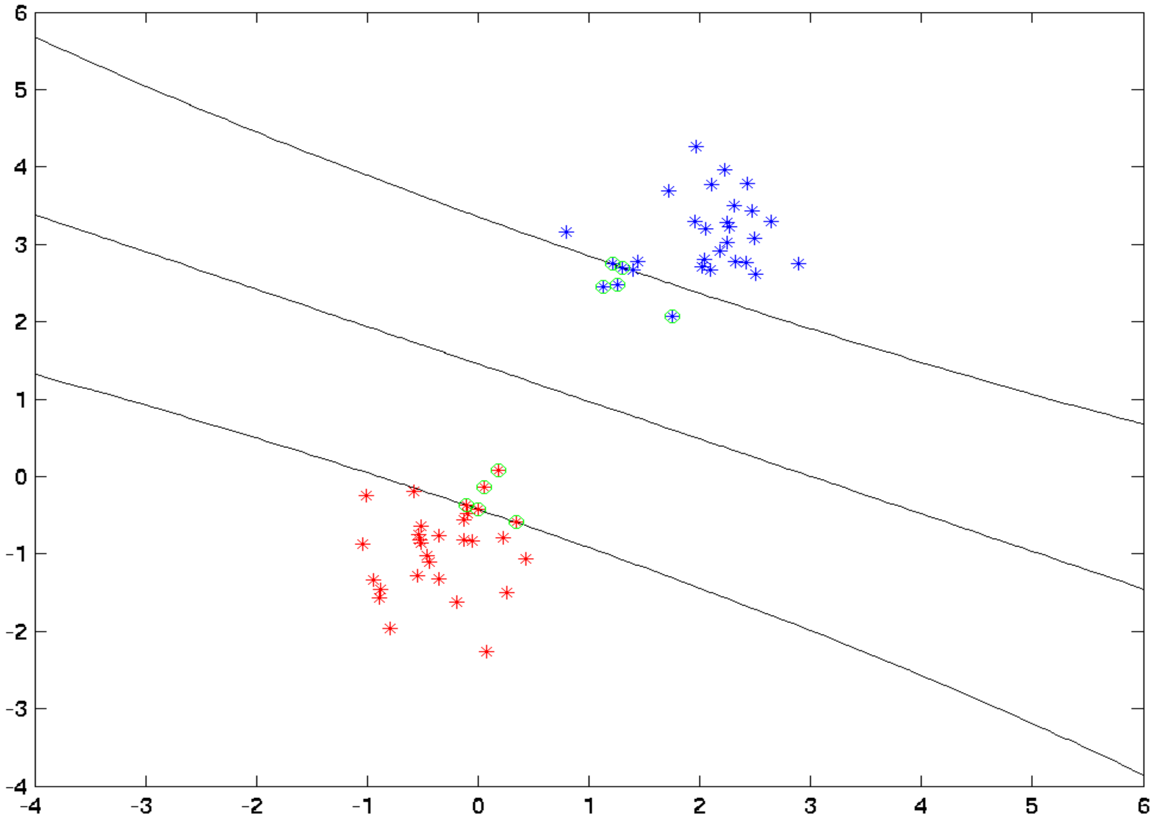
Gaussian RBF with $\sigma = 2$



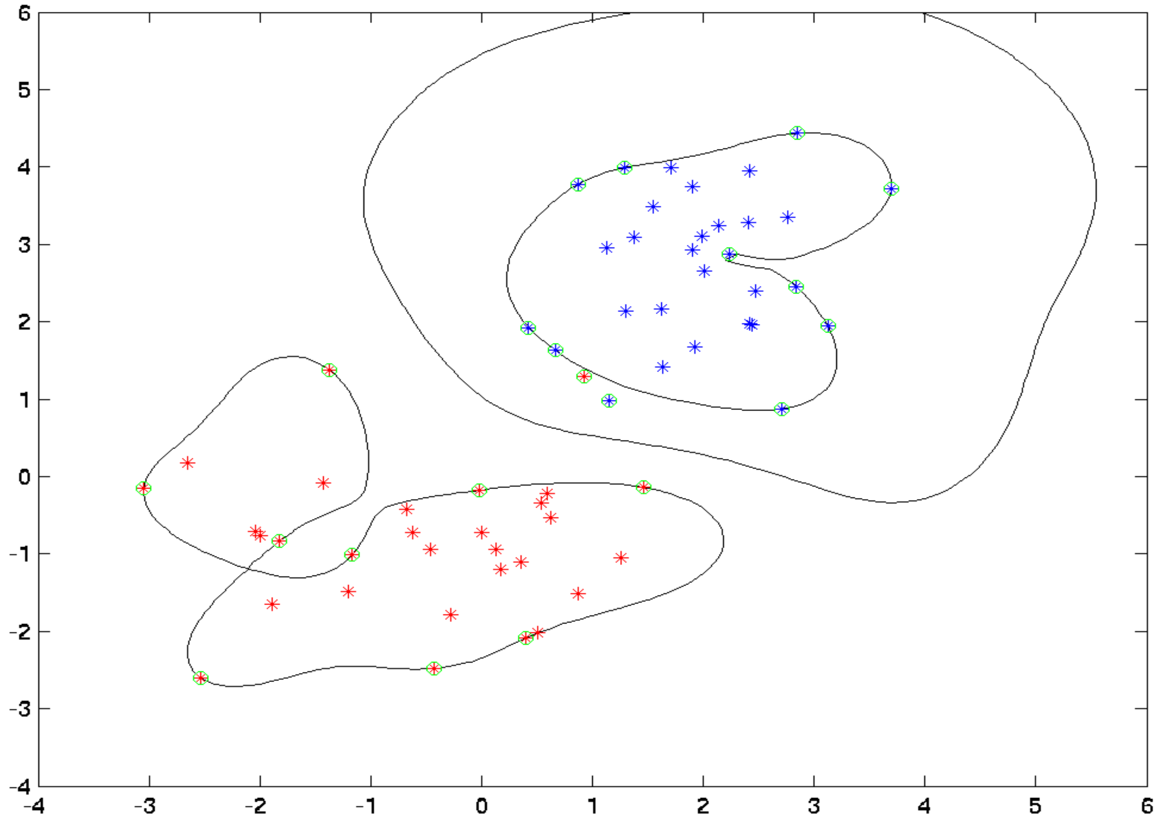
Gaussian RBF with $\sigma = 5$



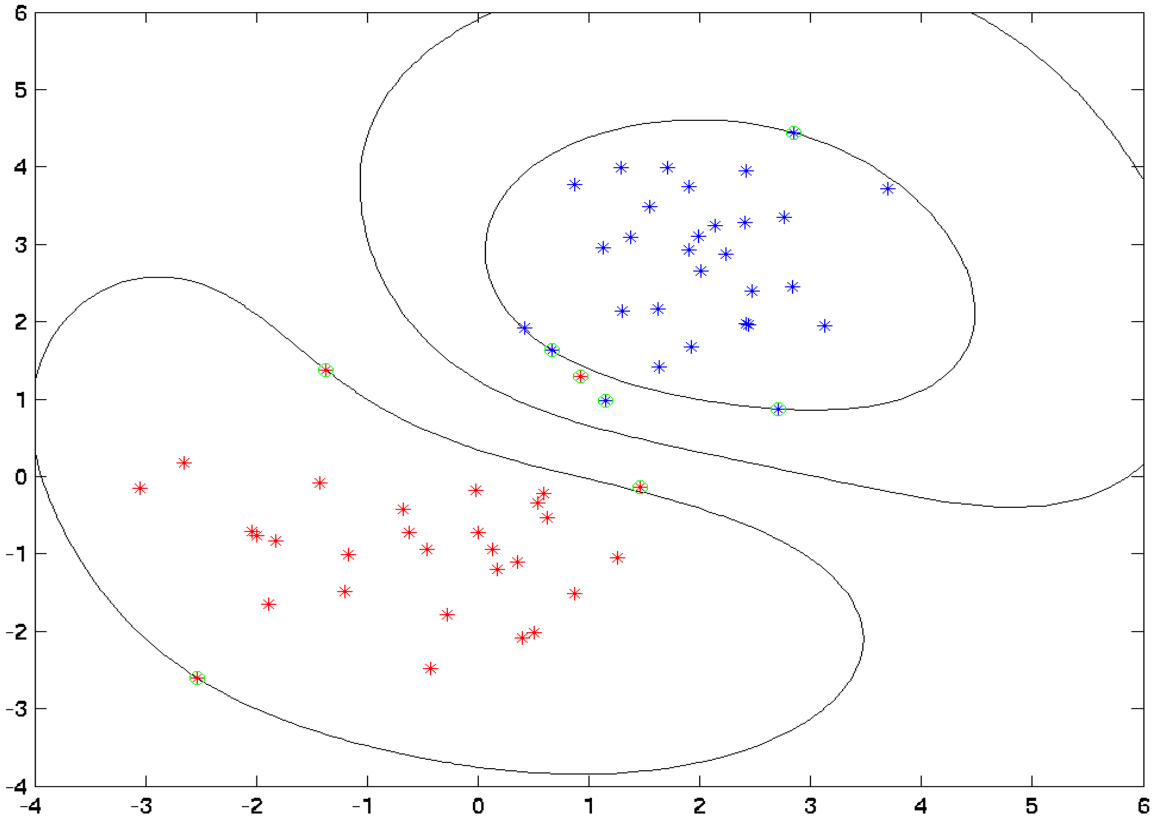
Gaussian RBF with $\sigma = 10$



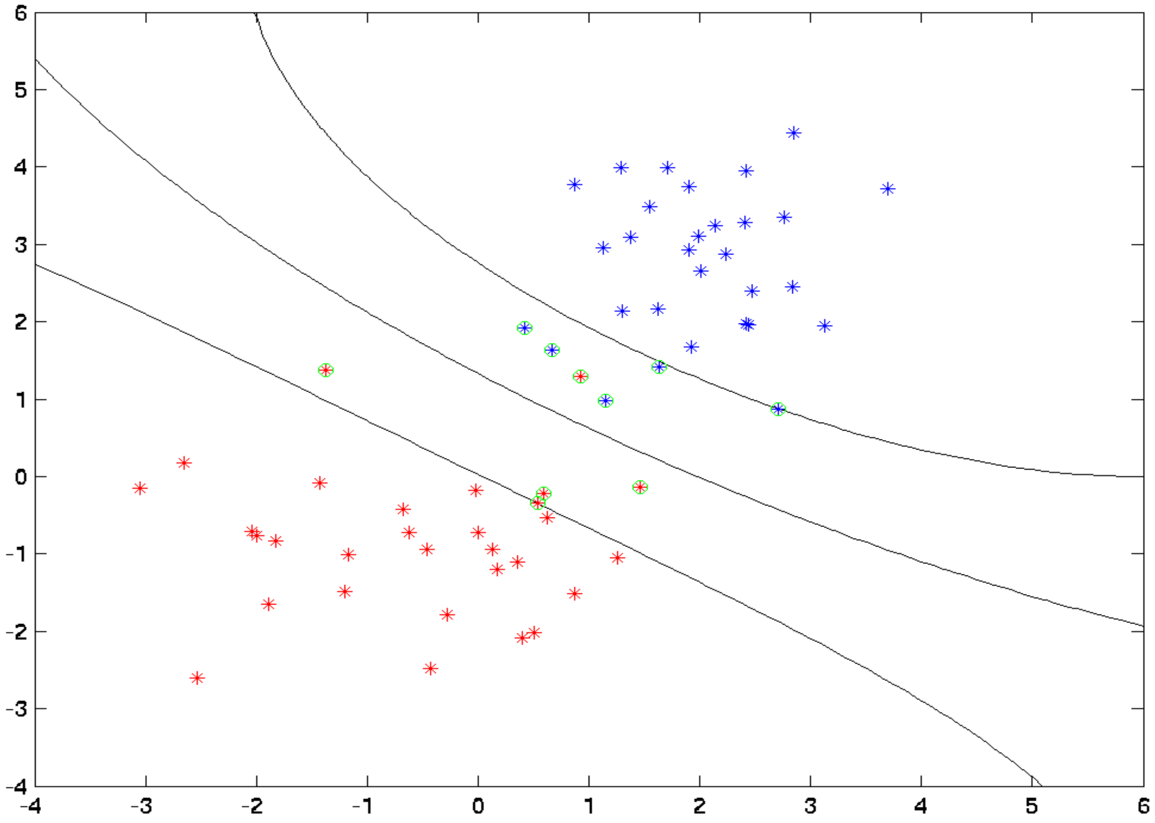
Gaussian RBF with $\sigma = 1$



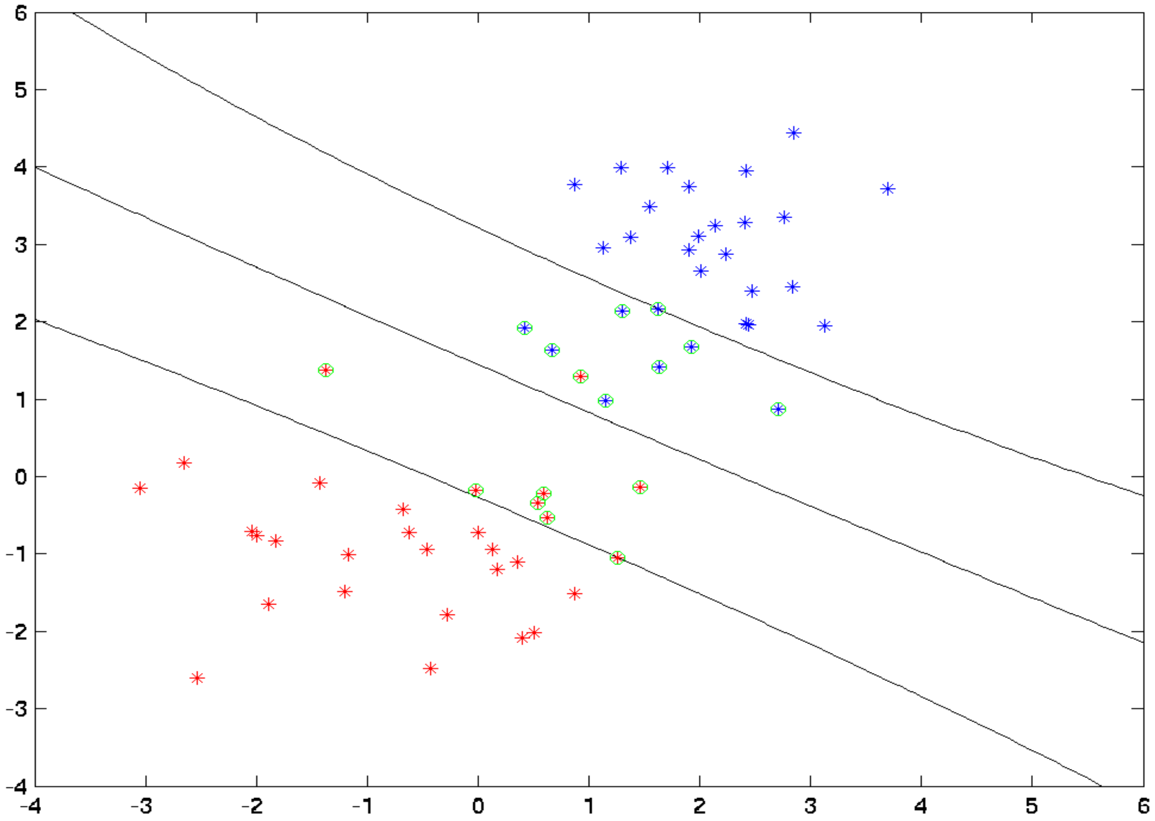
Gaussian RBF with $\sigma = 2$



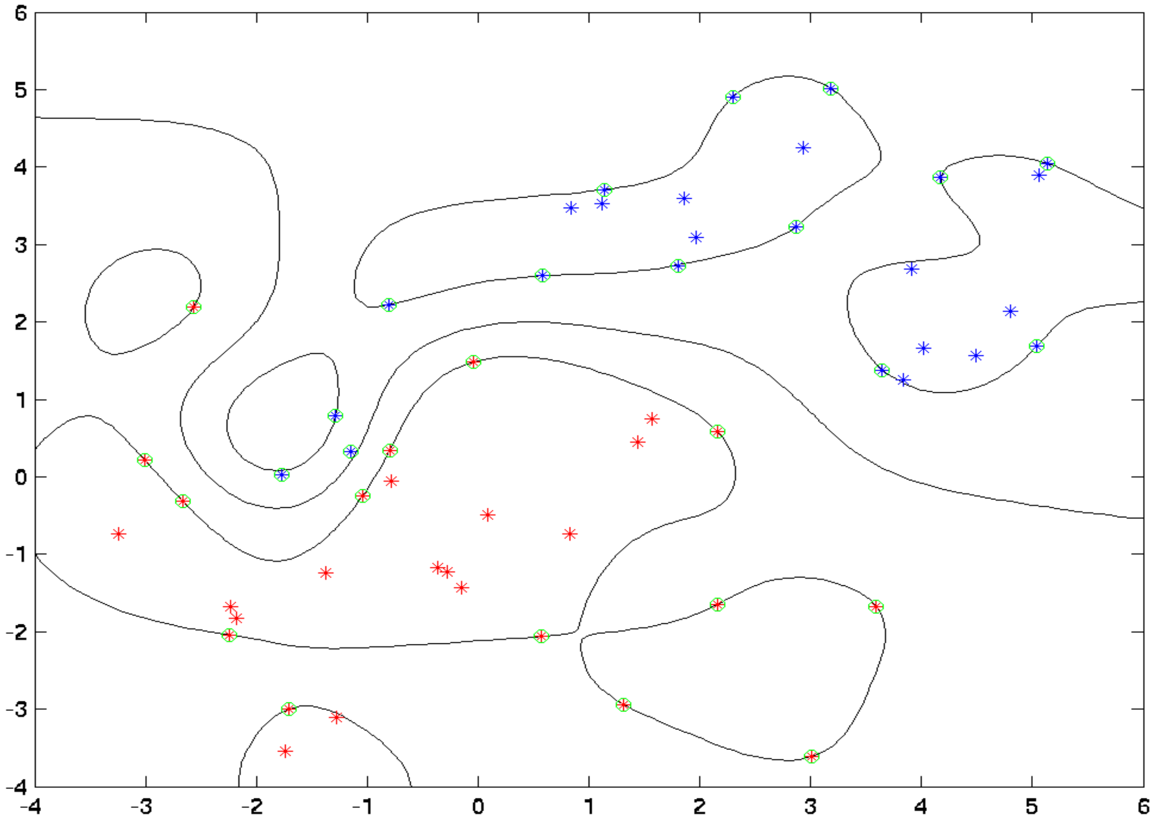
Gaussian RBF with $\sigma = 5$



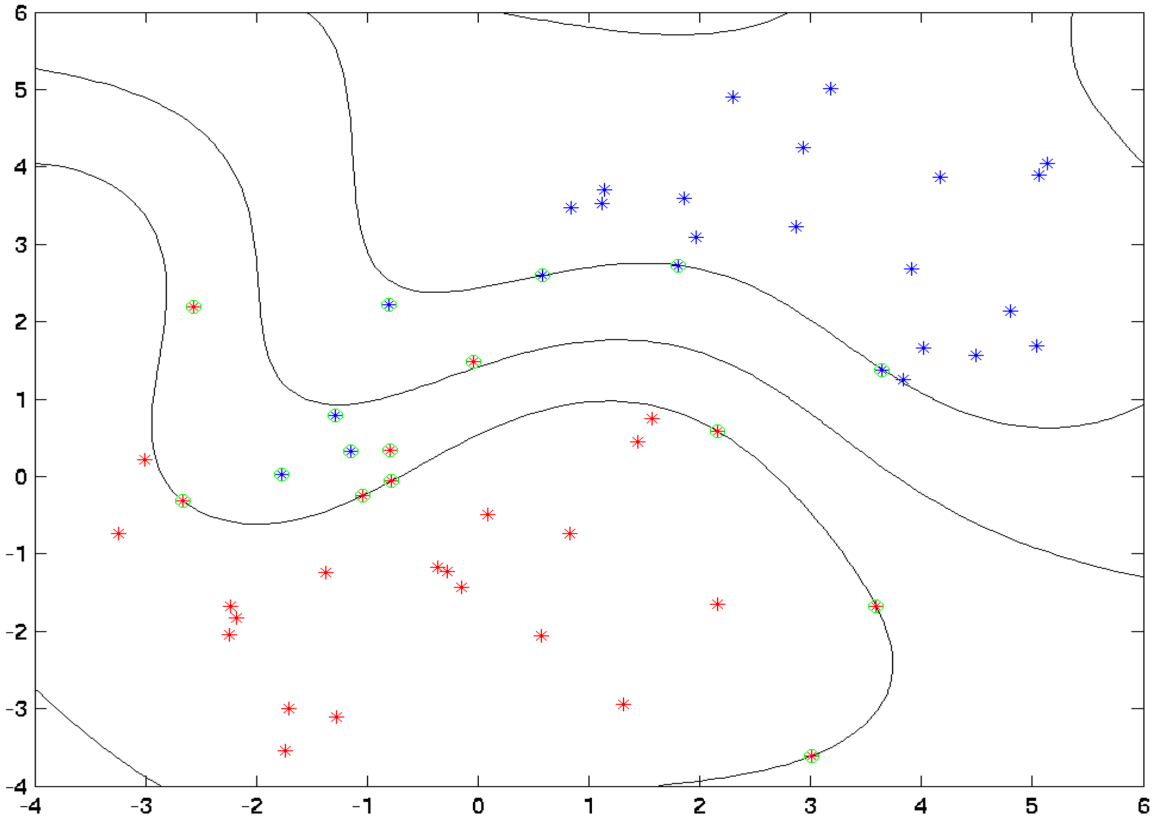
Gaussian RBF with $\sigma = 10$



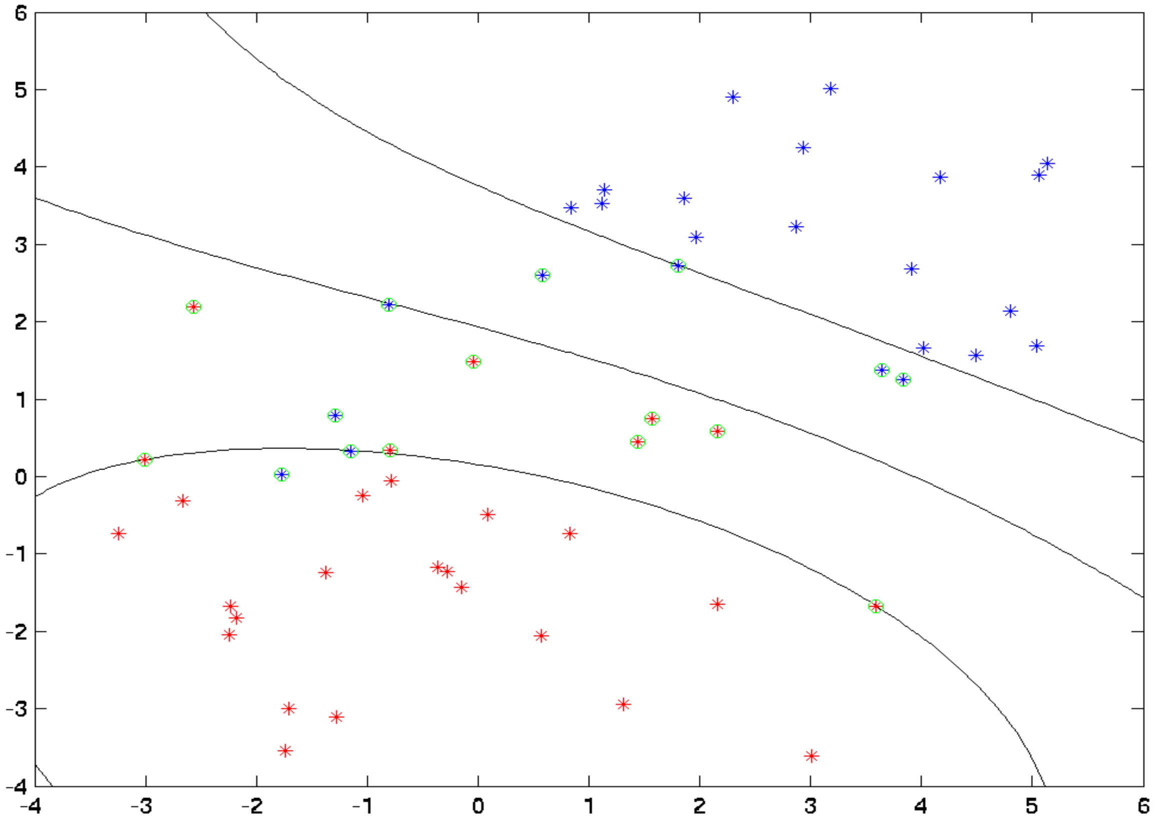
Gaussian RBF with $\sigma = 1$



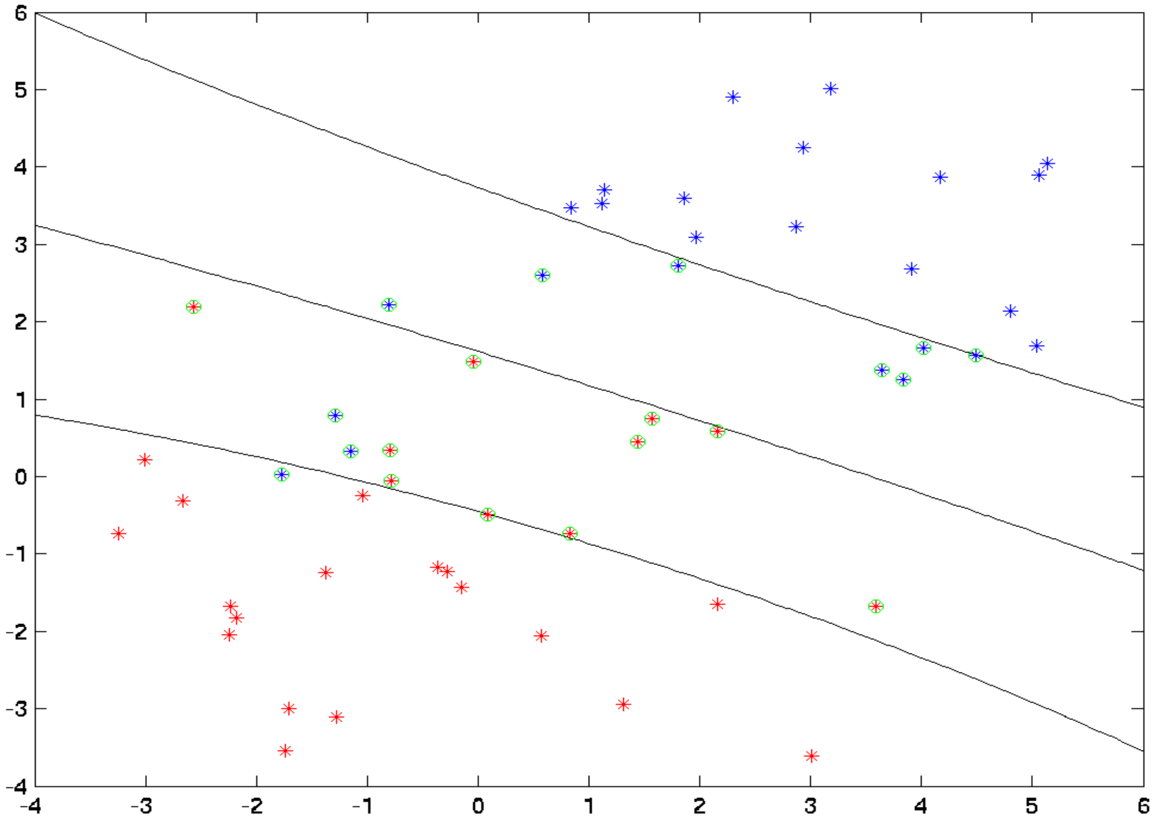
Gaussian RBF with $\sigma = 2$



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

Today's Summary

Machine learning and probability theory

Introduction to pattern recognition, classification, regression, novelty detection, probability theory, Bayes rule, inference

Density estimation and Parzen windows

Kernels and density estimation, Silverman's rule, Watson Nadaraya estimator, crossvalidation

Perceptron and kernels

Hebb's rule, perceptron algorithm, convergence, feature maps, kernel trick, examples

Support Vector classification

Geometrical view, dual problem, convex optimization, kernels and SVM