Artificial Intelligence and Machine Learning

Barbara Caputo

Perceptron: Non-linearity and preprocessing

Nonlinear Features

- Perceptron

Nonlinear Features

- Perceptron
    - Map data into feature space $x \rightarrow \phi(x)$

Nonlinear Features

- Perceptron
  - Map data into feature space $x \to \phi(x)$
  - Solve problem in this space

Nonlinear Features

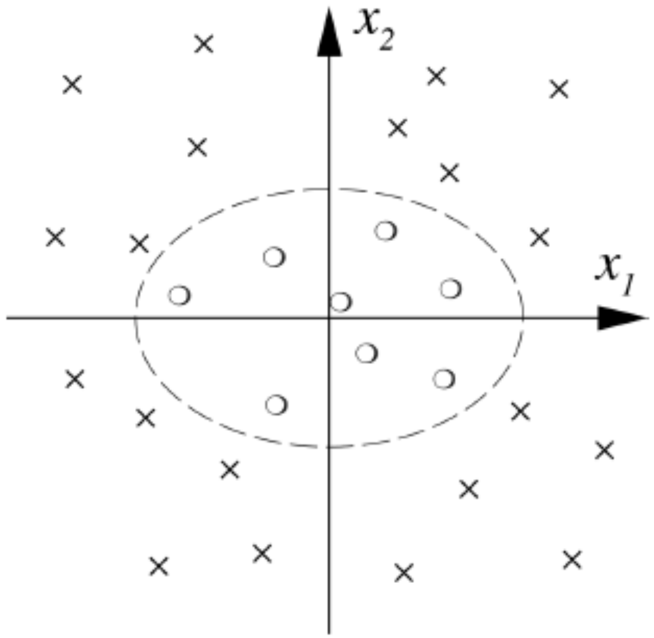
- Perceptron
  - Map data into feature space $x \to \phi(x)$
  - Solve problem in this space
  - Query replace $\langle x, x' \rangle$ by $\langle \phi(x), \phi(x') \rangle$ for code

Nonlinear Features

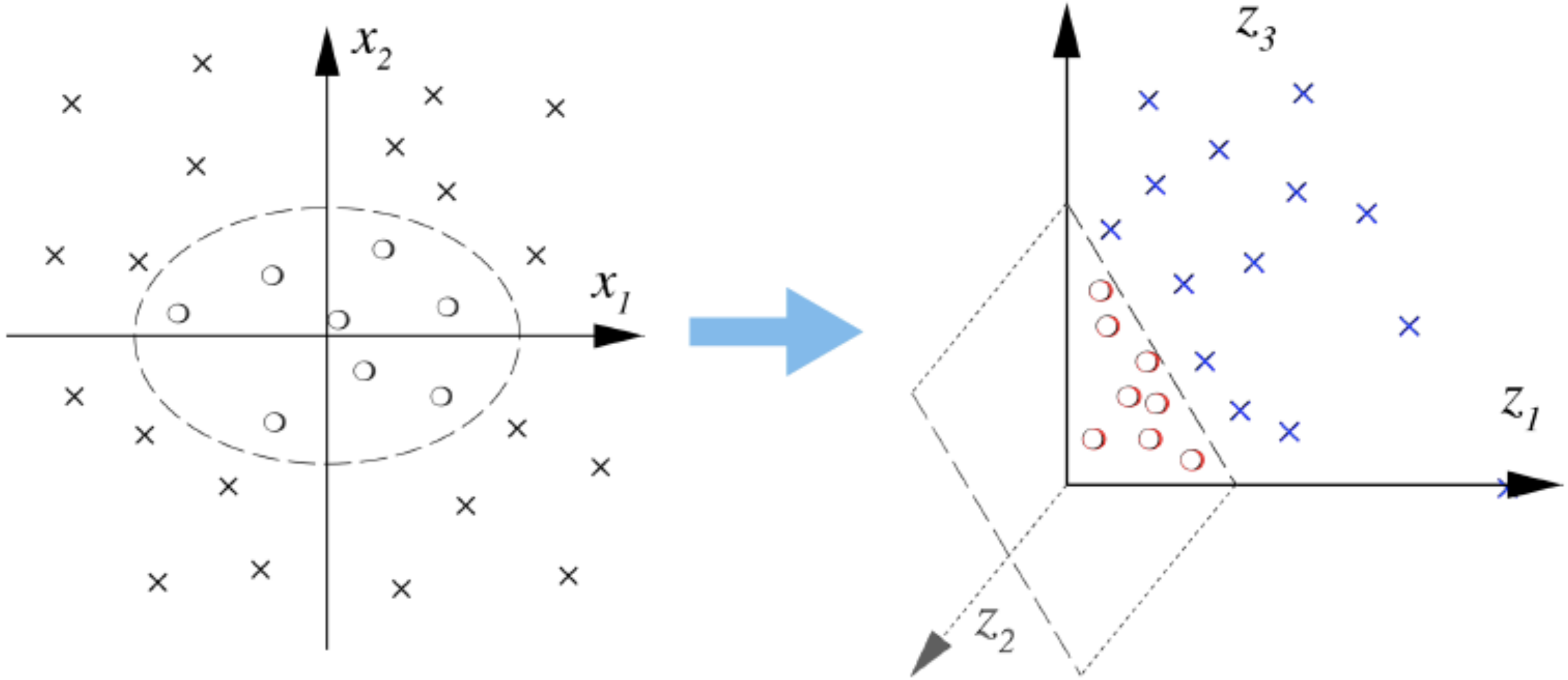
- Perceptron
  - Map data into feature space $x \to \phi(x)$
  - Solve problem in this space
  - Query replace $\langle x, x' \rangle$ by $\langle \phi(x), \phi(x') \rangle$ for code
- Feature Perceptron
  - Solution in span of $\phi(x_i)$

Quadratic Features

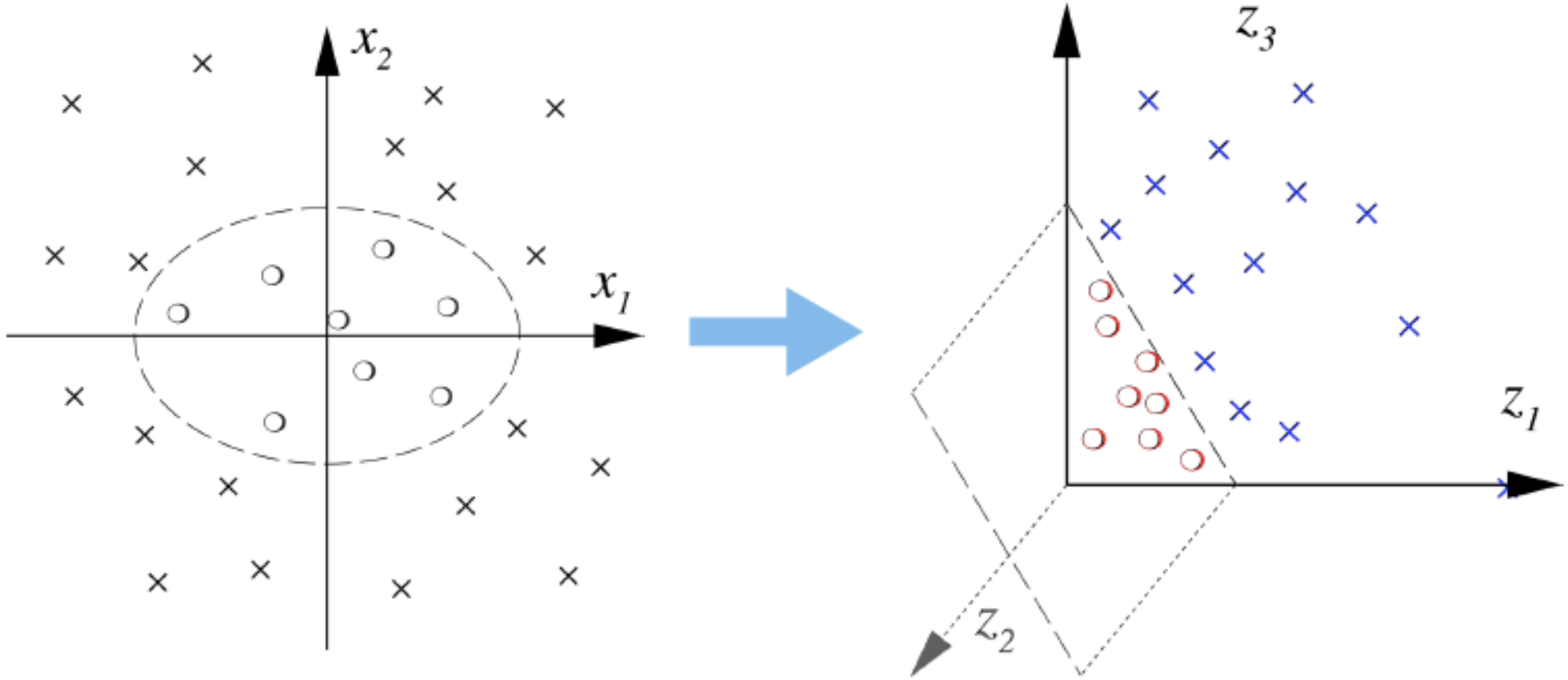
Quadratic Features



Quadratic Features



Quadratic Features



- Separating surfaces are
  Circles, hyperbolae, parabolae

Constructing Features (very naive OCR system)

	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

Feature Engineering for Spam Filtering

```
Delivered-To: alex.smola@gmail.com
Received: by 10.216.47.73 with SMTP id s51cs361171web;
        Tue, 3 Jan 2012 14:17:53 -0800 (PST)
Received: by 10.213.17.145 with SMTP id s17mr2519891eba.147.1325629071725;
        Tue, 03 Jan 2012 14:17:51 -0800 (PST)
Return-Path: <alex+caf_=alex.smola=gmail.com@smola.org>
Received: from mail-ey0-f175.google.com (mail-ey0-f175.google.com [209.85.215.175])
        by mx.google.com with ESMTPS id n4si29264232eef.57.2012.01.03.14.17.51
        (version=TLSv1/SSLv3 cipher=OTHER);
        Tue, 03 Jan 2012 14:17:51 -0800 (PST)
Received-SPF: neutral (google.com: 209.85.215.175 is neither permitted nor denied by best
guess record for domain of alex+caf_=alex.smola=gmail.com@smola.org) client-
ip=209.85.215.175;
Authentication-Results: mx.google.com; spf=neutral (google.com: 209.85.215.175 is neither
permitted nor denied by best guess record for domain of alex
+caf_=alex.smola=gmail.com@smola.org) smtp.mail=alex+caf_=alex.smola=gmail.com@smola.org;
dkim=pass (test mode) header.i=@googlemail.com
Received: by eaal1 with SMTP id l1so15092746eaa.6
        for <alex.smola@gmail.com>; Tue, 03 Jan 2012 14:17:51 -0800 (PST)
Received: by 10.205.135.18 with SMTP id ie18mr5325064bkc.72.1325629071362;
        Tue, 03 Jan 2012 14:17:51 -0800 (PST)
X-Forwarded-To: alex.smola@gmail.com
X-Forwarded-For: alex@smola.org alex.smola@gmail.com
Delivered-To: alex@smola.org
Received: by 10.204.65.198 with SMTP id k6cs206093bki;
        Tue, 3 Jan 2012 14:17:50 -0800 (PST)
Received: by 10.52.88.179 with SMTP id bh19mr10729402vdb.38.1325629068795;
        Tue, 03 Jan 2012 14:17:48 -0800 (PST)
Return-Path: <althoff.tim@googlemail.com>
Received: from mail-vx0-f179.google.com (mail-vx0-f179.google.com [209.85.220.179])
        by mx.google.com with ESMTPS id dt4si11767074vdb.93.2012.01.03.14.17.48
        (version=TLSv1/SSLv3 cipher=OTHER);
        Tue, 03 Jan 2012 14:17:48 -0800 (PST)
Received-SPF: pass (google.com: domain of althoff.tim@googlemail.com designates
209.85.220.179 as permitted sender) client-ip=209.85.220.179;
Received: by vcbf13 with SMTP id f13so11295098vcb.10
        for <alex@smola.org>; Tue, 03 Jan 2012 14:17:48 -0800 (PST)
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
        d=googlemail.com; s=gamma;
        h=mime-version:sender:date:x-google-sender-auth:message-id:subject
         :from:to:content-type;
        bh=WCbdZ5sXac25dpH02XcRyDOdts993hKwsAVXpGrFh0w=;
        b=WK2B2+ExWnf/gvTkw6uUvKuP4XeoKnlJq3USYTm0RARK8dSFjyOQsIHeAP9Yssxp60
         7ngGoTzYqd+ZsyJfvQcLAWp1PCJhG8AMcnqWkx0NMeoFvIp2HQooZwxSOCx5ZRgY+7qX
         uIbbdna4lUDXj6UFe16SpLDCkptd8OZ3gr7+o=
MIME-Version: 1.0
Received: by 10.220.108.81 with SMTP id e17mr24104004vcp.67.1325629067787;
 Tue, 03 Jan 2012 14:17:47 -0800 (PST)
Sender: althoff.tim@googlemail.com
Received: by 10.220.17.129 with HTTP; Tue, 3 Jan 2012 14:17:47 -0800 (PST)
Date: Tue, 3 Jan 2012 14:17:47 -0800
X-Google-Sender-Auth: 6bwi6D17HjZIkxOEol38NZzyeHs
Message-ID: <CAFJJHDGPBW+SdZg0MdAABiAKydDk9tpeMoDijYGjoGO-WC7osg@mail.gmail.com>
Subject: CS 281B. Advanced Topics in Learning and Decision Making
From: Tim Althoff <althoff@eecs.berkeley.edu>
To: alex@smola.org
Content-Type: multipart/alternative; boundary=f46d043c7af4b07e8d04b5a7113a

--f46d043c7af4b07e8d04b5a7113a
Content-Type: text/plain; charset=ISO-8859-1
```

- bag of words
- pairs of words
- date & time
- recipient path
- IP number
- sender
- encoding
- links
- ... secret sauce ...

More feature engineering

- Two Interlocking Spirals
  Transform the data into a radial and angular part
  $$(x_1, x_2) = (r \sin \phi, r \cos \phi)$$
- Handwritten Japanese Character Recognition
  - Break down the images into strokes and recognize it
  - Lookup based on stroke order
- Medical Diagnosis
  - Physician's comments
  - Blood status / ECG / height / weight / temperature ...
  - Medical knowledge

The Perceptron on features

The Perceptron on features

initialize $w, b = 0$
repeat

The Perceptron on features

initialize $w, b = 0$
repeat
    Pick (x_i, y_i) from data

The Perceptron on features

initialize $w, b = 0$
repeat
    Pick (x_i, y_i) from data
    if $y_i(w \cdot \Phi(x_i) + b) \leq 0$ then

$$\begin{aligned} w' &= w + y_i \Phi(x_i) \\ b' &= b + y_i \end{aligned}$$

until $y_i(w \cdot \Phi(x_i) + b) > 0$ for all i

The Perceptron on features

initialize $w, b = 0$
repeat
    Pick (x_i, y_i) from data
    if $y_i(w \cdot \Phi(x_i) + b) \leq 0$ then

$$\begin{aligned} w' &= w + y_i\Phi(x_i) \\ b' &= b + y_i \end{aligned}$$

until $y_i(w \cdot \Phi(x_i) + b) > 0$ for all i

- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum y_i \phi(x_i)$

The Perceptron on features

initialize $w, b = 0$
repeat
Pick (x_i, y_i) from data
if $y_i(w \cdot \Phi(x_i) + b) \leq 0$ then

$$\begin{aligned} w' &= w + y_i \Phi(x_i) \\ b' &= b + y_i \end{aligned}$$

until $y_i(w \cdot \Phi(x_i) + b) > 0$ for all i

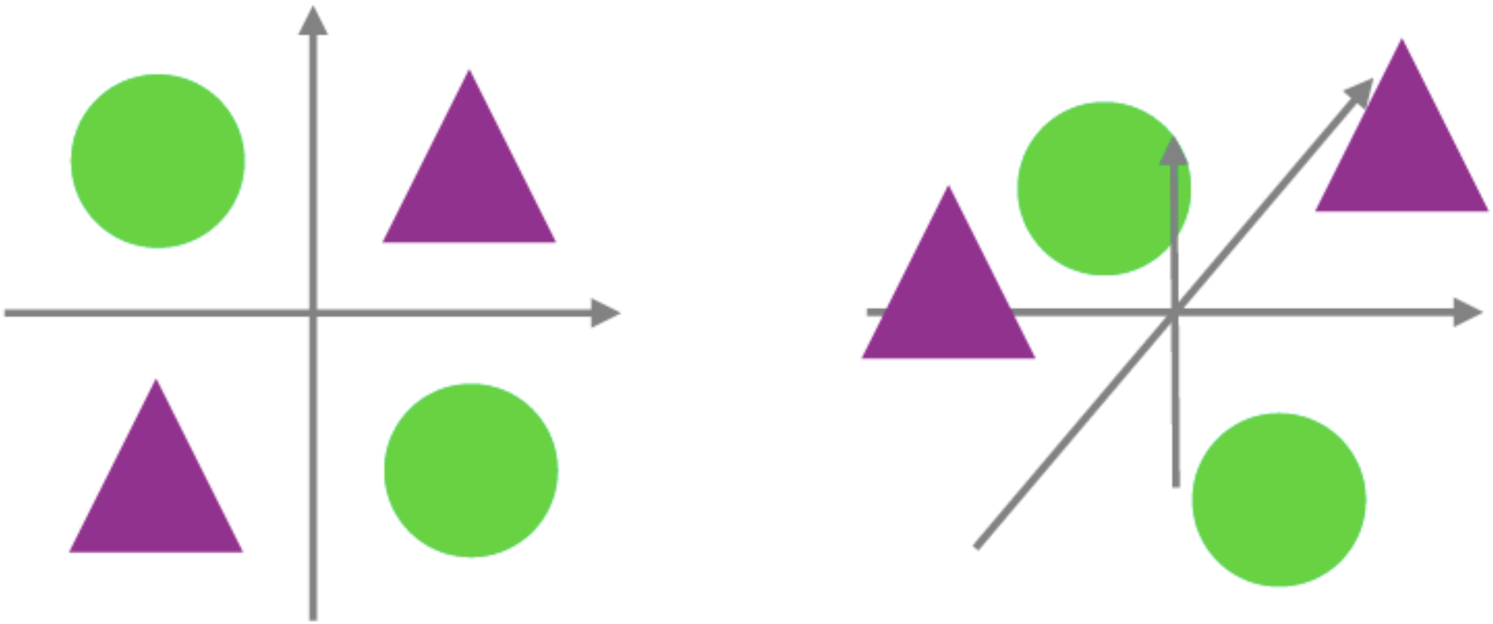
- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i \phi(x_i)$
- Classifier is linear combination of inner products $f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b$

Problems

- Problems
  - Need domain expert (e.g. Chinese OCR)
  - Often expensive to compute
  - Difficult to transfer engineering knowledge
- Shotgun Solution
  - Compute many features
  - Hope that this contains good ones
  - Do this efficiently

Kernels

Solving XOR



(x_1, x_2) $(x_1, x_2, x_1 x_2)$

- XOR not linearly separable
- Mapping into 3 dimensions makes it easily solvable

Quadratic Features

Quadratic Features

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

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

Quadratic Features

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

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

Dot Product

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

Quadratic Features

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

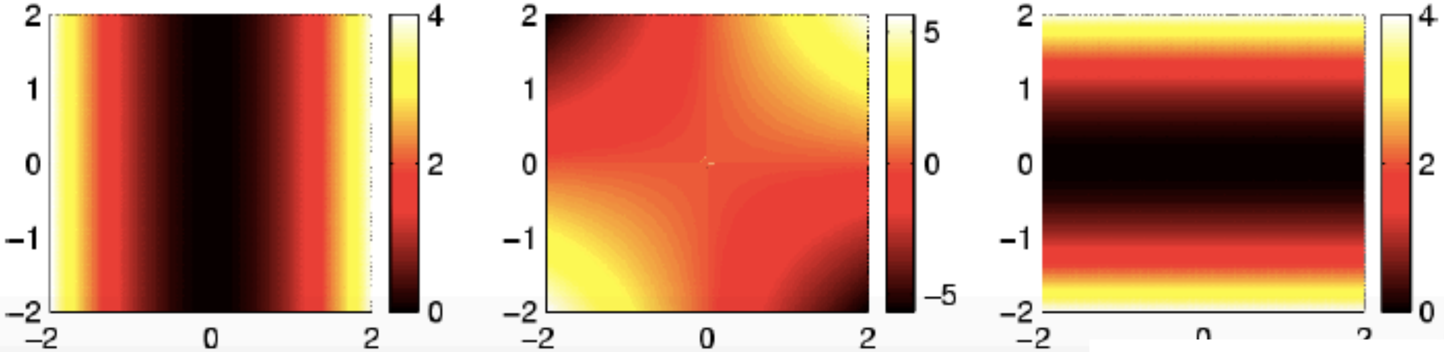
$$\Phi(x) := \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

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 $\langle x, x'\rangle^d$.



Computational Efficiency

Problem

- Extracting features can sometimes be very costly.
- Example: second order features in 1000 dimensions. This leads to $5 \cdot 10^5$ numbers. For higher order polynomial features much worse.

Computational Efficiency

Problem

- Extracting features can sometimes be very costly.
- Example: second order features in 1000 dimensions. This leads to $5 \cdot 10^5$ 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 . . .

Computational Efficiency

Problem

- Extracting features can sometimes be very costly.
- Example: second order features in 1000 dimensions. This leads to $5 \cdot 10^5$ 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 $\Phi(x)$ ...

The Kernel Perceptron

The Kernel Perceptron

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

The Kernel Perceptron

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

- Nothing happens if classified correctly

The Kernel Perceptron

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

- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i \phi(x_i)$

The Kernel Perceptron

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

- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i \phi(x_i)$
- Classifier is linear combination of inner products

$$f(x) = \sum_{i \in I} y_i \langle \phi(x_i), \phi(x) \rangle + b = \sum_{i \in I} y_i k(x_i, x) + b$$

5 minutes break

Polynomial Kernels

Idea

Polynomial Kernels

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.

Polynomial Kernels

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 Conditions

Computability

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

Kernel Conditions

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.

Kernel Conditions

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 $\langle \Phi(x), \Phi(x') \rangle = \langle \Phi(x'), \Phi(x) \rangle$.

Kernel Conditions

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

Mercer's Theorem

The Theorem

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

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

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

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

Mercer's Theorem

The Theorem

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

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

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

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

Interpretation

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

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

Properties

Properties

Distance in Feature Space

Distance between points in feature space via

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

Properties

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.

Properties

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

Properties

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:

Properties

K is Positive Semidefinite

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

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

Properties

K is Positive Semidefinite

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

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

Kernel Expansion

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

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

A Counterexample

A 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 . . .

A Counterexample

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.

Examples

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

Examples

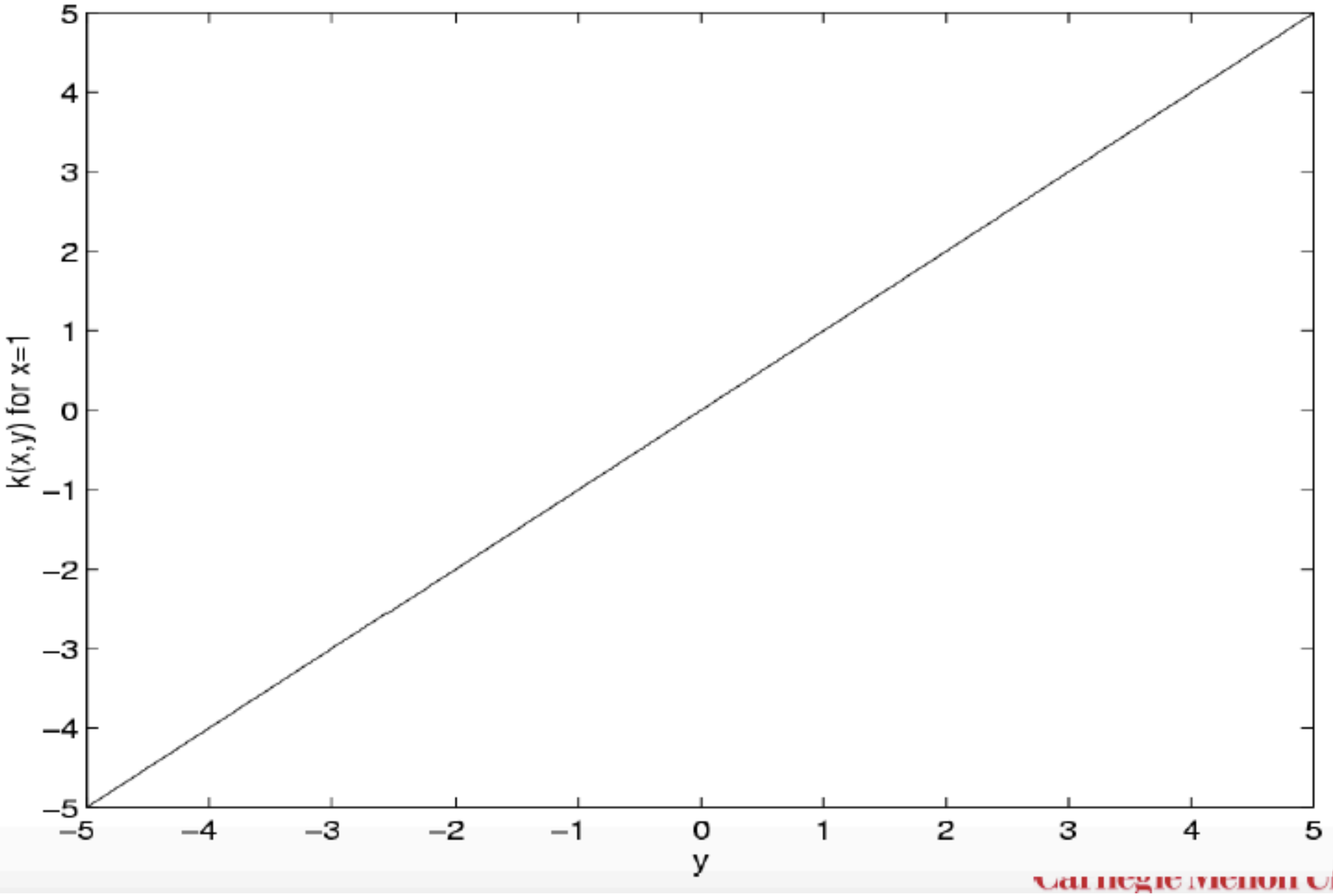
Examples of kernels $k(x, x')$

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

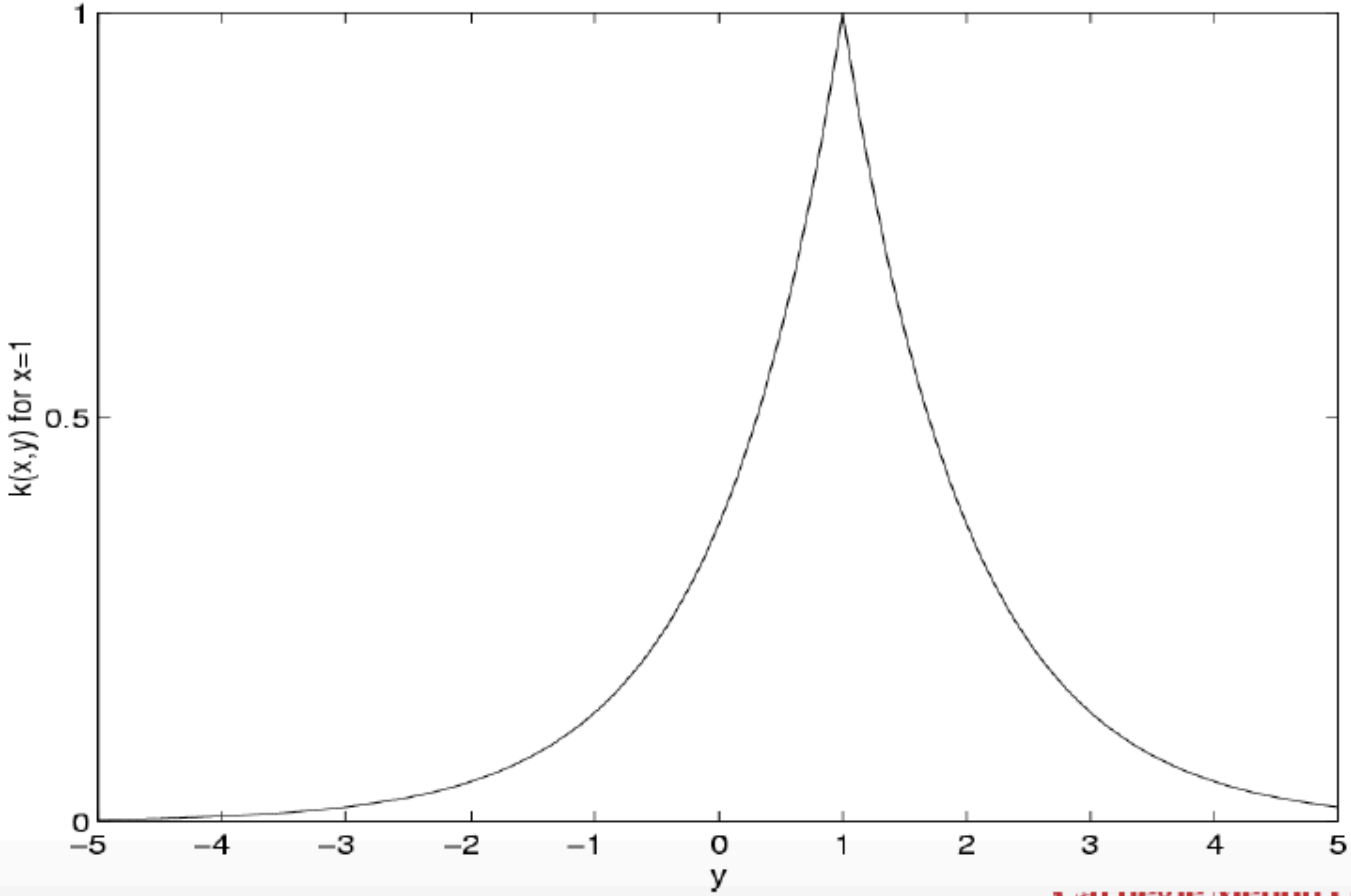
Simple trick for checking Mercer's condition

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

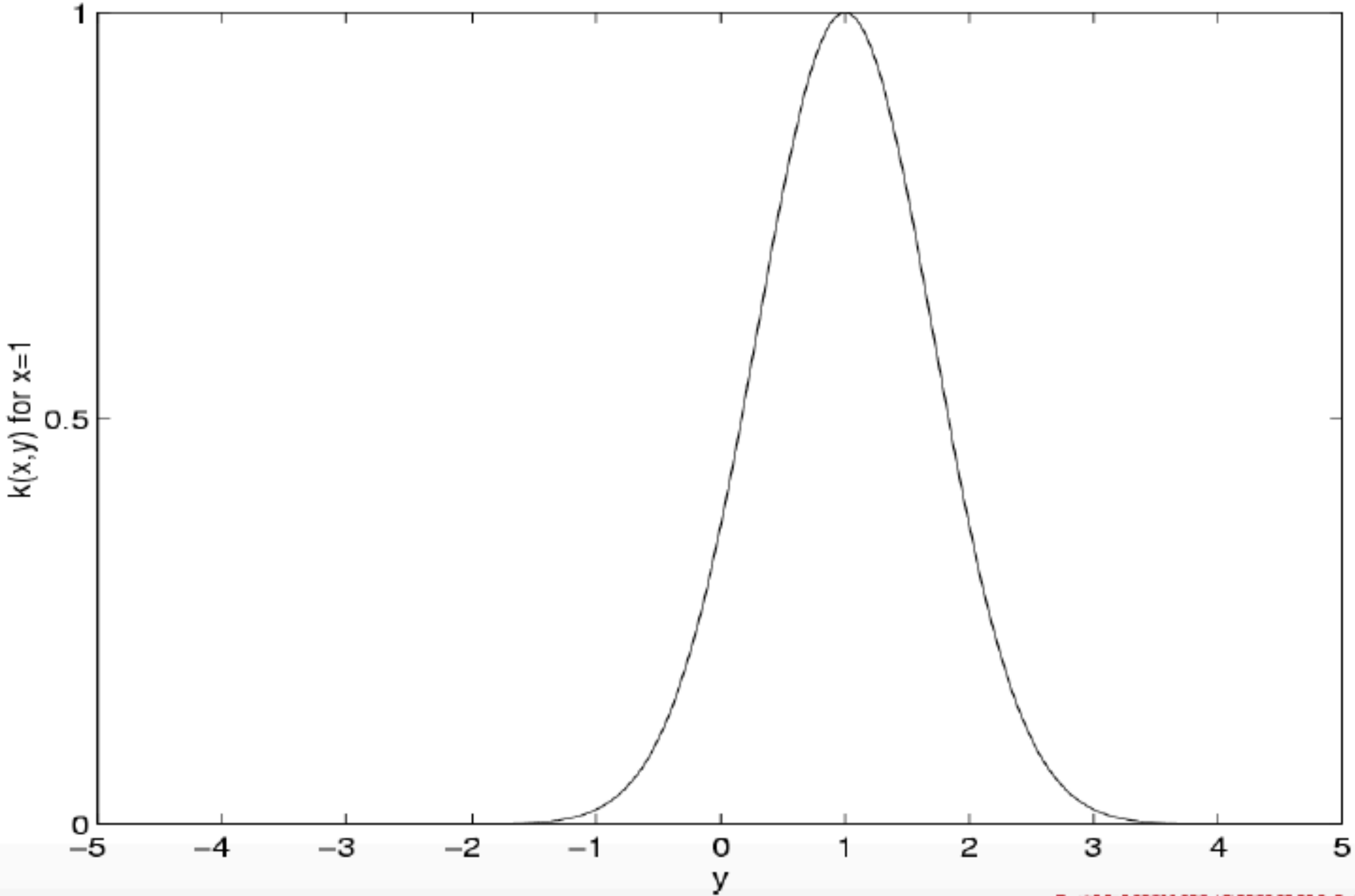
Linear Kernel



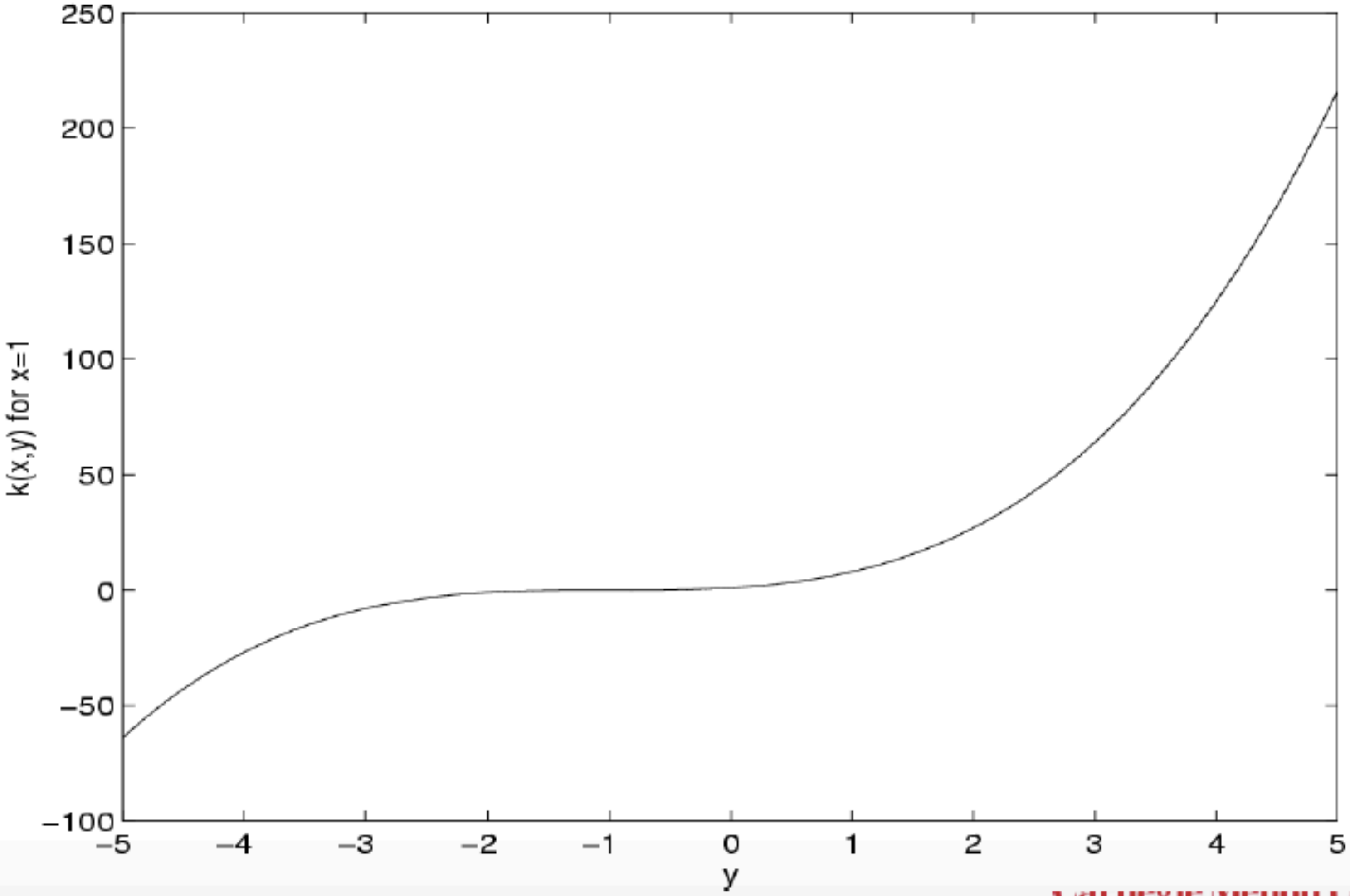
Laplacian Kernel



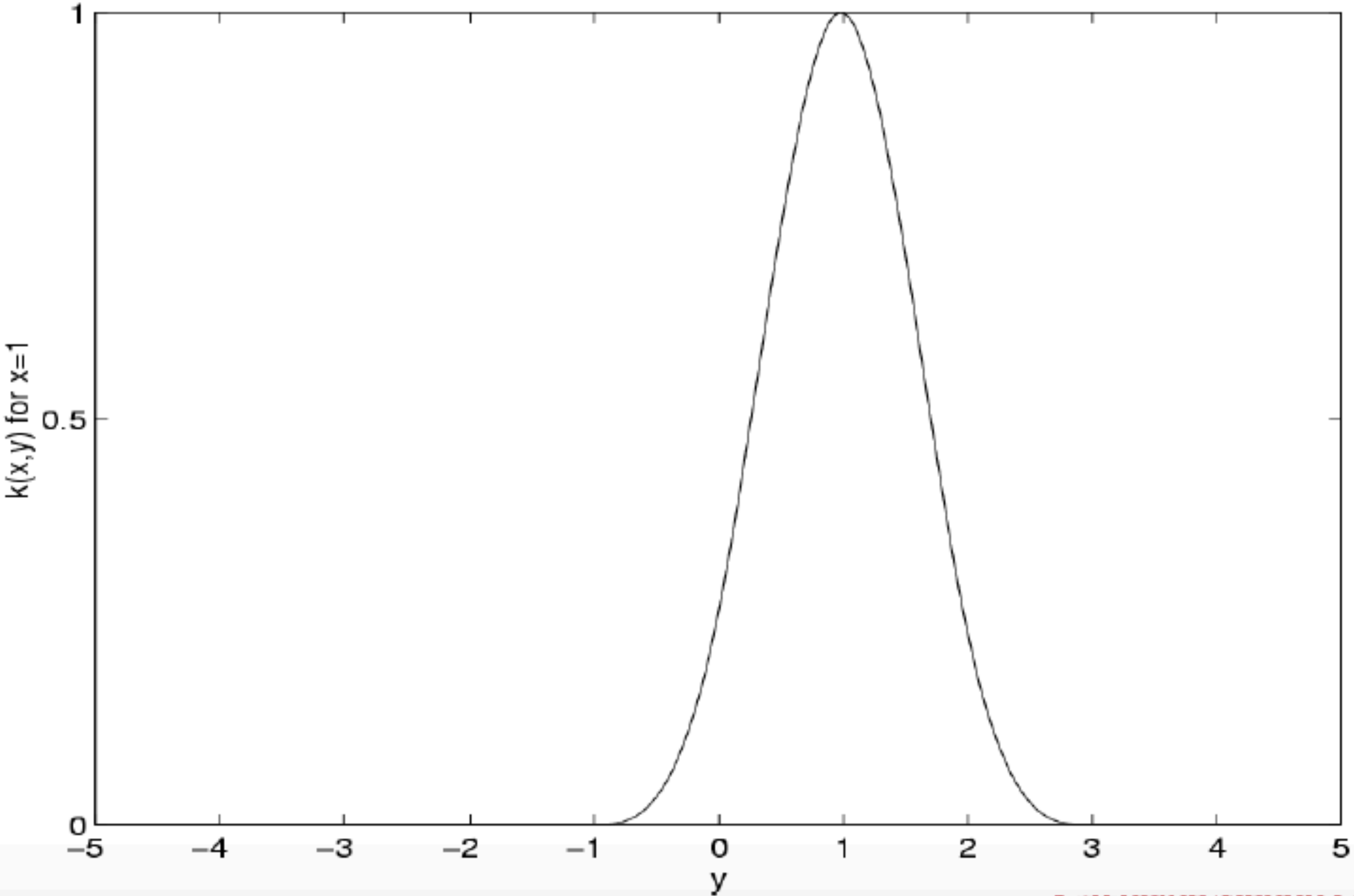
Gaussian Kernel



Polynomial of order 3



B_3 Spline Kernel



5 minutes break!

Parzen Windows

Density Estimation

- Observe some data x_i
- Want to estimate $p(x)$

Density Estimation

- Observe some data x_i
- Want to estimate p(x)
  - Find unusual observations (e.g. security)
  - Find typical observations (e.g. prototypes)

Density Estimation

- Observe some data x_i
- Want to estimate p(x)
  - Find unusual observations (e.g. security)
  - Find typical observations (e.g. prototypes)
  - Classifier via Bayes Rule

$$p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(x|y)p(y)}{\sum_{y'} p(x|y')p(y')}$$

- Need tool for computing p(x) easily

Bin Counting

- Discrete random variables, e.g.
  - English, Chinese, German, French, ...
  - Male, Female
- Bin counting (record # of occurrences)

25	English	Chinese	German	French	Spanish
male	5	2	3	1	0
female	6	3	2	2	1

Bin Counting

- Discrete random variables, e.g.
  - English, Chinese, German, French, ...
  - Male, Female
- Bin counting (record # of occurrences)

25	English	Chinese	German	French	Spanish
male	0.2	0.08	0.12	0.04	0
female	0.24	0.12	0.08	0.08	0.04

Bin Counting

- Discrete random variables, e.g.
  - English, Chinese, German, French, ...
  - Male, Female
- Bin counting (record # of occurrences)

25	English	Chinese	German	French	Spanish
male	0.2	0.08	0.12	0.04	0
female	0.24	0.12	0.08	0.08	0.04

Bin Counting

- Discrete random variables, e.g.
  - English, Chinese, German, French, ...
  - Male, Female
- Bin counting (record # of occurrences)

not enough data

25	English	Chinese	German	French	Spanish
male	0.2	0.08	0.12	0.04	0
female	0.24	0.12	0.08	0.08	0.04

Curse of dimensionality (lite)

- Discrete random variables, e.g.
  - English, Chinese, German, French, ...
  - Male, Female
  - ZIP code
  - Day of the week
  - Operating system
  - ...

#bins grows exponentially

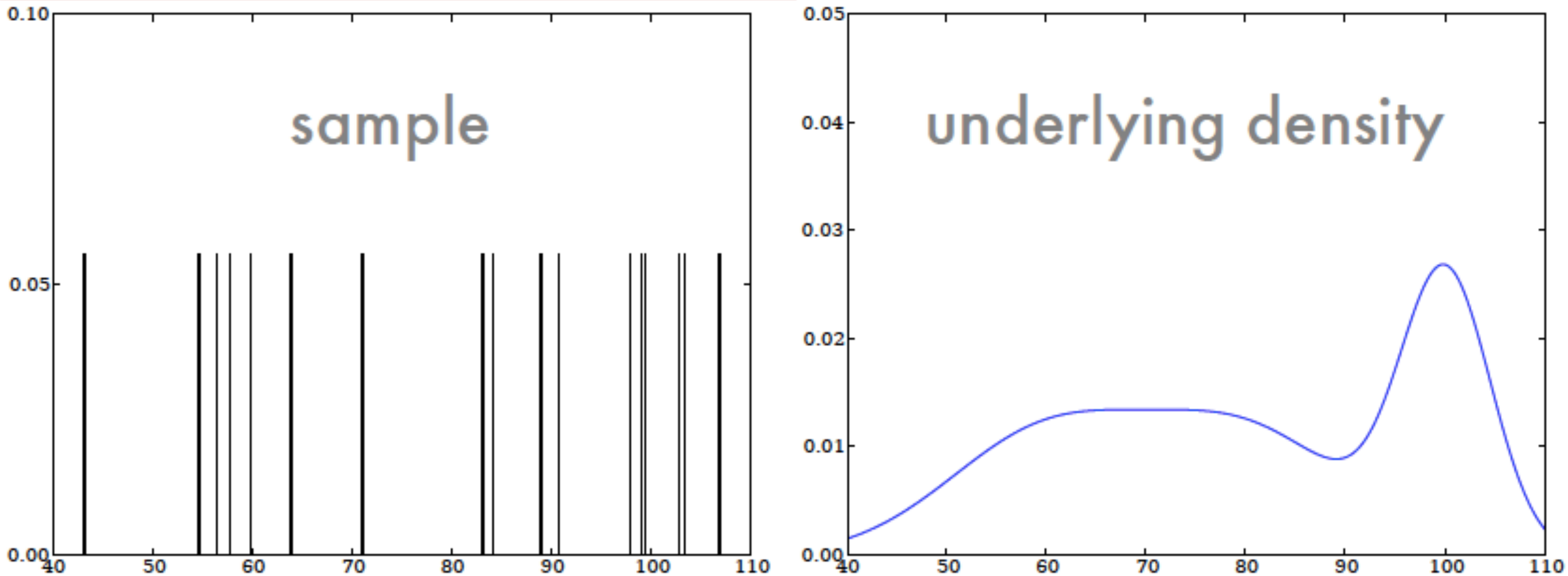
Curse of dimensionality (lite)

- Discrete random variables, e.g.
  - English, Chinese, German, French, ...
  - Male, Female
  - ZIP code
  - Day of the week
  - Operating system
  - ...
- Continuous random variables
  - Income
  - Bandwidth
  - Time

#bins grows exponentially

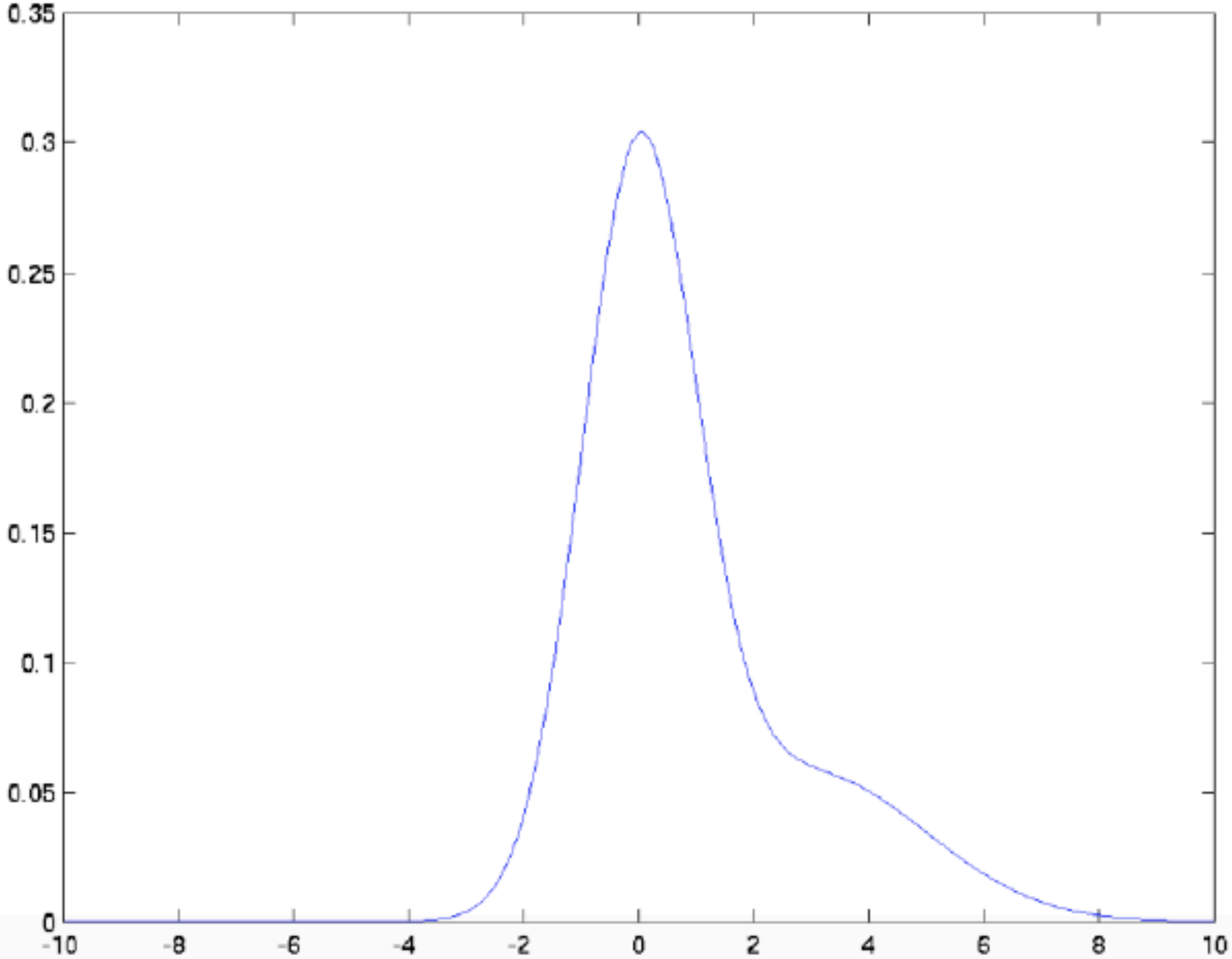
need many bins per dimension

Density Estimation

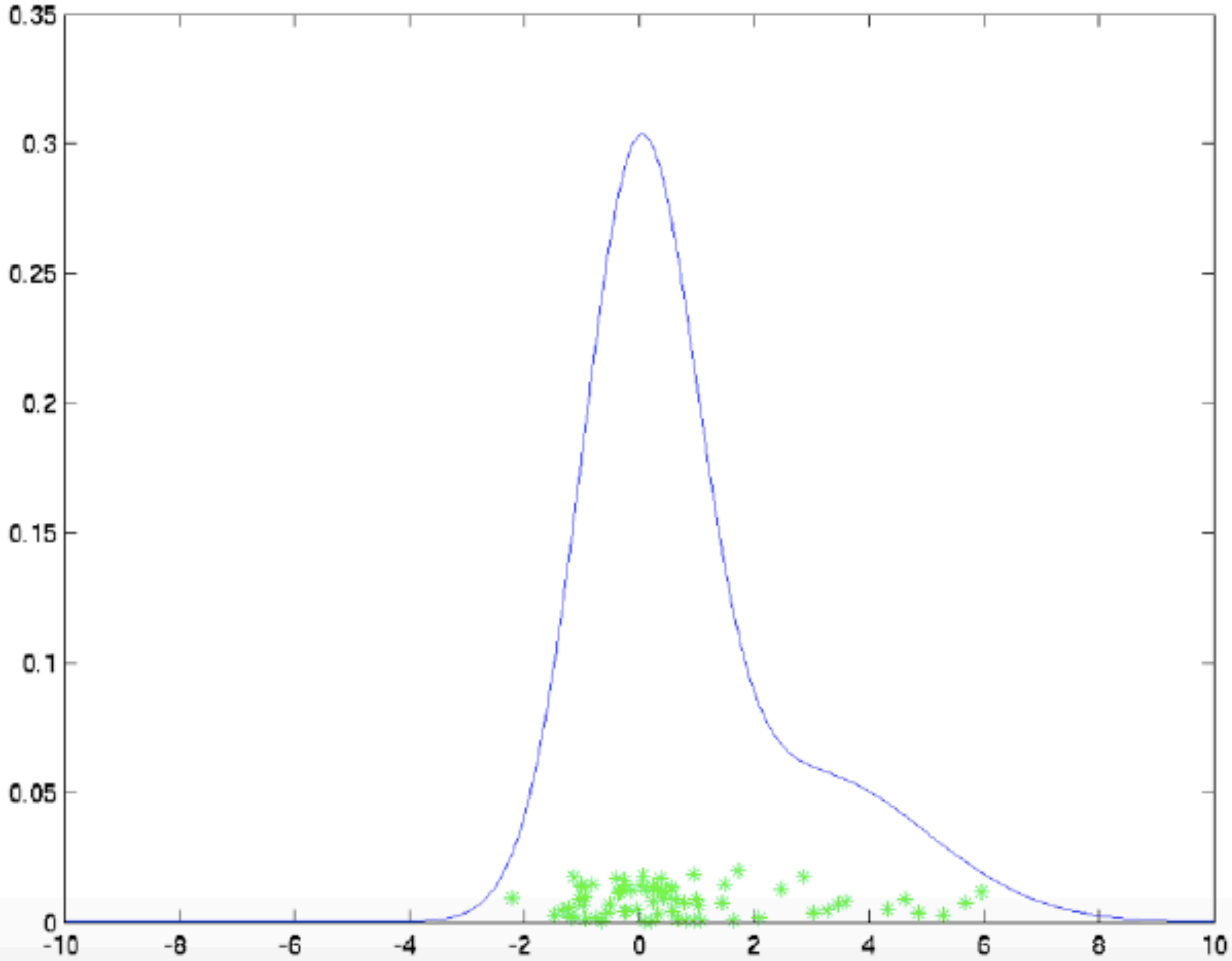


- Continuous domain = infinite number of bins
- Curse of dimensionality
  - 10 bins on [0, 1] is probably good
  - 10^{10} bins on $[0, 1]^{10}$ requires high accuracy in estimate: probability mass per cell also decreases by 10^{10}

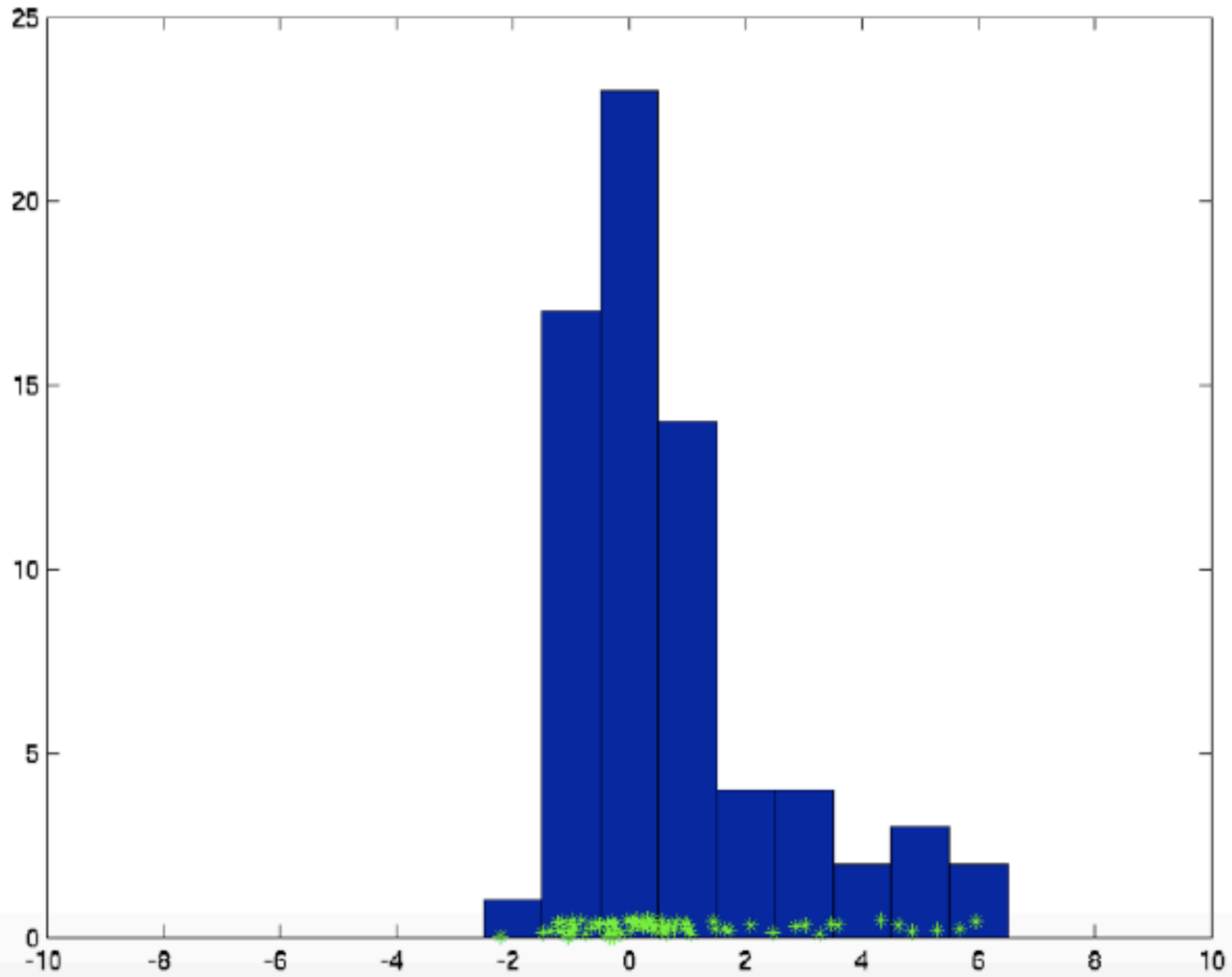
Bin Counting



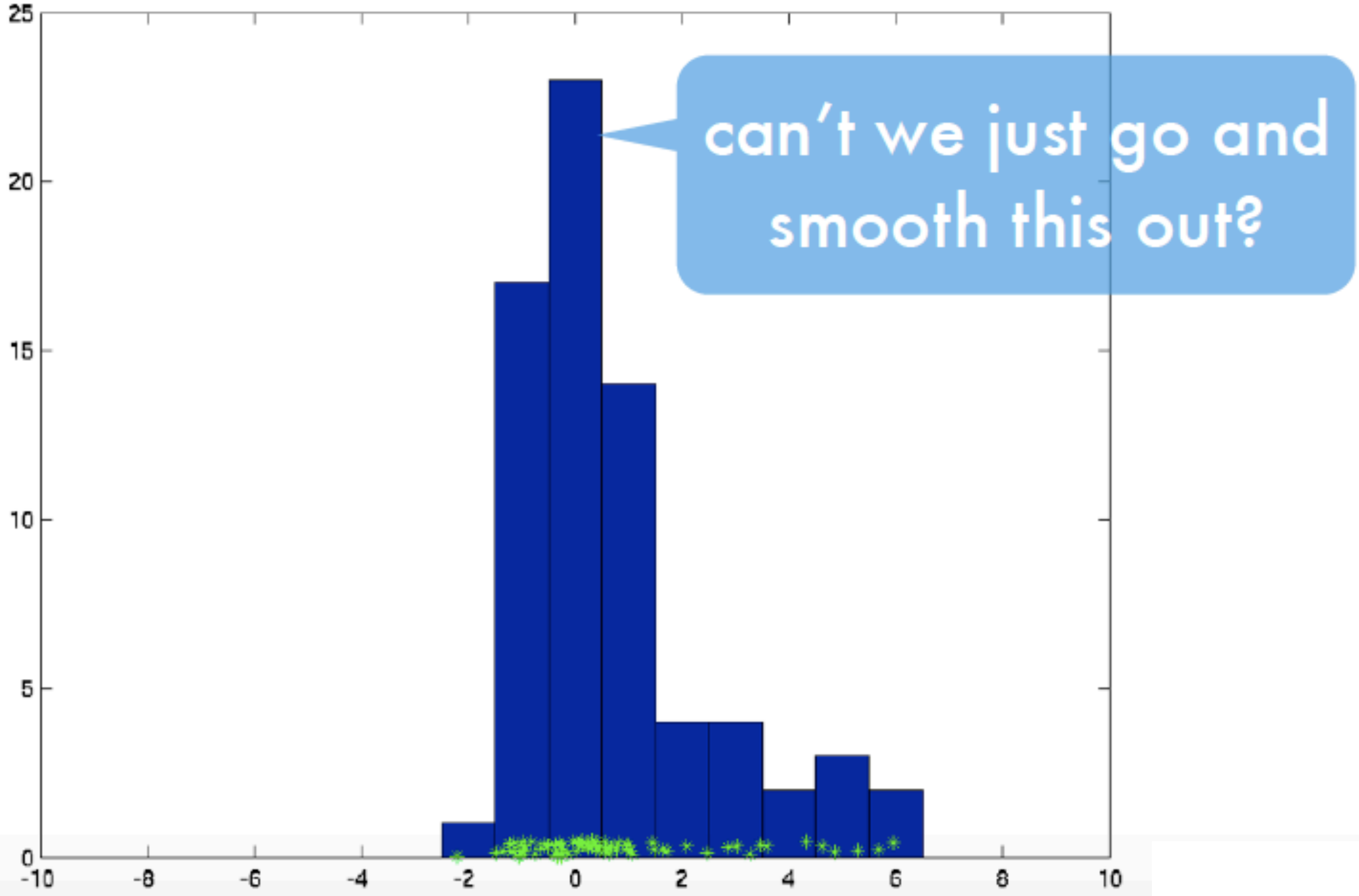
Bin Counting



Bin Counting



Bin Counting



Parzen Windows

- Naive approach
  Use empirical density (delta distributions)

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

Parzen Windows

- Naive approach
  Use empirical density (delta distributions)

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

- This breaks if we see slightly different instances
- Kernel density estimate

Parzen Windows

- Naive approach
  Use empirical density (delta distributions)

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

- This breaks if we see slightly different instances
- Kernel density estimate
  Smear out empirical density with a nonnegative smoothing kernel $k_x(x')$ satisfying

$$\int_{\mathcal{X}} k_x(x') dx' = 1 \text{ for all } x$$

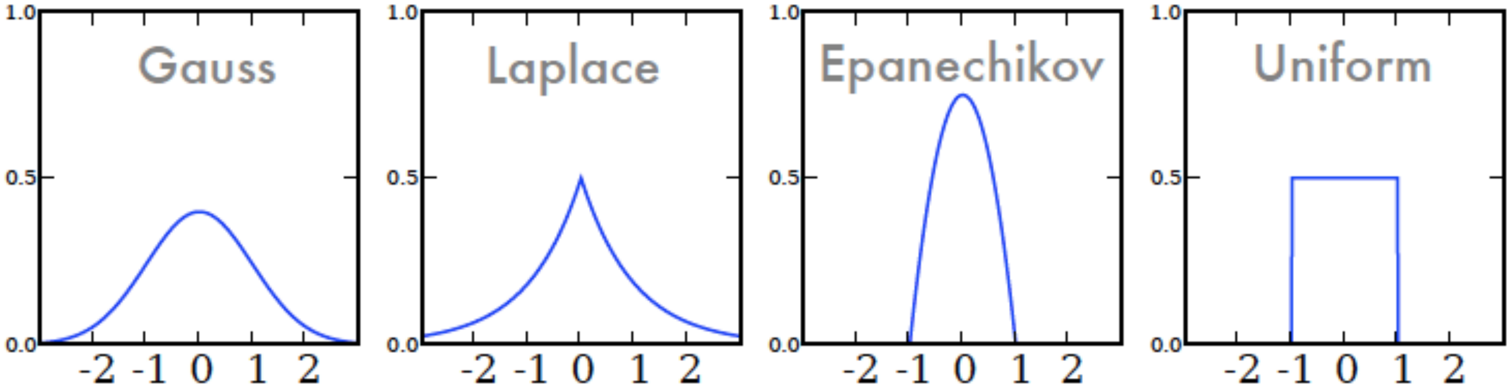
Parzen Windows

- **Density estimate**

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

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

- **Smoothing kernels**



$(2\pi)^{-\frac{1}{2}}e^{-\frac{1}{2}x^2}$ $\quad$ $\frac{1}{2}e^{-|x|}$ $\quad$ $\frac{3}{4}\max(0, 1-x^2)$ $\quad$ $\frac{1}{2}\chi_{[-1,1]}(x)$

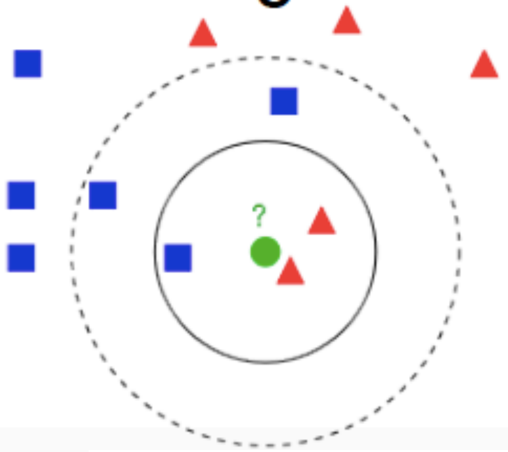
Nearest Neighbor

Nearest Neighbors

- Table lookup
  For previously seen instance remember label

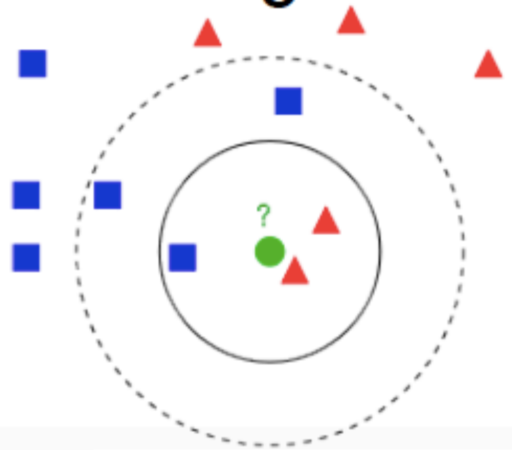
Nearest Neighbors

- Table lookup
  For previously seen instance remember label
- Nearest neighbor
  - Pick label of most similar neighbor
  - Slight improvement - use k-nearest neighbors



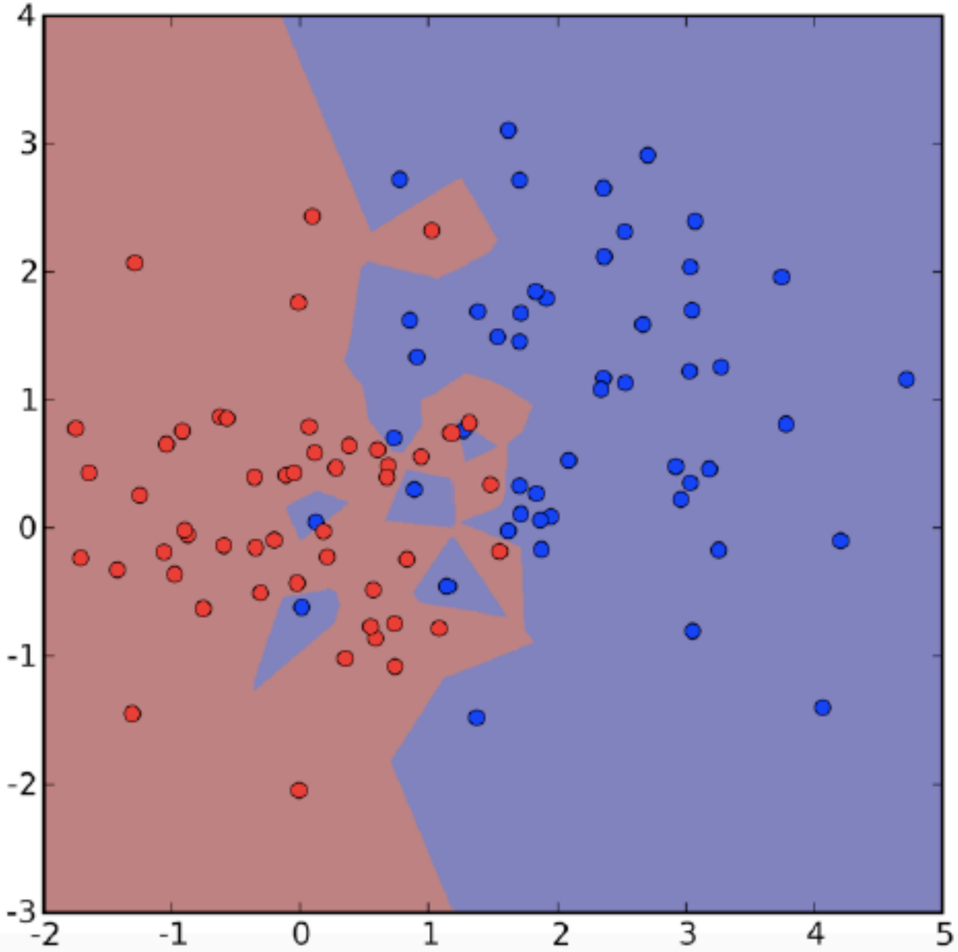
Nearest Neighbors

- Table lookup
  For previously seen instance remember label
- Nearest neighbor
  - Pick label of most similar neighbor
  - Slight improvement - use k-nearest neighbors

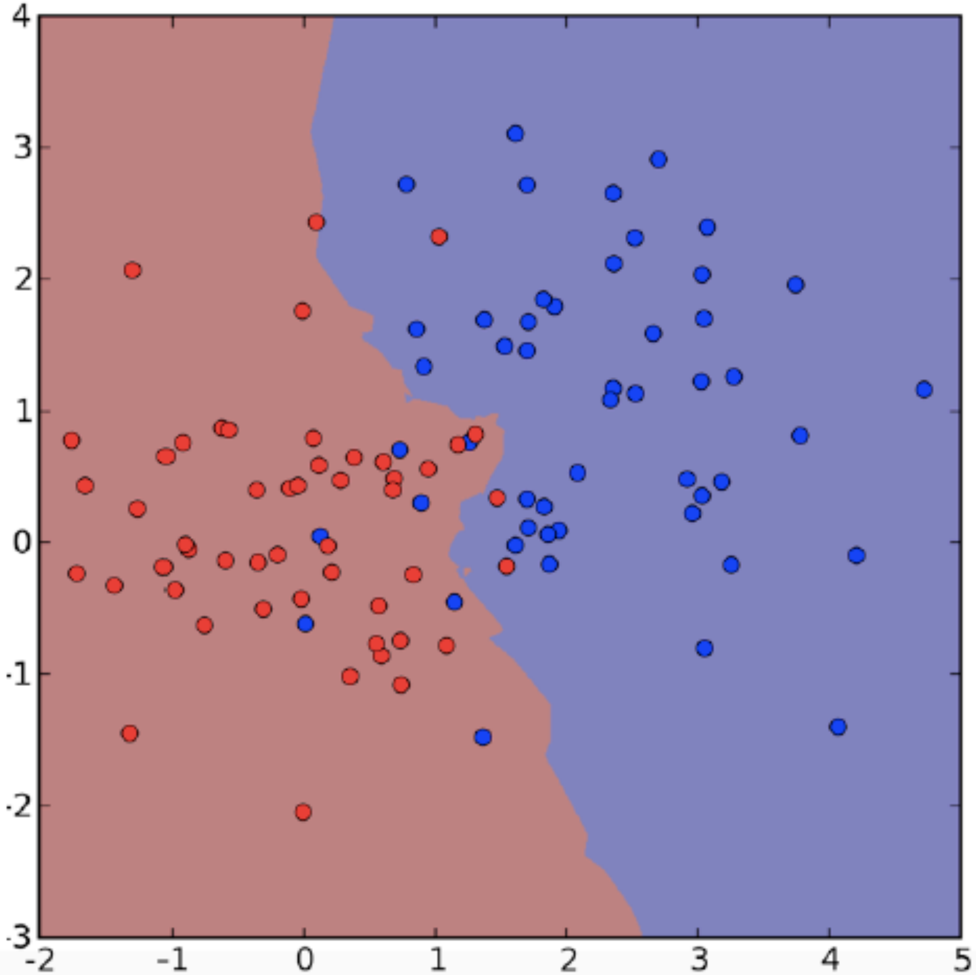


- Really useful baseline!
- Easy to implement for small amounts of data.

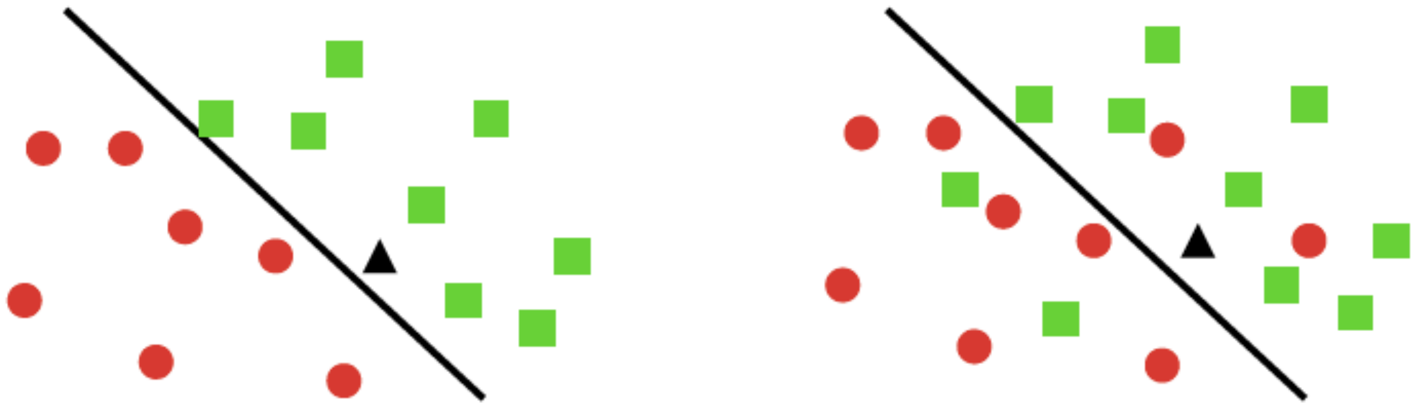
1-Nearest Neighbor



4-Nearest Neighbors Sign

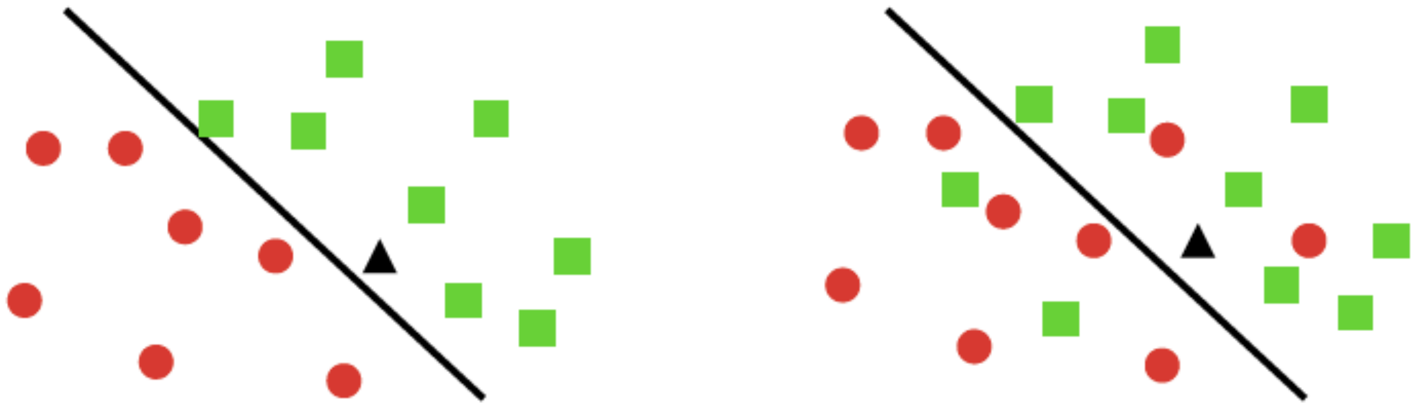


If we get more data



- 1 Nearest Neighbor
  - Converges to perfect solution if separation
  - Twice the minimal error rate 2p(1-p) for noisy problems

If we get more data



- 1 Nearest Neighbor
  - Converges to perfect solution if separation
  - Twice the minimal error rate 2p(1-p) for noisy problems
- k-Nearest Neighbor
  - Converges to perfect solution if separation (but needs more data)
  - Converges to minimal error min(p,1-p) for noisy problems (use increasing k)

That's all