An Introduction to Machine Learning with Kernels

Lecture 5

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

Statistical Machine Learning Program
National ICT Australia, Canberra

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

L5 Applications

Microarray Analysis

- Data
- Classification
- Gene Selection

Biological Sequence Analysis

- Protein functions
- Sequence annotation
- String kernels

Document Analysis

- Bag of words
- Document retrieval
- Ordinal regression and ranking

Microarrays for Dummies

Genes

- Think of them as “subroutines” of the cell
- Assume that activity of genes tells us something about cell status
- Can only measure amount of mRNA (messenger RNA), not genes directly.

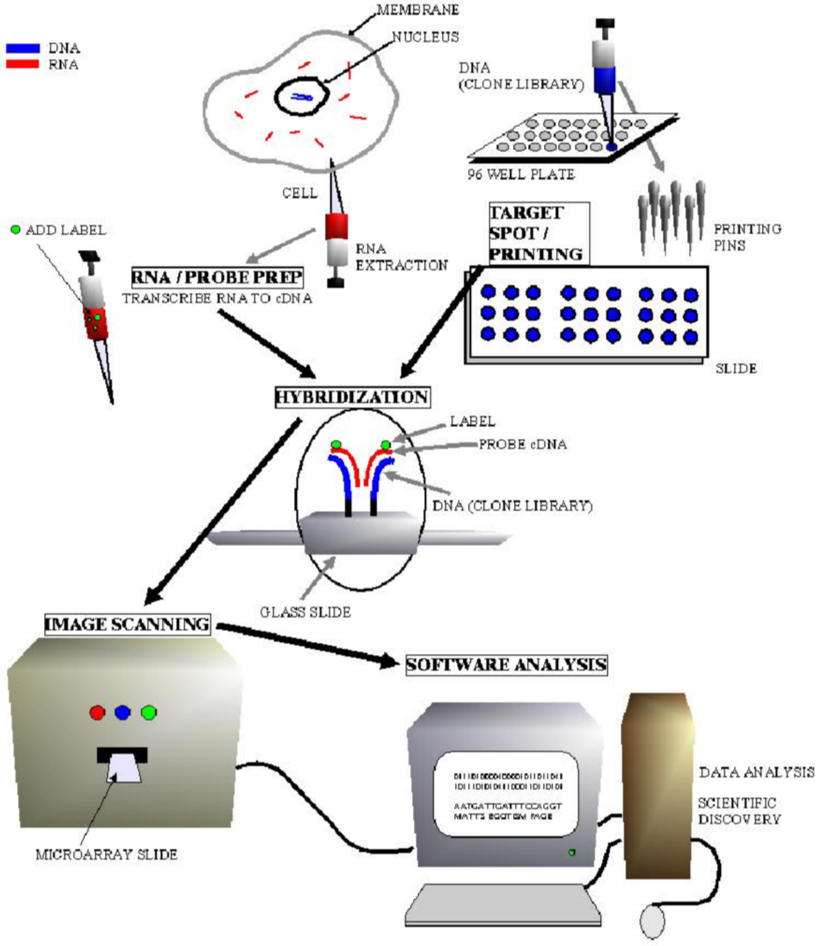
Goal

- Detect disease in cell (e.g. cancer)
- Understand cell activity
- Understand function of genes

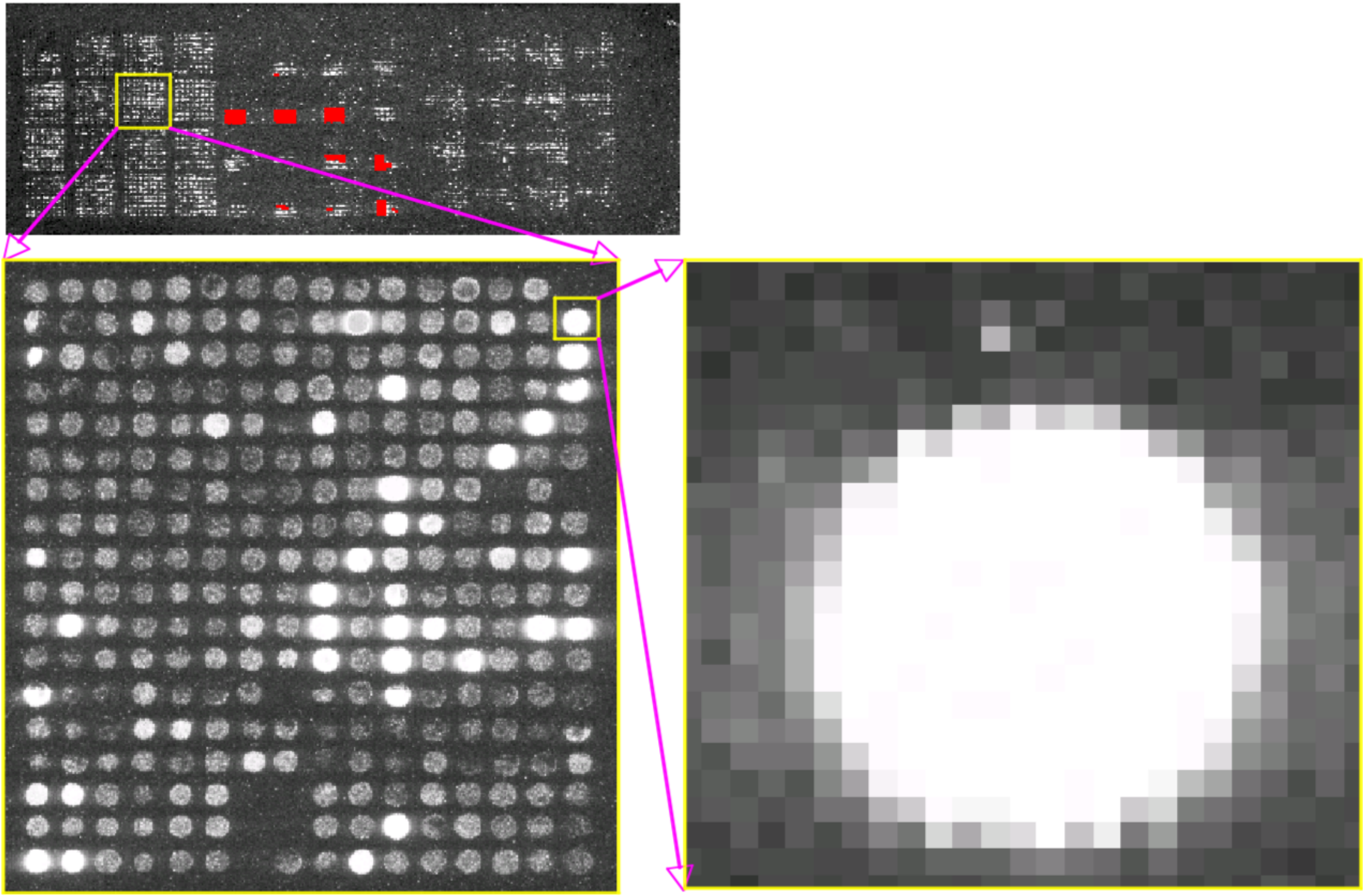
Method

- Print “detectors” for mRNA on a glass slide
- Pour cell content on it and let react
- Measure amount of substance

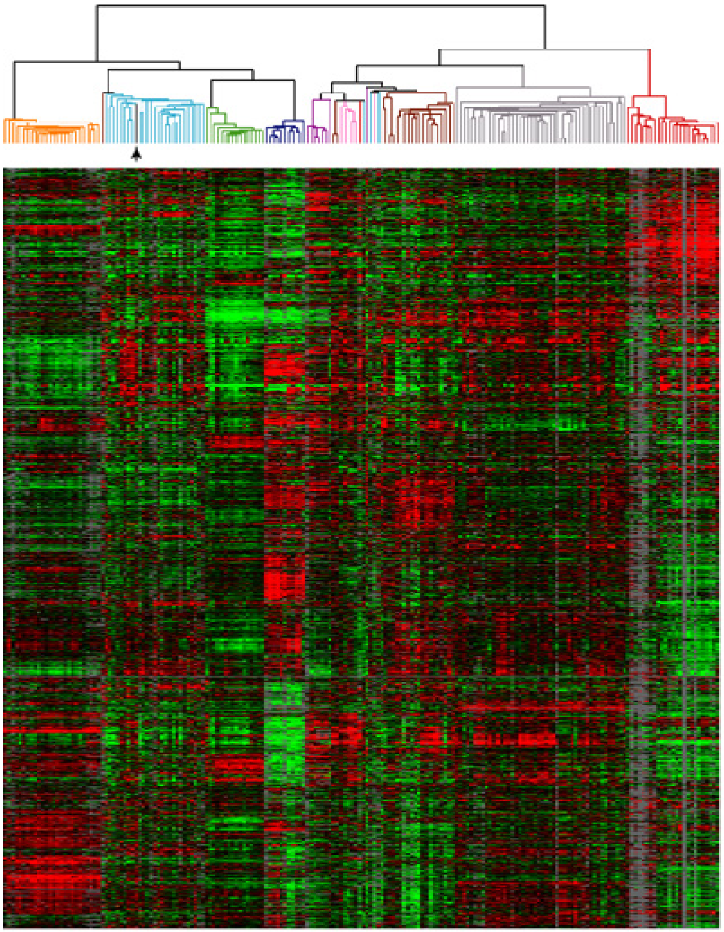
Microarray Process

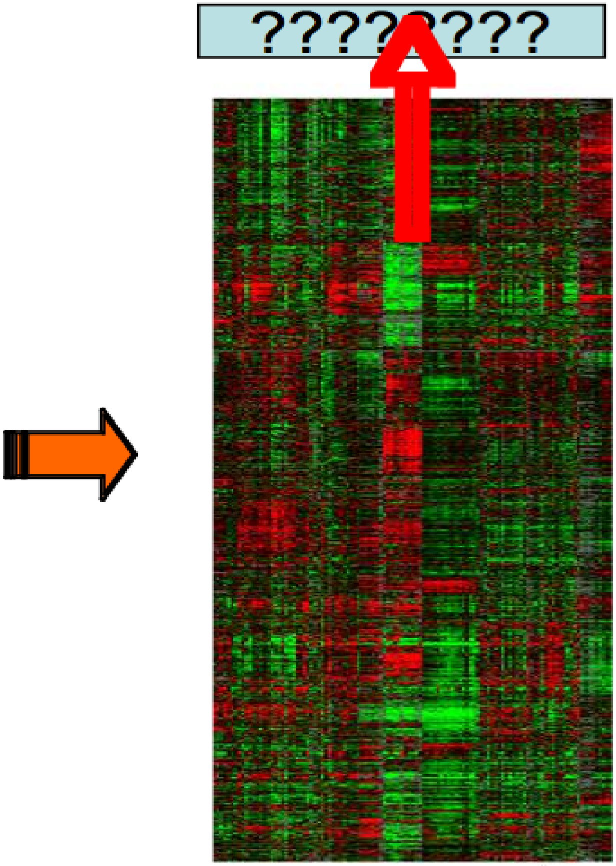


Raw Image



Processed Microarray Data





Dimensionality of the Data

Genes

- up to 100,000 on latest devices (Affymetrix)
- typically around 1,000 to 10,000, e.g. for cancer diagnosis, selective breeding (spotted arrays)
- noisy measurements
- missing data (measurement, processing, etc.)

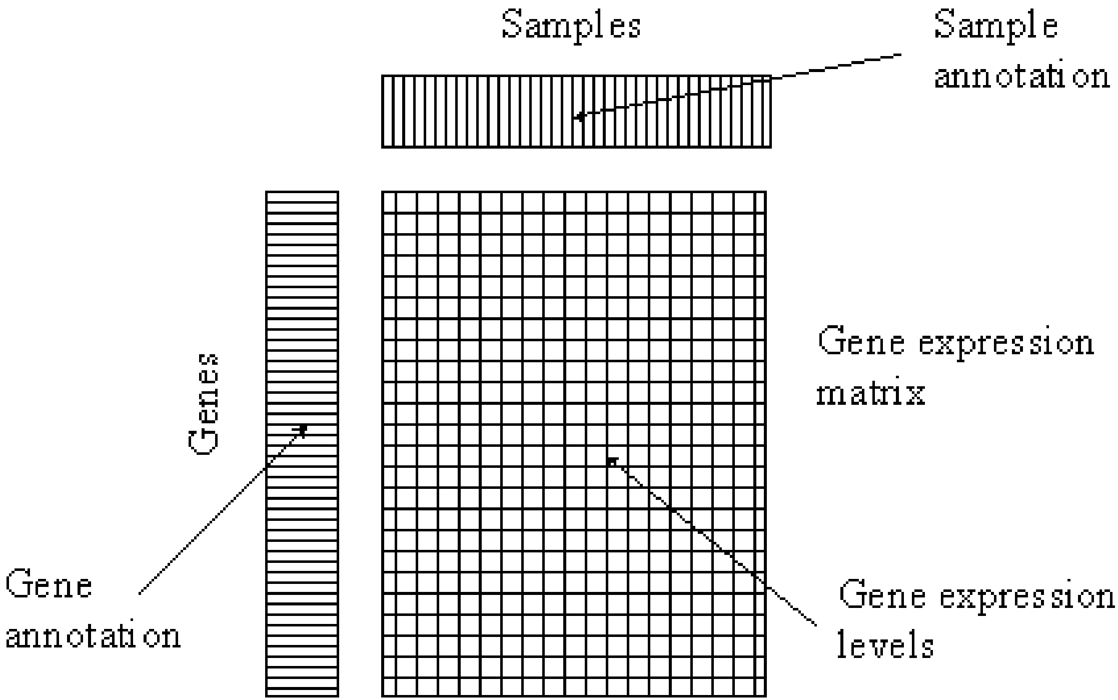
Observations

- 1 to 4 observations per patient, cell, plant, etc.
- Few patients, often different labs
- Typically 100-200 observations (privacy and ethics)
- Sometimes 1,000 observations (mainly plants)

Problems

- Data highdimensional, few observations
- Biologists want interpretation

Data


Samples
Sample annotation
Genes
Gene expression matrix
Gene annotation
Gene expression levels

Simple Approach

SVM Classification

- Linear classifier
- Solve SVM classification problem using inner product matrix between observations

Advantages

- Small Gram matrix, independent of number of genes:

$$K_{ij} = \sum_{l=1}^{n} x_{il} x_{jl} \text{ where } n \geq 10,000$$

- Easy optimization problem (< 0.1s on laptop)
- Solution involves all genes

Problems

- Solution involves all genes (bad for interpretation)
- Not very reliable

Feature Selection

Goal

- Select genes which are meaningful for problem
- Select genes such that tests are cheaper and faster
- Select genes to increase reliability of estimate

Problem

- Would get "meaningful" results even from random data (hint: try it with your friendly biologist and watch them explain random results ...)
- For most selection methods, can find datasets where it breaks.
- **Useful but use are your own peril!**

Iterative Selection Procedures

Basic Idea

- Solve original estimation problem (e.g. via SVM classification)
- Remove least meaningful genes
- Repeat procedure

Example: SVM Feature Selection, Guyon et al. 2000

- Want to find meaningful genes for classification
- Solve linear SVM optimization problem
- Pick smallest coordinates in

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

  and remove them. Remove 10-20% of them
- Repeat procedure with subset of genes

Iterative Selection Procedures

Example: Wavelet Denoising, Donoho et al. 1995

- Run wavelet transform
- Remove smallest coefficients. No repeat

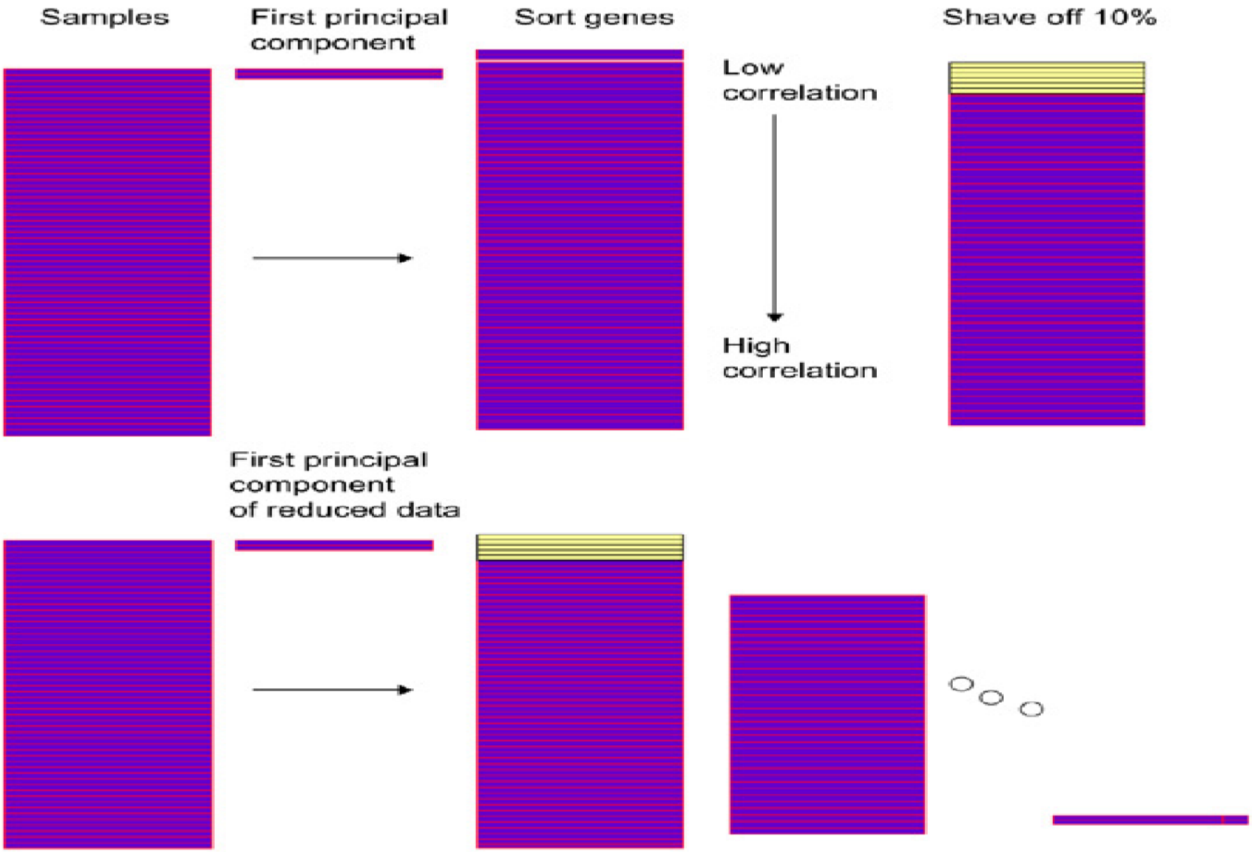
Example: Gene Shaving, Hastie et al. 2000

- Want to find meaningful genes, maybe also clustering
- Perform principal component analysis
- Remove genes with small coordinate projections along leading principal components
- Repeat procedure with subset of genes

Result

- Correlated genes (aligned with principal component)
- Criterion to stop shaving process (use variance)
- Repeat process on remainder: find new clusters

Gene Shaving



Regularization Selection Procedures

Basic Idea

- We want to classify well and that with as few genes as possible.
- Set up optimization problem to reflect that

Optimization Problem

- Generic setup

$$\text{minimize } C \sum_{i=1}^{m} \xi_i + \Omega[w]$$
$$\text{subject to } y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

- Regularizer $\Omega[w]$ such that many small coefficients and few large coefficients are preferred.
- Need penalty which increases **quickly** for small w_i.

Examples

SVM regularizer

$$\Omega[w] = \sum_{i=1}^{n} w_i^2$$

Feature selection regularizer (Boosting, etc.)

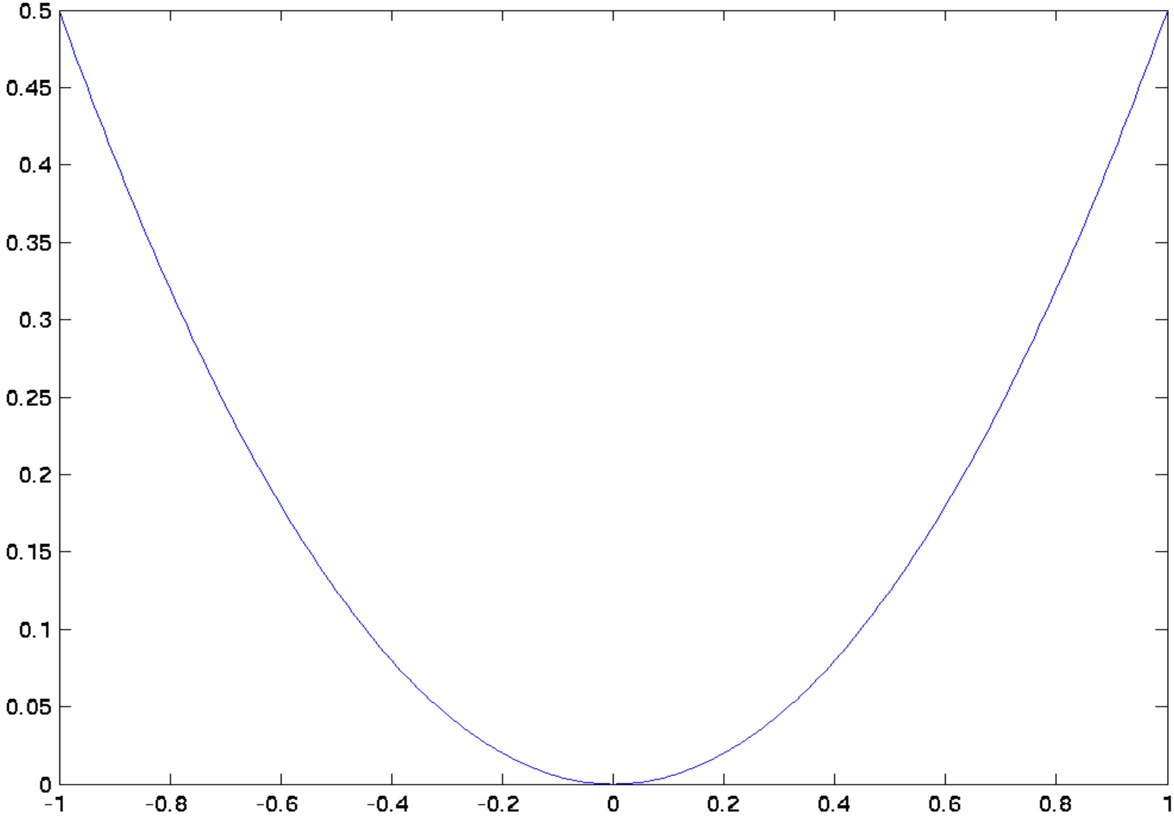
$$\Omega[w] = \sum_{i=1}^{n} |w_i|$$

Relevance vector machine regularizer

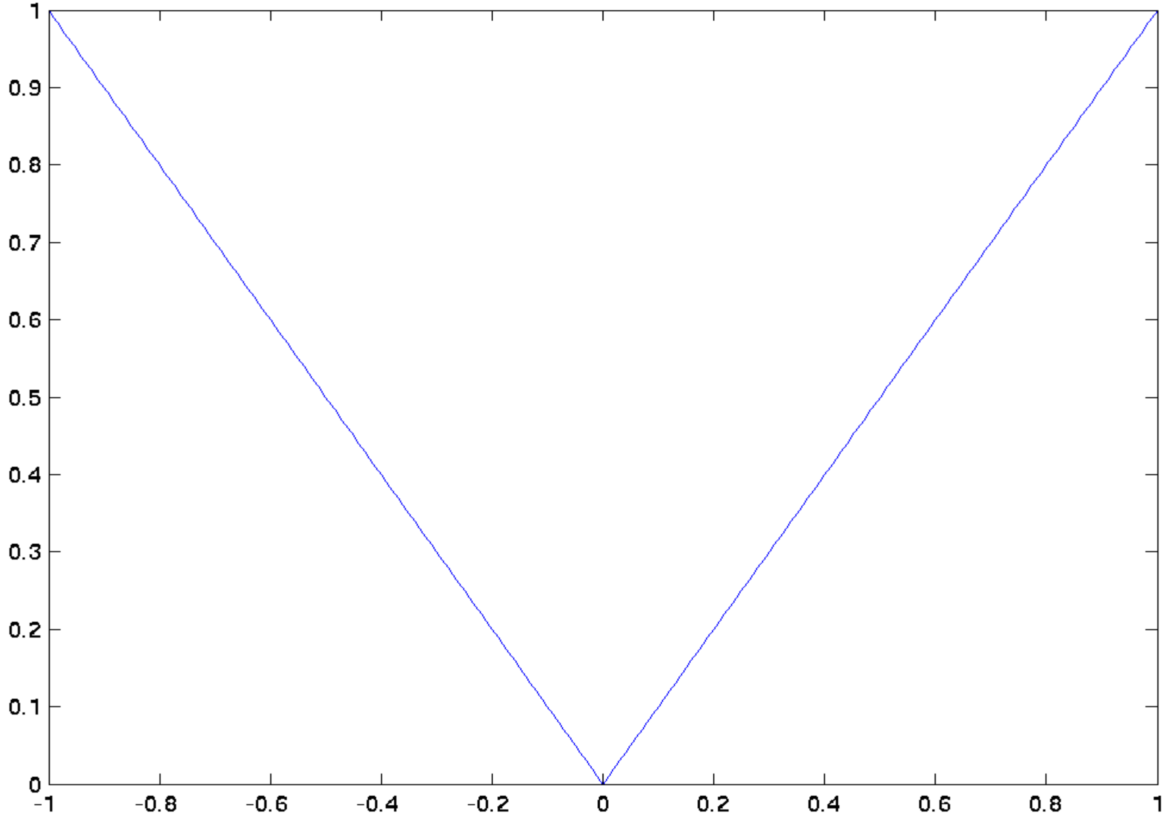
$$\Omega[w] = \sum_{i=1}^{n} -\log p_\gamma(w_i)$$

- $p_\gamma(w_i)$ is the Γ distribution
- For details see Tipping et al. 2001
- For microarrays see Campbell and Lin, 2003

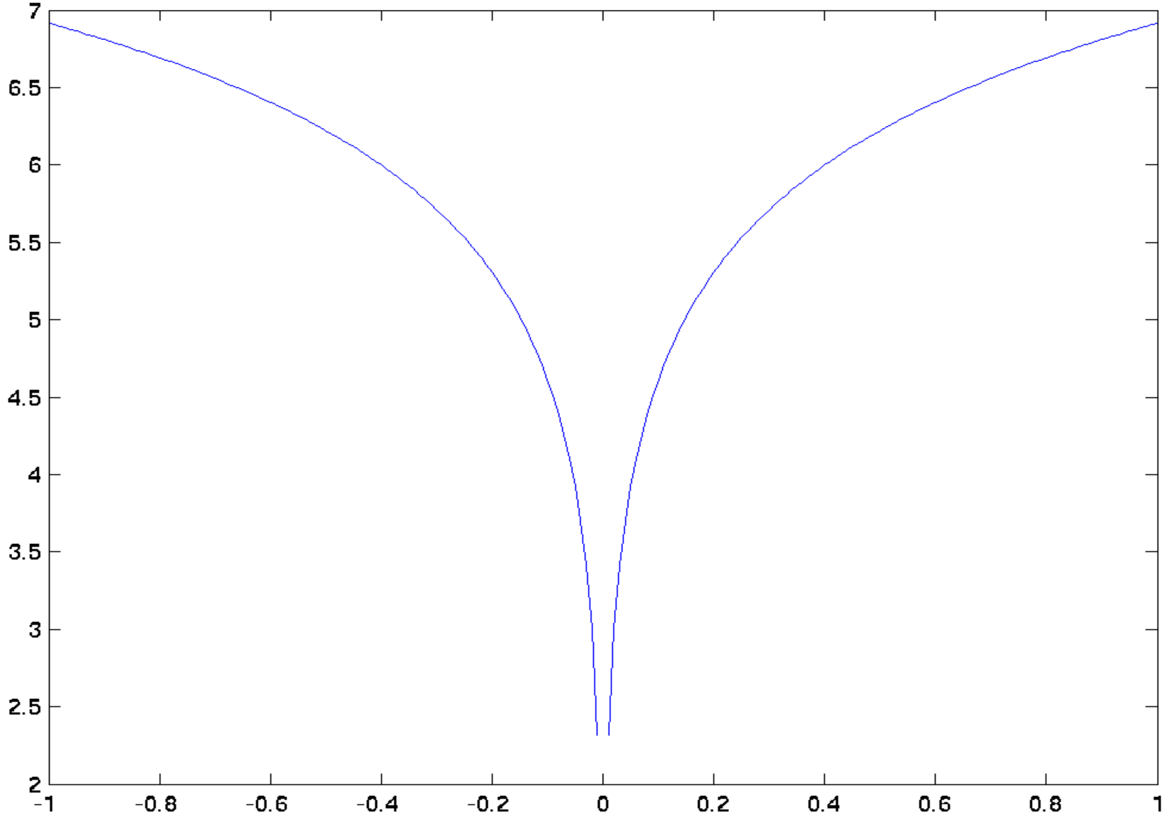
SVM Regularization



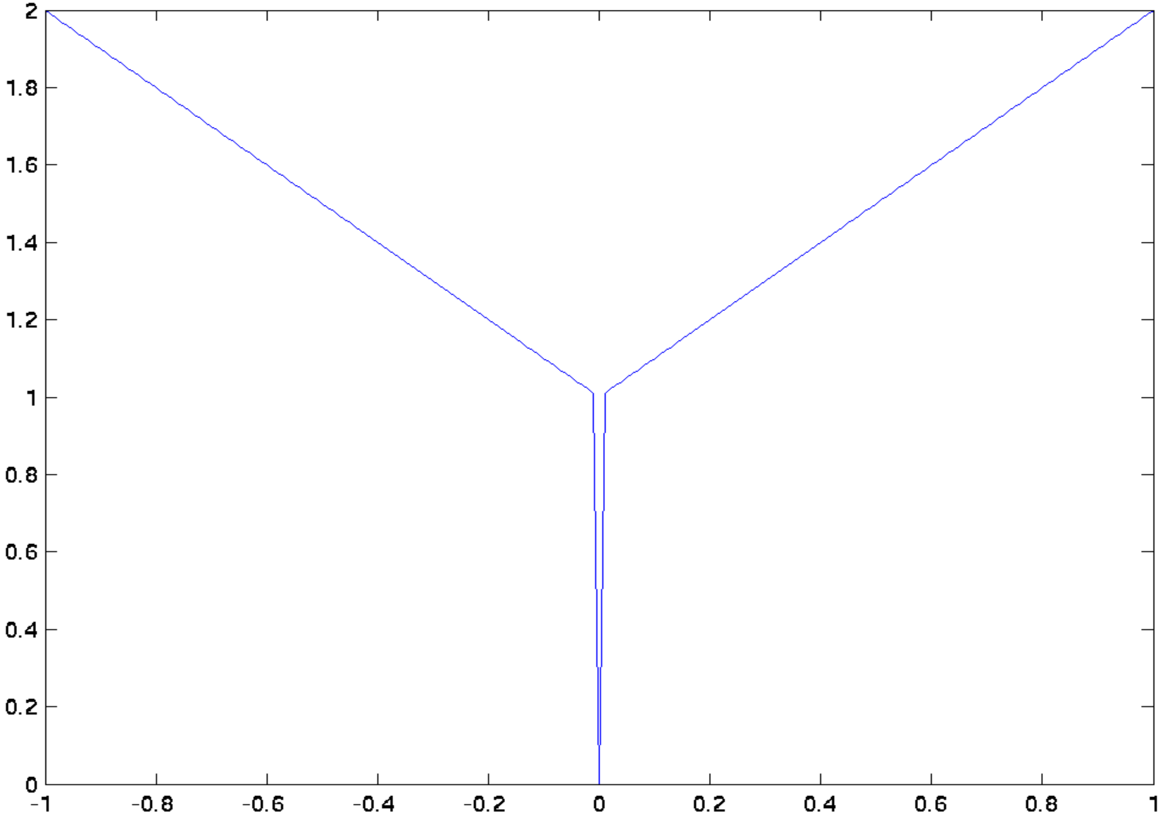
L1 Regularization



RVM Regularization



L0-L1 Regularization (Fung et al. 2002)



Coordinate Selection Procedures

Basic Idea

- Lots of genes, unreliable, use **really simple** criterion
- Check discriminative power for each gene separately and pick the top scoring ones

Examples

- Difference in means

$$s_j := \sum_{y_i=1} x_{ij} - \sum_{y_i=-1} x_{ij}$$

- Discriminative variance

$$s_j := \frac{\sum_{y_i=1} x_{ij} - \sum_{y_i=-1} x_{ij}}{\mathrm{std}\{x_{1j}, \ldots x_{mj}\}}$$

- 101 other and similar functions . . .

Mini Summary

Microarray Data

- Genes
- Data generation

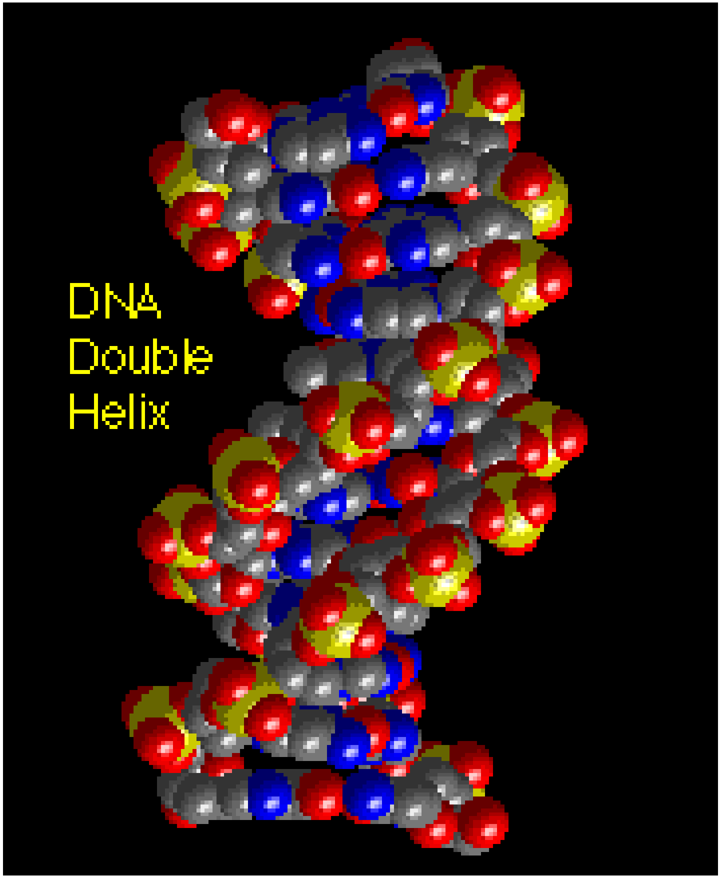
Problems

- High dimensional, few observations
- Need interpretability

Solution Approaches

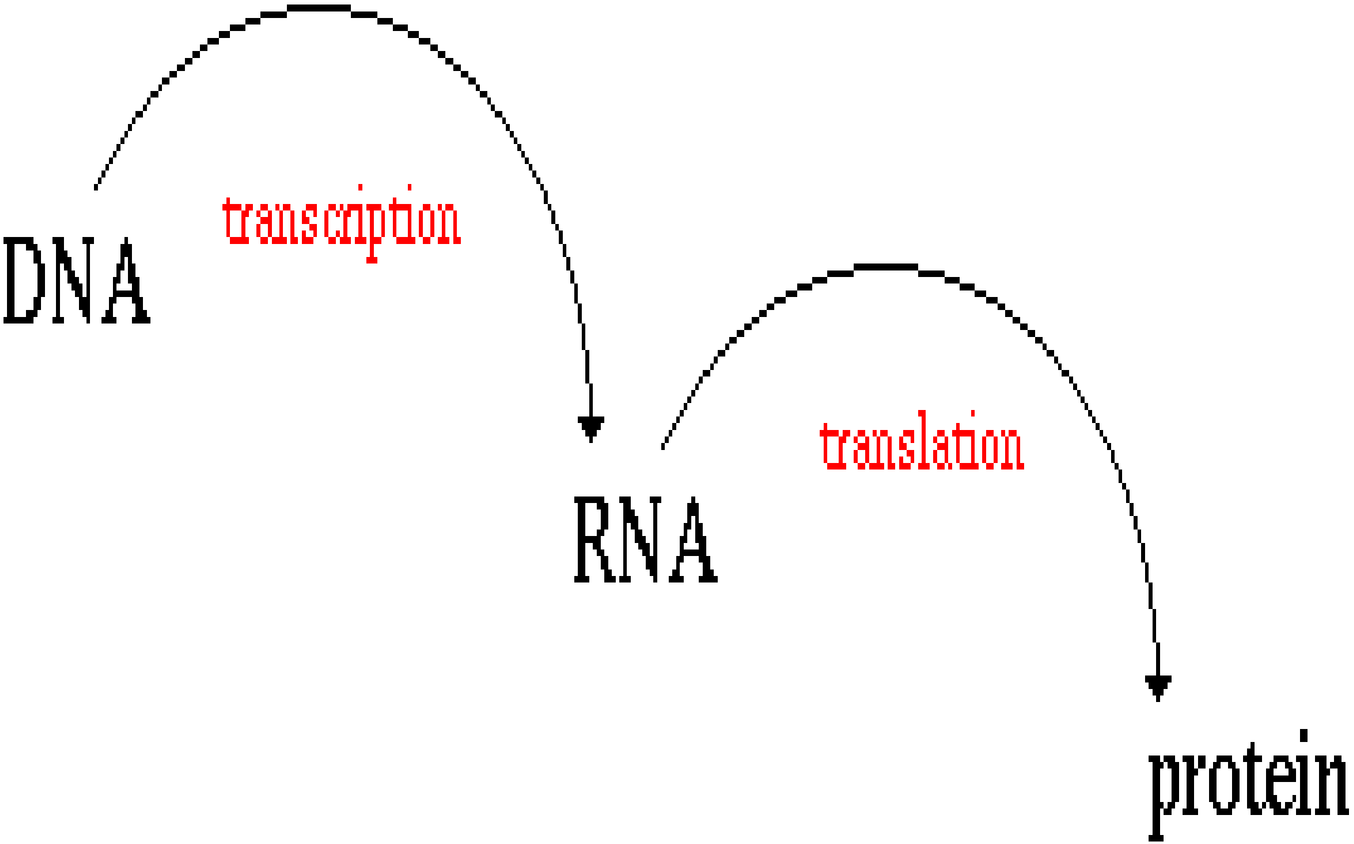
- Plain vanilla linear SVM
- Feature selection by iteration
- Feature selection by regularization
- Feature selection by coordinate wise choice

Biological Sequences



- Linear chain
  - Adenine
  - Guanine
  - Cytosine
  - Thymine
- Very long chain
  - 10^5 for bacteria
  - 10^9 for plants and mammals
- Store sequence
  `...GATTACA ...`

Central Dogma



Structure Prediction

Primary

Sequence itself

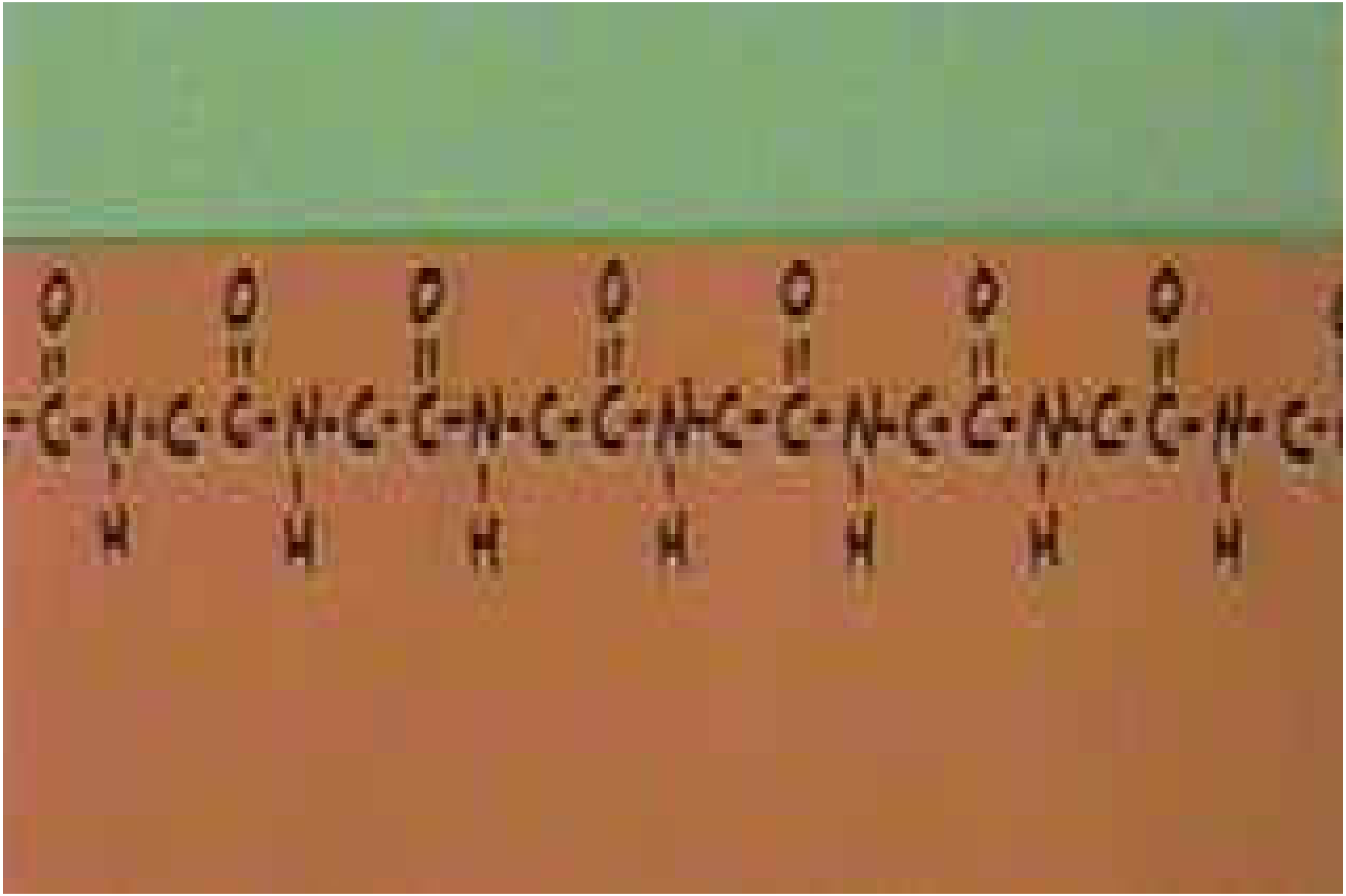
Secondary (structural motifs)

- α-helix
- β-sheet
- Loop, coil (or anything else that doesn't fit)

Tertiary

- 3D structure
- Packing of secondary structures
- Determines function

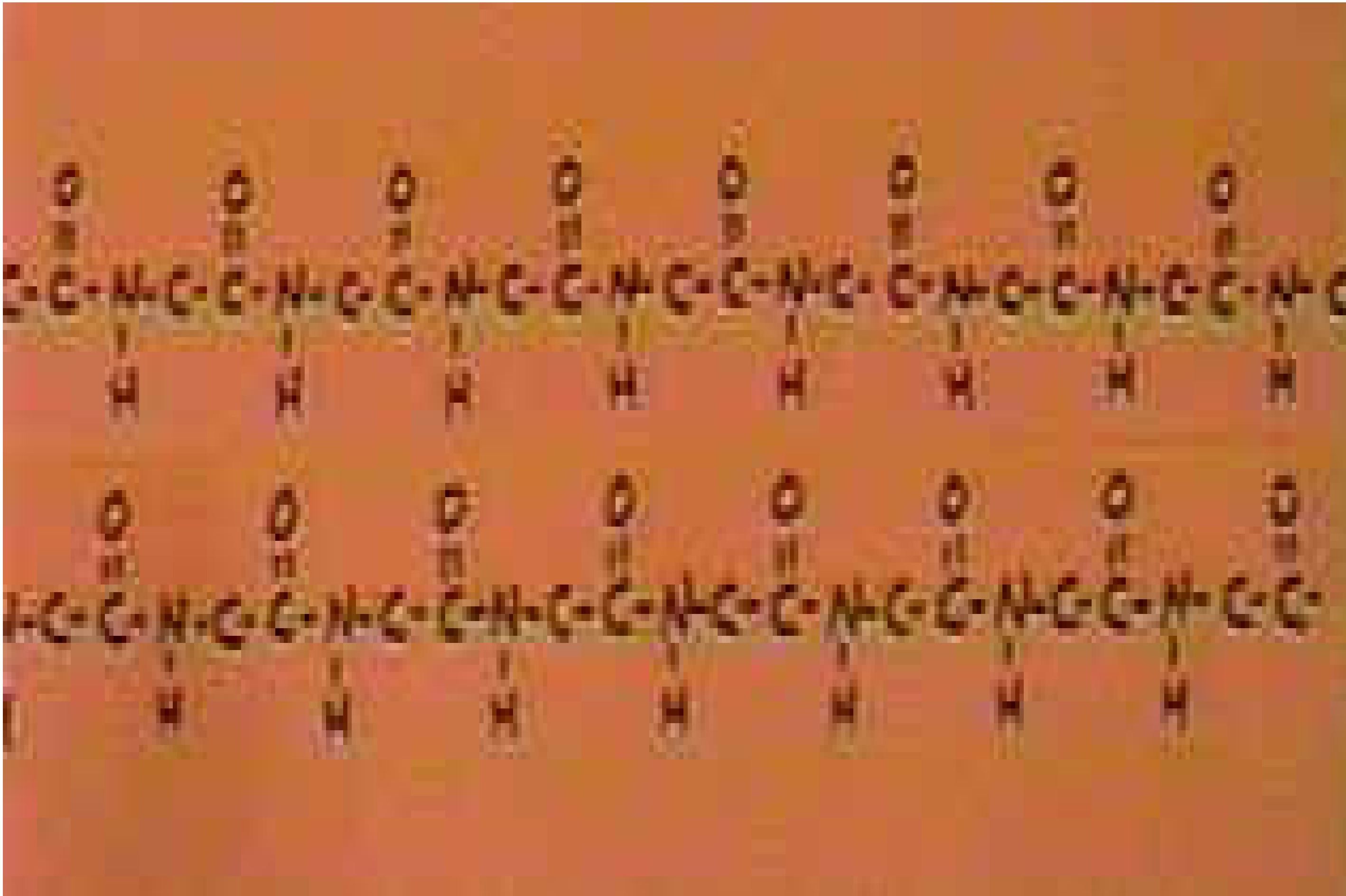
An Alpha Helix



An Alpha Helix

An Alpha Helix

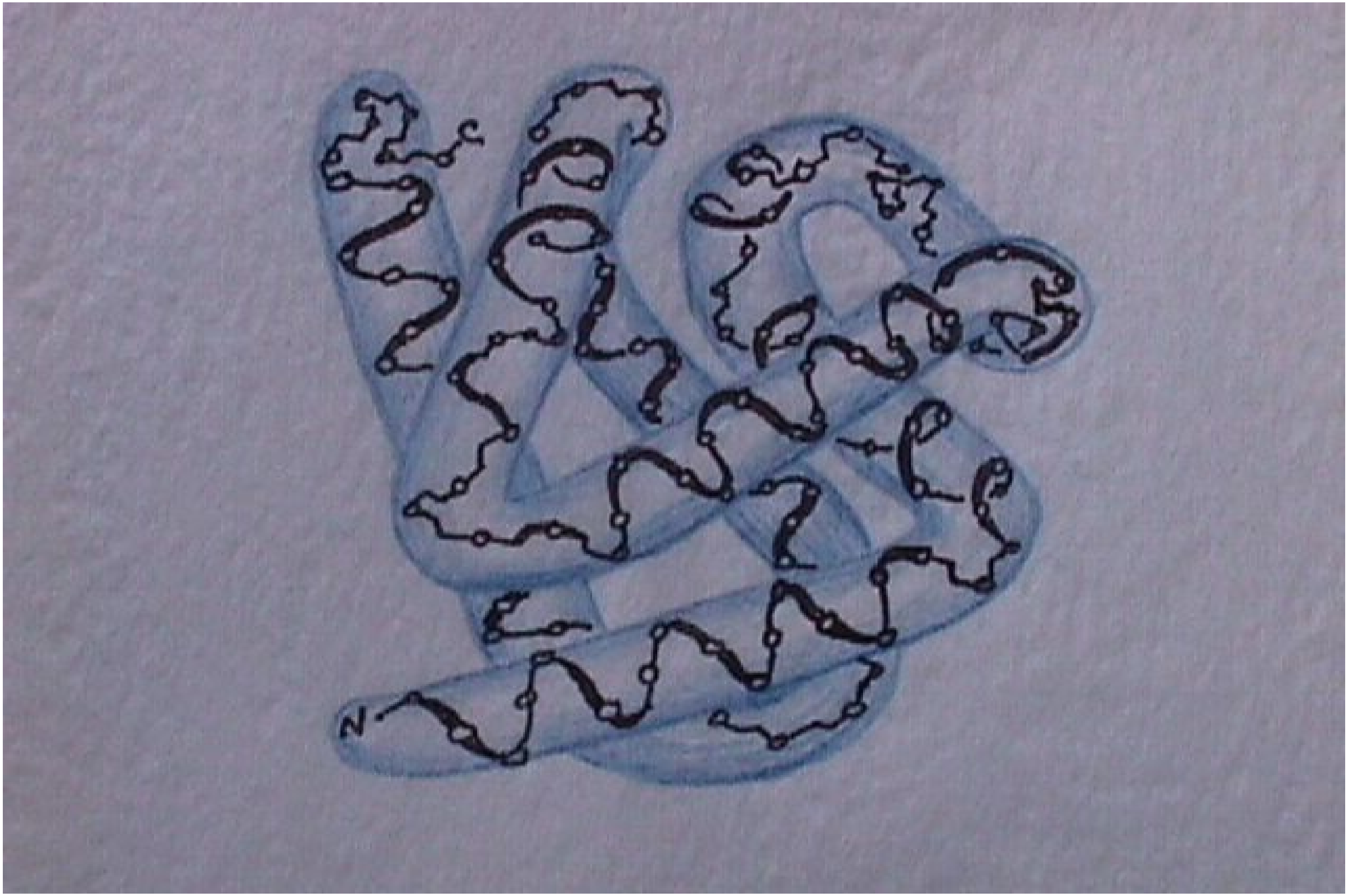
A Beta Sheet



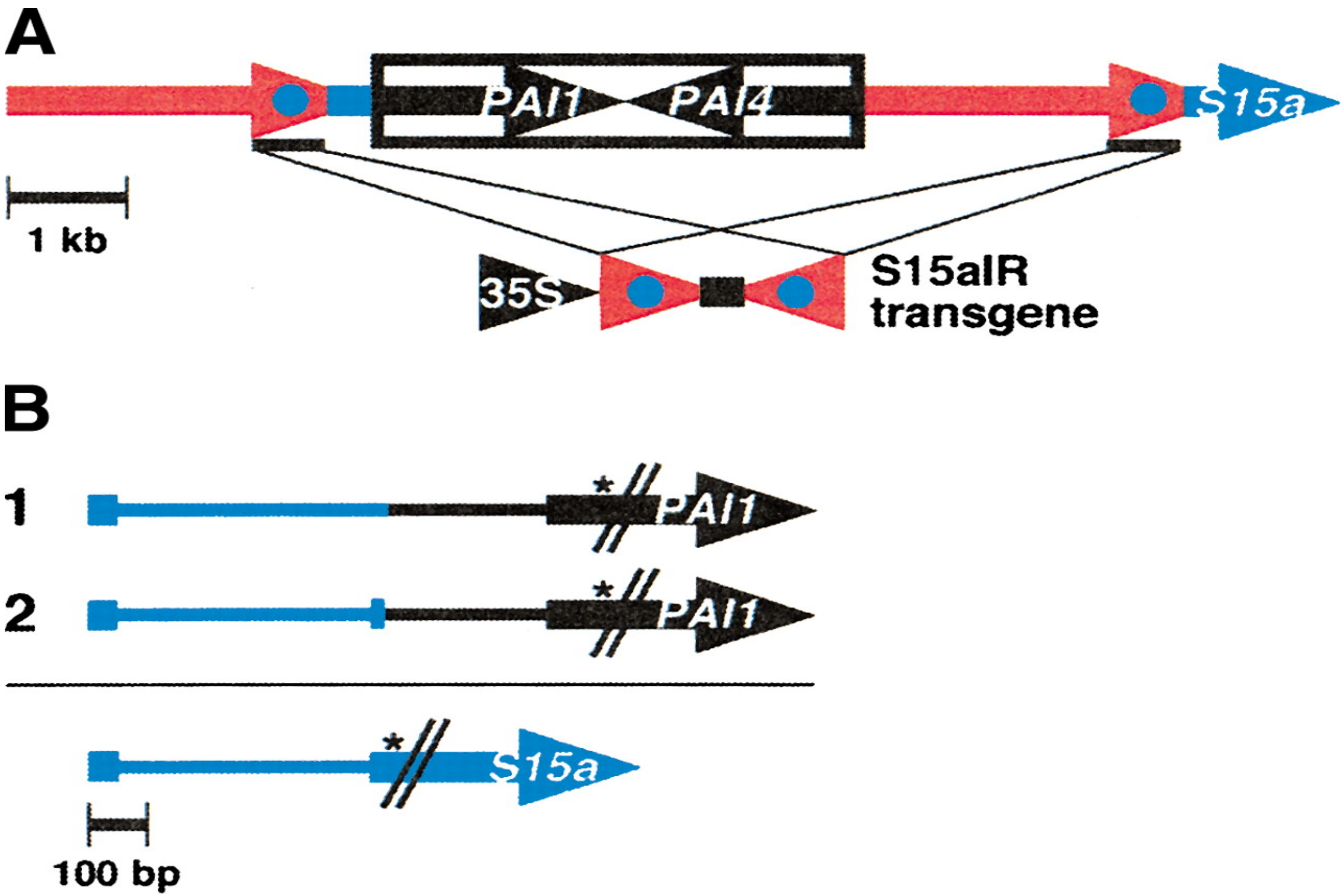
Tertiary Structure

Tertiary Structure

Myoglobin



Promoter


A
PAI1
PAI4
S15a
1 kb
35S
S15aIR
transgene
B
1
PAI1
2
PAI1
S15a
100 bp

The Problem

Data

- Sequence, something like . . . GATTACA . . .
- Information about window length (sometimes)
- Information about genes (sometimes)

Goal

- secondary structure estimates (local)
- locations of promoters and splice sites (local)
- 3D structure (global)
- function (global)
- location of genes (local)

Abstract Problem

- Given a sequence
- Find annotation of it

Challenges

Why use machine learning

- Too complex to be solved from first principles
- Some labeled data available
- Labeling is very expensive (lots of people in labs needed)

Challenges for machine learning

- Large amounts of data
- Large amounts of **unlabeled** data
- Data with lots of structure (sequences, graphs)
- Output with lots of structure (trees, sequences, graphs)
- Combination of different data sources and data types

Polynomial Kernel

Simple Idea

- Use polynomial kernel on symbols
- Treat symbols as dummy variables
- Use window around area of interest

Kernel

- Polynomial function

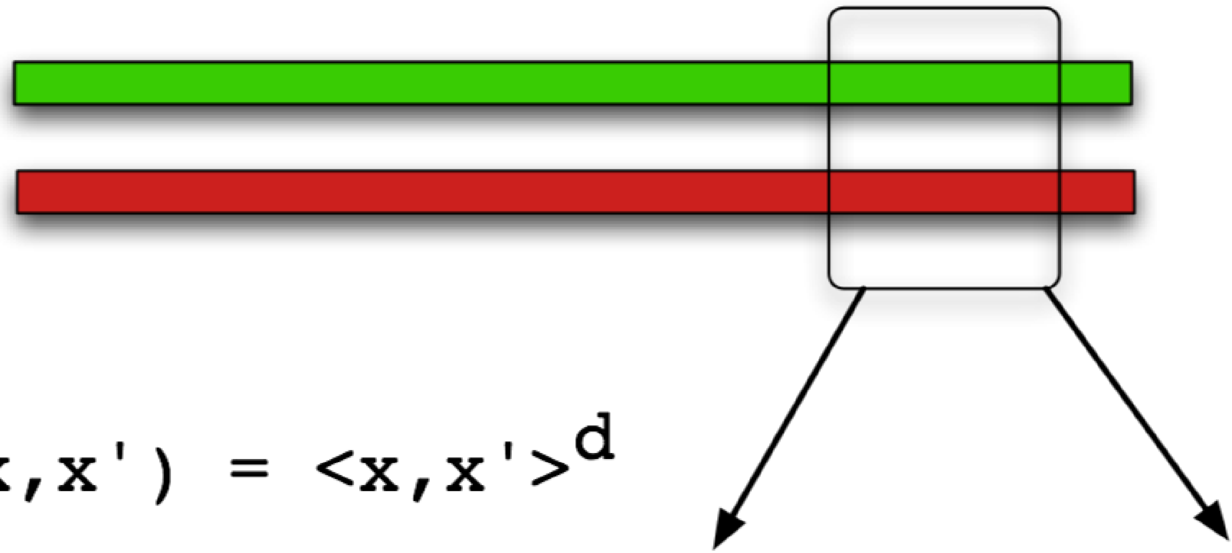
$$k(x, x') = (\langle x, x' \rangle + c)^d$$

- This counts the number of matches between sequences, raised to the power of d.
- Kernel in the space of all matches up to length d.

Improvements

- Weigh local matches around region of interest
- Use additional side information (e.g. from HMM)

Polynomial Kernel



$$k(x,x') = \langle x,x' \rangle^d$$

ATCAGCGAGA

=.=..==.=.

AGCTTCGTGC

1010011010

String Kernels

Basic Idea

- Local matches more important than long-range
- Want to count matches between sequences according to their importance (e.g. frequent sequences are probably not so meaningful)
- Want to have flexible weighting function
- Do this efficiently

Connection to Natural Language Processing

- Biological strings and texts look very similar
- Similar problems: annotate and label sequences

Insight

- Use the same tools for NLP and Bioinformatics
- Works amazingly well

More about the NLP motivation later

Mini Summary

Data

- DNA sequences
- Secondary structure sequences
- Graphs (alignment between sequences)

Goal

- Annotate the sequence
- Do it efficiently (large datasets)

Tools

- Simple similarity measure
- Polynomial kernels
- String kernels

String Kernel Basics

Some Notation

Alphabet: what we build strings from
Sentinel Character: usually $\$$, it terminates the string
Concatenation: xy obtained by assembling strings x, y
Prefix / Sufix: If $x = yz$ then y is a prefix and z is a suffix

Exact Matching Kernels

$$k(x, x') := \sum_{s \sqsubseteq x, s' \sqsubseteq x'} w_s \delta_{s,s'} = \sum_{s \in \mathcal{A}^*} \#_s(x) \#_s(x') w_s.$$

Inexact Matching Kernels

$$k(x, x') := \sum_{s \sqsubseteq x, s' \sqsubseteq x'} w_{s,s'} = \sum_{s \in \mathcal{A}^*} \#_s(x) \#_s(x') w_{s,s'}.$$

Counting mismatch much more expensive . . .

String Kernel Examples

Bag of Characters

$w_s = 0$ for all $|s| > 1$ counts single characters. Can be computed in linear time and linear-time predictions

Bag of Words

s is bounded by whitespace. Linear time

Limited Range Correlations

$w_s = 0$ for all $|s| > n$ for length n limited range

K-spectrum kernel

This takes into account substrings of length k (Eskin et al., 2002), where $w_s = 0$ for all $|s| \neq k$. Linear time kernel computation, and quadratic time prediction.

General Case

Quadratic time kernel computation (Haussler, 1998, Watkins, 1998), cubic time prediction.

Tree Kernels

Definition (Colins and Duffy, 2001)

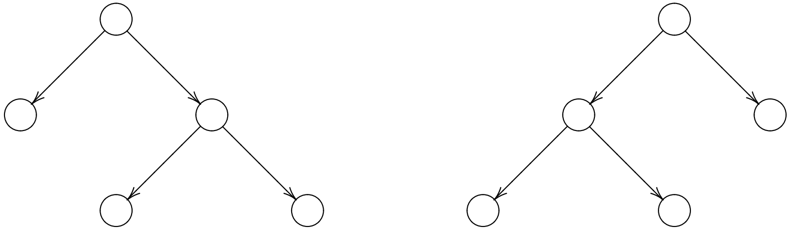
Denote by T, T' trees and by $t \models T$ a subtree of T, then

$$k(T, T') = \sum_{t \models T, t' \models T'} w_t \delta_{t,t'}.$$

We count matching subtrees (other definitions possible, will come to that later).

Problem

We want permutation invariance of unordered trees.



Solution

Sort trees before computing kernel

Sorting Trees

Sorting Rules

- Assume existence of lexicographic order on labels
- Introduce symbols '[', ']' satisfy '[' < ']', and that ']', '[' < $\mathrm{label}(n)$ for all labels.

Algorithm

- For an unlabeled leaf n define $\mathrm{tag}(n) := \texttt{[]}$.
- For a labeled leaf n define $\mathrm{tag}(n) := \texttt{[}\mathrm{label}(n)\texttt{]}$.
- For an unlabeled node n with children $n_1, \ldots, n_c$ sort the tags of the children in lexicographical order such that $\mathrm{tag}(n_i) \leq \mathrm{tag}(n_j)$ if $i < j$ and define

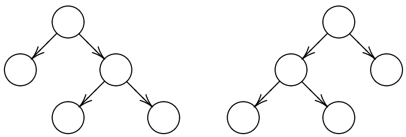
$$\mathrm{tag}(n) = \texttt{[}\mathrm{tag}(n_1)\mathrm{tag}(n_2)\ldots\mathrm{tag}(n_c)\texttt{]}.$$

- For a labeled node same operations as above

$$\mathrm{tag}(n) = \texttt{[}\mathrm{label}(n)\mathrm{tag}(n_1)\mathrm{tag}(n_2)\ldots\mathrm{tag}(n_c)\texttt{]}.$$

Sorting Trees in Linear Time

Example

The trees  have label `[[][[][]]]`.

Theorem

1. $\text{tag}(\text{root})$ can be computed in $(\lambda + 2)(l \log_2 l)$ time and linear storage in l.
2. Substrings s of $\text{tag}(\text{root})$ starting with '$[$' and ending with a balanced '$]$' correspond to subtrees T' of T where s is the tag on T'.
3. Arbitrary substrings s of $\text{tag}(\text{root})$ correspond to subset trees T' of T.
4. $\text{tag}(\text{root})$ is invariant under permutations of the leaves and allows the reconstruction of an unique element of the equivalence class (under permutation).

Tree to String Conversion

Consequence

We can compute tree kernel by

1. Converting trees to strings
2. Computing string kernels

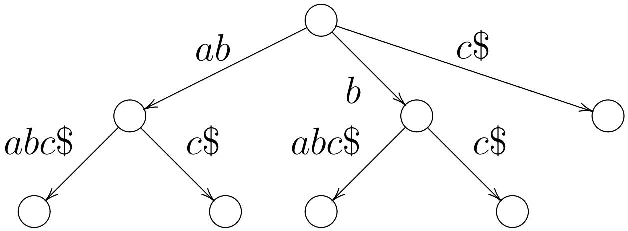
Advantages

- More general subtree operations possible: we may include non-balanced subtrees (cutting a slice from a tree).
- Simple storage and simple implementation (dynamic array suffices)
- All speedups for strings work for kernels, too (XML documents, etc.)

Suffix Trees

Definition

Compact tree built from all the suffixes of a word. Suffix tree of `ababc`



Properties

- Can be built and stored in linear time (Ukkonen, 1995)
- Leaves on subtree $\equiv$ matching substrings

Suffix Links

Connections *across* the tree. Vital for parsing strings (e.g., if we parsed `abracadabra` this speeds up the parsing of `bracadabra`).

Matching Statistics

Definition

Given strings x, y with $|x| = n$ and $|y| = m$, the matching statistics of x with respect to y are defined by $v, c \in \mathbb{N}^n$, where

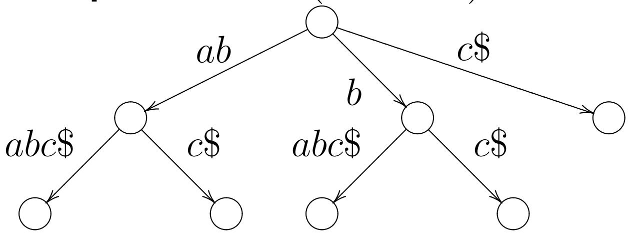
- v_i is the length of the longest substring of y matching a prefix of $x[i:n]$
- $\overline{v_i} := i + v_i - 1$
- c_i is a pointer to $\mathrm{ceil}(x[i:\overline{v_i}])$ in $S(y)$.

Computable in linear time (Chang and Lawler, 1994).

Example

Matching statistic of `abba` with respect to $S(\texttt{ababc})$.

String	a	b	b	a
v_i	2	1	2	1
$\mathrm{ceil}(c_i)$	ab	b	babc$	ab



Matching Substrings

Prefixes

w is a substring of x iff there is an i such that w is a prefix of $x[i : n]$. The number of occurrences of w in x can be calculated by finding all such i.

Substrings

The set of matching substrings of x and y is the set of all prefixes of $x[i : \overline{v_i}]$.

Next Step

If we have a substring w of x, prefixes of w may occur in x with higher frequency. We need an efficient computation scheme.

Key Trick

Assumptions

x and y strings, c and v matching statistics of x w.r.t. y.

$$W(y, t) = \sum_{s \in \mathrm{pref}(v)} w_{us} - w_u \text{ where } u = \mathrm{ceil}(t) \text{ and } t = uv.$$

can be computed in $O(1)$ time for any t.

Theorem

$k(x, y)$ can be computed in $O(|x| + |y|)$ time via

$$\begin{aligned} k(x, y) &= \sum_{i=1}^{|x|} \mathrm{val}(x[i : \overline{v_i}]) \\ &= \sum_{i=1}^{|x|} \mathrm{val}(c_i) + \mathrm{lvs}(\mathrm{floor}(x[i : \overline{v_i}]))W(y, x[i : \overline{v_i}]) \end{aligned}$$

where $\mathrm{val}(t) := \mathrm{lvs}(\mathrm{floor}(t)) \cdot W(y, t) + \mathrm{val}(\mathrm{ceil}(t))$

$W(y,t)$ in Constant Time

Length-Dependent Weights

Assume that $w_s = w_{|s|}$, then

$$W(y,t) = \sum_{j=|\text{ceil}(t)|}^{|t|} w_j - w_{|\text{ceil}(t)|} = \omega_{|t|} - \omega_{|\text{ceil}(t)|}$$

where $\omega_j := \sum_{i=1}^{j} w_j$

Examples

- Correlations up to length s. Simply set all weights after w_s to 0.
- Exponentially decaying weight
- Bounded range
- Fixed length correlations (e.g. only of length s)

$W(y,t)$ in Constant Time

Generic Weights

- Simple option: pre-compute the annotation of all suffix trees beforehand.
- Better: build suffix tree on all strings (linear time) and annotate this tree.
- Simplifying assumption for TFIDF weights

$$w_s = \phi(|s|)\psi(\#s)$$

$$W(y,t) = \sum_{s \in \mathrm{pref}(t)} w_s - \sum_{s \in \mathrm{pref}(\mathrm{ceil}(t))} w_s$$

$$= \phi(\mathrm{freq}(t)) \sum_{i=|\mathrm{ceil}(t)|+1}^{|t|} \phi(i)$$

Linear Time Prediction

Problem

For prediction we need to compute $f(x) = \sum_i \alpha_i k(x_i, x)$.

- This depends on the number of SVs.
- Bad for large databases (e.g., spam filtering). The classifier degrades in runtime, the more data we have.
- We are repeatedly parsing s

Idea

We can merge matching weights from all the SVs. All we need is a compressed lookup function.

Linear Time Prediction

- Merge all SVs into one suffix tree Σ.
- Compute matching statistics of x wrt. Sigma.
- Update weights on every node of Σ as

$$\text{weight}(\overline{w}) = \sum_{i=1}^{m} \alpha_i \text{lvs}_{x_i}(\overline{w})$$

- Extend the definition of $\text{val}(x)$ to Σ via

$\text{val}_\Sigma(t) := \text{weight}(\text{floor}(t)) \cdot W(\Sigma, t) + \text{weight}(\text{ceil}(t))$ and $\text{val}_\Sigma(\text{root}$

- Here $W(\Sigma, t)$ denotes the sum of weights between $\text{ceil}(t)$ and t, with respect to Σ rather than $S(y)$. We only need to sum over $\text{val}_\Sigma(x[i : \overline{v_i}])$ to compute f.

We can classify texts in linear time regardless of the size of the SV set!

Mini Summary

- Redux of Tree to String kernels (heaps, stacks, bags, etc. trivial)
- Linear prediction and kernel computation time (previously quadratic or cubic). Makes things practical.
- Storage of SVs needed. Can be greatly reduced if redundancies abound in SV set. E.g. for `anagram` and `analphabet` we need only `analphabet` and `gram`.
- Coarsening for trees (can be done in linear time, too)
- Approximate matching and wildcards
- Automata and dynamical systems
- Do “expensive” things with string kernel classifiers.

Documents

Data

- Plain text
- HTML/XML documents
- WWW graph
- Structured database records

Goal

- Categorize them
- Preference relations
- Authorship
- Annotate them (named entity tagging)

HTML Documents

```
<html>

<head>
<meta http-equiv="content-type" content="text/html;charset=iso-8859-1">
<title>canberra.yourguide</title>

     <script language="JavaScript">
          function LOver(cobj,lobj,bgclr,lclr)
          {
          cobj.backgroundColor=bgclr;
          lobj.color=lclr;
          }

          function openWin(fname,winname)
          {
          ModalWin = window.open(fname,winname,"resizeable=no, scrollbars=yes, width=480, height=300");
          }
     </script>

  <style type="text/css"><!--
   a { text-decoration: none }-->
  </style>

 </head>


<body bgcolor="white" topmargin="0" leftmargin="0" marginheight="0" marginwidth="0" vlink="#003366"">

<!-- Include virtual="/_includes/grey_header.inc" -->



<table border="0" cellpadding="0" cellspacing="0" width="100%" bgcolor="#ffffff">
     <tr>
          <td valign="top" bgcolor="#333366">
               <table border="0" cellpadding="0" cellspacing="0" width="180" bgcolor="#666699">
                    <tr>
                         <td width="10" bgcolor="#666699"><img src="/images/general/space.gif" width="1" height="20"></td>
                         <td bgcolor="#666699"><a href="http://www.yourguide.com.au" target="_top"><font
face="Verdana,Geneva,Swiss,Helvetica,Arial" color="#FFFFFF" size="2">www.yourguide.com.au</font></a></td>
                         <td width="8" bgcolor="#666699"> </td>
                         <td bgcolor="#666699" valign="bottom"><img src="/images/general/tab_fronttop_r.gif" width="10" height="20"></td>
```

XML Data


```
<!DOCTYPE lewis SYSTEM "lewis.dtd">
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="5544" NEWID="1">
<DATE>26-FEB-1987 15:01:01.79</DATE>
<TOPICS><D>cocoa</D></TOPICS>
<PLACES><D>el-salvador</D><D>usa</D><D>uruguay</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;C T
&#22;&#22;&#1;f0704&#31;reute
u f BC-BAHIA-COCOA-REVIEW   02-26 0105</UNKNOWN>
<TEXT>&#2;
<TITLE>BAHIA COCOA REVIEW</TITLE>
<DATELINE>    SALVADOR, Feb 26 - </DATELINE><BODY>Showers continued throughout the week in
the Bahia cocoa zone, alleviating the drought since early
January and improving prospects for the coming temporao,
although normal humidity levels have not been restored,
Comissaria Smith said in its weekly review.
    The dry period means the temporao will be late this year.
    Arrivals for the week ended February 22 were 155,221 bags
of 60 kilos making a cumulative total for the season of 5.93
mln against 5.81 at the same stage last year. Again it seems
that cocoa delivered earlier on consignment was included in the
arrivals figures.
    Comissaria Smith said there is still some doubt as to how
much old crop cocoa is still available as harvesting has
practically come to an end. With total Bahia crop estimates
around 6.4 mln bags and sales standing at almost 6.2 mln there
```

Named Entity Recognition

Wolff	B-PER
,	O
currently	O
a	O
journalist	O
in	O
Argentina	B-LOC
,	O
played	O
with	O
Del	B-PER
Bosque	I-PER
in	O
the	O
final	O
years	O
of	O
the	O
seventies	O
in	O
Real	B-ORG
Madrid	I-ORG
.	O

Authorship

Wednesday, October 15, 2003

New releases from mozilla.org

As many of you have probably already noticed, mozilla.org has released new versions of the Mozilla Application Suite (1.5), Mozilla Firebird (0.7) and Mozilla Thunderbird (0.3). Check them out if you not already haven't.

I will now go and try to clean up the firebird part of bugzilla a little bit, before it is overran with duplicate bugs by newbies ;-)
posted by Simon : 12:31

Thursday, October 09, 2003

My odyssey in trying to build Mozilla Sunbird

Ok, so yesterday I went out to look for a current windows sunbird build. But the only build I found was the original testing build from over two month ago. So I thought, you have a working Mozilla build environment, why not build it yourself? So I asked in the newsgroups, but before anyone could answer, I found Mostafah on IRC today and asked him, if he could provide me with some instructions, which he did.

So after some initial hassle, he pointed me to a newsgroup posting which had detailed instructions. So I changed my .mozconfig and started building ('**make -f client.mk build**'), but in the last quarter of the compile run, I ran into bug 214940, which is a duplicate of bug 210791. So I asked Mostafah, who told me how to get around this bug.

Here's what you have to do directly after the compile run failed:

1. Goto the xpfe/components/autocomplete-directory and run plain '**make**'
2. Goto the mailnews-directory and run plain '**make**'
3. Now you have to have the recent calendar code checked out under mozilla/calendar, which I hadn't. So I had to do an '**cvs update**' in the calendar-directory
4. Goto the calendar/sunbird-directory
5. '**Copy Makefile.in Makefile**' and then run plain '**make**'
6. Run '**make -f client.mk build**' again
7. Goto the calendar/sunbird-directory and run '**make clean**' and then plain '**make**'

You should now have Sunbird in your dist/bin directory.

Unfortunately Sunbird doesn't quite work. The Sunbird-distribution also contains a mozillafirebird.exe and running mozillasunbird.exe brings up a firebird window. Running '**mozillasunbird.exe -calendar**' brings up a Mozilla Calendar window with a lot of missing chrome and most of the existing chrome not working.

Mostafah told me, that he hasn't built sunbird after the first release, so something in the dist may have changed an the code may need update. He'll try to take a look at the code as soon as he can. This is fine with me. Mostafah is a very nice guy and was very helpful. So a big '**Kudos**' goes out to him. Keep up the good work, Mostafah!
posted by Simon : 23:17

Features

Bag of Words

- Count number of occurrence of words in document
- Useful when trying to detect topics

Example

- Mr. Kerry, with strategists in both parties saying he had helped himself in the first of three debates with Mr. Bush, acted at campaign rallies in Florida as though he had instantly taken the upper hand. He told thousands of screaming Democrats that Mr. Bush thought he could "fool you all the time" on everything from Iraq to the economy.
- 2 Bush, 1 Democrats, 1 Florida, 1 Iraq, 1 Kerry, 3 Mr., 1 acted, 1 all, 1 as, 1 at, 1 both, 1 campaign, 1 could, 1 debates, 1 economy , 1 everything, 1 first, 1 fool, 1 from, 2 had, 1 hand, 4 he, 1 helped, 1 himself, 3 in, 1 instantly, 2 of, 1 on, 1 parties, 1 rallies, 1 saying, 1 screaming, 1 strategists, 1 taken, 1 that, 4 the, 1 though, 1 thought, 1 thousands, 1 three, 1 time, 1 to, 1 told, 1 upper, 2 with, 1 you

Features

- Sparse feature vector (lots of words do not occur)
- Length of document matters
- Some frequent words which do not contain much in-

Feature transformations

Problems

- Most words are missing
- Word frequency counts unreliable
- Frequent words are often not informative
- Words in various forms (e.g. wish, wishes, wished, big problem in other languages, e.g. German)
- Taking things out of context (bag of words)

Fixes

- Use Laplace rule for word counts (i.e. pseudocounts)
- Divide document by length
- Weigh by inverse document frequency
- Use stemming
- Use longer range correlations (StringKernel)

Practical Concerns

Sparse Feature Vector

- Store as sparse vector **(do not store the zeros)** and use sparse vector-vector multiplication.
- SVMLight is pretty good for that

Document Categorization

- Use multiclass classifier for multiple categories (SMV-Light supports it now)
- Use hierarchy of classes if available (e.g. for DMOZ: clothing - formal wear - jackets - dinner jackets)

Named entity tagging

- Use window around word to be tagged.
- Better method available now (but more complicated: conditional random fields and Max-Margin-Markov networks)

Authorship Identification

Problem

- Two collections of documents
- Determine whether written by same author
- May have tried to obscure identity (or different topics)
- No complete set of “other authors” available

Solution

- Try distinguishing both collections via SVM classifier
- Compute crossvalidation error
- Remove top scoring features, Repeat

Result

- For identical author, documents are hard to distinguish
- For different authors removing top scoring features does not degrade matters too much.
- Train SVM on same-same, same-different pairs.

Naive Bayes Classifier

Properties

- Super simple to implement
- Fast
- Mediocre performance
- Runs on many spam filters

Key Ingredient

- Estimator of $p(x_i|y)$, that is, probability of occurrence of words. We get this from individual documents.
- Often use Poisson distribution as model

Naive Bayes HOWTO

Fundamental assumption

$$p(x|y) = \prod_{i=1}^{m} p(x_i|y)$$

That is, word frequency only depends on class label

Bayes Rule

- Invoke it to use $p(x|y)$ for

$$p(y|x) \propto \prod_{i=1}^{m} p(x_i|y)p(y)$$

- Classifier via odds-ratio

$$\frac{p(y=1|x)}{p(y=-1|x)} = \frac{\prod_{i=1}^{m} p(x_i|y=1)p(y=1)}{\prod_{i=1}^{m} p(x_i|y=-1)p(y=-1)}$$

Mini Summary

Features

- Bag of words
- Long range correlations
- Use string kernels

Applications

- Document categorization
- Named entity tagging
- Authorship verification

Cheap Alternatives

- Naive Bayes classifier
- Mediocre performance but very fast and simple

Summary

Microarray Analysis

- Data
- Classification
- Gene Selection

Biological Sequence Analysis

- Protein functions
- Sequence annotation
- String kernels
- Efficient computation via suffix trees

Document Analysis

- Bag of words
- Document retrieval
- Ordinal regression and ranking

An Introduction to Machine Learning with Kernels

Lecture 6

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

Statistical Machine Learning Program
National ICT Australia, Canberra

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

L6 Optimization

Convex Optimization Basics

- Convex functions
- Optimality and uniqueness
- Subspace descent
- Numerical math basics

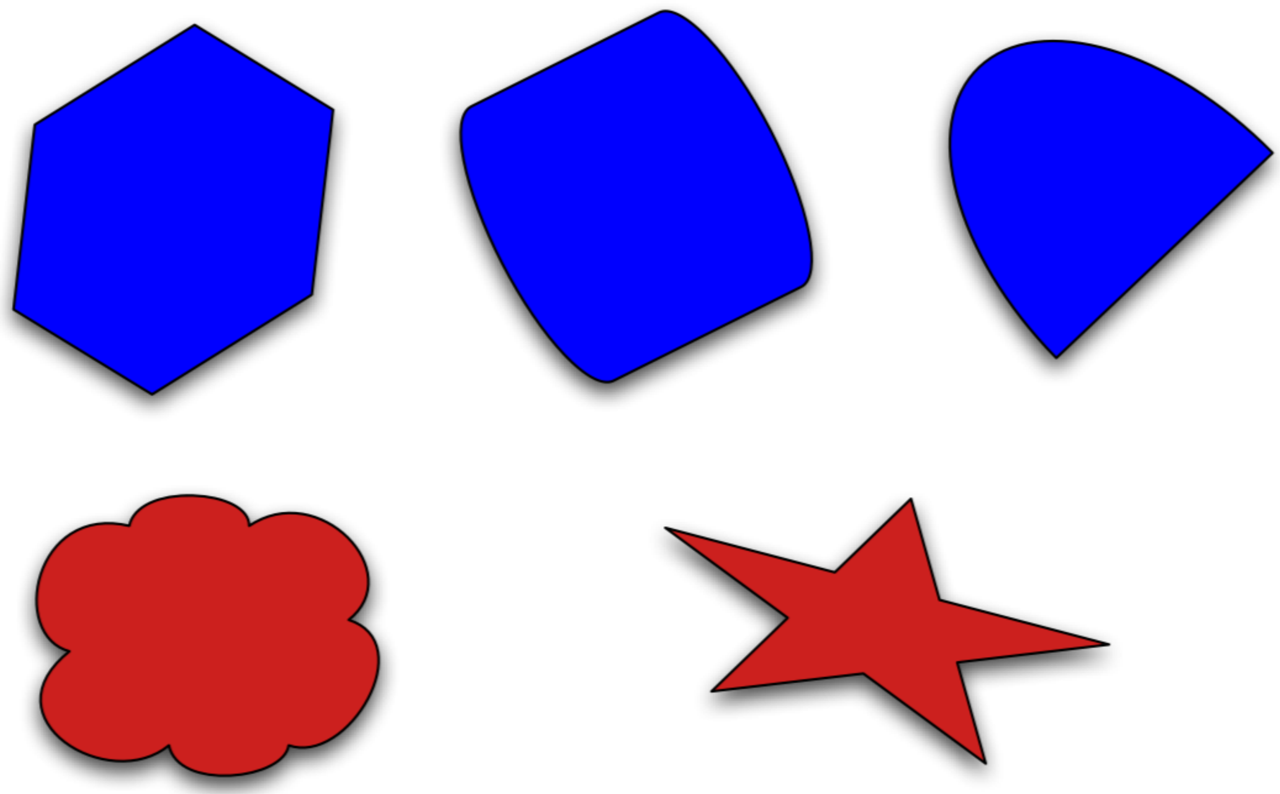
Sequential Minimal Optimization and Chunking

- Chunking
- Explicit solution
- Selection strategy

Stochastic gradient descent

- Basic idea
- Online SVM
- Further applications

Convexity



Convexity

Convex Set

A set X is called convex if for any $x, x' \in X$ and any $\lambda \in [0, 1]$ we have

$$\lambda x + (1 - \lambda)x' \in X.$$

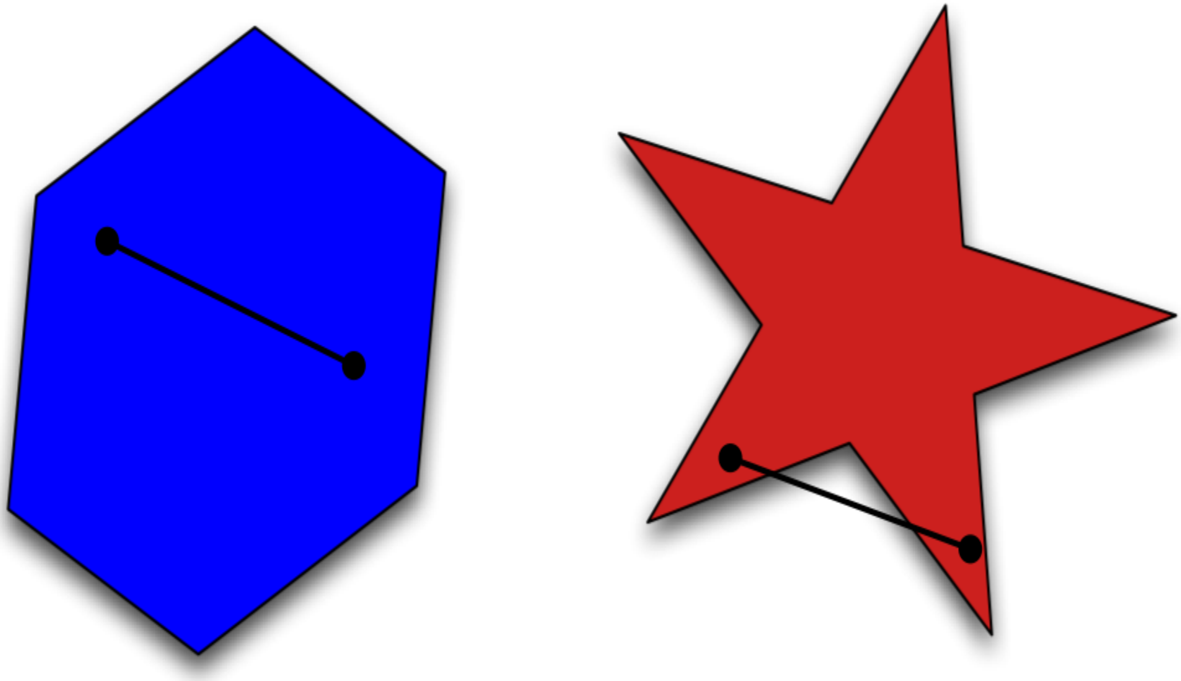
Convex Function

A function f defined on a set X (note that X need not be convex itself) is called convex if for any $x, x' \in X$ and any $\lambda \in [0, 1]$ such that $\lambda x + (1 - \lambda)x' \in X$ we have

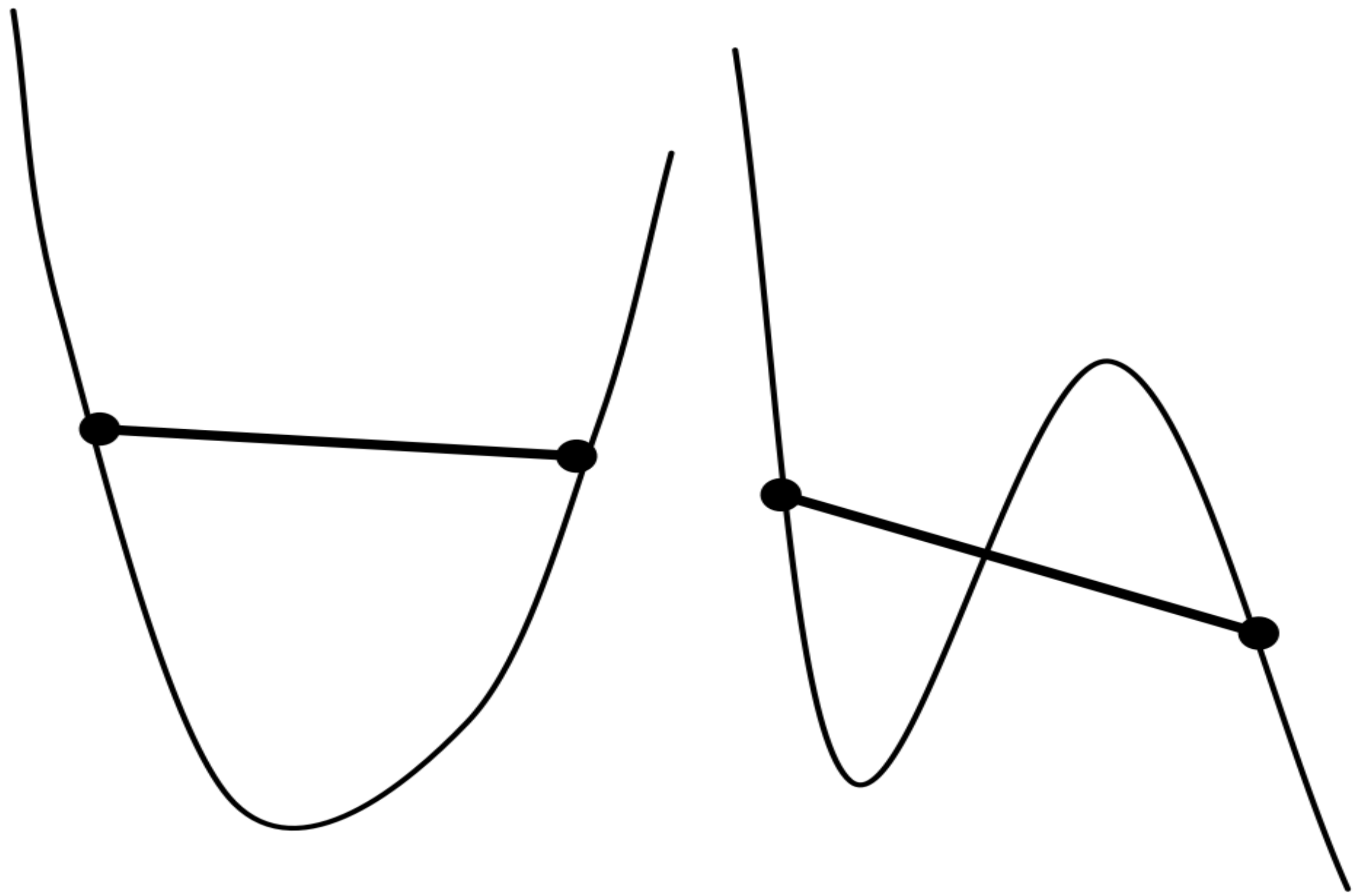
$$f(\lambda x + (1 - \lambda)x') \leq \lambda f(x) + (1 - \lambda)f(x').$$

A function f is called *strictly* convex if the inequality is strict for $\lambda \in (0, 1)$.

Convex and Nonconvex Sets



Convex and Nonconvex Functions



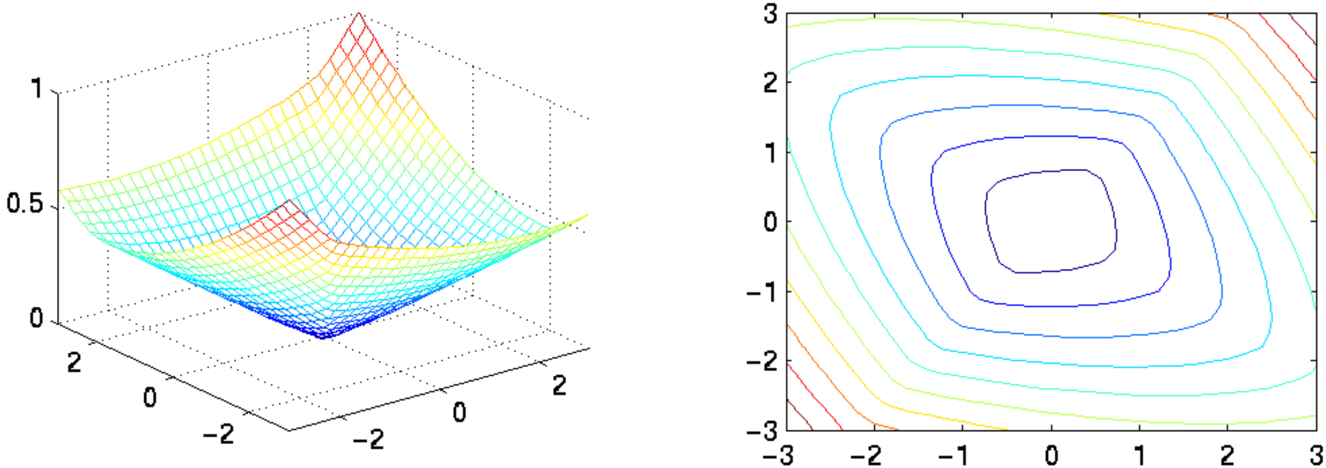
Convex Sets as Below Sets

Lemma

If $f : \mathfrak{X} \to \mathbb{R}$ is a convex function. Then the set

$$X := \{x | x \in \mathfrak{X} \text{ and } f(x) \leq c\} \text{ for some } c \in \mathbb{R}$$

is convex.



Uniqueness of Minimum

Key Theorem

- Convex function f on convex set X
- Consequently f has unique minimum on X

Proof Idea

- Assume that there are two minima x and x'
- Draw line between them
- Use the fact that the function is convex which gives contradiction.

Newton Method

Basic Idea

- Minimize $f(x)$ using quadratic approximation

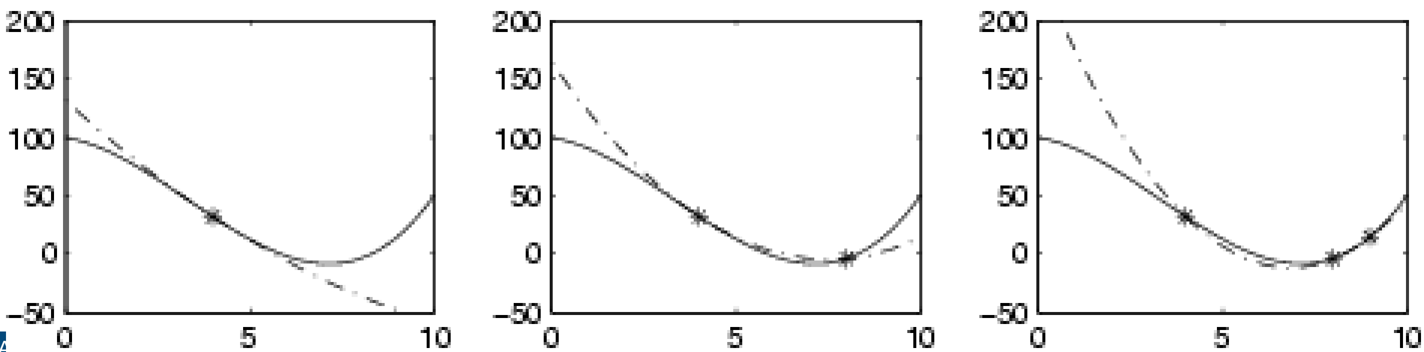
$$f(x+\delta x) \approx f(x) + \delta x f'(x) + \frac{1}{2}(\delta x)^2 f''(x)$$

- Solve at each step for the minimum explicitly

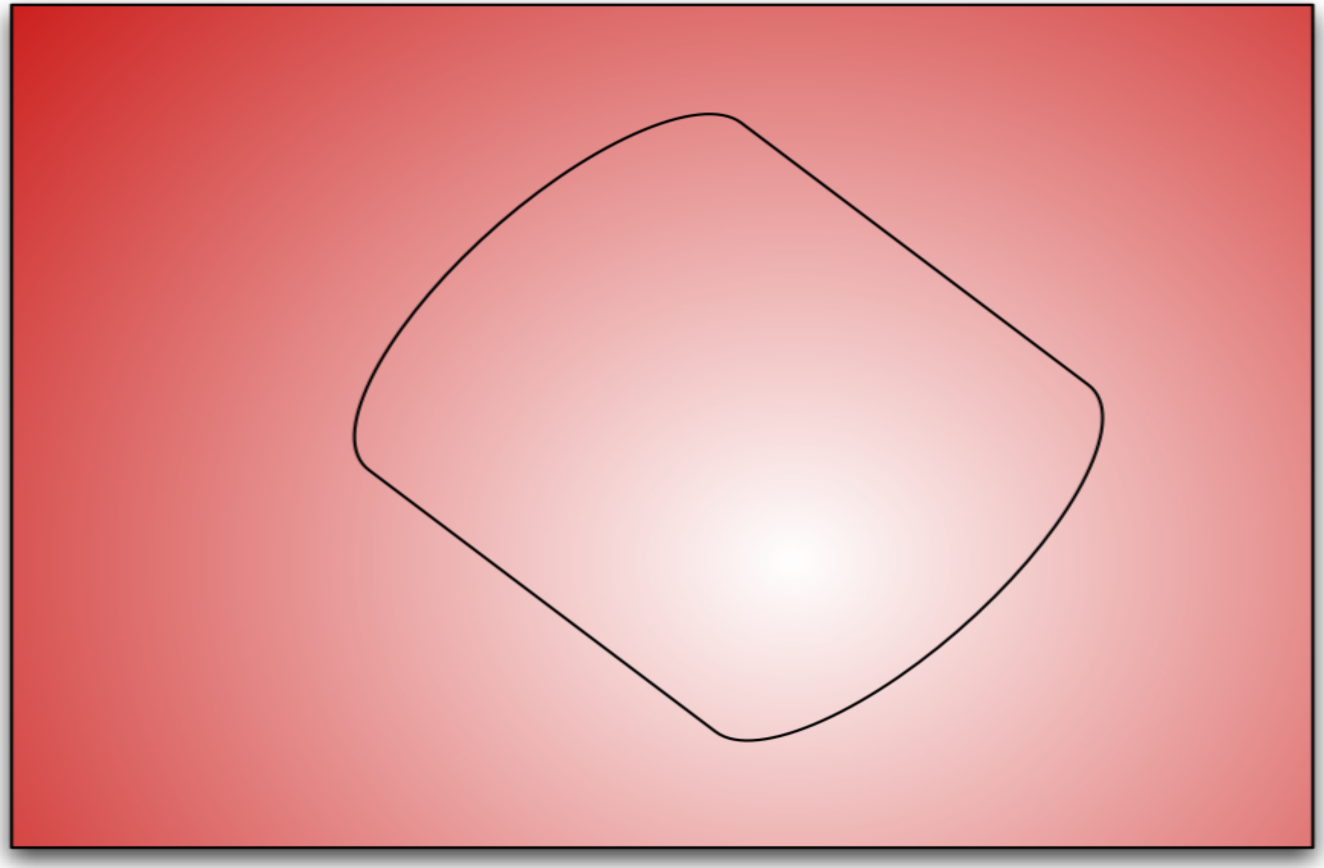
Repeat $x = x - \frac{f'(x)}{f''(x)}$ **until** $\|f'(x)\| \leq \epsilon$

Convergence of Newton Method

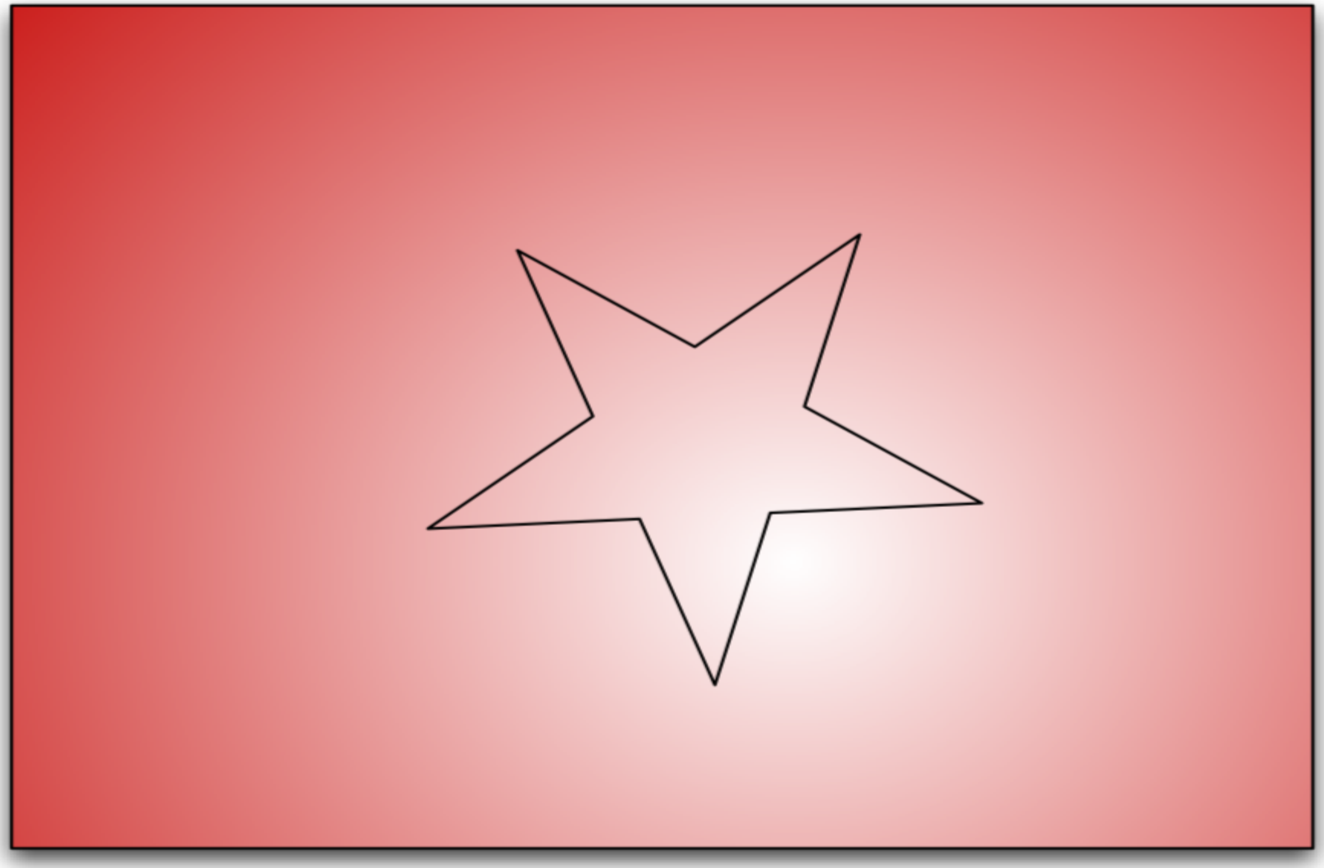
For some region around x^* it converges quadratically.



Convex Function on Convex Set



Convex Function on Non-convex Set



Constrained Optimization

Optimization Problem

$$\text{minimize } f(x) \text{ subject to } c_i(x) \leq 0 \text{ for all } i \in [n]$$

Here $c_i(x)$ and f are all **convex** functions.

Lagrange Function

Convert the constrained optimization problem into **saddlepoint problem** of the Lagrange function

$$L(x, \alpha) := f(x) + \sum_{i=1}^{n} \alpha_i c_i(x) \text{ where } \alpha_i \geq 0.$$

Key Theorem

The saddlepoint of $L(x, \alpha)$ is achieved at optimality $(\bar{x}, \bar{\alpha})$ of the original problem.

$$L(\bar{x}, \alpha) \leq L(\bar{x}, \bar{\alpha}) \leq L(x, \bar{\alpha})$$

Quadratic Programs

Primal Objective

$$\text{minimize } \frac{1}{2}x^\top Kx + c^\top x \text{ subject to } Ax + b \leq 0$$

Convexity

- Constraints are linear, hence they are convex.
- Objective function has derivatives

$$\partial_x[\ldots] = Kx + c$$
$$\partial_x^2[\ldots] = K$$

  This is convex whenever K has no negative eigenvalues (OK if K is a kernel matrix).

Good News

- Optimizers exist for this.
- SVM optimization problem looks exactly like that

Constrained Quadratic Program

Mini Summary

Convexity

- Definition
- Convex functions have unique minimum
- Solve by Newton method

Constraints

- Need convex constraints
- Lagrange function
- Saddlepoint property

Quadratic Programs

- Quadratic function in objective
- Linear constraints
- Solvers exist: CPLEX, YALMIP (great MATLAB frontend for lots of other solvers, plus good pointers), MATLAB solver (terrible performance)

Active Set Problem

Problem

- Optimization in all variables is really difficult
- Big problem, lots of variables, not enough memory, . . .

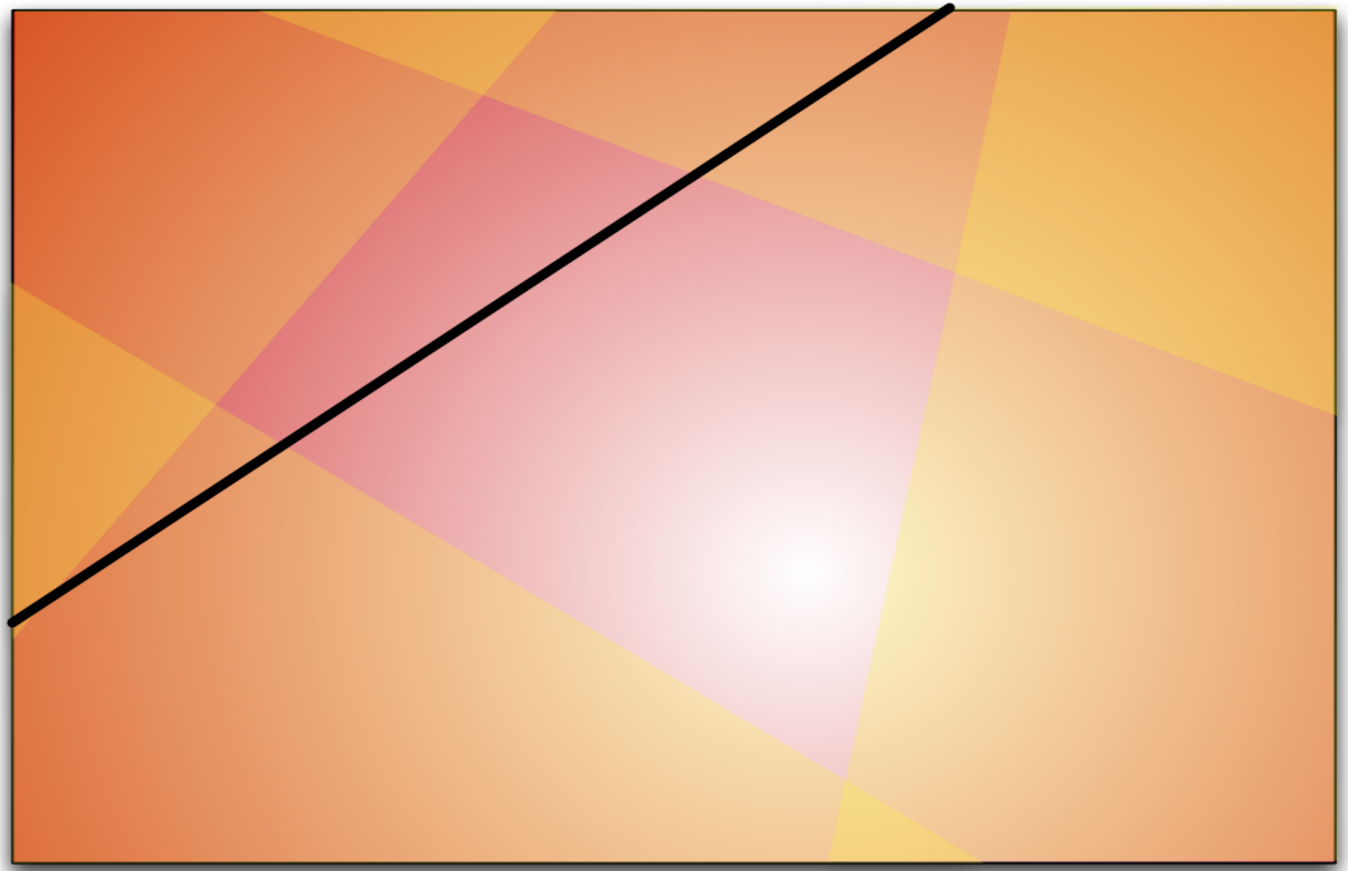
Idea

- Pick subset of variables (fix the rest) and minimize over them

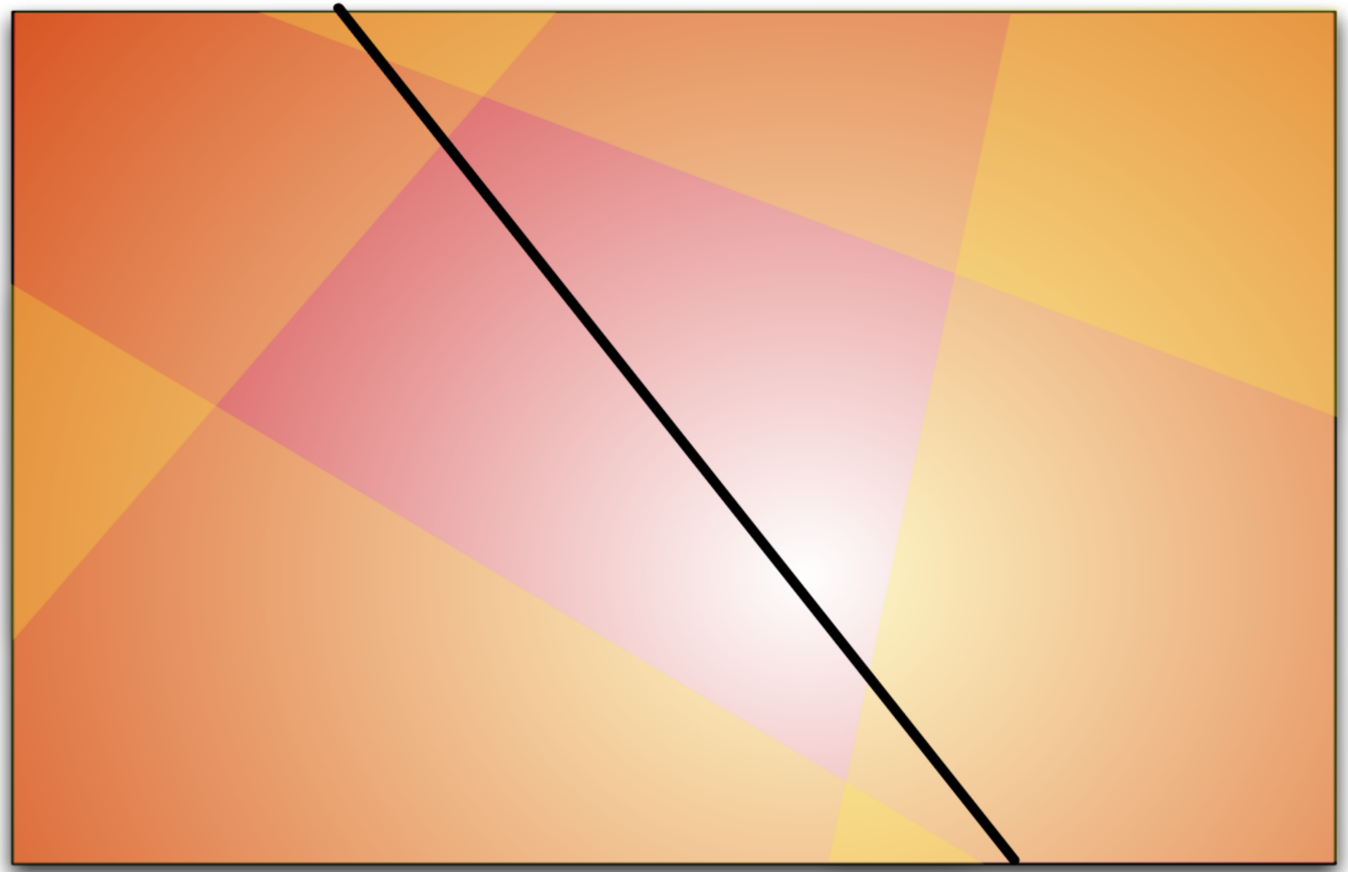
Result

- Smaller problem, few variables, sometimes can be solved in closed form.
- We always make progress
- Iterative procedure for minimization

Active Set Method



Active Set Method



Chunking

Full problem (using $\bar{K}_{ij} := y_i y_j k(x_i, x_j)$)

$$\text{minimize } \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j \bar{K}_{ij} - \sum_{i=1}^{m} \alpha_i$$

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

Constrained problem: pick subset S

$$\text{minimize } \frac{1}{2}\sum_{i,j \in S} \alpha_i \alpha_j \bar{K}_{ij} - \sum_{i \in S} \alpha_i \left[1 - \sum_{j \notin S} K_{ij} \alpha_j\right] + \text{const.}$$

$$\text{subject to } \sum_{i \in S} \alpha_i y_i = -\sum_{i \notin S} \alpha_i y_i \text{ and } \alpha_i \in [0, C] \text{ for all } i \in S$$

Chunking Variants

Simple Version

- Start with small set, train, keep SVs, add patterns
- Great for clean data, e.g. USPS and NIST OCR

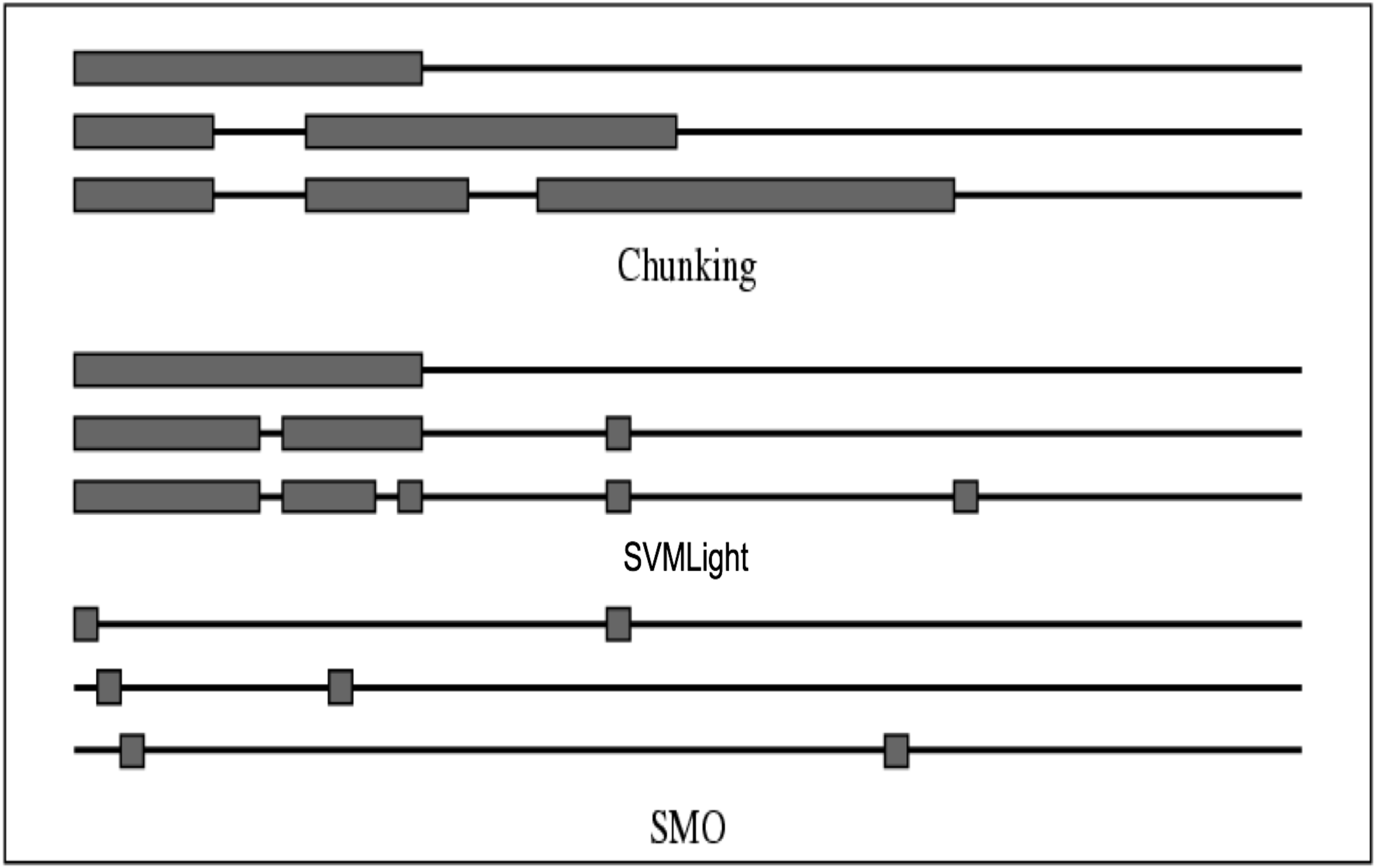
General Idea

- Take subset S of variables, optimize over them, then pick next set of variables, repeat.
- Good implementation is SVMLight. Works great on text.

Common Problems

- Convergence can be slow. **Highly problem dependent**.
- Which active set should you pick?
- Performance degrades with the number of additional linear constraints (e.g. ν-trick).

Chunking Strategies



Sequential Minimal Optimization (SMO)

Basic Idea

- Optimize only over pairs of variables.
- Need pairs to keep the equality constraints satisfied

Advantage

- Analytic solution of subproblems is possible
- Simple one-dimensional convex minimization problem

Scaling Behaviour

- Large problems solved at only $O(m)$ storage cost
- May need to wait for a long time (time scales with $O(m^\gamma)$ where $\gamma > 2$)

Problems

- Some formulations are hard to deal with in SMO, e.g. many nonzero start variables, several constraints at the same time (as in ν-SVM).

The ugly details

Quadratic function in one variable

- Minimize over x

$$f(x) = \frac{1}{2}ax^2 + bx + c$$

- Compute first derivative

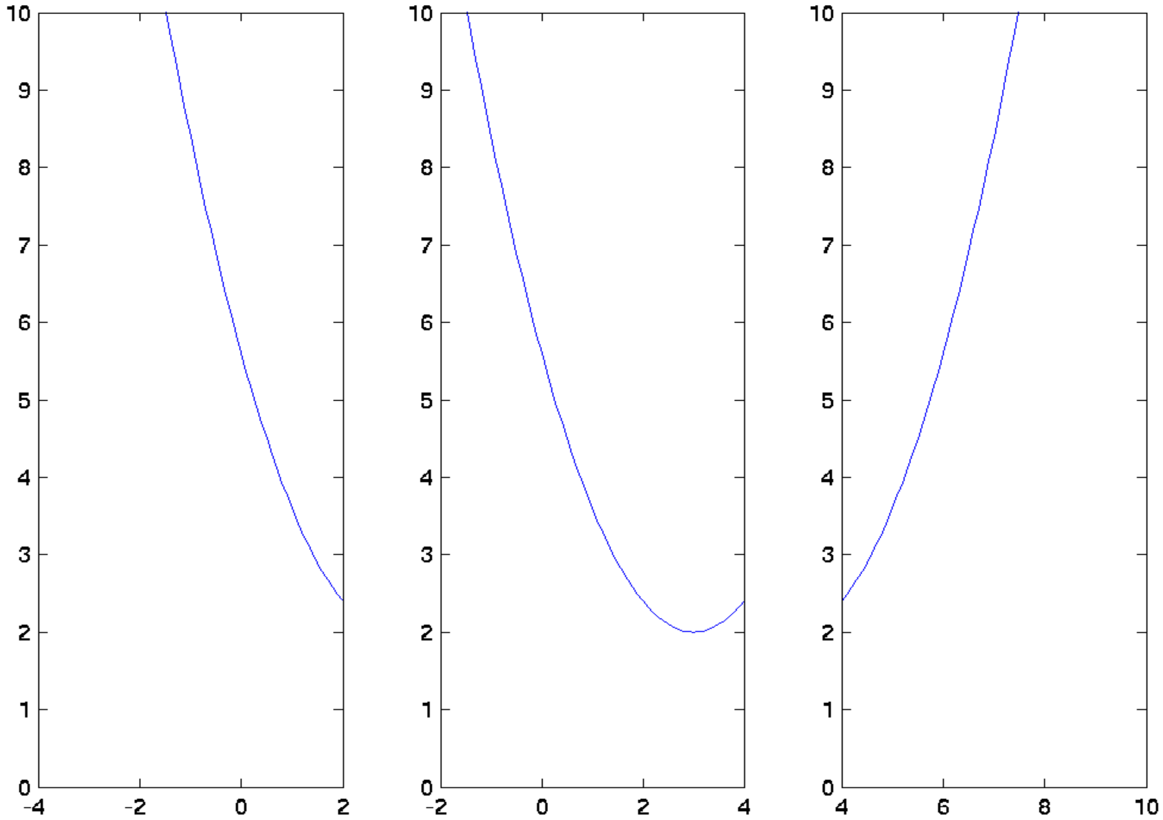
$$f'(x) = ax + b$$

and set it to zero $x = -b/a$

Quadratic function with constraints

- Same function as above, just with additional constraints $C_1 \leq x \leq C_2$.
- Case 1: $-b/a \leq C_1$. Here we pick $x = C_1$.
- Case 2: $C_1 \leq -b/a \leq C_2$. Here we pick $x = -b/a$.
- Case 3: $C_2 \leq -b/a$. Here we pick $x = C_2$.

Three cases



The ugly details

Optimization problem in two variables

$$\underset{\alpha_i,\alpha_j}{\text{minimize}} \ \frac{1}{2}\left[\alpha_i^2 Q_{ii} + \alpha_j^2 Q_{jj} + 2\alpha_i\alpha_j Q_{ij}\right] + c_i\alpha_i + c_j\alpha_j$$

$$\text{subject to } s\alpha_i + \alpha_j = \gamma$$

$$0 \leq \alpha_i \leq C_i \text{ and } 0 \leq \alpha_j \leq C_j.$$

Q, c, γ are obtained from kernel matrix via subproblem.

Key insight

- Constrained problem in two variables with linear constraint reduces to constrained problem in one variable without linear constraint.
- Can be solved by minimizing quadratic function.
- Details see Platt, 1998 or Schölkopf and Smola, 2002

The very ugly details

A Warmup

- Constraints

	$y_i = y_j$	$y_i \neq y_j$
L	$\max(0, \alpha_i^{\text{old}} + \alpha_j^{\text{old}} - C_j)$	$\max(0, \alpha_i^{\text{old}} - \alpha_j^{\text{old}})$
H	$\min(C_i, \alpha_i^{\text{old}} + \alpha_j^{\text{old}})$	$\min(C_i, C_j + \alpha_i^{\text{old}} - \alpha_j^{\text{old}})$

- More definitions

$$\chi := K_{ii} + K_{jj} - 2sK_{ij} \text{ where } s = y_i y_j$$
$$\delta := y_i((f(x_j) - y_j) - (f(x_i) - y_i))$$

Unconstrained Solution

$$\bar{\alpha} = \alpha_i^{\text{old}} + \chi^{-1}\delta \text{ if } \chi \neq 0 \text{ otherwise } \bar{\alpha} = -\operatorname{sgn}(\delta)\infty.$$

Truncated solution

$$\alpha_i = \min(\max(\bar{\alpha}, L, H)) \text{ and } \alpha_j = s(\alpha_i^{\text{old}} - \alpha_i) - \alpha_j^{\text{old}}.$$

Selecting Points

Major Loop

- Loop through data cyclically until all data approximately satisfies optimality conditions:
- $\alpha_i = 0 \implies x_i$ is correct with margin at least 1.
- $\alpha_i = C \implies x_i$ is on the margin or a margin error.
- $0 < \alpha_i < C \implies x_i$ is on the margin.

Selection of second point

- Errors should be balanced, to make step large:

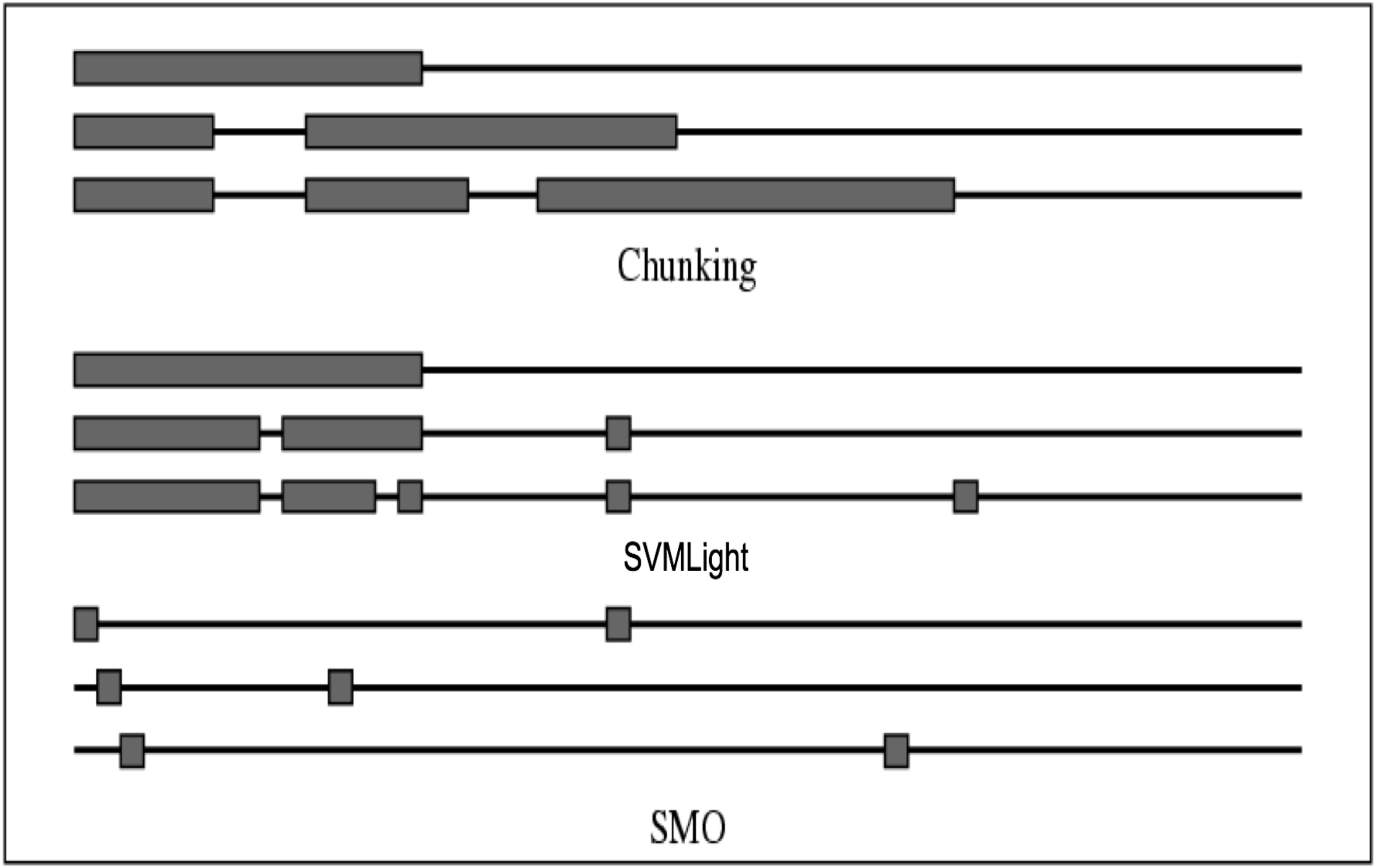
$$f(x_j) - y_j - f(x_i) + y_i$$

- Determinant should be small $(K_{ii} + K_{jj} - 2sK_{ij})$. But that is expensive to check.

Important Trick

Cache function values $f(x_i)$ and update them when the α_i, α_j change.

Recall: Chunking Strategies



SVMLight

Modification

- Take more than 2 variables at a time
- Invoke an off-the-shelf optimizer on the small problems

Selection Strategy

- Pick points for which α_i and the position wrt. the margin do not match.
- Balance selection such that errors are evenly distributed, i.e. margin errors with $\alpha_i < C$ and correct points with $\alpha_i > 0$.
- Cycle through data.

Convergence

Can be shown, see Thorsten Joachims or Chi-Jen Lin.

Mini Summary

Chunking

- Pick subset of the problem and solve.
- Smaller problem is easier to solve.
- Need to iterate

Sequential Minimal Optimization (SMO)

- Pick only two variables at a time
- Solve small problem analytically
- Pick balanced pair (outer loop sweeping through data, inner loop, looking for maximum discrepancy)
- Easy to implement
- Small storage requirement

Other Chunking Variants

- SVMLight (picks larger numbers of variables at a time)
- Simple chunking (adds support vectors as you go)

Gradient Descent

Objective Function

Some function $f : \mathbb{R}^n \to \mathbb{R}$.

Gradient Descent

initial value x_0, learning rate λ

repeat

$$x_{i+1} = x_i - \lambda \nabla f(x_i)$$

until $\|\nabla f(x_{i+1})\| \leq \epsilon$

Find direction of steepest descent, take a step, repeat.

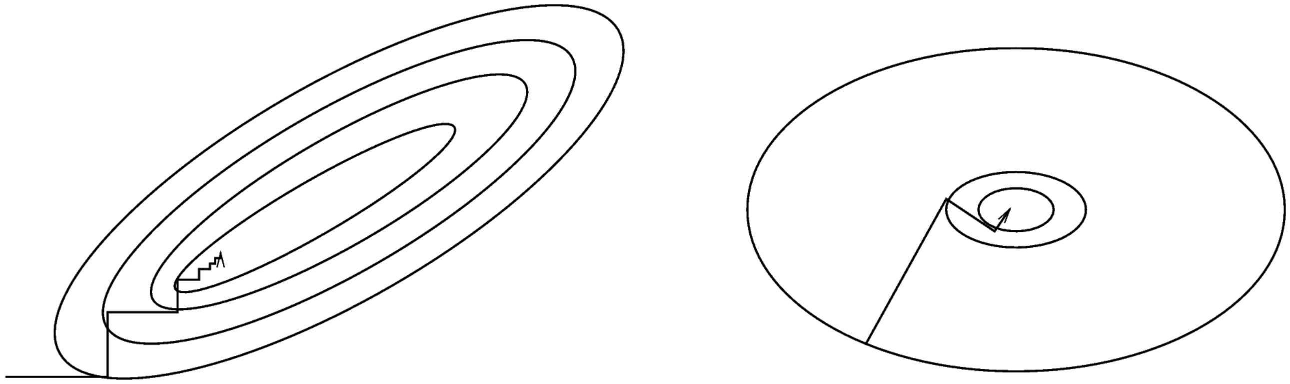
Line Search Variant

Replace the update by

$$x_{i+1} = x_i - \hat{\lambda} \nabla f(x_i) \text{ where } \hat{\lambda} = \operatorname*{argmin}_{\lambda} f(x_i - \lambda \nabla f(x_i))$$

Find direction of steepest descent, walk downhill until it goes uphill again, repeat.

Problems with Gradient Descent



Left

- Gradient descent takes a long time to converge
- Gets trapped in a long and narrow valley (zig-zagging along the walls of the valley).

Right

- Homogeneous structure of the objective function
- Gradient descent converges quickly

Fixing It

Conjugate Directions

- Distort the space such that the coordinates become homogeneous.
- Do that in an on-line fashion.
- Conjugate gradient descent does that (used a lot for minimizing quadratic functions).
- If function is not quadratic, need to restart periodically.

Stochastic Gradient Descent

- Use noisy estimates of gradient
- Cheaper to compute (if overall gradient is average of terms)
- Inherent noise gets it out of (not too big) local minima
- Often still convergent

Stochastic Gradient Descent

Stochastic Approximation

- Function $f : \mathfrak{X} \to \mathbb{R}$ made up of many individual terms

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

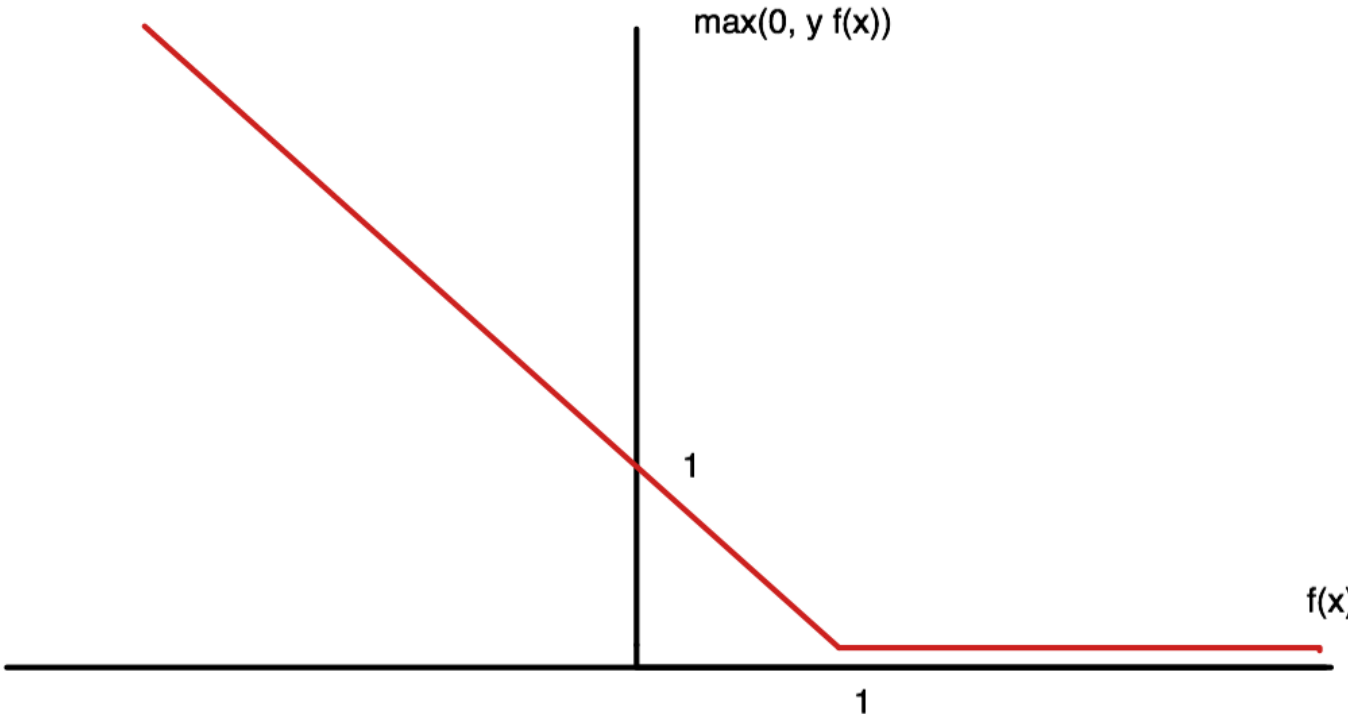
- Randomly select one f_j at a time and perform gradient descent with respect to f_i.
- Update rule

$$x_{i+1} = x_i - \lambda \nabla f_j(x)$$

Advantage

- Much cheaper to compute than ∇f
- If all f_i are somewhat similar less wasteful.
- Use f_i as loss functions

Margin Loss Function



Margin loss $\max(0, y(\langle w, \phi(x) \rangle + b))$

SVM Online Learning

Rewriting the SVM problem

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

Perform stochastic approximation

Replace sum by single term

Compute gradient w.r.t. stochastic approximation

$$\max(0, 1 - y_i(\langle\phi(x_i), \theta\rangle + b) + \frac{\lambda}{2}\|w\|^2$$

$$\partial_w[\ldots] = \lambda w - \begin{cases} y_i\phi(x_i) & \text{if } y_i\left(\langle\phi(x_i), \theta\rangle + b\right) < 1 \\ 0 & \text{otherwise} \end{cases}$$

SVM Online Learning

Kernel Expansion

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

Update in Coefficient Space

$$\begin{aligned} \alpha_t &= -\eta \begin{cases} y_i & \text{if } y_i \left(\langle \phi(x_i), \theta \rangle + b \right) < 1 \\ 0 & \text{otherwise} \end{cases} \\ \alpha_i &= (1 - \eta\lambda)\alpha_i \text{ for } i < t \end{aligned}$$

Finite Time Horizon

- $\frac{\lambda}{2}\|w\|^2$ ensures that coefficients decay over time
- Drop α_i after t steps with error at most $(1 - \eta\lambda)^t \sqrt{k(x_i, x_i)}$.
- Learning rate η and regularization λ govern length of history.

Mini Summary

Problems with gradient descent

- Expensive to compute
- May not converge quickly
- Long valley problem

Stochastic Approximation

- Take only one loss term at a time
- Perform update in this direction
- Quadratic penalty bounds the time horizon
- Decrease learning rate for convergence

Kernel expansion

- Kernels allow for efficient computation (no feature space needed)
- Only store α_i for margin errors

Summary

Convex Optimization Basics

- Convex functions
- Optimality and uniqueness
- Subspace descent
- Numerical math basics

Sequential Minimal Optimization and Chunking

- Chunking
- Explicit solution
- Selection strategy

Stochastic gradient descent

- Basic idea
- Online SVM
- Further applications

An Introduction to Machine Learning with Kernels

Lecture 7

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

Statistical Machine Learning Program
National ICT Australia, Canberra

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

L7 Novelty Detection and Regression

Novelty Detection

- Basic idea
- Optimization problem
- Stochastic Approximation
- Examples

LMS Regression

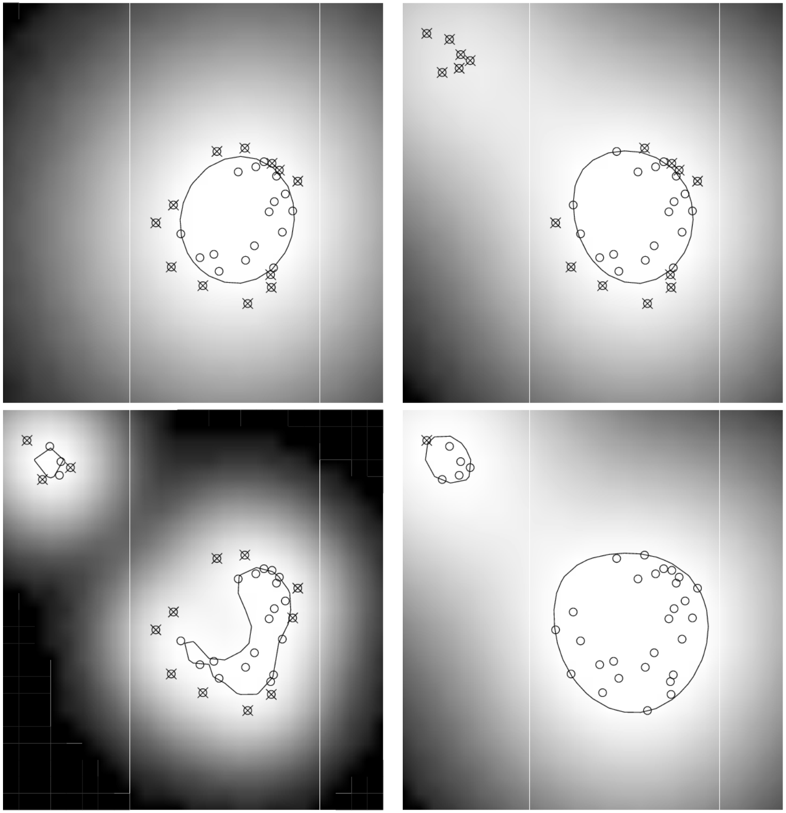
- Additive noise
- Regularization
- Examples
- SVM Regression

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

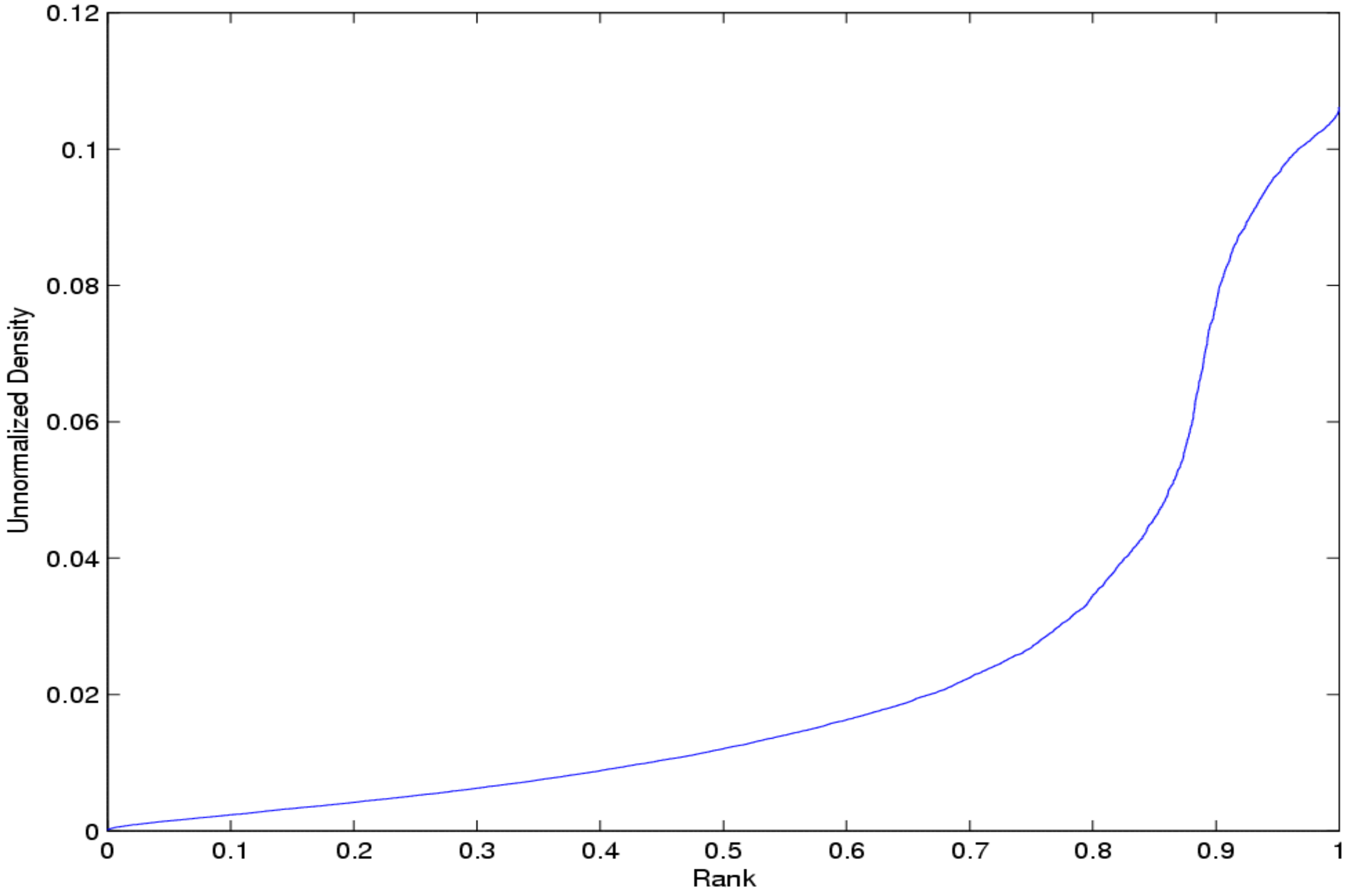
Step 2: Thresholding the density

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

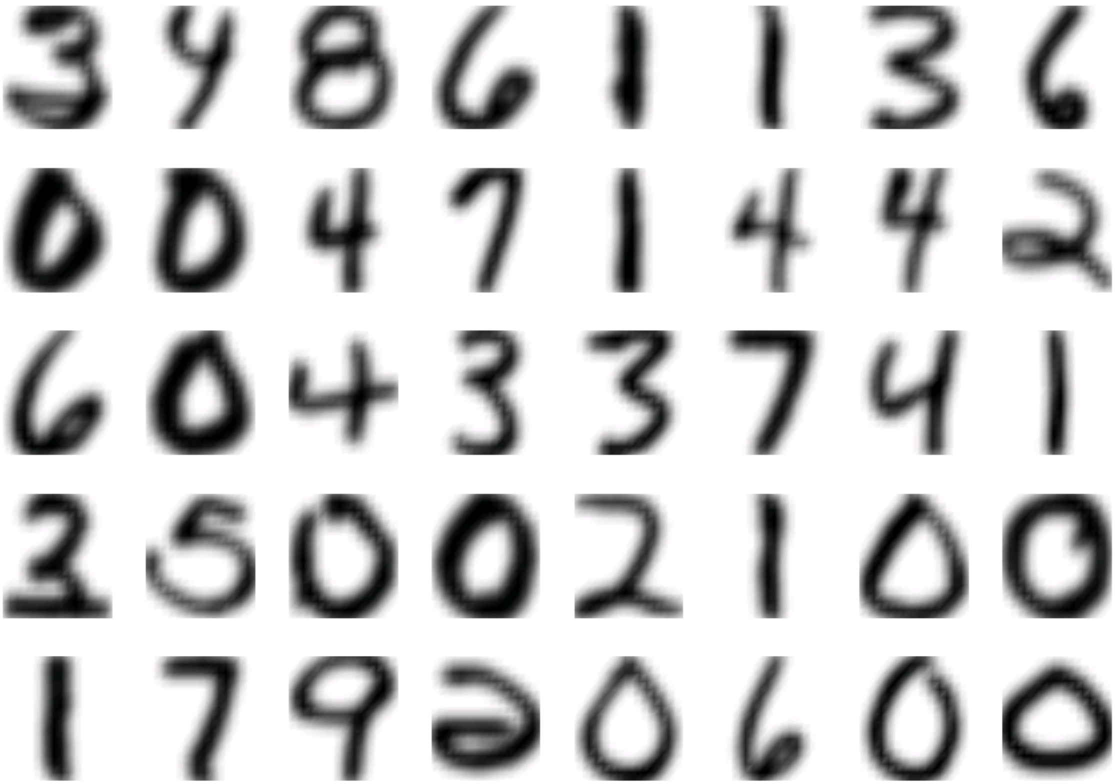
$$p(x_i) = \frac{1}{m} \sum_j k(x_i, x_j) \text{ for all } i$$

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

Order Statistic of Densities



Typical Data



Outliers

A better way . . .

Problems

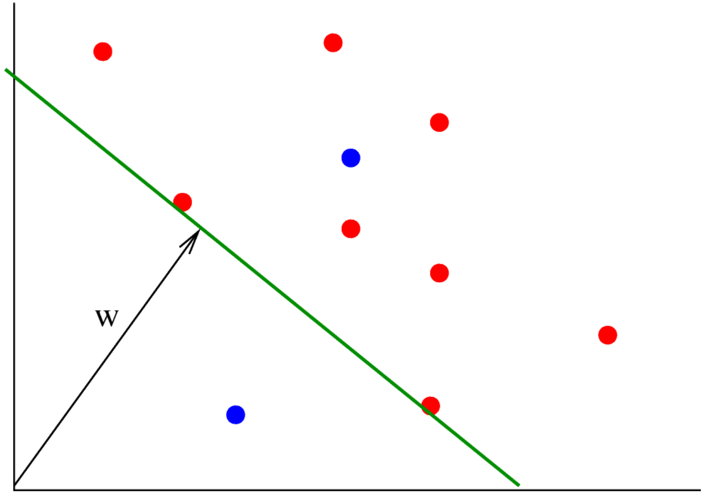
- We do not care about estimating the density properly in regions of high density (waste of capacity).
- We only care about the relative density for thresholding purposes.
- We want to eliminate a certain fraction of observations and tune our estimator specifically for this fraction.

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

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

Dual Problem

Primal Problem

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

$$\text{subject to} \quad \langle w, x_i\rangle - 1 + \xi_i \geq 0 \text{ and } \xi_i \geq 0$$

Lagrange Function L

- Subtract constraints, multiplied by Lagrange multipliers (α_i and η_i), from Primal Objective Function.
- Lagrange function L has **saddlepoint** at optimum.

$$L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left(\langle w, x_i\rangle - 1 + \xi_i\right) - \sum_{i=1}^{m}\eta_i\xi_i$$

subject to $\alpha_i, \eta_i \geq 0$.

Dual Problem

Optimality Conditions

$$\begin{aligned}\partial_w L &= w - \sum_{i=1}^m \alpha_i x_i = 0 \implies w = \sum_{i=1}^m \alpha_i x_i \\ \partial_{\xi_i} L &= C - \alpha_i - \eta_i = 0 \implies \alpha_i \in [0, C]\end{aligned}$$

Now **substitute** the optimality conditions **back into** L.

Dual Problem

$$\begin{aligned}\text{minimize} \quad & \frac{1}{2}\sum_{i=1}^m \alpha_i \alpha_j \langle x_i, x_j \rangle - \sum_{i=1}^m \alpha_i \\ \text{subject to} \quad & \alpha_i \in [0, C]\end{aligned}$$

All this is only possible due to the convexity of the primal problem.

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

$$\text{minimize } \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \xi_i - m\nu\rho$$
$$\text{where } \langle w, x_i \rangle - \rho + \xi_i \geq 0$$
$$\xi_i \geq 0$$

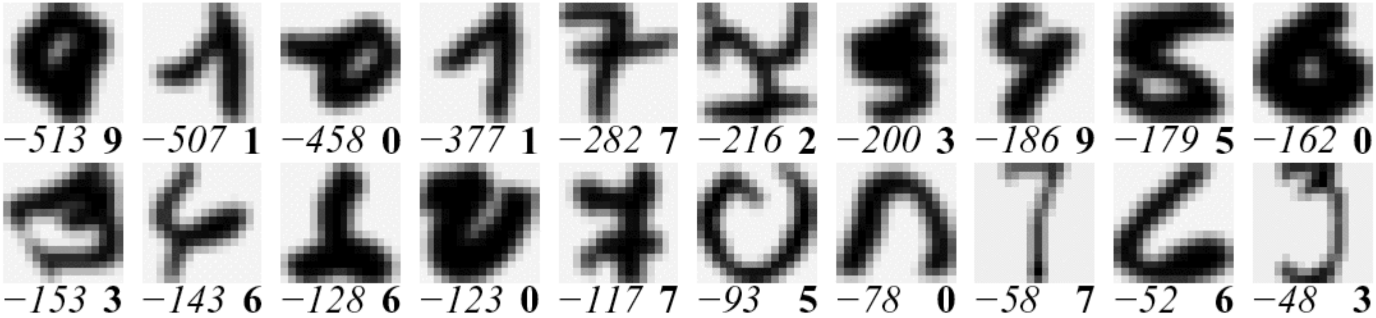
Dual Problem

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

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

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

$$\partial_w[\ldots] = \begin{cases} w - \phi(x_i) & \text{if } \langle w, \phi(x_i)\rangle < \rho \\ 0 & \text{otherwise} \end{cases}$$

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

Practical Implementation

Update in coefficients

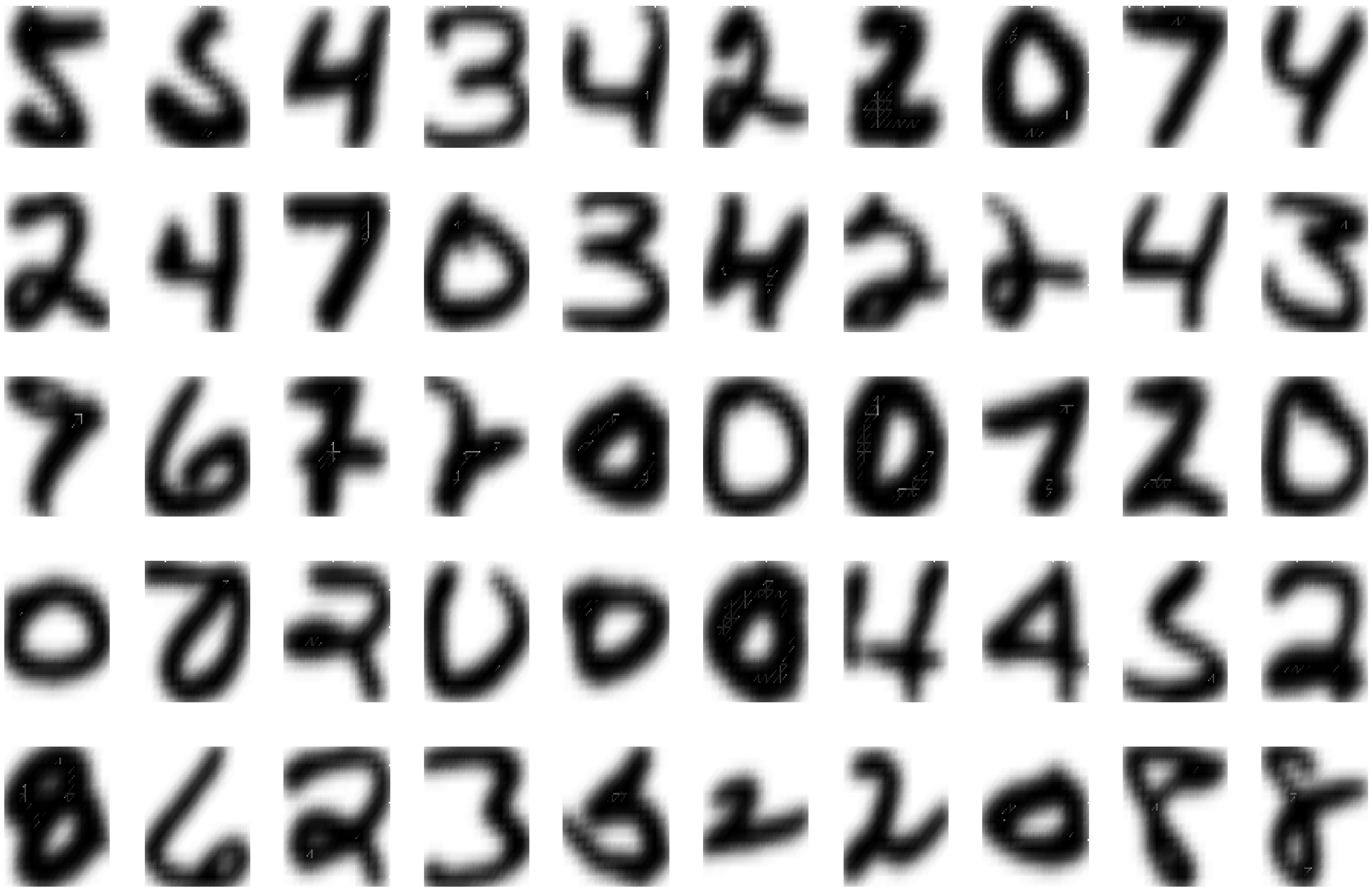
$$\alpha_j \leftarrow (1-\eta)\alpha_j \text{ for } j \neq i$$

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

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

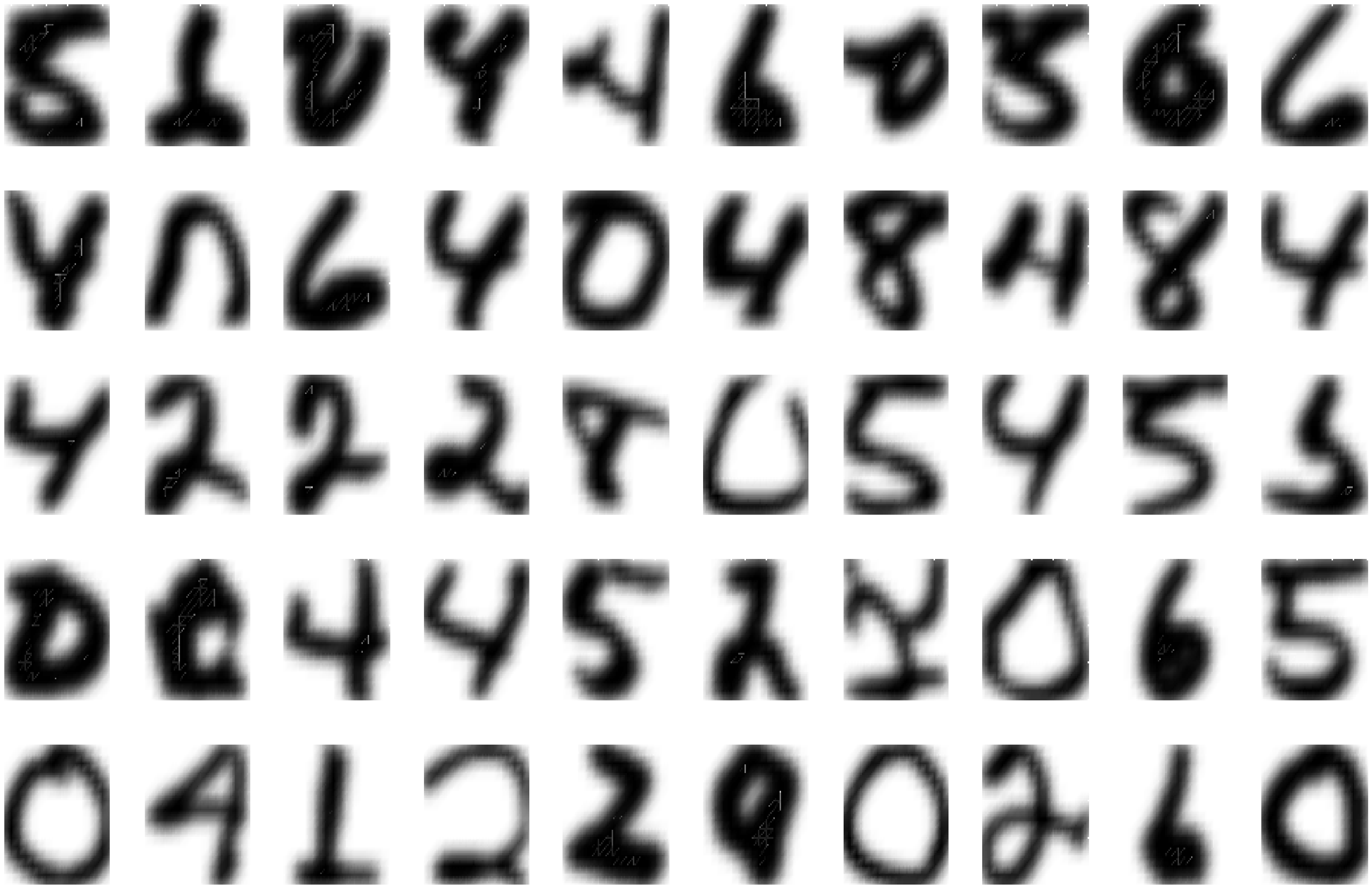
Using learning rate η.

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

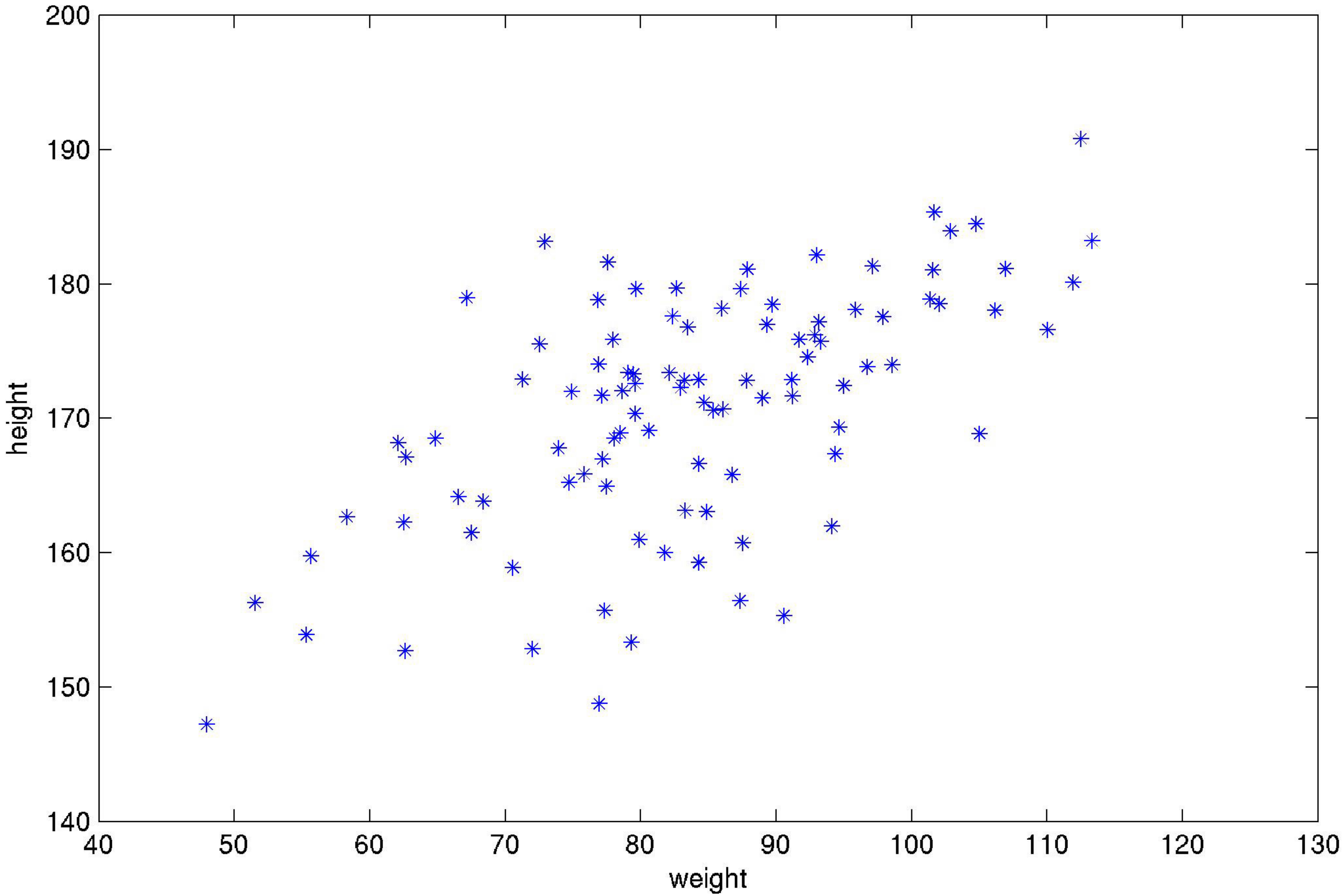
Novelty Detection via SVM

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

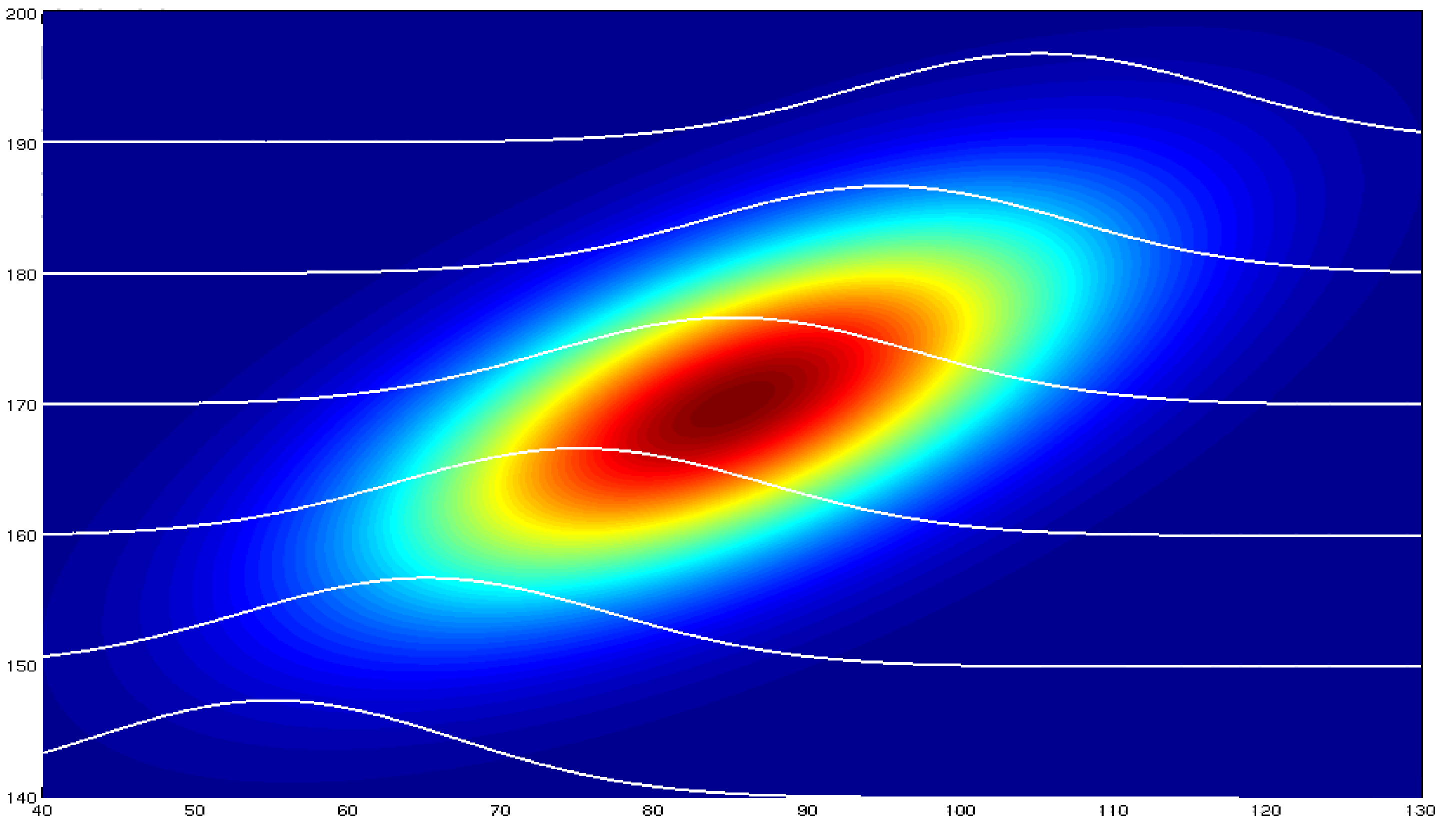
Online Version

- 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

Conditional probability

- If we have conditional probability $p(y|x)$ we can estimate y (here x are the observations and y is what we want to compute).
- For instance, we can get the regression by computing the mean of $p(y|x)$.

Joint to conditional probability

- Joint can be used to get conditional, via Bayes rule

$$p(x, y) = p(y|x)p(x) \text{ and hence } p(y|x) = \frac{p(x, y)}{p(x)} \propto p(x, y)$$

- Expression only depends on y for fixed x in $p(x, y)$.

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

Normal Distribution in $\mathbb{R}$

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

with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$.

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. Note that this is now a matrix.
- Σ has only nonnegative eigenvalues (i.e. the variance is never negative).

Inference in Normal Distributions

Correlated Observations

Assume that the random variables $t \in \mathbb{R}^n, t' \in \mathbb{R}^{n'}$ are jointly normal with mean (μ, μ') and covariance matrix K

$$p(t, t') \propto \exp\left(-\frac{1}{2}\begin{bmatrix} t-\mu \\ t'-\mu' \end{bmatrix}^\top \begin{bmatrix} K_{tt} & K_{tt'} \\ K_{tt'}^\top & K_{t't'} \end{bmatrix}^{-1} \begin{bmatrix} t-\mu \\ t'-\mu' \end{bmatrix}\right).$$

Inference

Given t, estimate t' via $p(t'|t)$. Translation into machine learning language: we learn t' from t.

Practical Solution

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

$$\tilde{K} = K_{t't'} - K_{tt'}^\top K_{tt}^{-1} K_{tt'} \text{ and } \tilde{\mu} = \mu' + K_{tt'}^\top \underbrace{\left[K_{tt}^{-1}(t-\mu)\right]}_{\text{independent of } t'}$$

Gaussian Process

Key Idea

Instead of a fixed set of random variables t, t' we assume a stochastic process $t : \mathcal{X} \to \mathbb{R}$, e.g. $\mathcal{X} = \mathbb{R}^n$.

Previously we had $\mathcal{X} = \{\text{age}, \text{height}, \text{weight}, \ldots\}$.

Definition of a Gaussian Process

A stochastic process $t : \mathcal{X} \to \mathbb{R}$, where all $(t(x_1), \ldots, t(x_m))$ are normally distributed.

Parameters of a GP

Mean $\quad \mu(x) := \mathbf{E}[t(x)]$

Covariance Function $\quad k(x, x') := \mathrm{Cov}(t(x), t(x'))$

Simplifying Assumption

We assume knowledge of $k(x, x')$ and set $\mu = 0$.

Some Covariance Functions

Observation

Any function k leading to a symmetric matrix with non-negative eigenvalues is a valid covariance function.

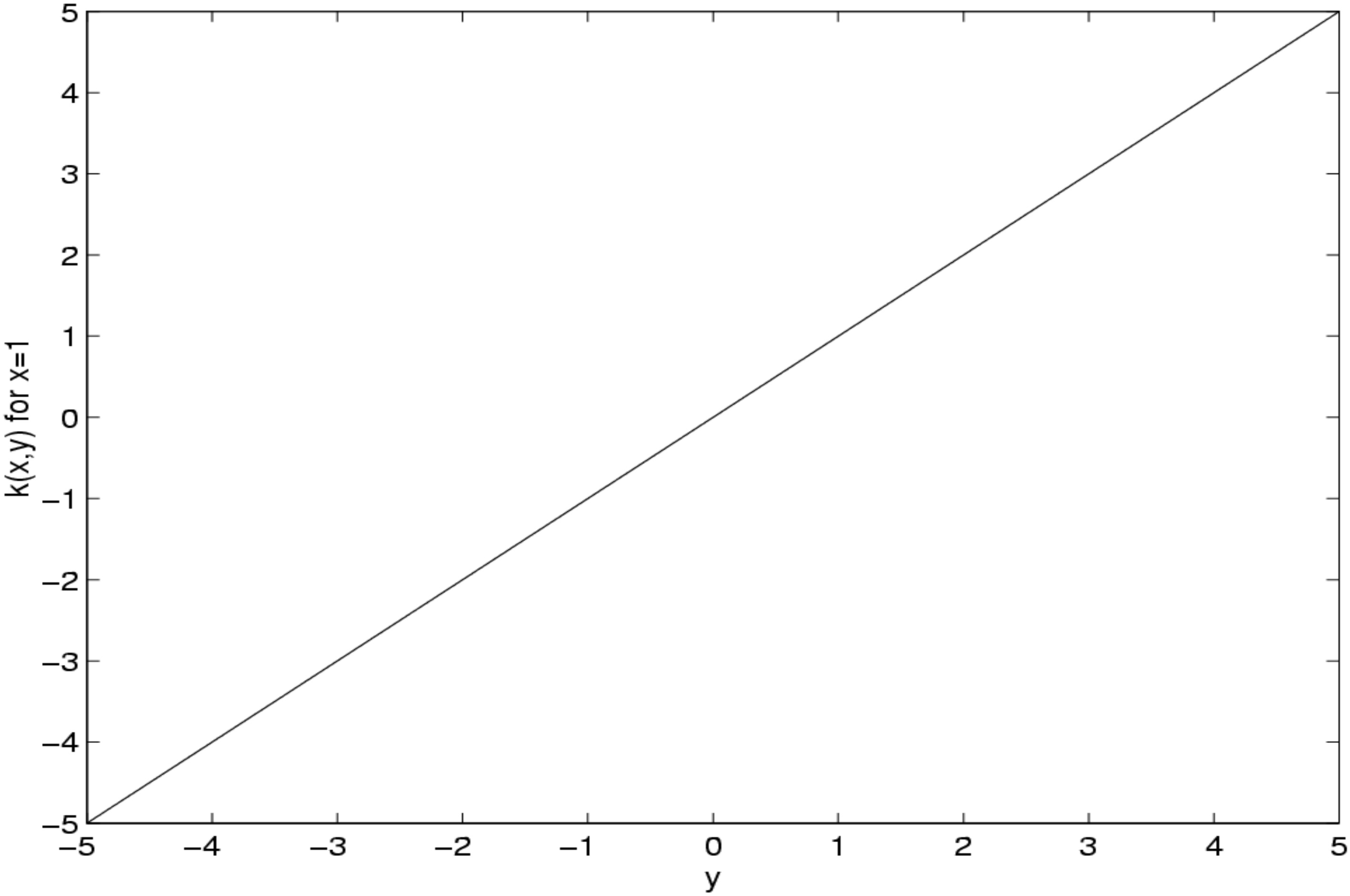
Necessary and sufficient condition (Mercer's Theorem)

k needs to be a nonnegative integral kernel.

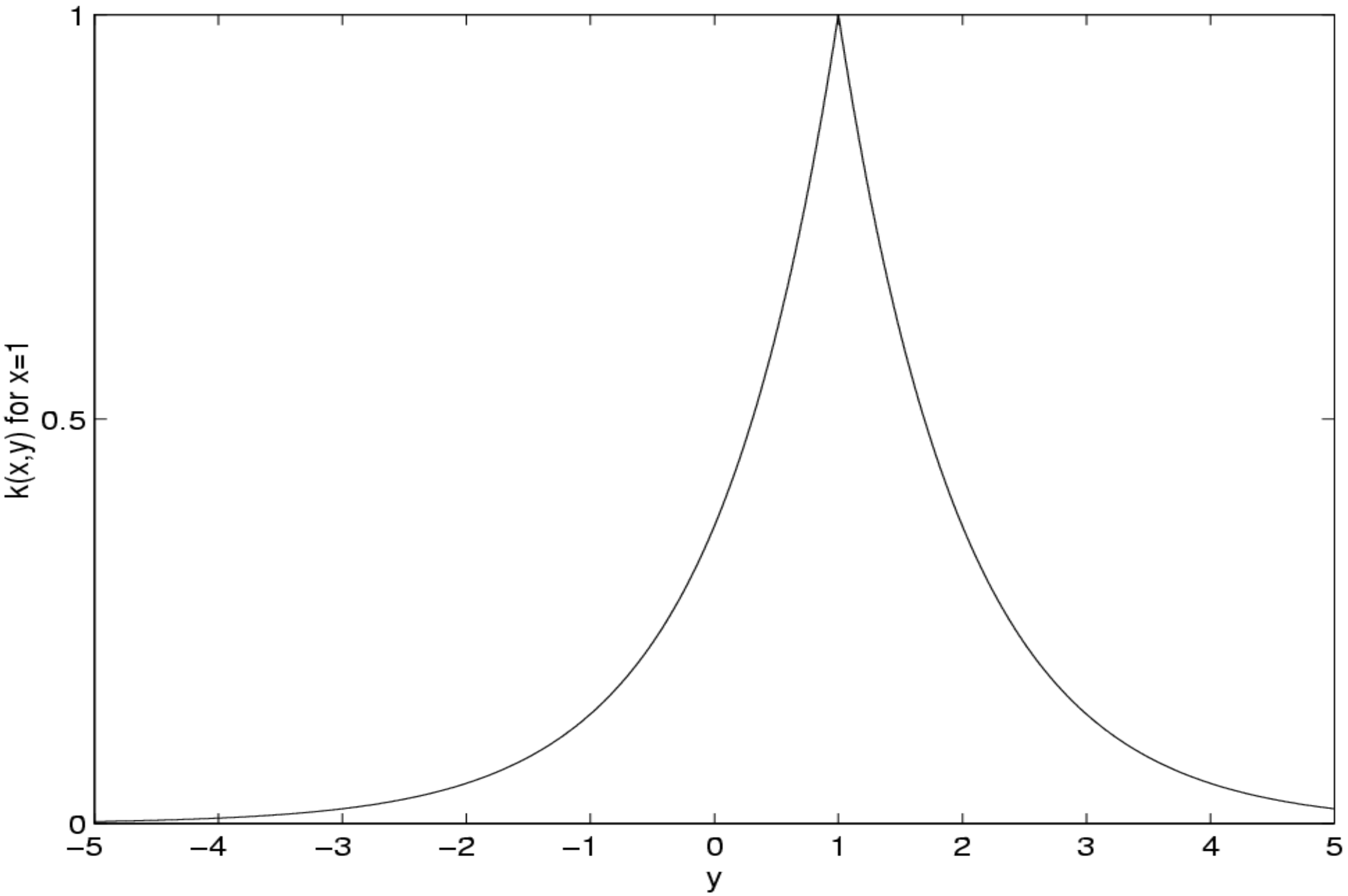
Examples of kernels $k(x, x')$

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	$\mathbf{E}_c[p(x\|c)p(x'\|c)]$

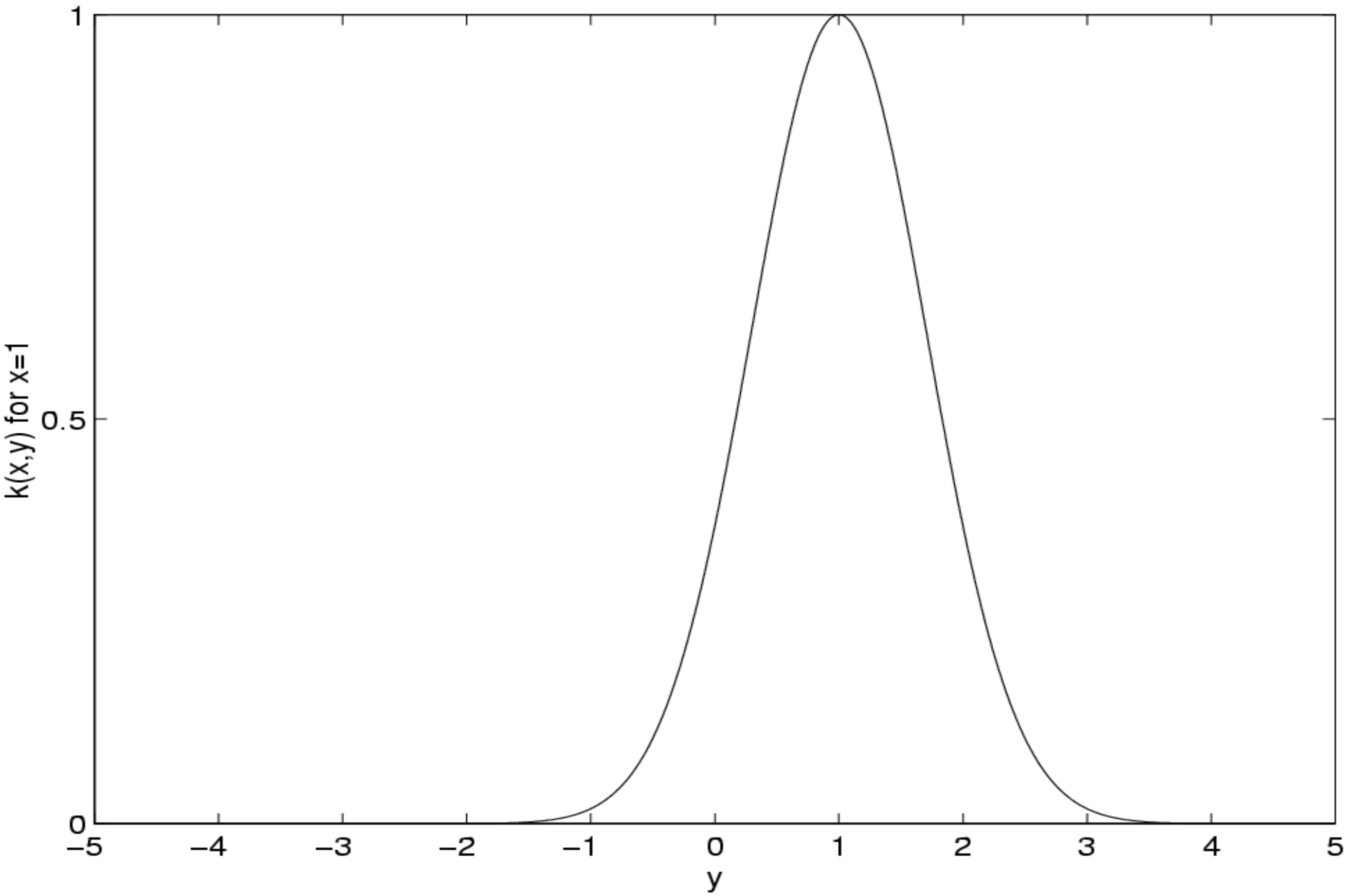
Linear Covariance



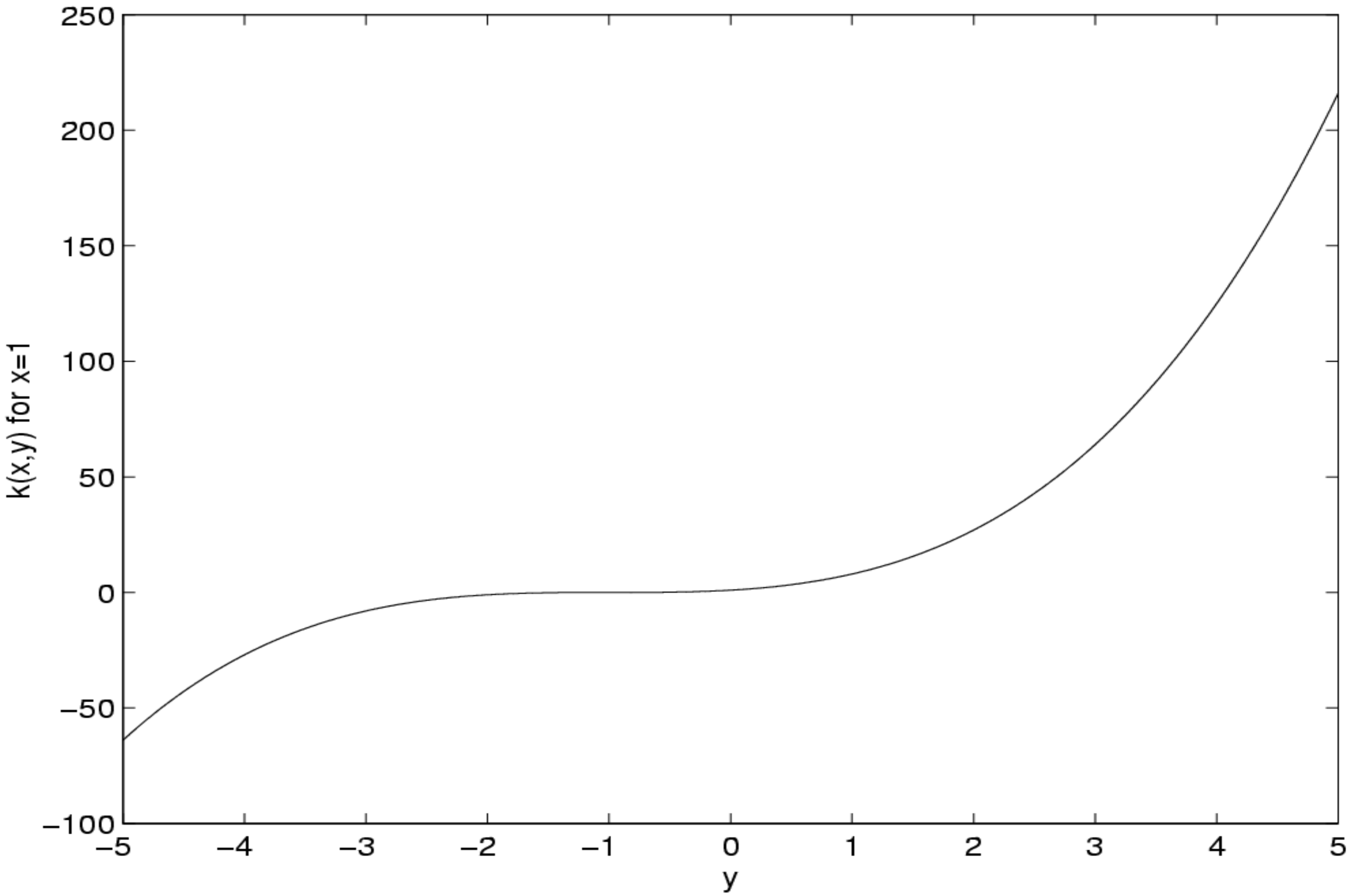
Laplacian Covariance



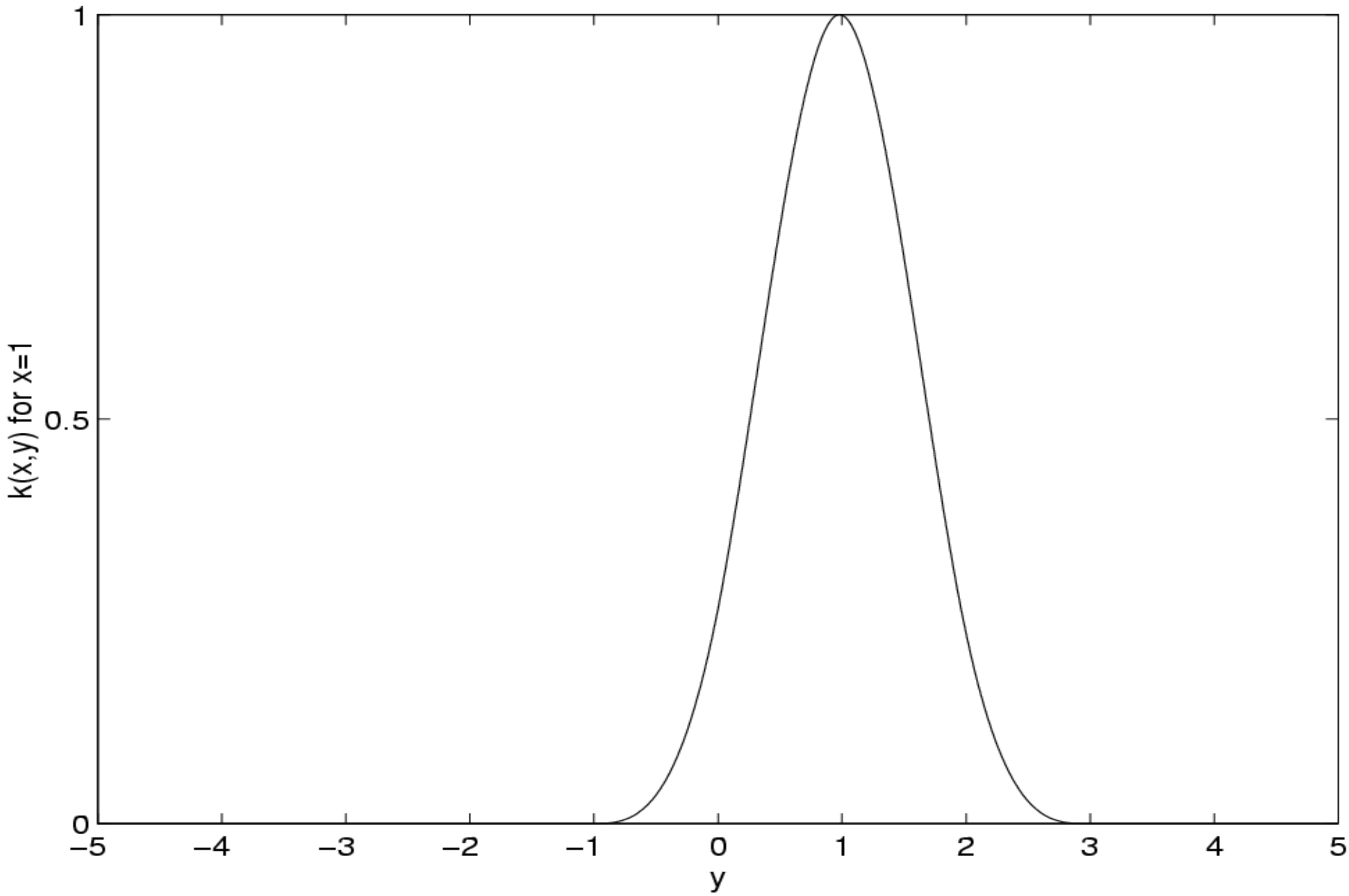
Gaussian Covariance



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 . . .

The Support Vector Connection

Gaussian Process on Parameters

$$t \sim \mathcal{N}(\mu, K) \text{ where } K_{ij} = k(x_i, x_j)$$

Linear Model in Feature Space

$$t(x) = \langle \Phi(x), w \rangle + \mu(x) \text{ where } w \sim \mathcal{N}(0, \mathbf{1})$$

The covariance between $t(x)$ and $t(x')$ is then given by

$$\mathbf{E}_w \left[\langle \Phi(x), w \rangle \langle w, \Phi(x') \rangle \right] = \langle \Phi(x), \Phi(x') \rangle = k(x, x')$$

Conclusion

A small weight vector in "feature space", as commonly used in SVM amounts to observing t with high $p(t)$.

Log prior $-\log p(t) \iff$ **Margin** $\|w\|^2$

Will get back to this later again.

Regression

Simple Model

Assume correlation between $t(x)$ and $t(x')$ via $k(x, x')$, so we can perform regression on $t(x')$, given $t(x)$.

Recall

$$p(t, t') \propto \exp\left(-\frac{1}{2}\begin{bmatrix} t \\ t' \end{bmatrix}^\top \begin{bmatrix} K_{tt} & K_{tt'} \\ K_{tt'}^\top & K_{t't'} \end{bmatrix}^{-1} \begin{bmatrix} t \\ t' \end{bmatrix}\right)$$

yields $t'|t \sim \mathcal{N}(\tilde{\mu}, \tilde{K})$, where

$$\tilde{K} = K_{t't'} - K_{tt'}^\top K_{tt}^{-1} K_{tt'} \text{ and } \tilde{\mu} = K_{tt'}^\top K_{tt}^{-1} t$$

Proof Idea

- $t'|t$ is normally distributed, hence we need only get the linear and quadratic terms in t'.
- Quadratic term via inverse of big covariance matrix.
- Linear term (for the mean) has cross terms with t.

Example: Linear Regression

Linear kernel: $k(x, x') = \langle x, x' \rangle$

- Kernel matrix $X^\top X$
- Mean and covariance

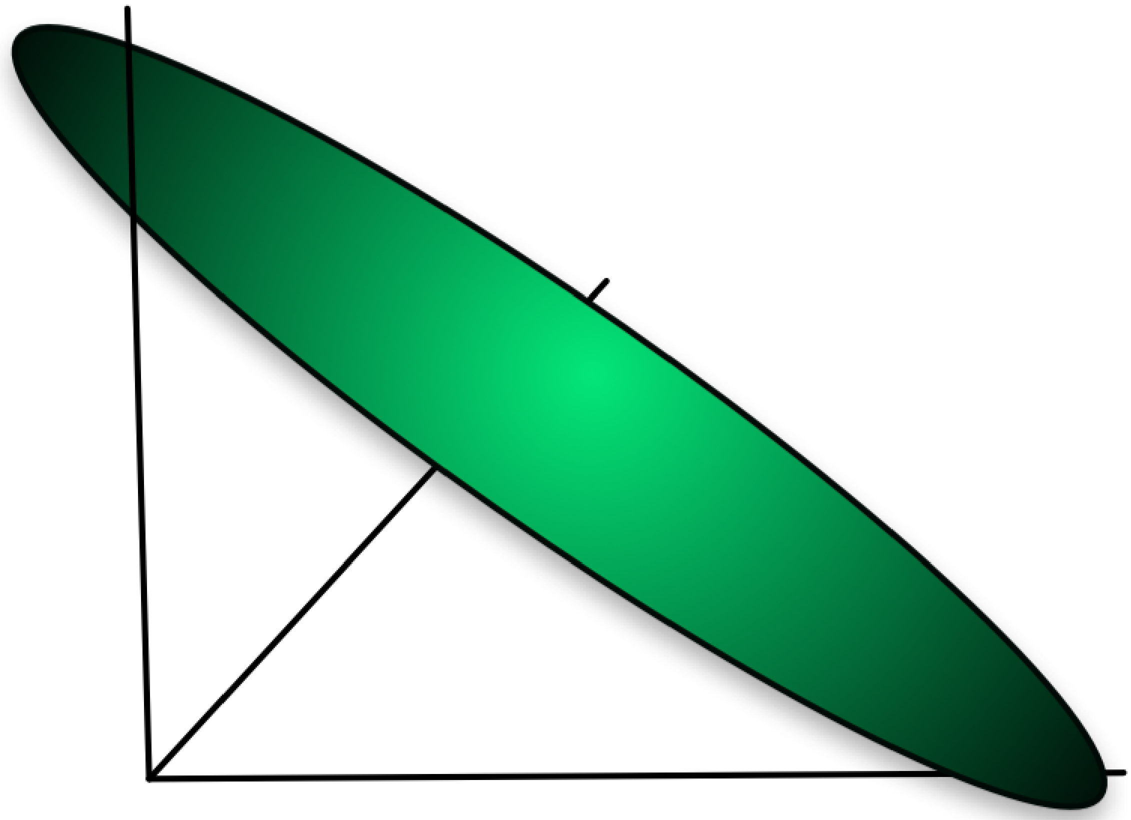
$$\tilde{K} = X'^\top X' - X'^\top X (X^\top X)^{-1} X^\top X' = X'^\top (\mathbf{1} - P_X) X'.$$
$$\tilde{\mu} = X'^\top \left[X (X^\top X)^{-1} t \right]$$

- $\tilde{\mu}$ is a **linear function of** X'.

Problem

- The covariance matrix $X^\top X$ has at most rank n.
- After n observations ($x \in \mathbb{R}^n$) the **variance vanishes**. This is **not realistic**.
- "Flat pancake" or "cigar" distribution.

Degenerate Covariance



Additive Noise

Indirect Model

Instead of observing $t(x)$ we observe $y = t(x) + \xi$, where ξ is a nuisance term. This yields

$$p(Y|X) = \int \prod_{i=1}^{m} p(y_i|t_i) p(t|X) dt$$

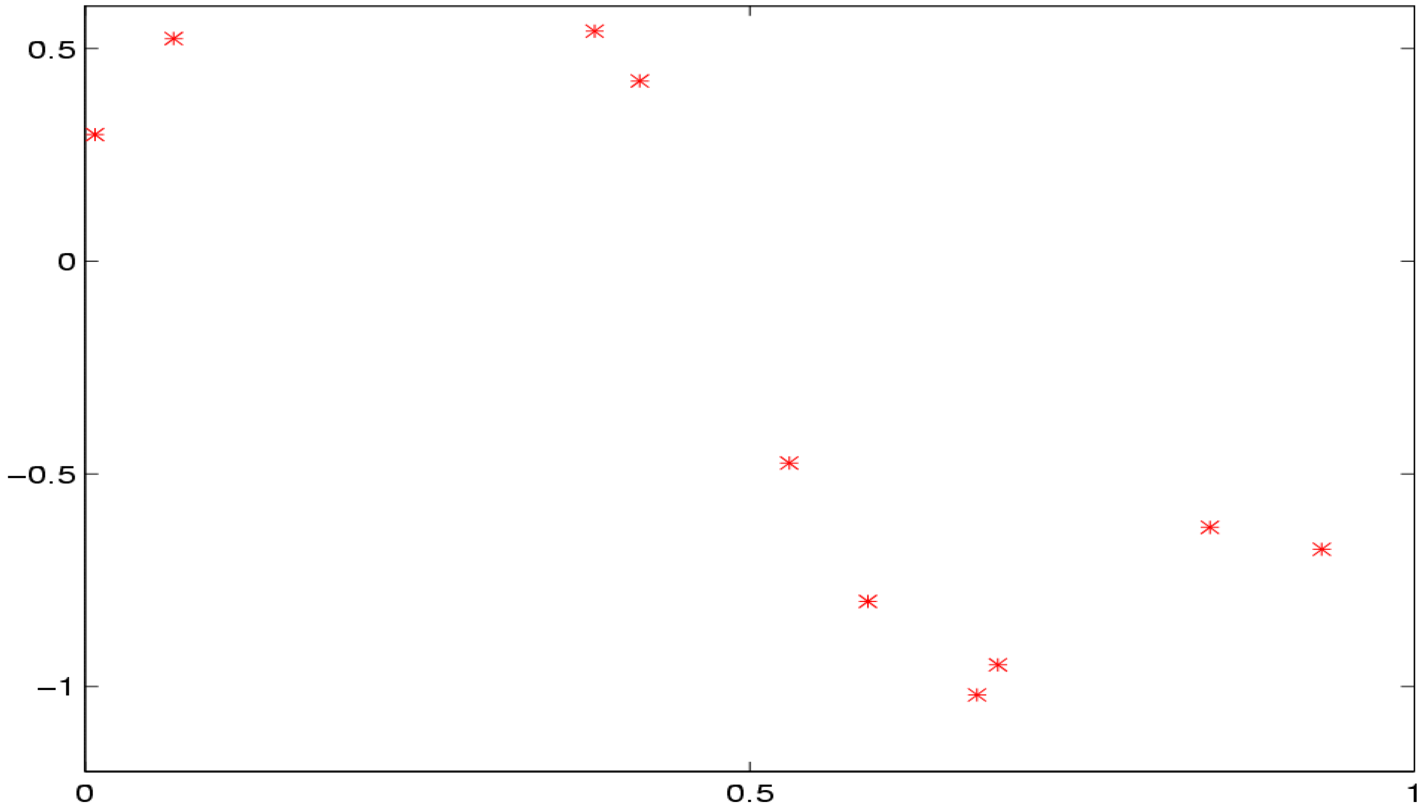
where we can now find a maximum a posteriori solution for t by maximizing the integrand (we will use this later).

Additive Normal Noise

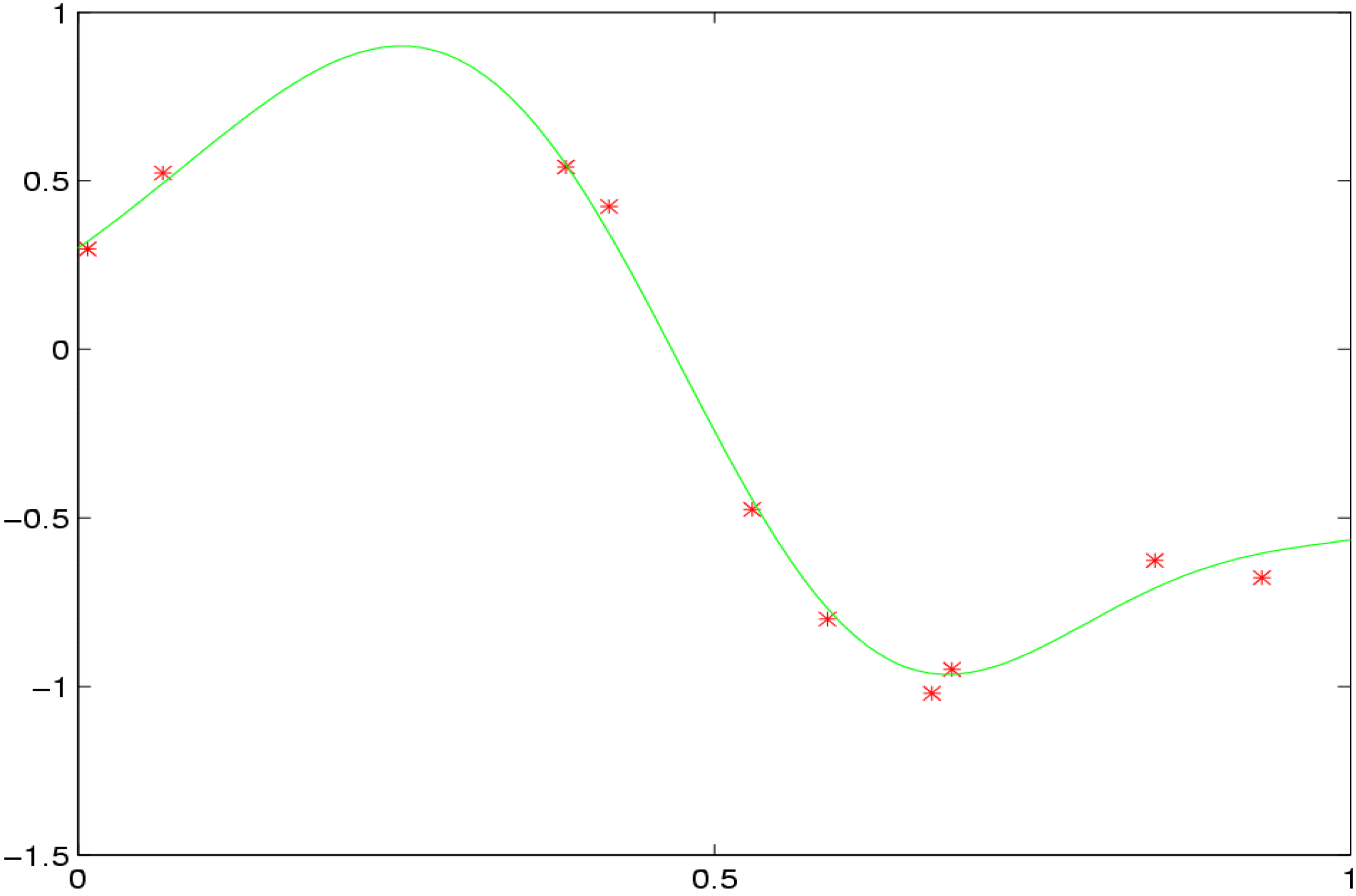
- If $\xi \sim \mathcal{N}(0, \sigma^2)$ then y is the sum of two Gaussian random variables.
- Means and variances **add up**.

$$y \sim \mathcal{N}(\mu, K + \sigma^2 \mathbf{1}).$$

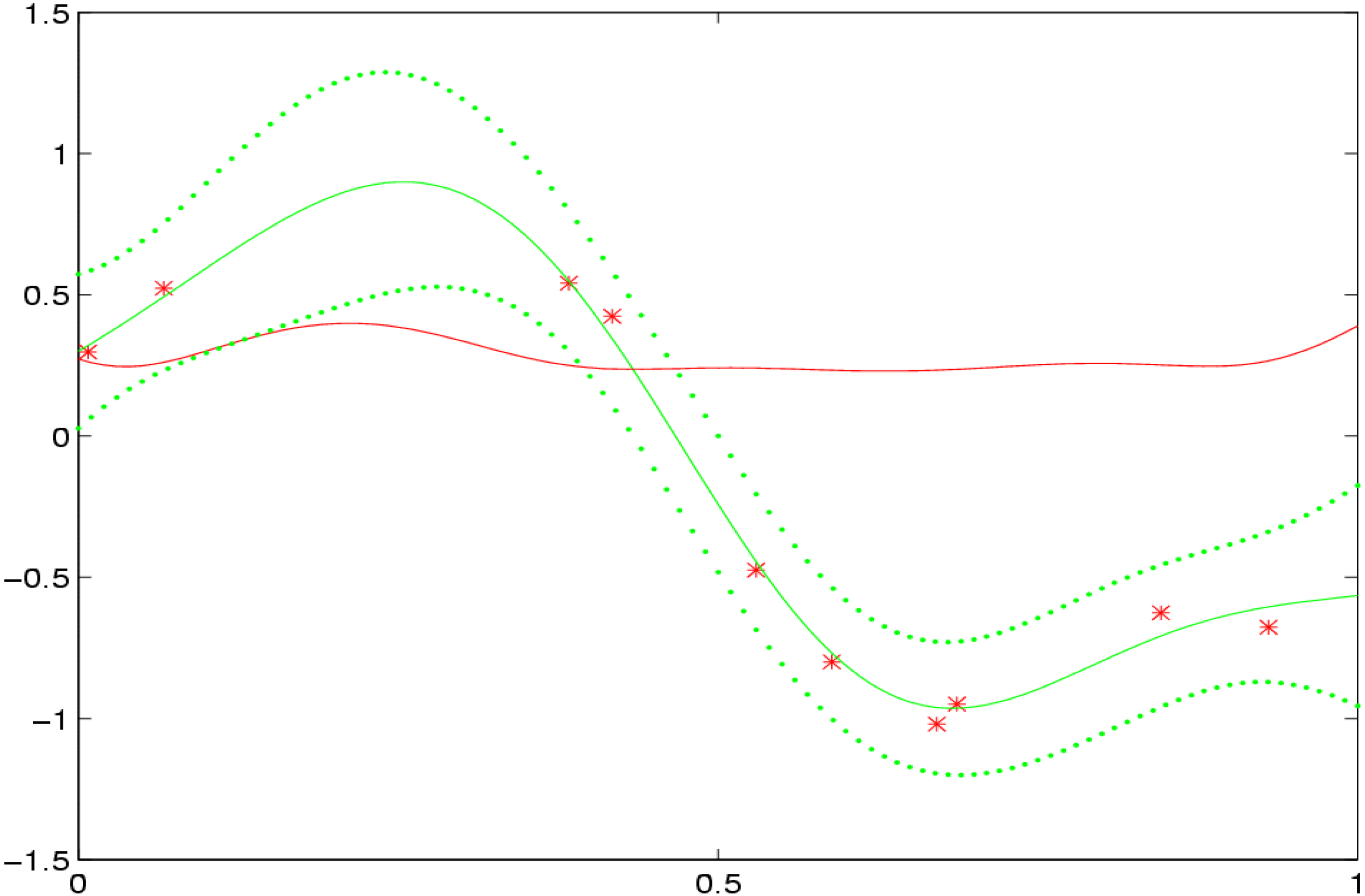
Training Data



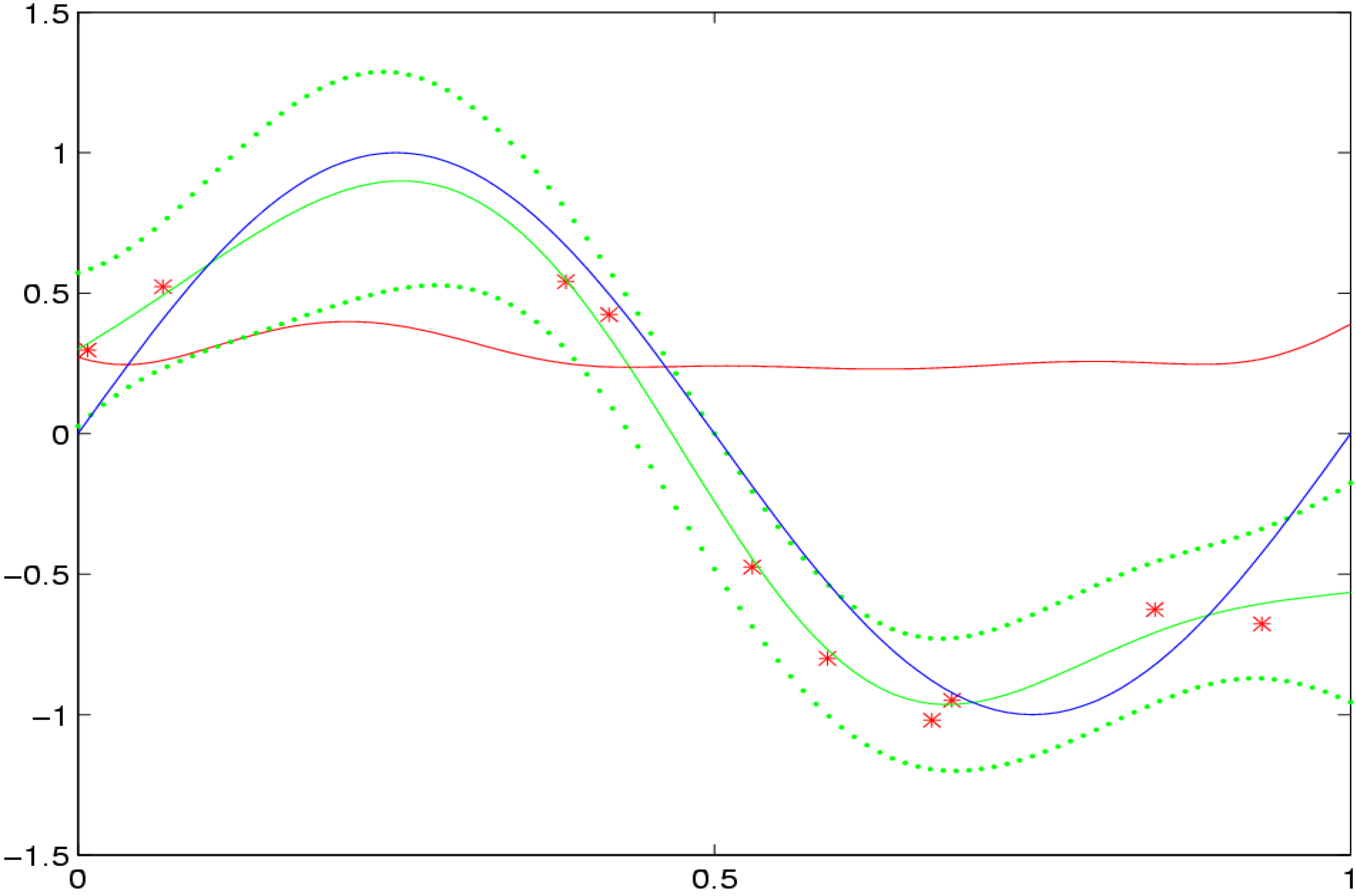
Mean $\vec{k}^\top(x)(K + \sigma^2\mathbf{1})^{-1}y$



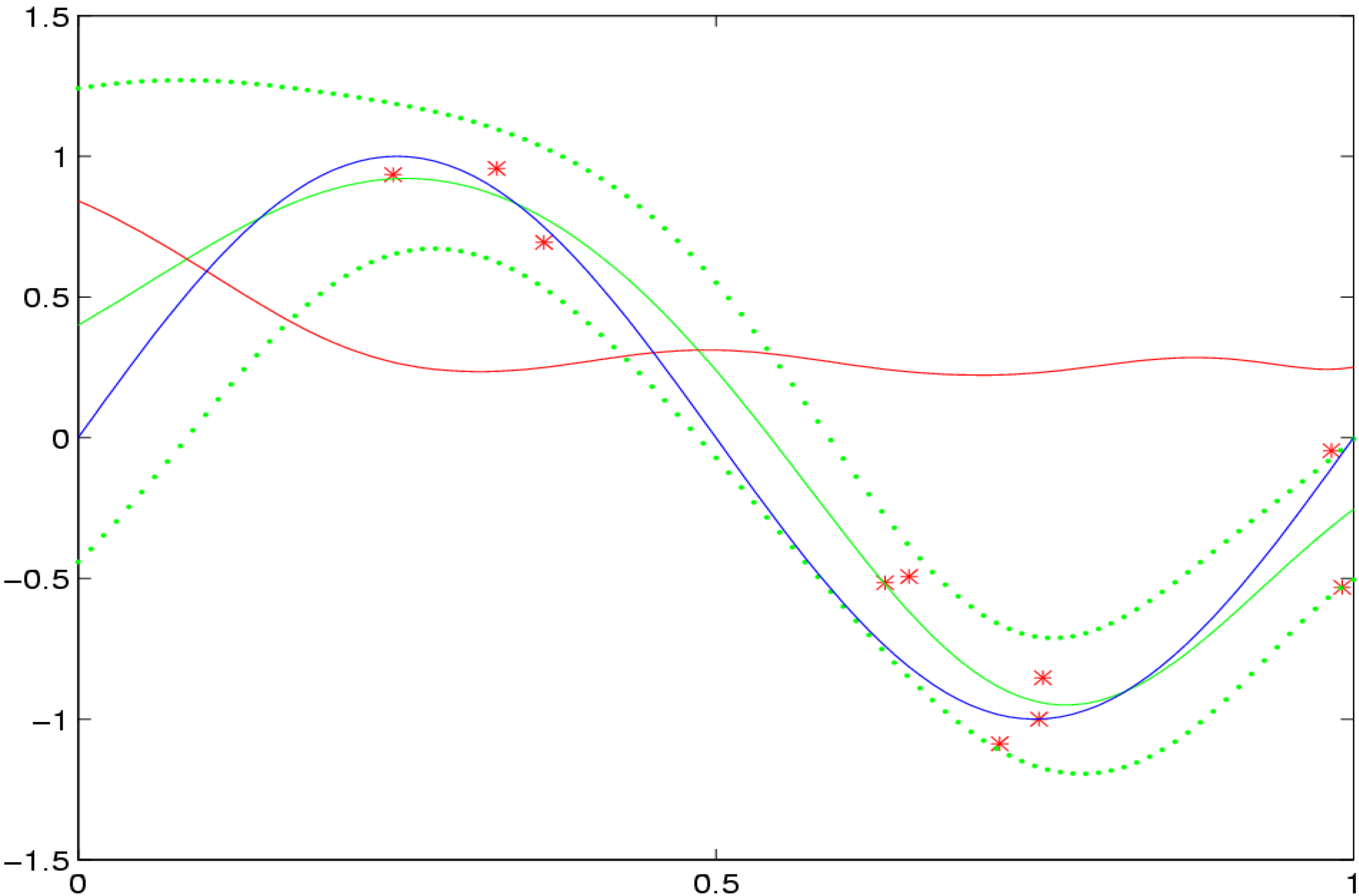
Variance $k(x,x) + \sigma^2 - \vec{k}^\top(x)(K + \sigma^2 \mathbf{1})^{-1}\vec{k}(x)$



Putting everything together . . .



Another Example



The ugly details

Covariance Matrices

- Additive noise

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

- Predictive mean and variance

$$\tilde{K} = K_{t't'} - K_{tt'}^\top K_{tt}^{-1} K_{tt'} \text{ and } \tilde{\mu} = K_{tt'}^\top K_{tt}^{-1} t$$

Pointwise prediction

$$\begin{aligned} K_{tt} &= K + \sigma^2 \mathbf{1} \\ K_{t't'} &= k(x, x) + \sigma^2 \\ K_{tt'} &= (k(x_1, x), \ldots, k(x_m, x)) \end{aligned}$$

Plug this into the mean and covariance equations.

The Support Vector Connection

SV Optimization Problem

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \mathrm{loss}(x_i, y_i, w)$$

Quadratic Loss

- Least mean squares regression

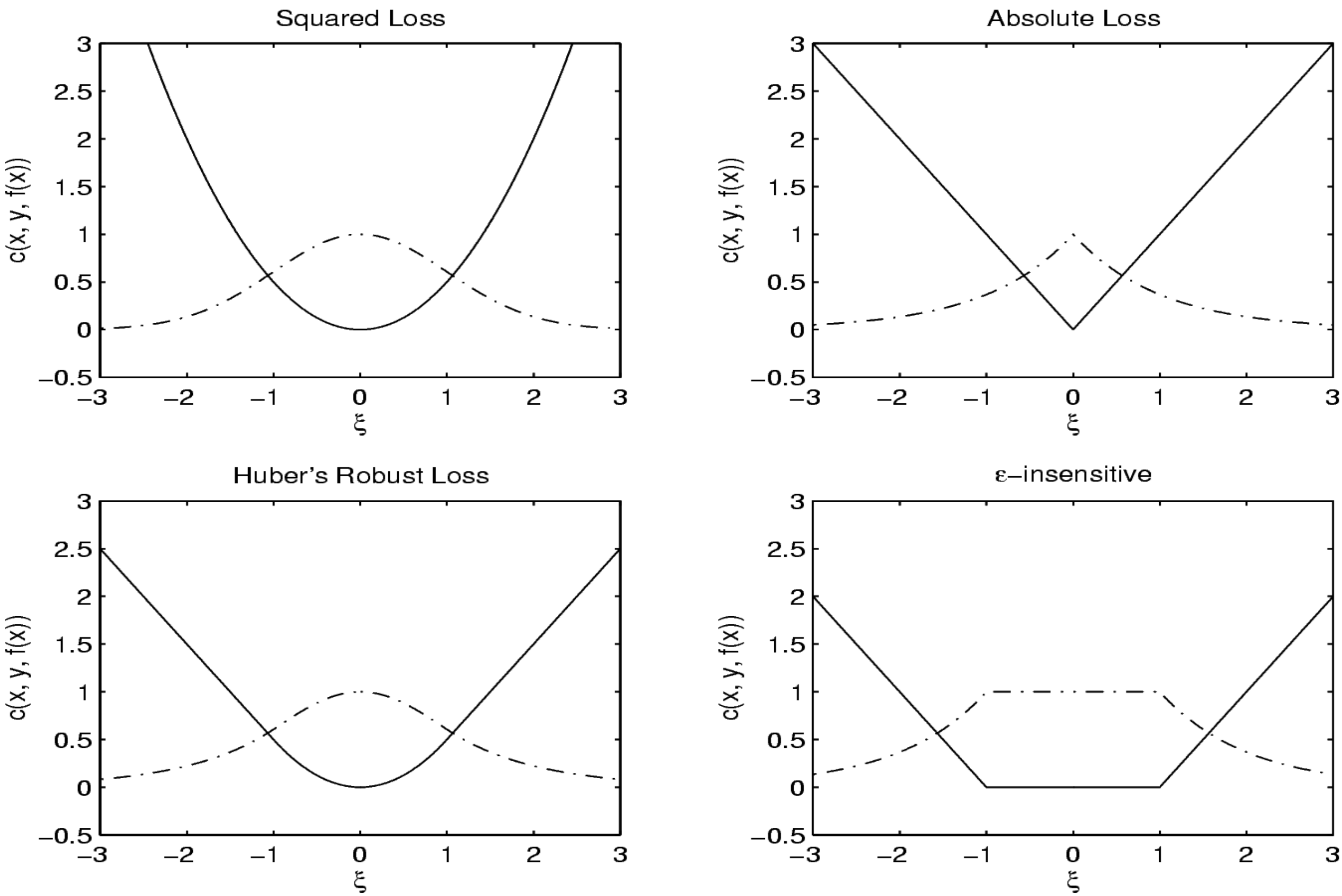
$$\text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \mathrm{loss}(y_i - \langle \phi(x_i), w\rangle)^2$$

- Solution

$$w = \sum_{i=1}^{m} \alpha_i \phi(x_i) \text{ where } \alpha = (K + C^{-1}\mathbf{1})y$$

This is identical to the GP regression (where $C = \sigma^{-2}$).

Regression loss functions



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

SV and GP connection

- GP kernel and SV kernel are the same
- Just different 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 with Kernels

Lecture 8

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

Statistical Machine Learning Program
National ICT Australia, Canberra

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

L8 Practical Tricks

Setting Parameters by Crossvalidation

- Leave one out estimation again
- Overdoing it

ν-trick

- Automatically adjusting the margin
- Optimization problems

Median trick and data scaling

- Scale matters
- Encoding data
- RBF-kernels

Smoothness and kernels

- Fourier transforms
- Frequency filters and smoothness

Adjusting Parameters

Problem

- Parameters can have huge impact on performance (number of errors, LMS error, etc.)
- Usually we don't have the test set when we tune the parameters (e.g. kernel width, value of C, learning rate)
- Huge bias if we only use training set to adjust terms

Solutions

- Use fancy learning theory (too complicated unless you really know what you're doing)
- Use Bayesian prior (too difficult until you understand statistics quite well)
- Use validation methods (easy to check in practice, works quite well)

Best number on a dice

Goal

- We want to win at gambling . . .
- Determine best face of a dice after observing it m times (and probability of occurrence).

Idea

- Pick best number after watching it m times.
- So we choose among n hypotheses

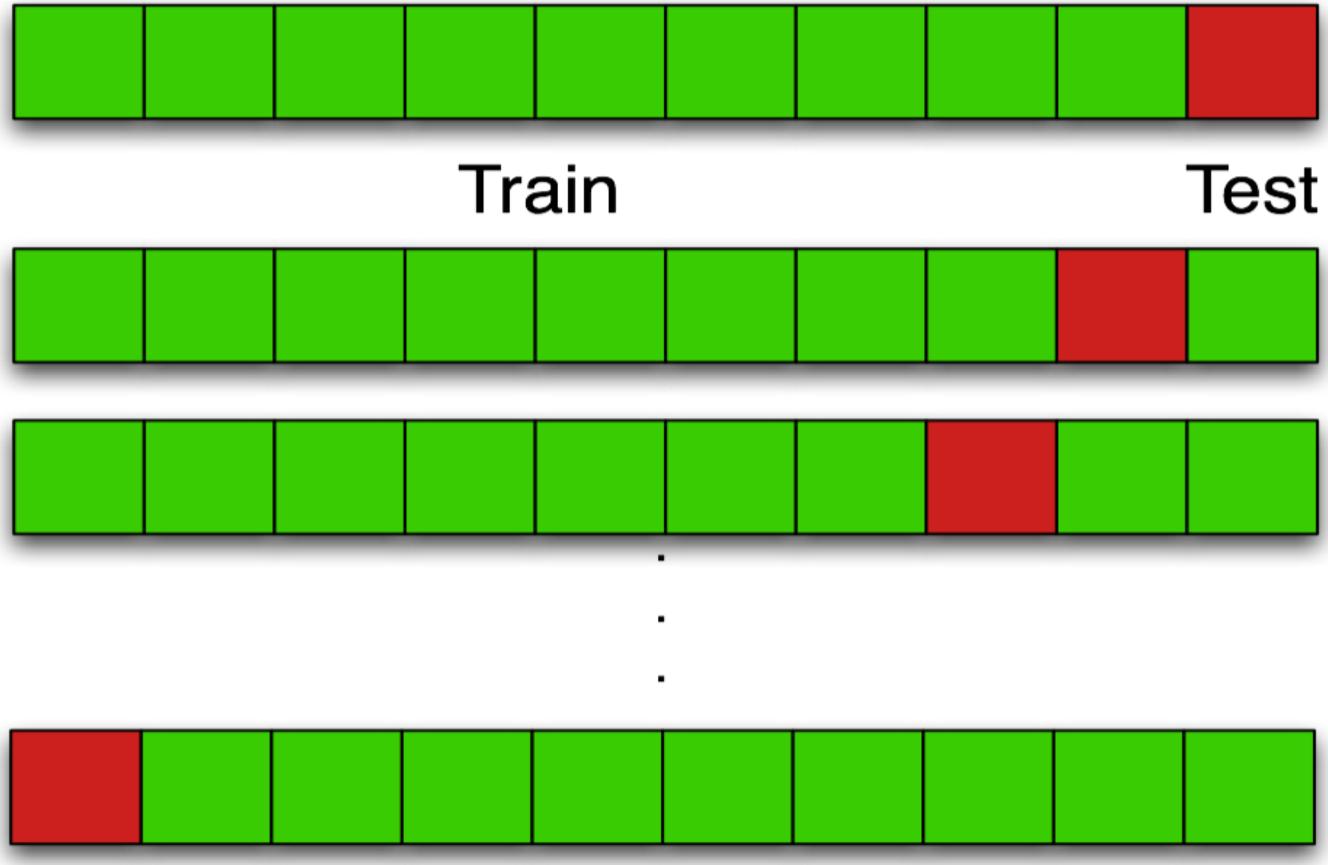
Problem

- Number of such occurrences is biased
- We want to know how reliable this is

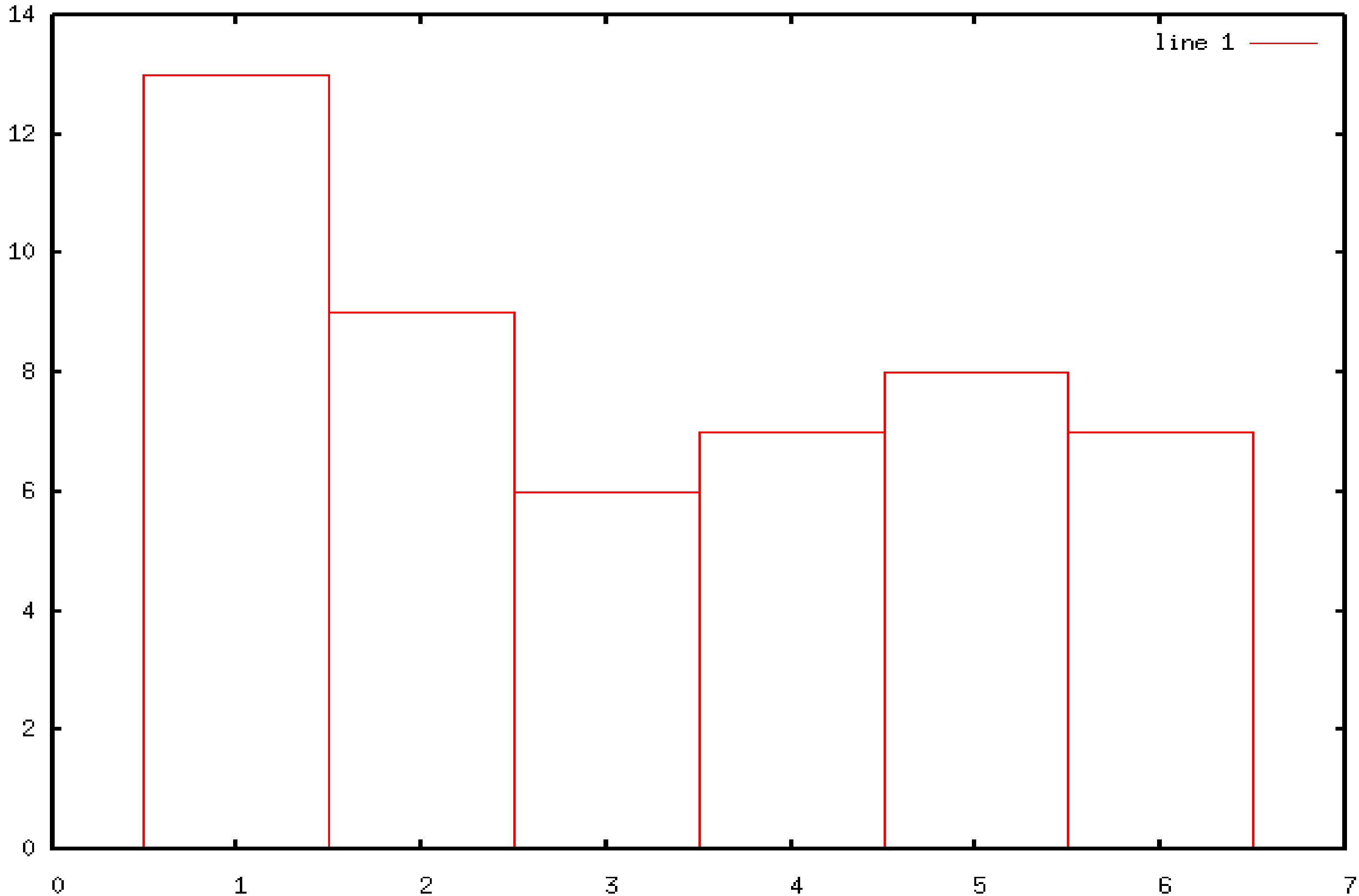
Solution

- Use 10-fold crossvalidation
- Estimate best number on 9/10 of the data and test on remaining 1/10. Repeat this for all partitions.

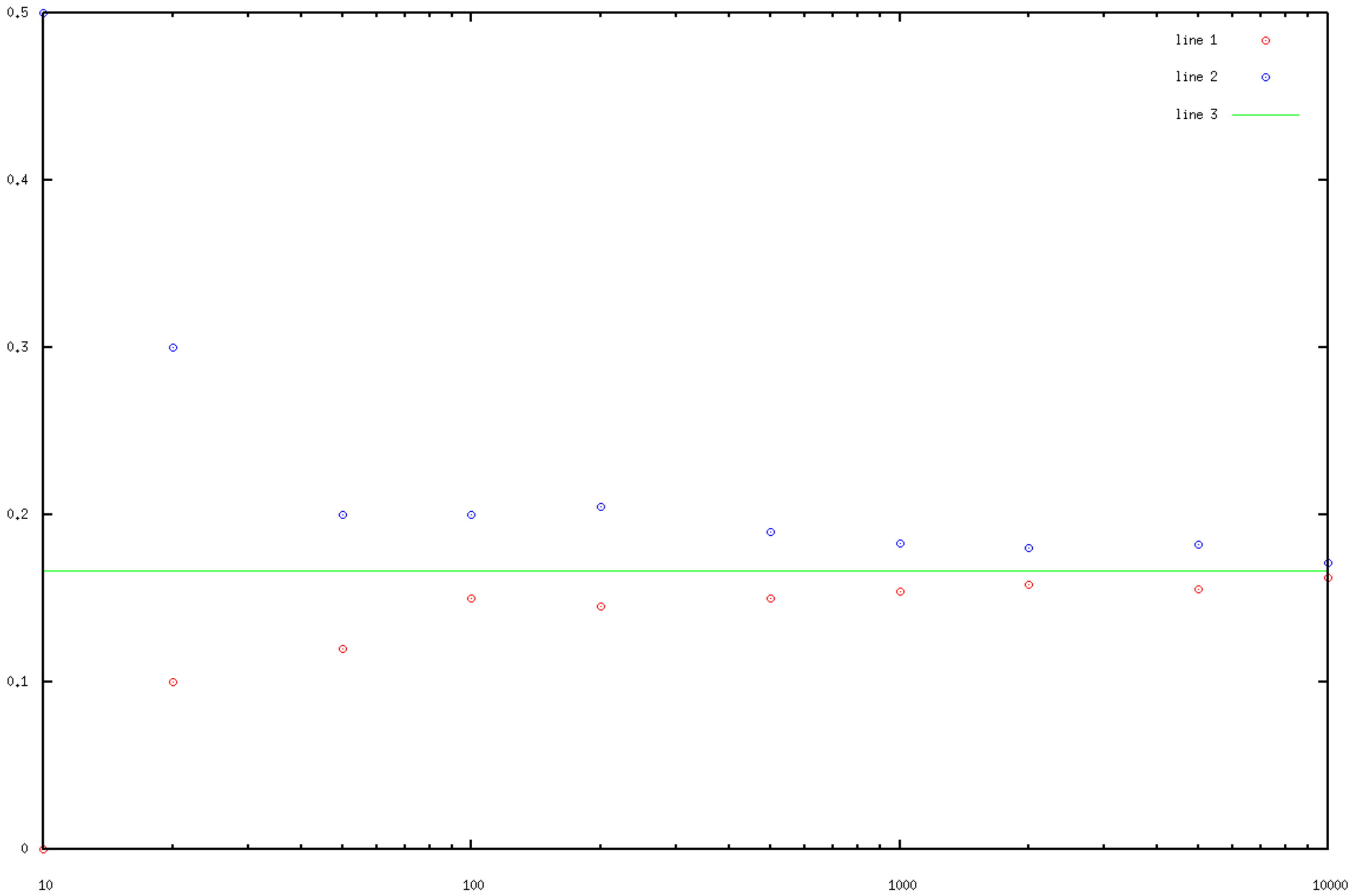
Crossvalidation



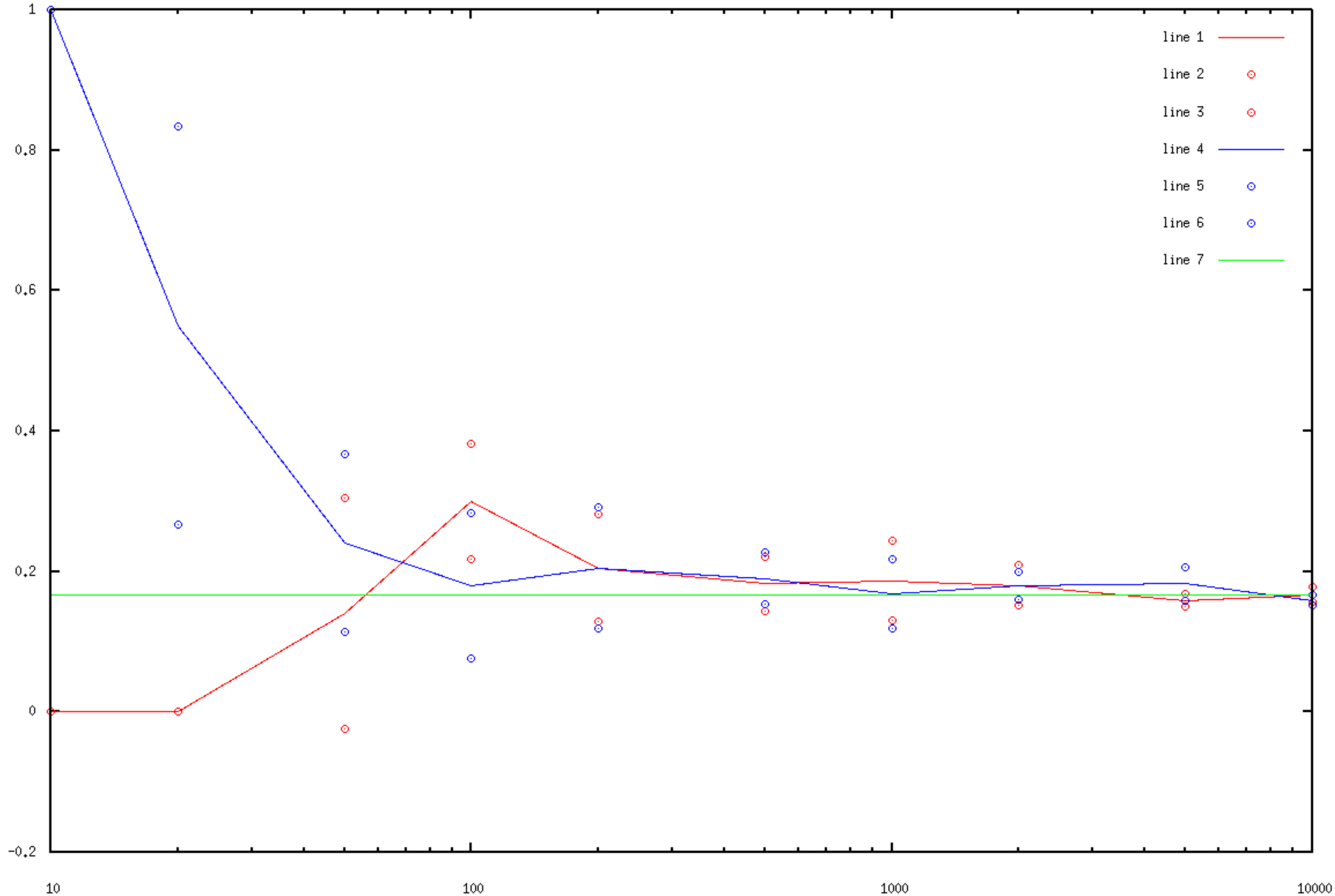
Casting a dice



Best number on a dice



Best number on a dice (Crossvalidation)



Morale of the story

Confidence intervals

- We get 10 noisy estimates of the error, e.g.

  0.1 | 0.2 | 0.2 | 0.2 | 0.4 | 0.2 | 0.2 | 0.0 | 0.2 | 0.1
- Compute variance and use it as confidence intervals. In the above case we get

  mean 0.18 and standard deviation 0.1

Overdoing it

- Testing too many options gives lousy estimates
- If possible, use at least 500 observations per parameter combination.

Using it in practice

Parameters

- Pick a set of parameters, e.g. $\sigma^2 \in \{0.1, 0.5, 1, 2, 5\}$

Cross validation

- Compute crossvalidation error for all the parameters
- Compute error bars
- Pick best one of the figures (within error bars)

Final estimate

- Use this set of parameters for final estimate (using all the data)
- Mission accomplished

Mini Summary

Crossvalidation

- Need to set parameters
- Use it to estimate performance of method
- Theory is complicated
- Quick and esay to implement hack

Practical Implementation

- Leave out 1/10 of the data and use it as validation set
- Cycle through data
- Average out and compute variance

Caveat

- Don’t compute too many cross validation estimates
- Estimates can be very noisy

The ν-trick

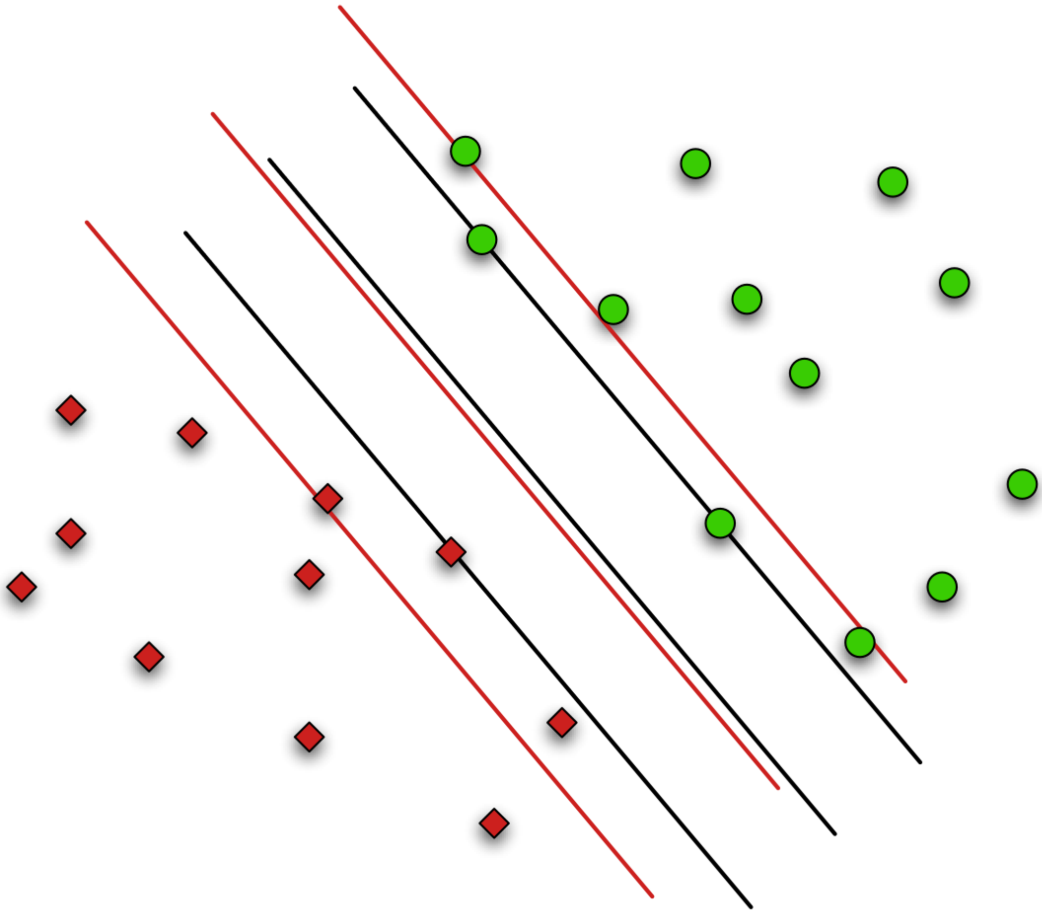
Problem

- Which value of C is right for the data
- Too small C means lots of training errors (end up using too simple a classifier, novelty detector, regression estimator, etc.)
- Too large C leads to overfitting (we believe even in the noisy data).

Solution

- Automatic capacity adjustment
- Adjust margin such that a certain *fraction* of points is an error
- Set this fraction to be roughly the expected number of errors.

Different margins



Large Margin Classifier

Standard Formulation

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

$$\text{subject to } y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

Capacity control by adjusting C

ν-Formulation

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

$$\text{subject to } y_i\left(\langle w, x_i\rangle + b\right) \geq \rho - \xi_i \text{ and } \xi_i \geq 0$$

Capacity control by adjusting $\nu$where $\nu \in [0, 1]$

The ν-property

Optimizing ρ

- Optimization Problem

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

$$\text{subject to } y_i\left(\langle w, x_i\rangle + b\right) \geq \rho - \xi_i \text{ and } \xi_i \geq 0$$

- Increasing ρ up to where a fraction of at most ν points violate the constraint decreases the objective function.
- Decreasing ρ up to where a fraction of at least $1-\nu$ satisfy the constraint decreases the objective function.
- In the limit ($m \to \infty$) the fractions become exact.

Interpretation

Dual Problem

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

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

$$\sum_{i=1}^{m} \alpha_i = \nu m$$

$$0 \leq \alpha_i \leq 1$$

Properties

- A large number of coefficients needs to be nonzero
- At least νm of them
- Different initialization than standard SMO (e.g. via Parzen windows)

Recall: novelty detection

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

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

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

Almost same problem as before, just with all $y_i = 1$ and no offset b.

Using it

Training errors vs. test errors

- Want to have similar number of training and test errors (then the estimator is not overfitting)
- Do not try learning training data perfectly (if it is noisy)
- Set ν to be order of expected test errors

Good news

- Adjusting ν is **very robust.**
- One parameter less to worry about

Bad news

- Not every optimizer supports it (choose LibSVM to have an optimizer)
- Some SV books don't cover it (choose one which does instead)

Mini Summary

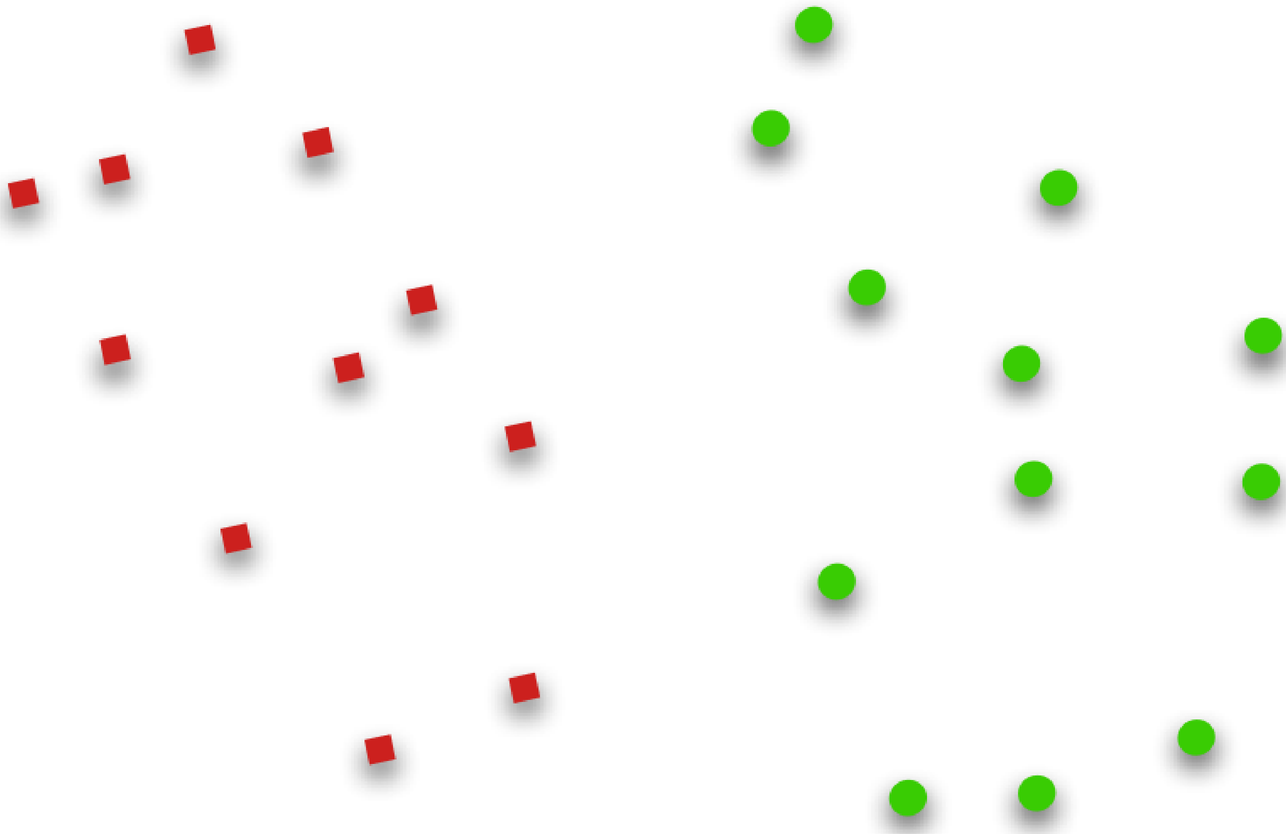
Basic Idea

- Adjust margin automatically.
- Optimization solution will follow.

Why

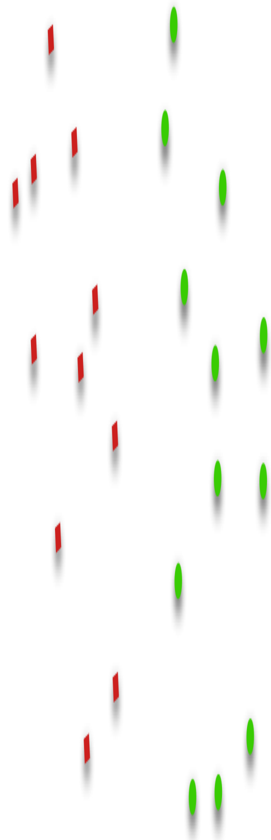
- No worries parameter setting.
- Safeguard against overfitting.
- Control number of novel points in novelty detection (we set the threshold).
- Easier to understand than setting C.

Normal Problem



Easy Problem

Hard Problem



Rescaling Data

Why

- Rescaling can make problem easy or difficult
- Often data is heterogeneous (e.g. height, weight, age, blood pressure, etc.) and not pre-scaled

How

- Do not rotate data unless you can assume that coordinates do not matter.
- Corollary: do not use PCA unless you've got a good reason to believe that it is useful.
- Simply rescale coordinates to zero mean and variance of same order of magnitude (typically 1, or bounded variation, etc.)

Example

Observations

- Age: between 10 and 90 years
- Weight: between 40 and 150 kg
- Height: between 1.4 and 2.2 m
- Blood pressure: between 60 and 200

Rescaling to bounded range

- Age: subtract 10 and divide by 80
- Weight: subtract 40 and divide by 110
- ...

Rescaling to given variance

- Subtract mean
- Divide by variance

Kernel width

RBF kernel parameters

- Kernel $k(x, x') = k(\sigma \|x - x'\|)$
- Keep σ in range where $k(\sigma \|x - x'\|)$ is interesting.

Example: Gaussian RBF kernel

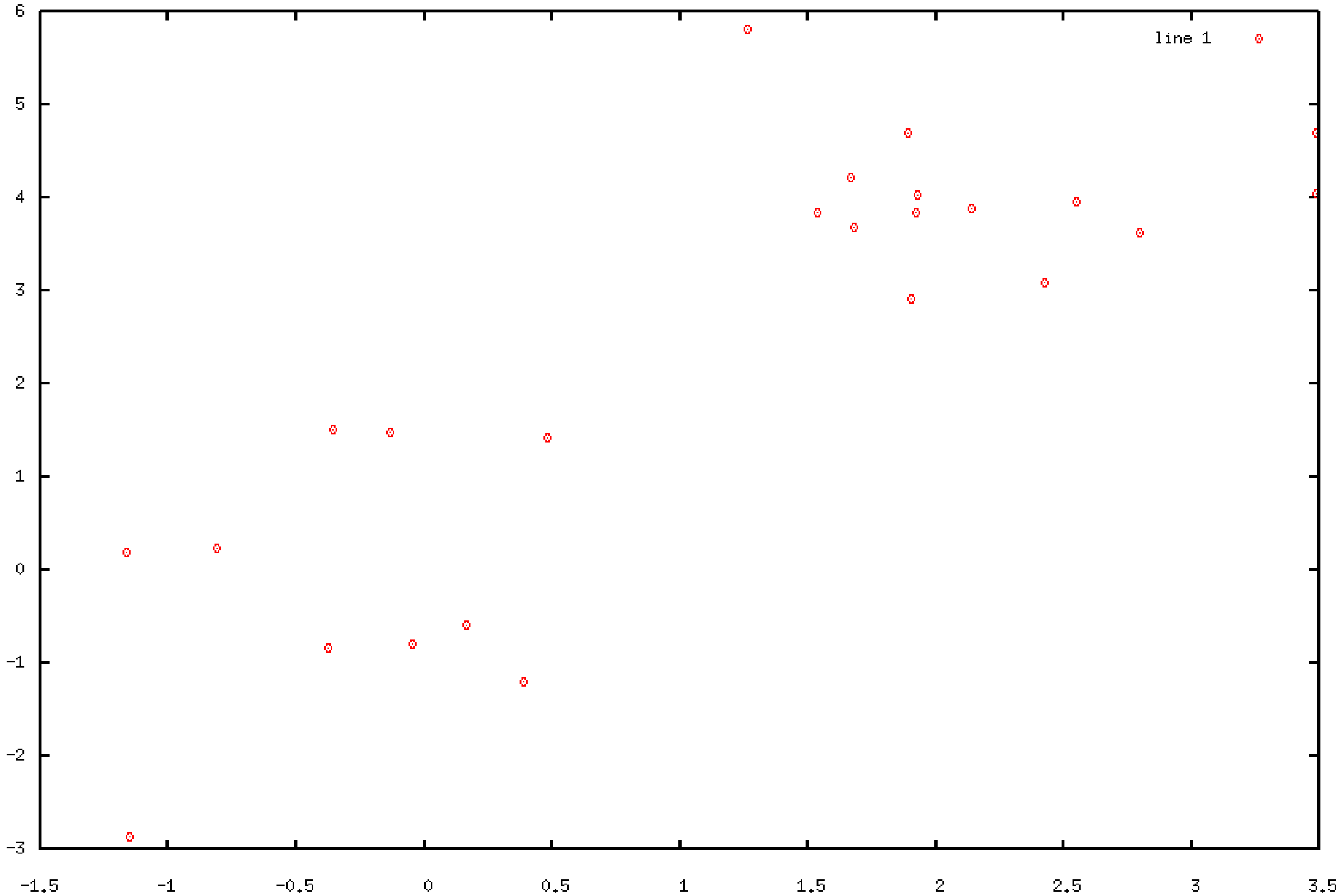
$$k(x, x') = \exp\left(-\frac{1}{\sigma^2}\|x - x'\|^2\right)$$

Set σ such that $\sigma^{-2}\|x - x'\|^2$ is in the order of 1.

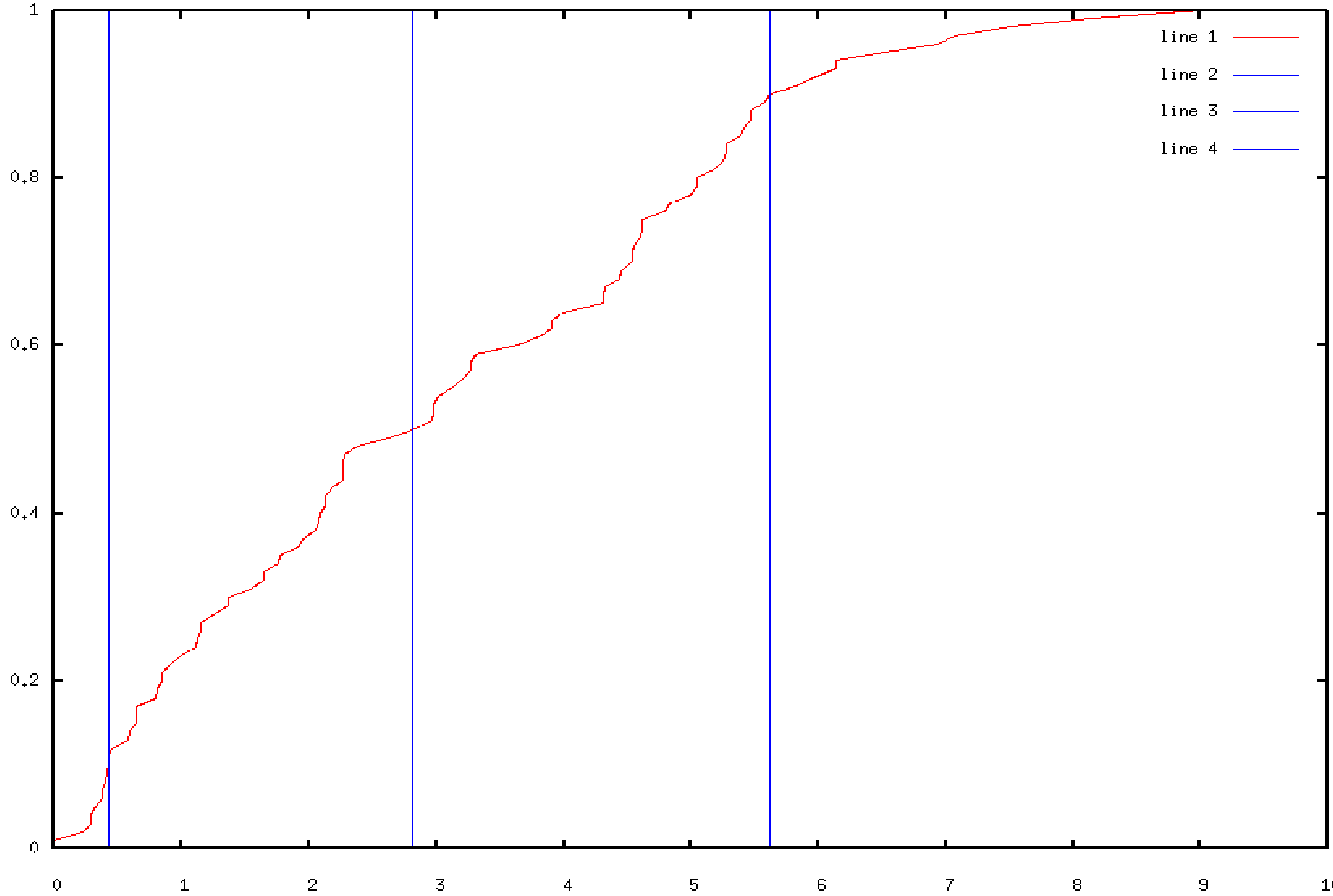
Solution

- Measure median pairwise distance $\|x_i - x_j\|$
- Practical implementation: pick 1000 random pairs (x_i, x_j) and compute distances.
- Pick $0.1, 0.5$, and 0.9 quantile as candidates for σ
- Fine-tuning with crossvalidation

Sample Data



Pairwise Distances



Categorical Data

Rule of thumb

- Pick dummy-variable code
- Pick Gaussian-RBF kernel

Rationale

- Dummy variable code maps data onto points on hypercube
- Diffusion process on hypercube corresponds to Gaussian RBF kernel.

Example

- Data: {Married, Single}, {English, French, German}
- (Married, French) $\mapsto$ (1,0,0,1,0)
- (Single, German) $\mapsto$ (0,1,0,0,1)
- (Single, English) $\mapsto$ (0,1,1,0,0)

Mini Summary

Data Scaling

- Wrong scales make problem difficult
- Comparable inputs are important

Kernel Width

- Adjust to interesting scale for RBF kernel
- Fine tuning via crossvalidation

Categorical Data

- If no relation, map into dummy variables
- If relation, map into thermometer code

Smoothness and Kernels

Conundrum

- SVM map data into highdimensional space
- Still SVM work well and (usually) do not overfit
- How do SVMs "choose" the "right complexity"

Solution

- Norm in feature space $\|w\|^2$ corresponds to smoothness of functional
- For RBF kernels this means higher order derivatives of the function $f(x) = \langle w, \phi(x) \rangle$.
- Example: Laplacian kernel corresponds to

$$\|w\|^2 \propto \|f\|_{L_2}^2 + \|\partial_x f\|_{L_2}^2$$

Smoothness and Fourier Transforms

RBF Kernels

- Kernels of type $k(x, x') = k(\|x - x'\|)$
- Representation in Fourier Domain (translation invariant setting)

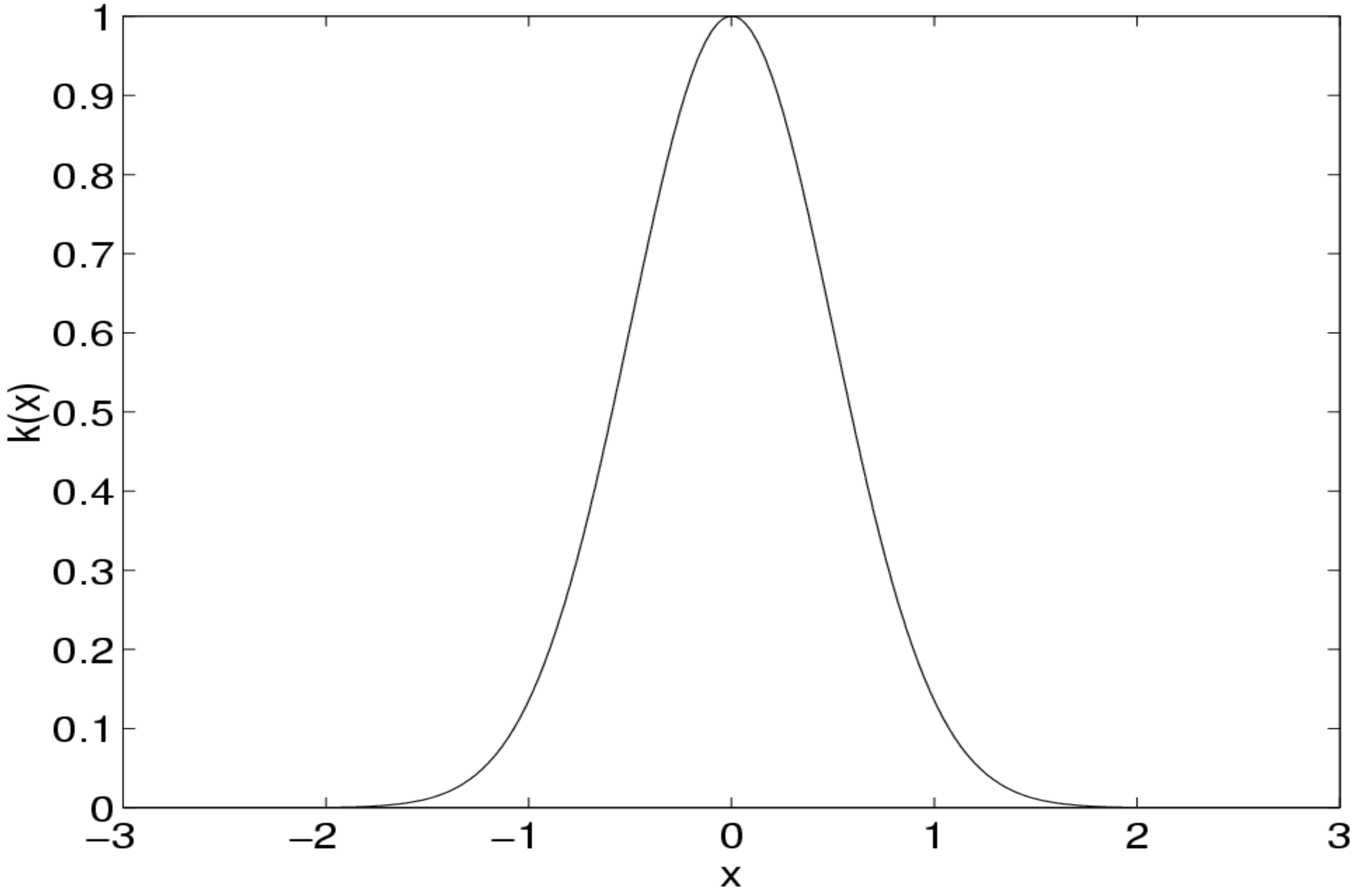
Kernels in Fourier Domain

- Compute Fourier transform $\tilde{k}$ of $k(\cdot)$
- This determines the smoothness of k
- Functional is

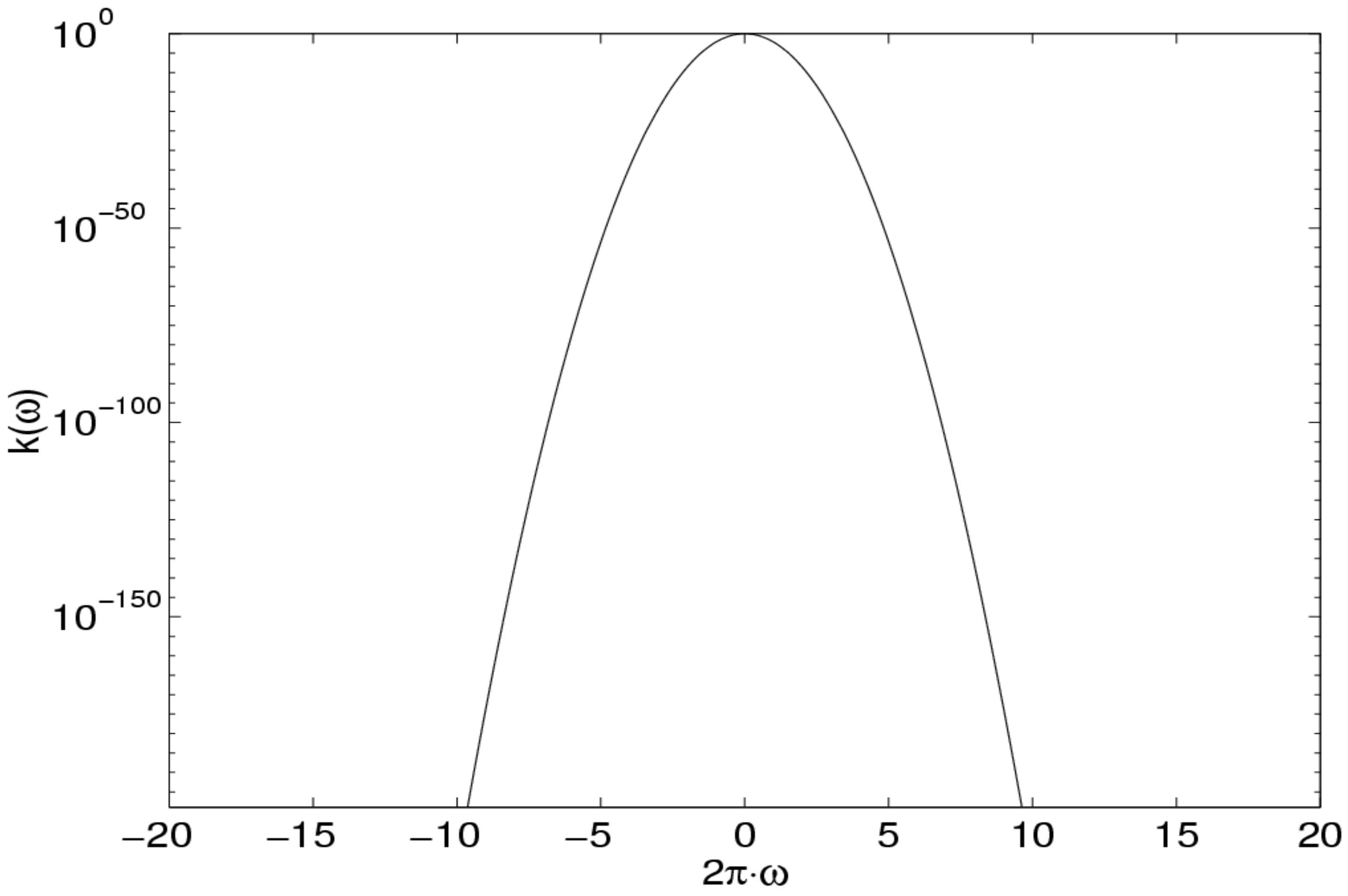
$$\|f\|^2 = \int \frac{\|\tilde{f}(\omega)\|^2}{\tilde{k}(\omega)} d\omega$$

- Small values of $\tilde{k}(\omega)$ require small terms in $\tilde{f}(\omega)$.
- Acts like frequency filter

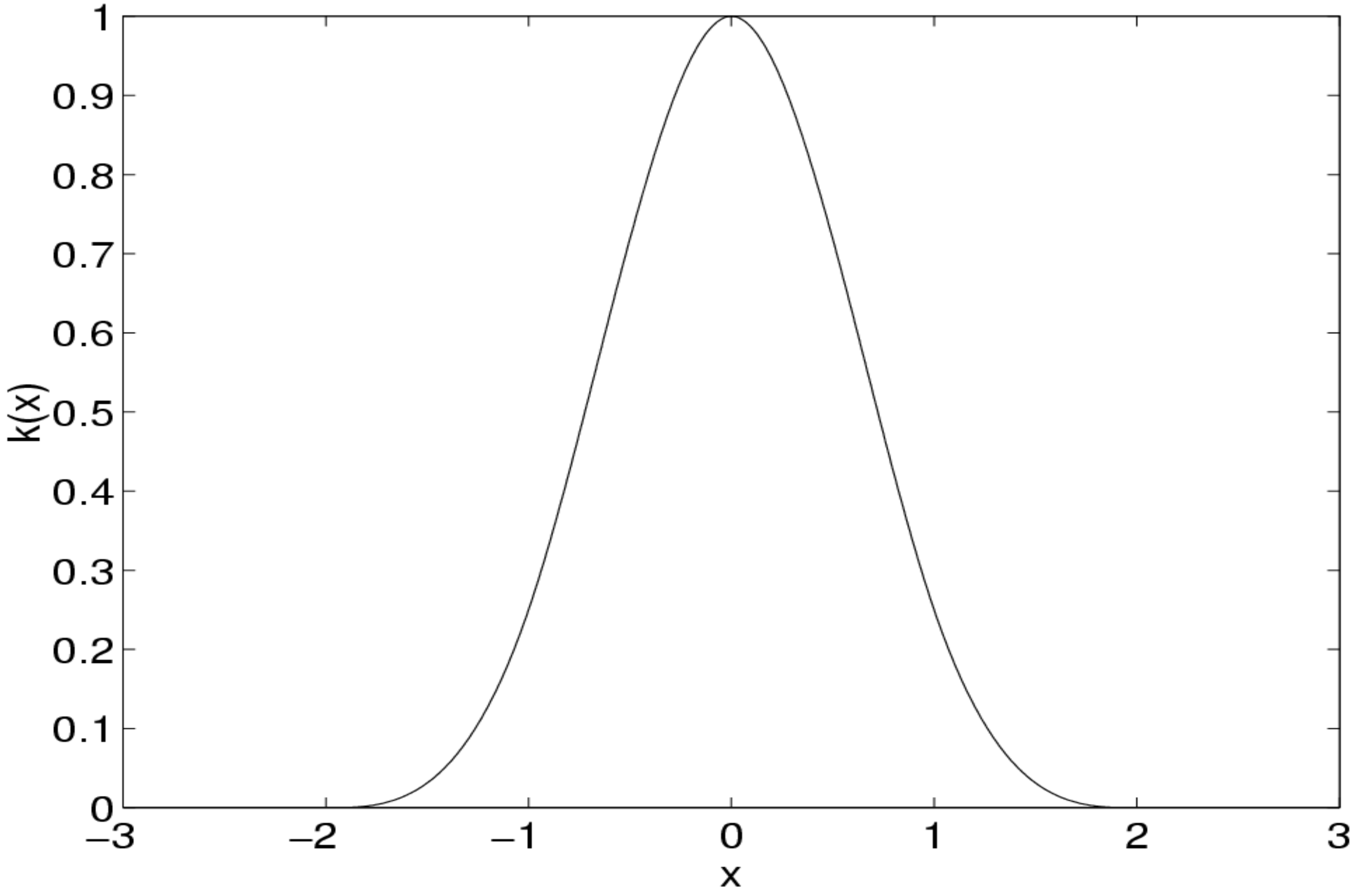
Gaussian RBF Kernel



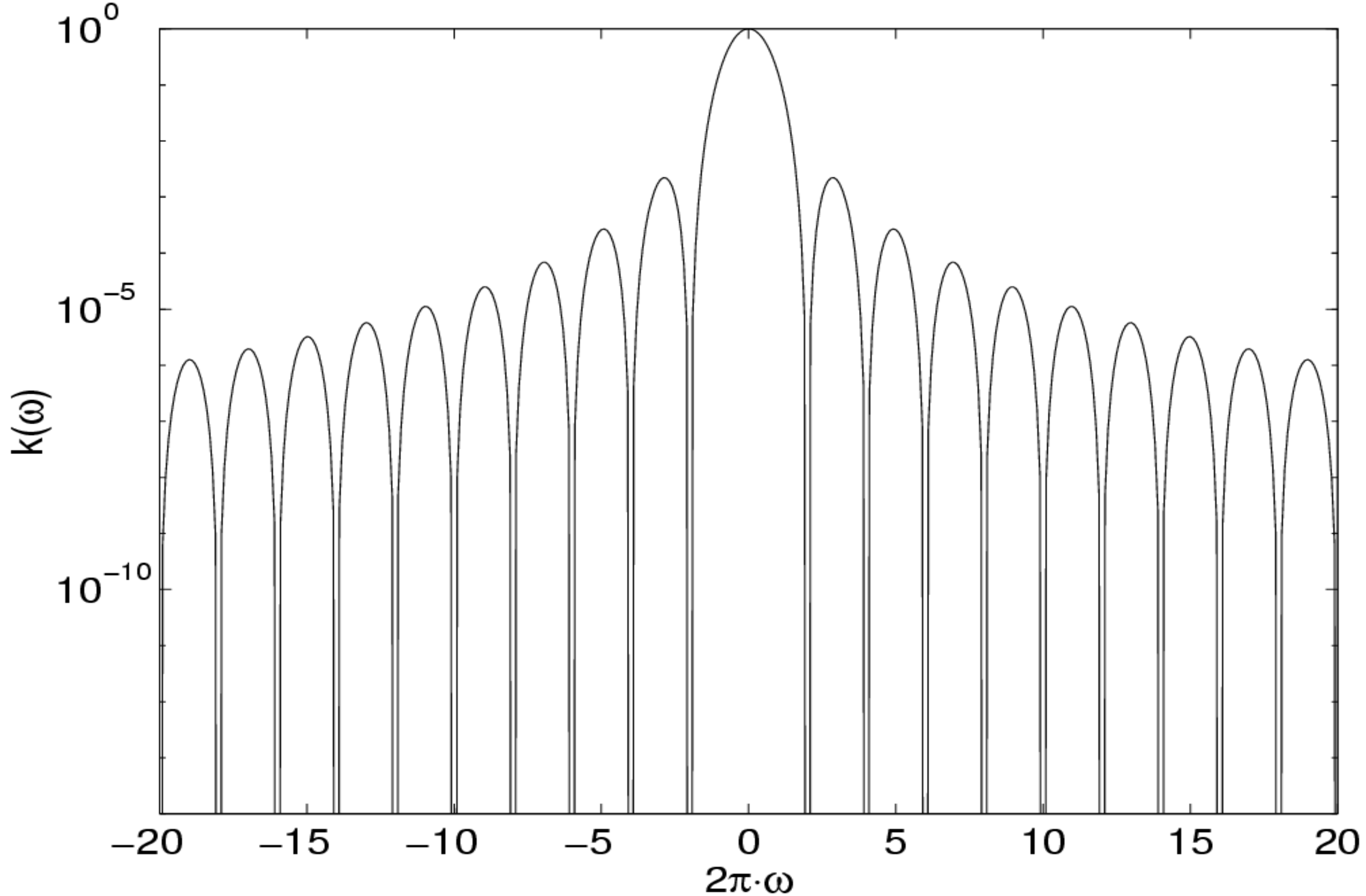
Gaussian RBF Kernel - FFT



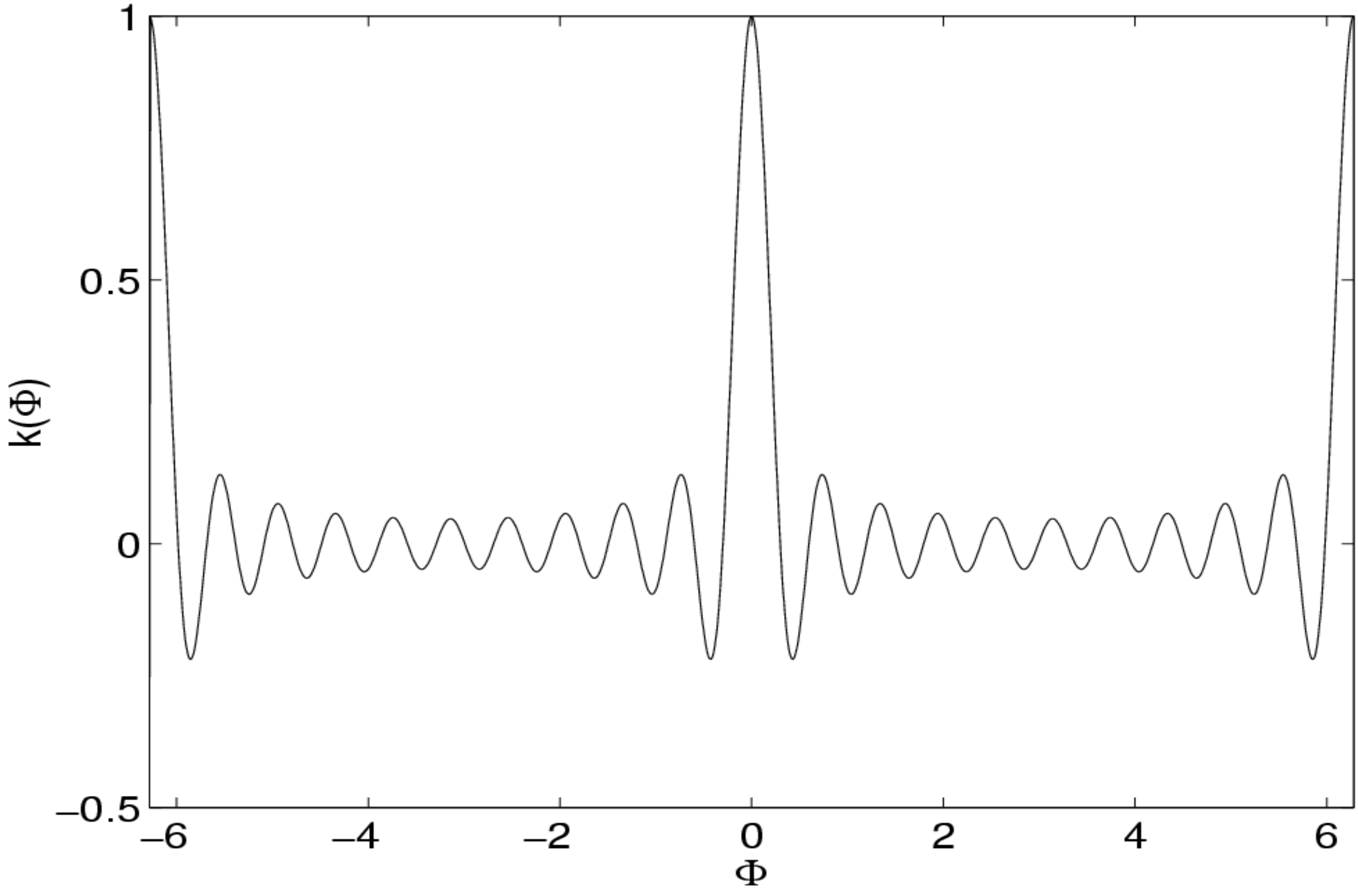
B3 Spline Kernel



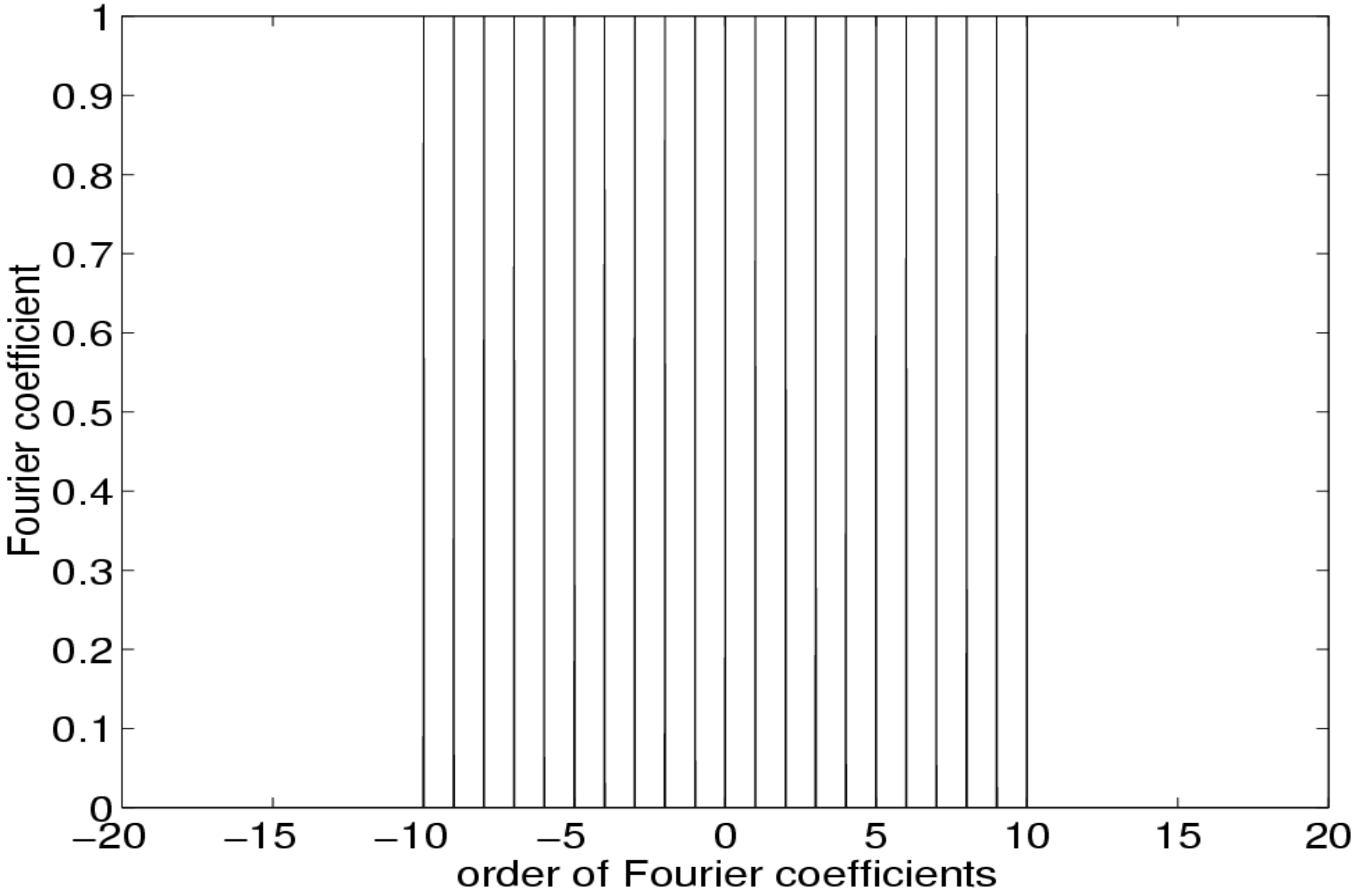
B3 Spline Kernel - FFT



Dirichlet Kernel order 10



Dirichlet Kernel order 10 - FFT



Mini Summary

Smoothness

- Norm in feature space is degree of smoothness
- SVM solves classification problem while keeping a simple function

Fourier Transform

- Fourier transform shows smoothness of kernel
- Positive sign of Fourier transform ensures Mercer property
- Easy to check engineering solution for Mercer's theorem
- Good intuition for suitable kernels

Summary

Setting Parameters by Crossvalidation

- Leave one out estimation again
- Overdoing it

ν-trick

- Automatically adjusting the margin
- Optimization problems

Median trick and data scaling

- Scale matters
- Encoding data
- RBF-kernels

Smoothness and kernels

- Fourier transforms
- Frequency filters and smoothness