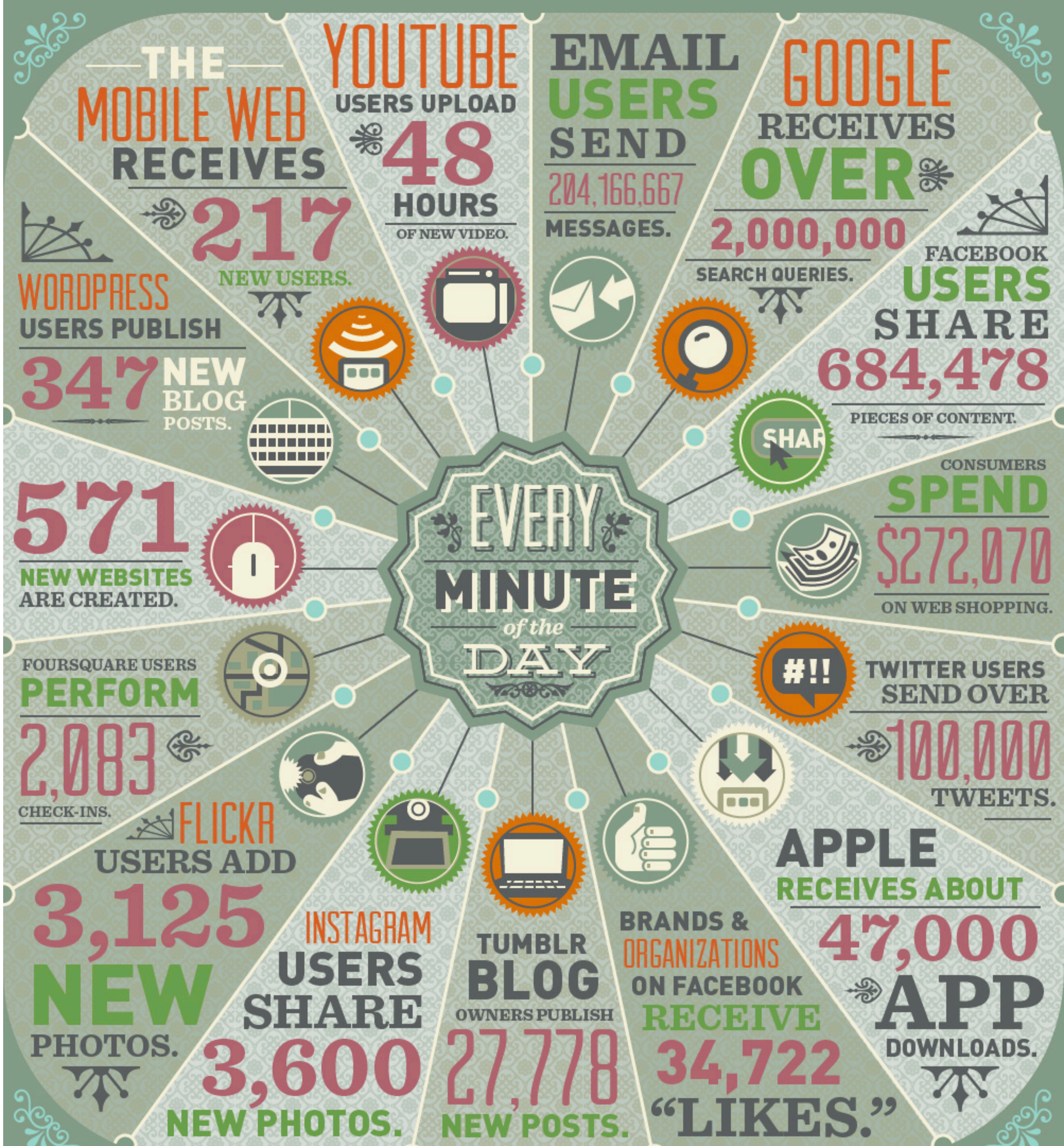
# Introduction to Machine Learning

## 2. Basic Tools

Alex Smola & Geoff Gordon
Carnegie Mellon University

http://alex.smola.org/teaching/cmu2013-10-701x
10-701

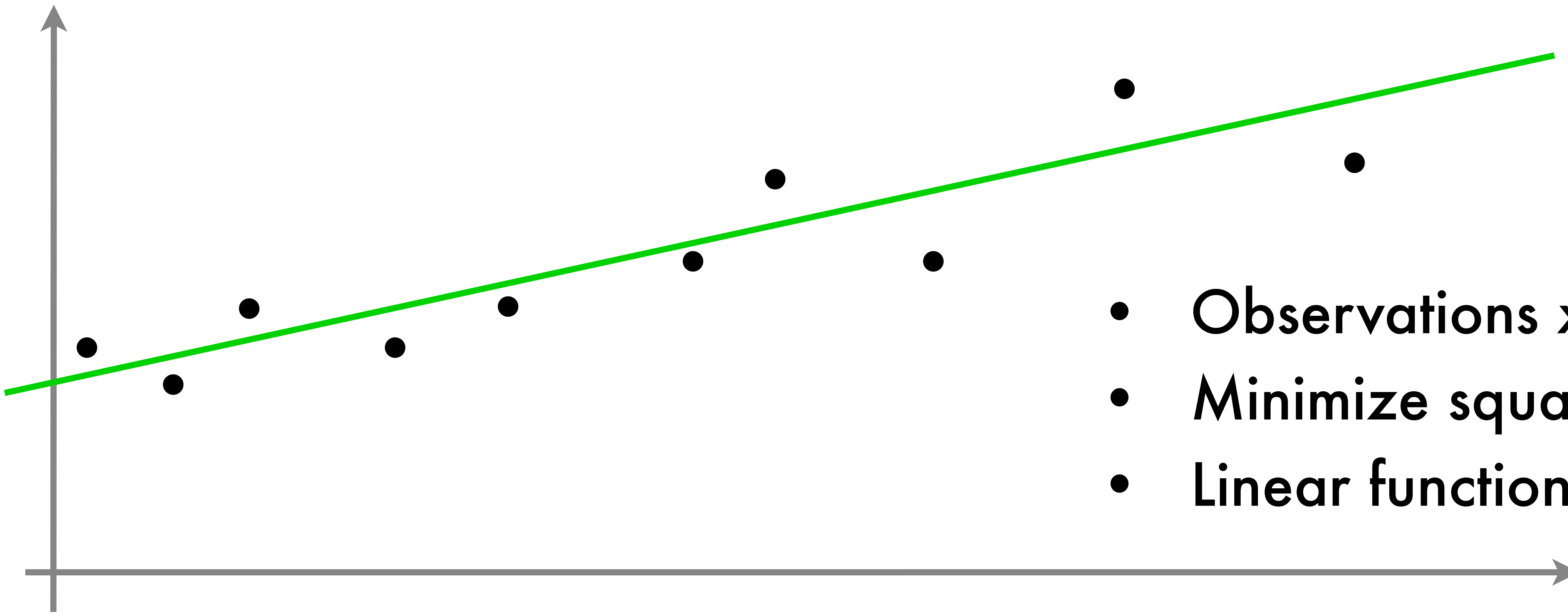
# This is not a toy dataset




http://wadejohnston1962.files.wordpress.com/2012/09/datainoneminute.jpg

# Linear Regression

# Linear Regression



- Observations x, labels y
- Minimize squared distance
- Linear function

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

$$\underset{a,b}{\text{minimize}} \sum_{i=1}^{m} \frac{1}{2}(ax_i + b - y_i)^2$$

$$\partial_a [\ldots] = 0 = \sum_{i=1}^{m} x_i(ax_i + b - y_i)$$

$$\partial_b [\ldots] = 0 = \sum_{i=1}^{m} (ax_i + b - y_{)}$$

# Linear Regression

- Optimization Problem

$$f(x) = \langle a, x \rangle + b = \langle w, (x, 1) \rangle$$

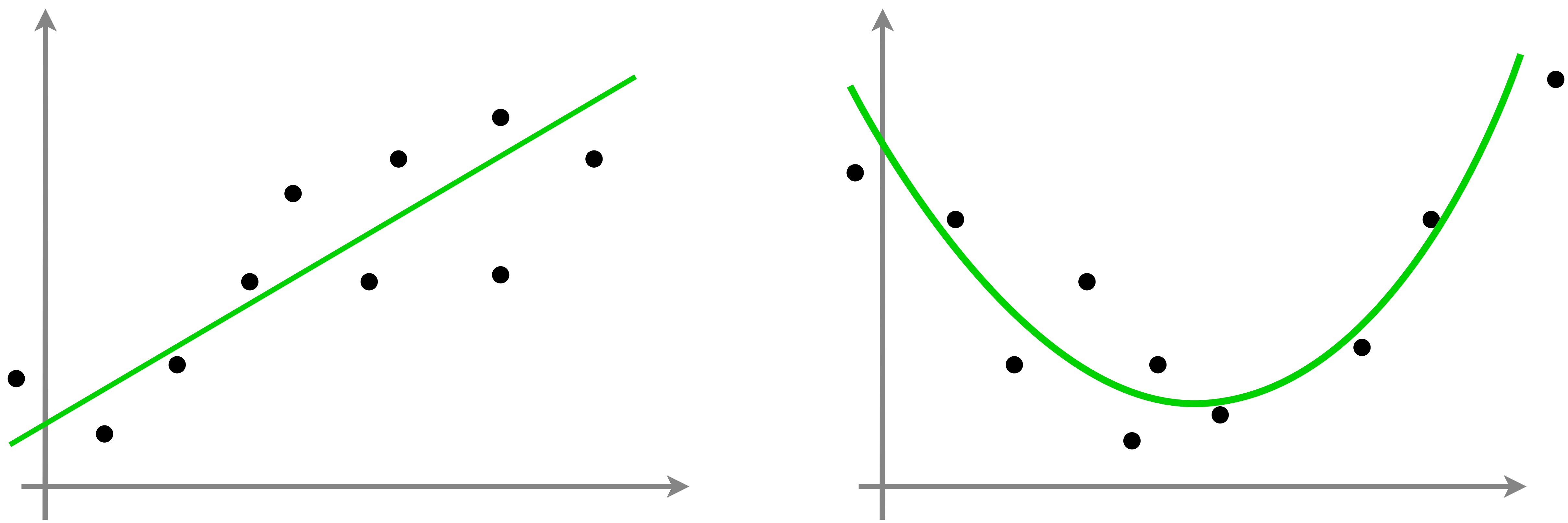
$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} \frac{1}{2} \left( \langle w, \bar{x}_i \rangle - y_i \right)^2$$

- Solving it

$$0 = \sum_{i=1}^{m} \bar{x}_i \left( \langle w, \bar{x}_i \rangle - y_i \right) \Longleftrightarrow \left[ \sum_{i=1}^{m} \bar{x}_i \bar{x}_i^\top \right] w = \sum_{i=1}^{m} y_i \bar{x}_i$$

only requires a matrix inversion.

# Nonlinear Regression



- Linear model $f(x) = \langle w, (1, x) \rangle$
- Quadratic model $f(x) = \left\langle w, (1, x, x^2) \right\rangle$
- Cubic model $f(x) = \left\langle w, (1, x, x^2, x^3) \right\rangle$
- Nonlinear model $f(x) = \langle w, \phi(x) \rangle$

# Linear Regression

- Optimization Problem

$$f(x) = \langle a, x \rangle + b = \langle w, (x, 1) \rangle$$

$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} \frac{1}{2} (\langle w, \bar{x}_i \rangle - y_i)^2$$

- Solving it

$$0 = \sum_{i=1}^{m} \bar{x}_i (\langle w, \bar{x}_i \rangle - y_i) \iff \boxed{\left[ \sum_{i=1}^{m} \bar{x}_i \bar{x}_i^\top \right]} w = \sum_{i=1}^{m} y_i \bar{x}_i$$

only requires a matrix inversion.

# Nonlinear Regression

- Optimization Problem

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

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

- Solving it

$$\sum_{i=1}^{m} \phi(x_i)(\langle w, \phi(x_i) \rangle - y_i) \Longleftrightarrow \left[ \sum_{i=1}^{m} \phi(x_i)\phi(x_i)^\top \right] w = \sum_{i=1}^{m} y_i \phi(x_i)$$

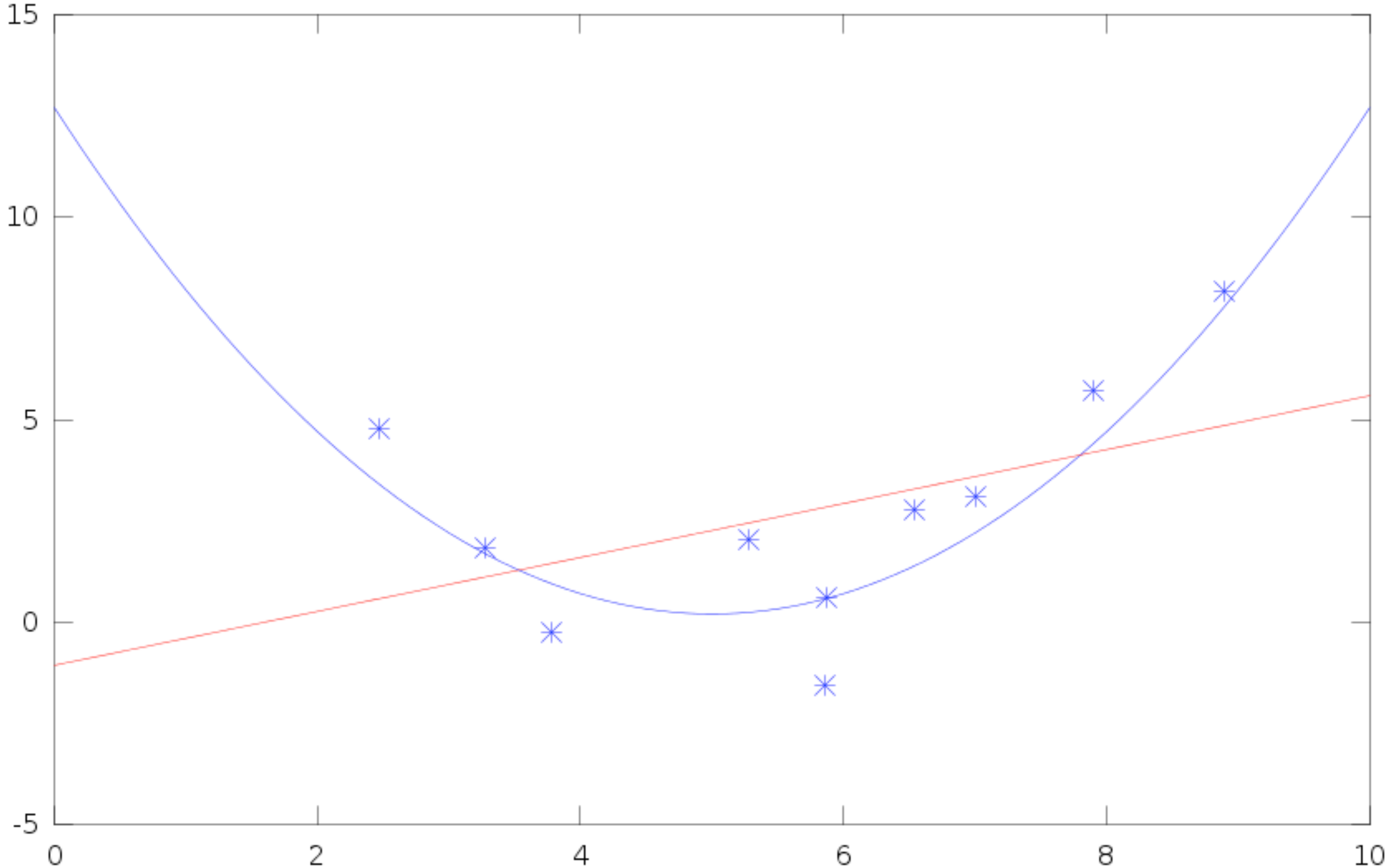
only requires a matrix inversion.

# Pseudocode (degree 4)

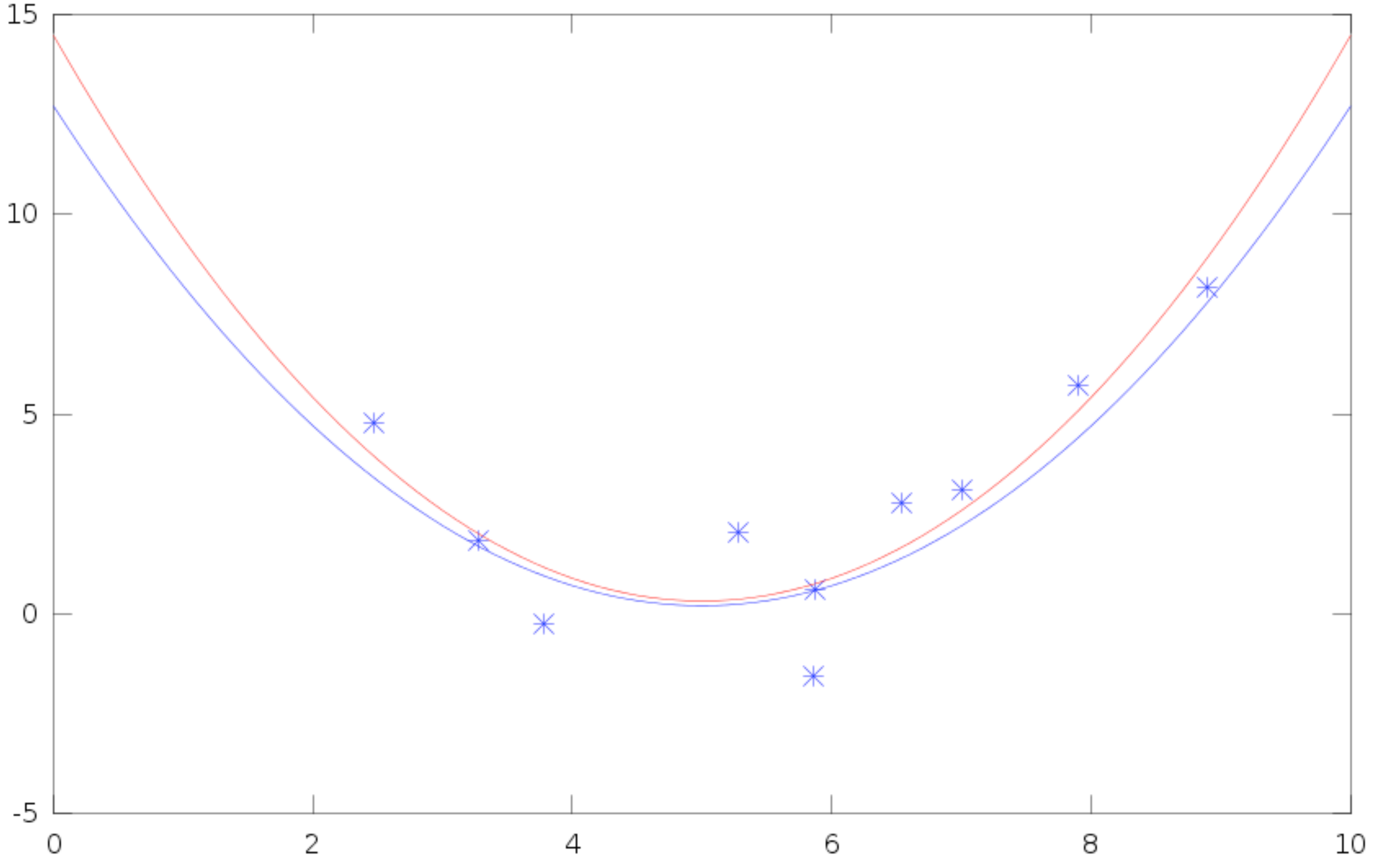
```
Training
phi_xx = [xx.^4, xx.^3, xx.^2, xx, 1.0 + 0.0 * xx];
w = (yy' * phi_xx) / (phi_xx' * phi_xx);

Testing
phi_x = [x.^4, x.^3, x.^2, x, 1.0 + 0.0 * x];
y = phi_x * w';
```

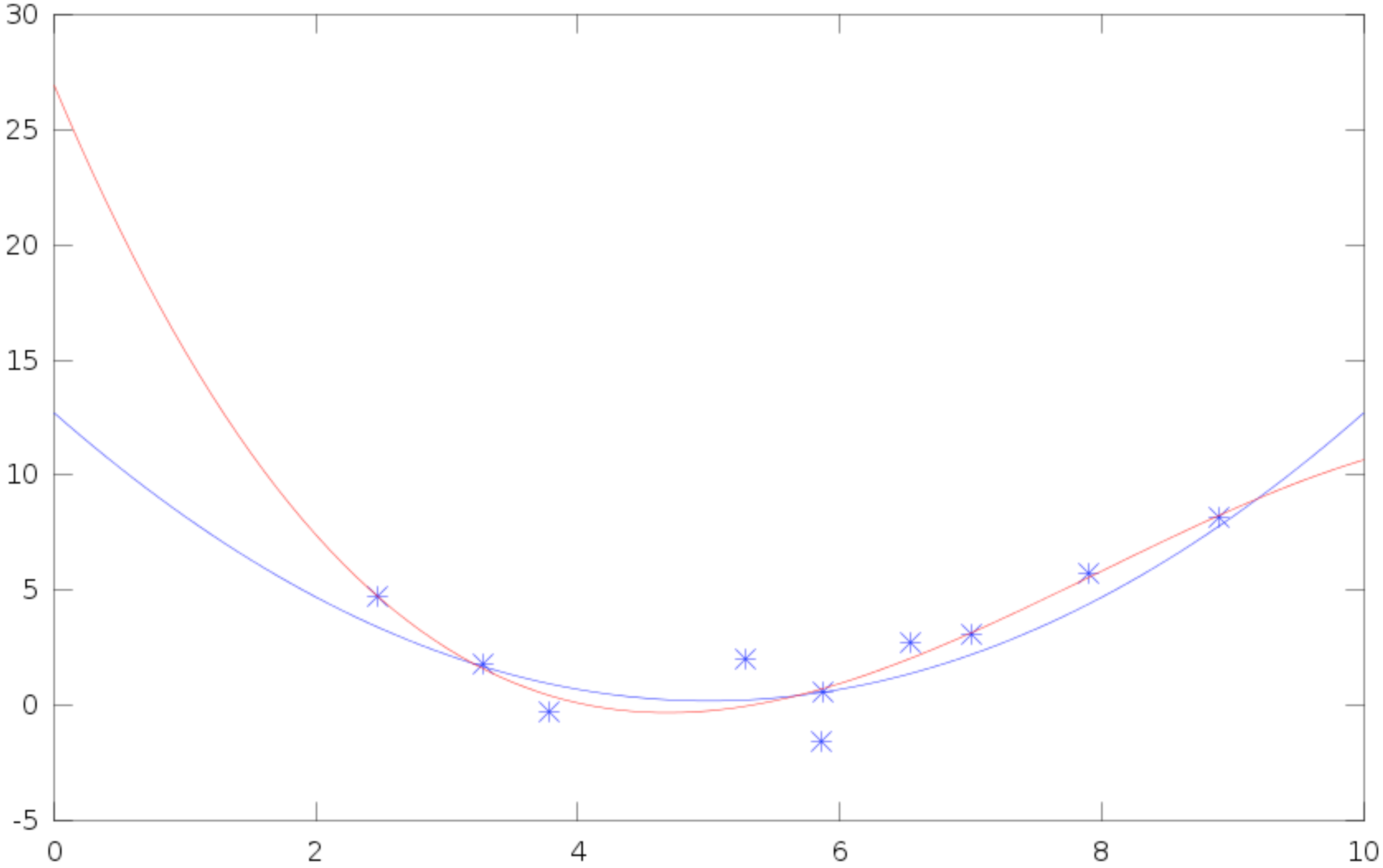
# Regression (d=1)



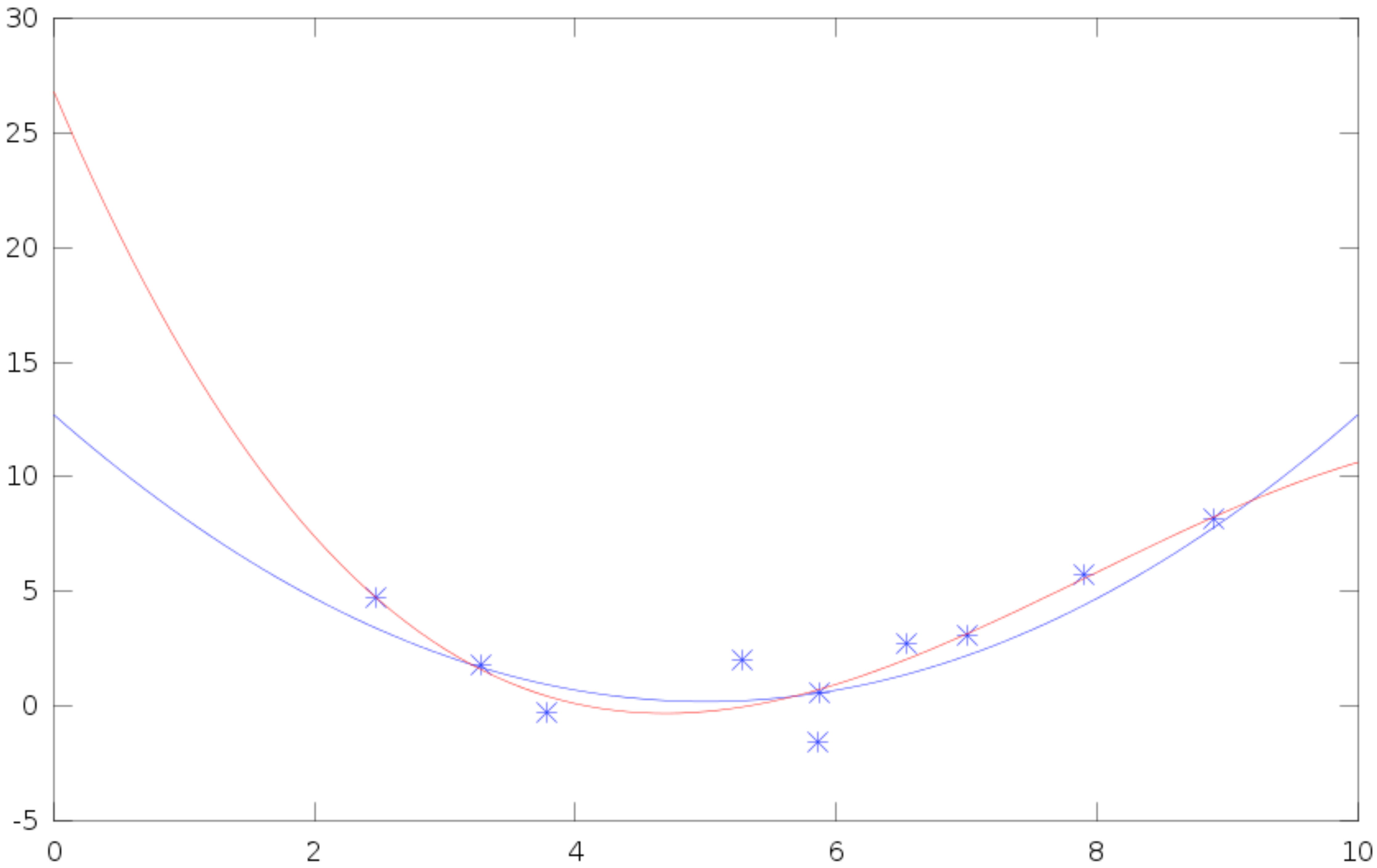
# Regression (d=2)



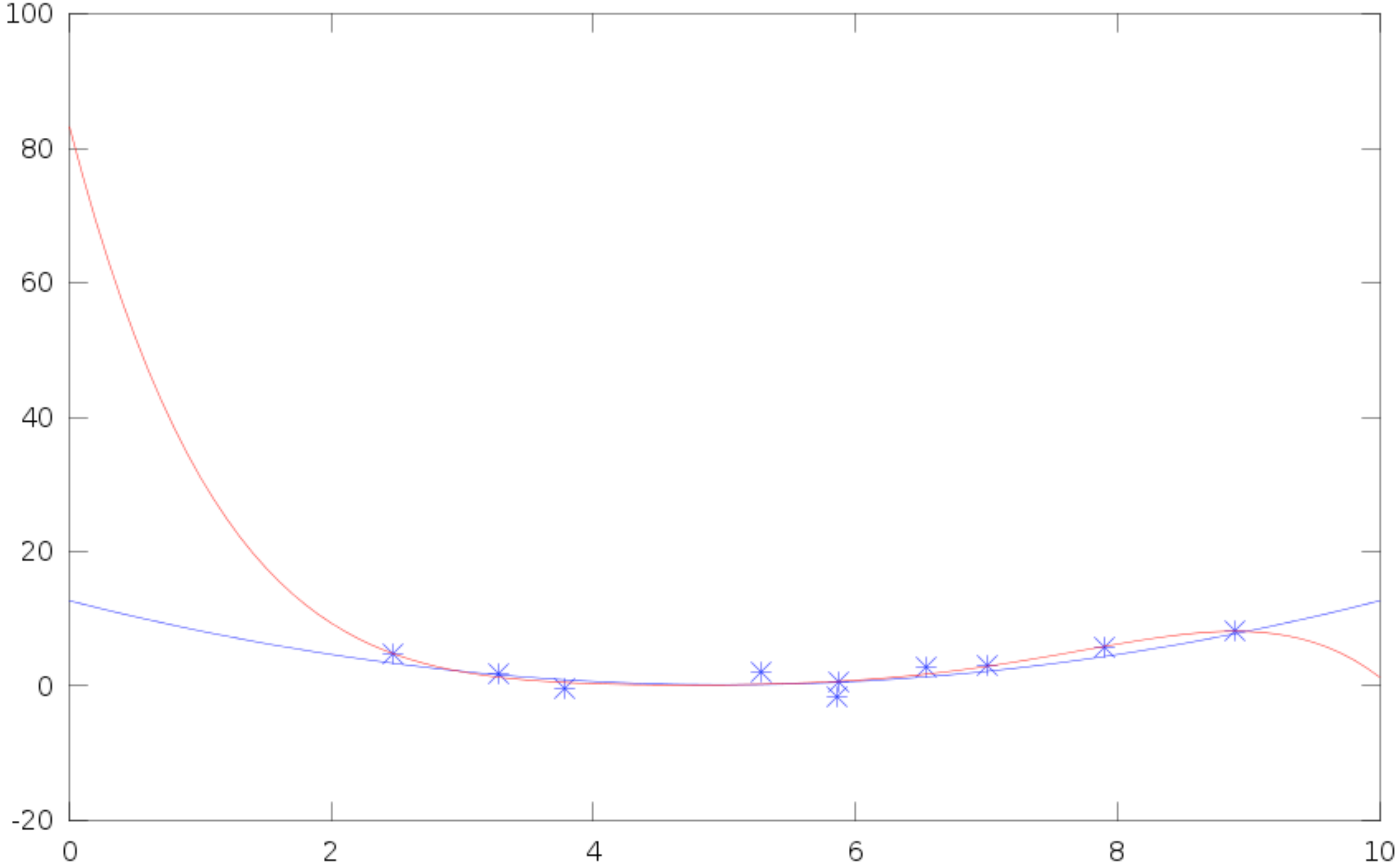
# Regression (d=3)



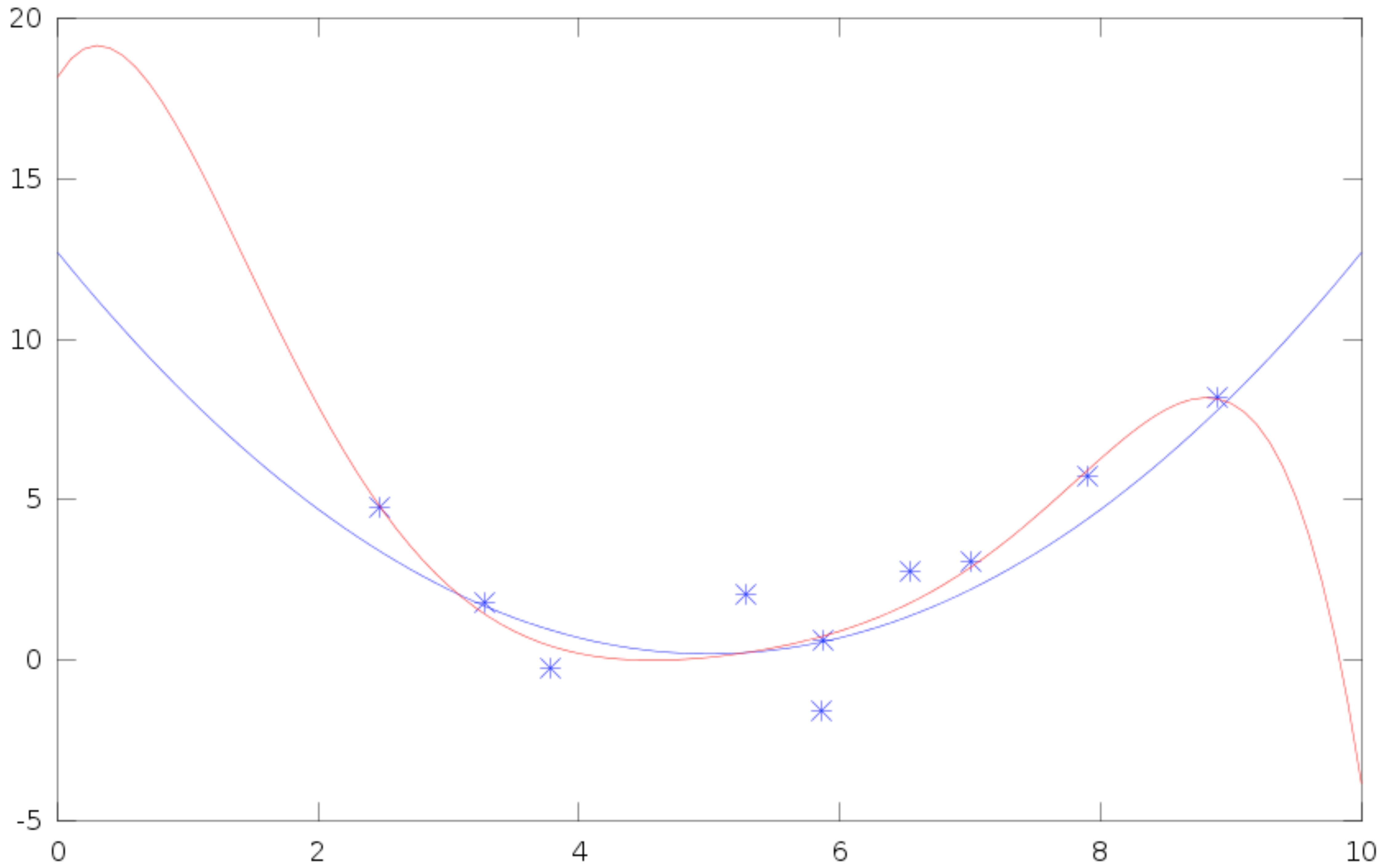
# Regression (d=4)



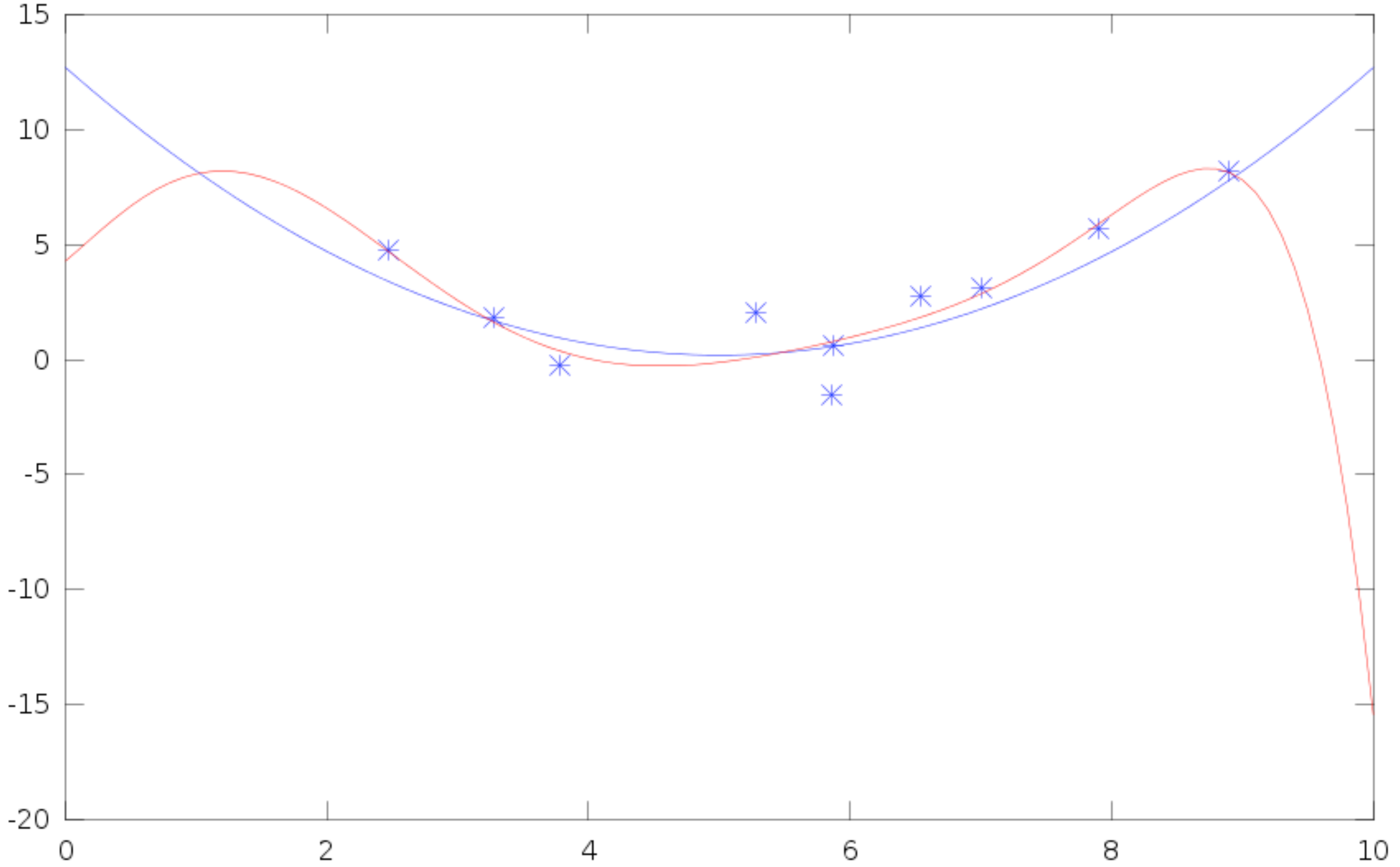
# Regression (d=5)



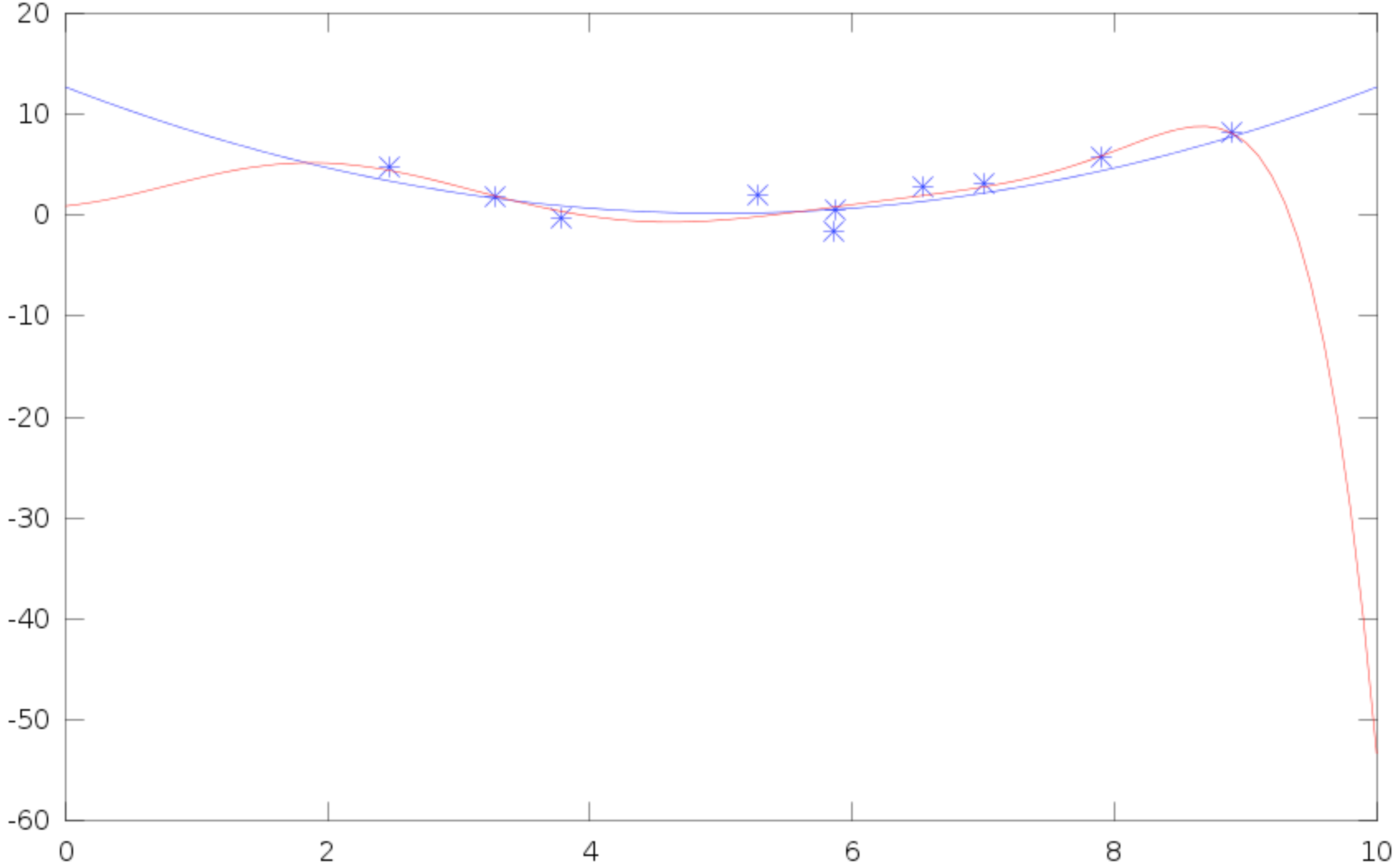
# Regression (d=6)



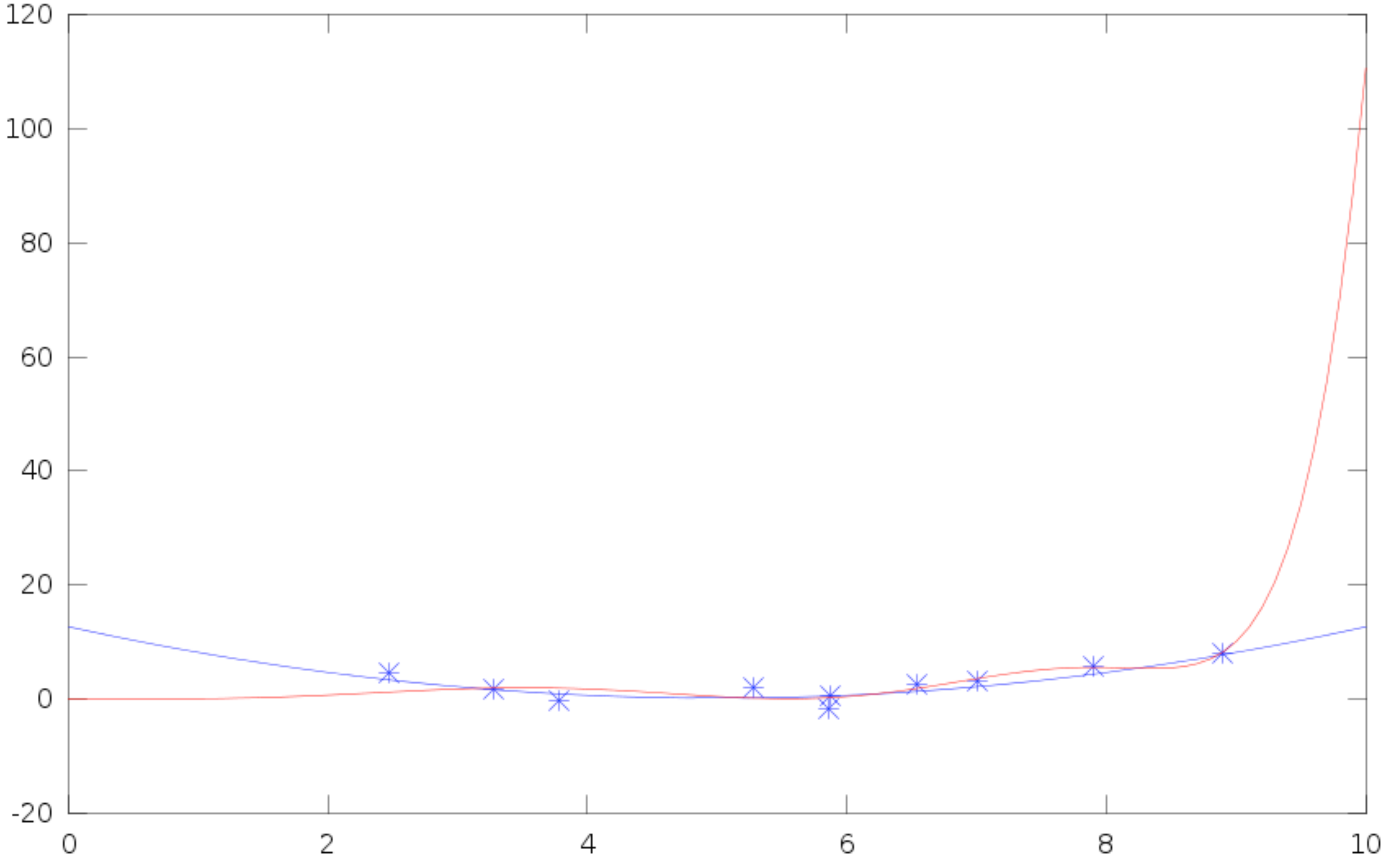
# Regression (d=7)



# Regression (d=8)



# Regression (d=9)



# Nonlinear Regression

```
warning: matrix singular to machine precision, rcond = 5.8676e-19
warning: attempting to find minimum norm solution
warning: matrix singular to machine precision, rcond = 5.86761e-19
warning: attempting to find minimum norm solution
warning: dgelsd: rank deficient 8x8 matrix, rank = 7
warning: matrix singular to machine precision, rcond = 1.10156e-21
warning: attempting to find minimum norm solution
warning: matrix singular to machine precision, rcond = 1.10145e-21
warning: attempting to find minimum norm solution
warning: dgelsd: rank deficient 9x9 matrix, rank = 6
warning: matrix singular to machine precision, rcond = 2.16217e-26
warning: attempting to find minimum norm solution
warning: matrix singular to machine precision, rcond = 1.66008e-26
warning: attempting to find minimum norm solution
warning: dgelsd: rank deficient 10x10 matrix, rank = 5
```

# Nonlinear Regression

```
warning: matrix singular to machine precision, rcond = 5.8676e-19
warning: attempting to find minimum norm solution
warning: matrix singular to machine precision, rcond = 5.86761e-19
warning: attempting to find minimum norm solution
warning: dgelsd: rank deficient 8x8 matrix, rank = 7
warning: matrix singular to machine precision, rcond = 1.10156e-21
warning: attempting to find minimum norm solution
warning: matrix singular to machine precision, rcond = [illegible]e-21
warning: attempting to find minimum norm solution
warning: dgelsd: rank deficient 9x9 matrix, rank = 6
warning: matrix singular to machine precision, rcond = 2.16217e-26
warning: attempting to find minimum norm solution
warning: matrix singular to machine precision, rcond = 1.66008e-26
warning: attempting to find minimum norm solution
warning: dgelsd: rank deficient 10x10 matrix, rank = 5
```

## Why does it fail?

# Model Selection

- Underfitting
  (model is too simple to explain data)
- Overfitting
  (model is too complicated to learn from data)
  - E.g. too many parameters
  - Insufficient confidence to estimate parameter
    (failed matrix inverse)
  - Often training error decreases nonetheless
- Model selection
  Need to quantify model complexity vs. data
- This course - algorithms, model selection, questions

# Parzen Windows

Parzen

# 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

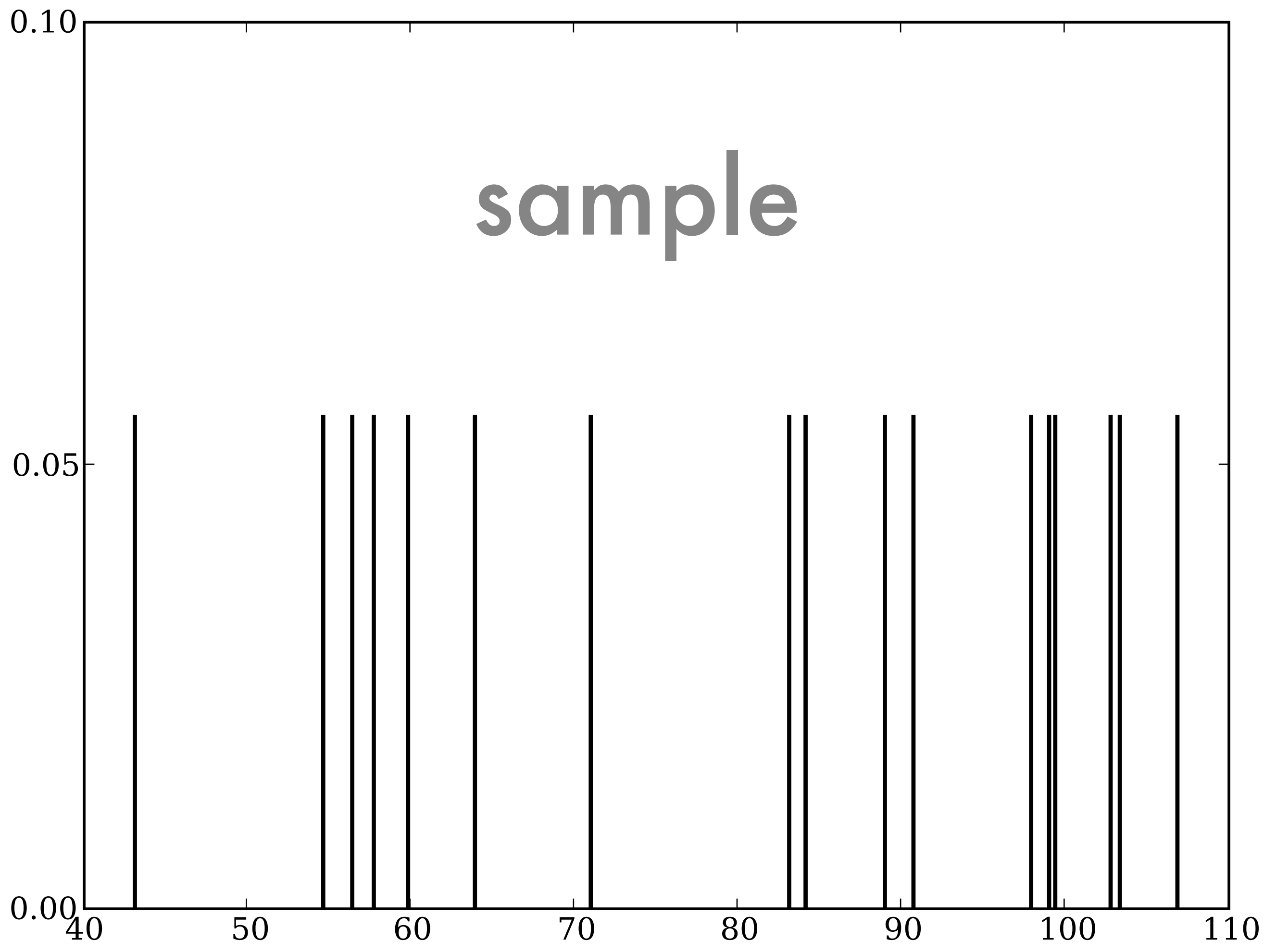
# Curse of dimensionality (lite)

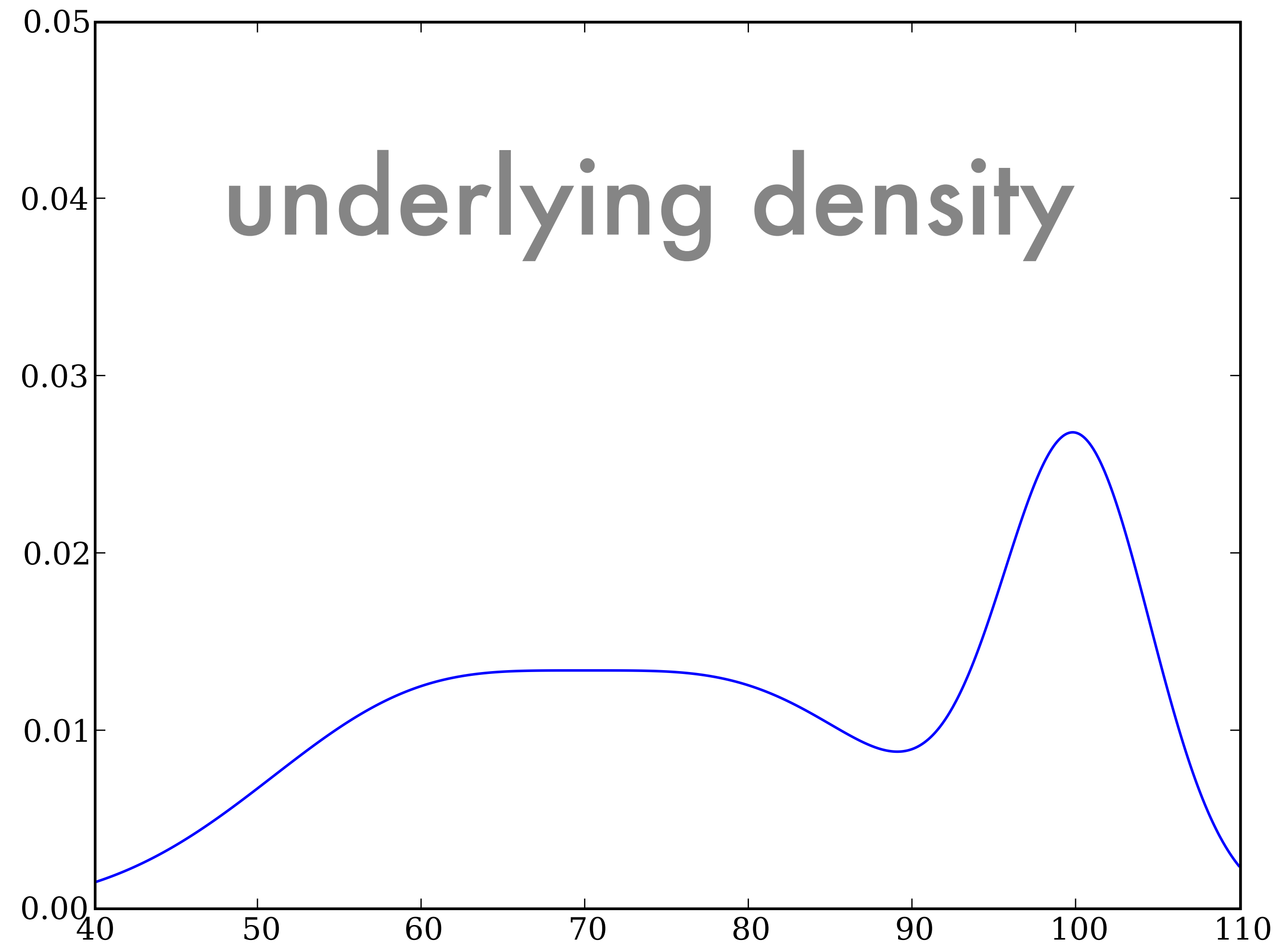
- Discrete random variables, e.g.
  - English, Chinese, German, French, ...
  - Male, Female
  - ZIP code
  - Day of the week
  - Operating system
  - ...
- Continuous random variables
  - Income
  - Bandwidth
  - Time

#bins grows exponentially

need many bins per dimension

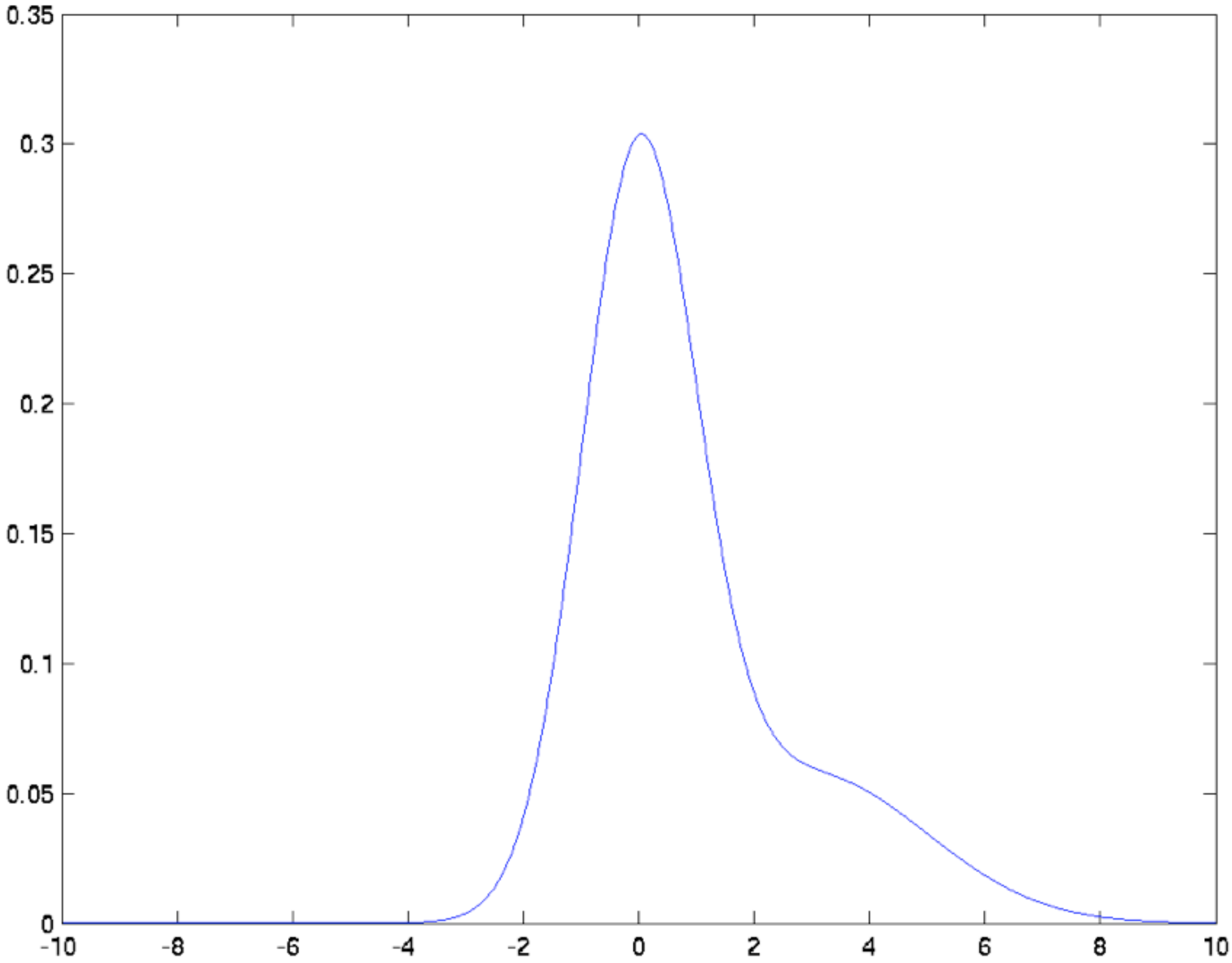
# Density Estimation



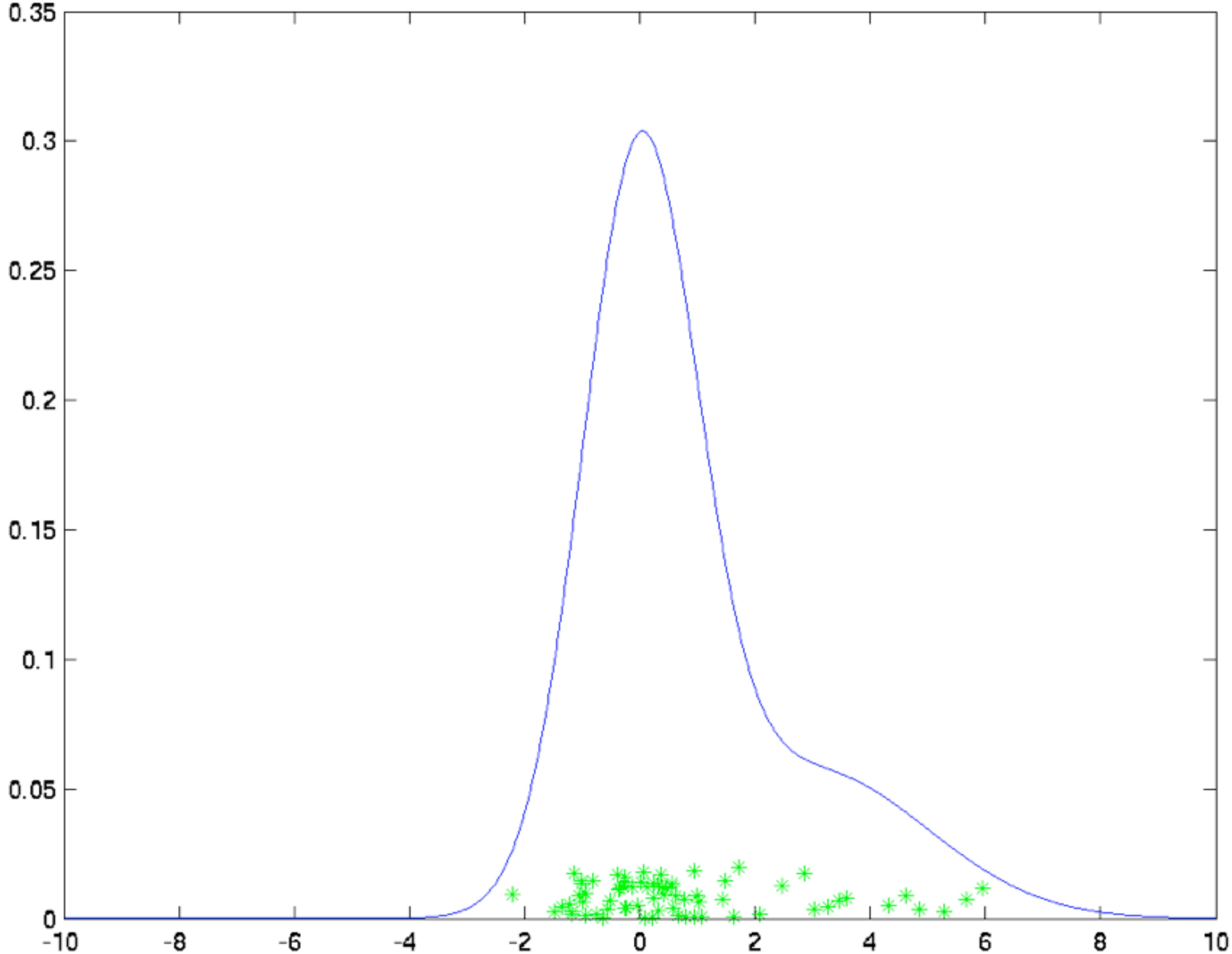

- Continuous domain = infinite number of bins
- Curse of dimensionality
  - 10 bins on [0, 1] is probably good
  - $10^{10}$ bins on $[0, 1]^{10}$ requires high accuracy in estimate: probability mass per cell also decreases by $10^{10}$

# Bin Counting

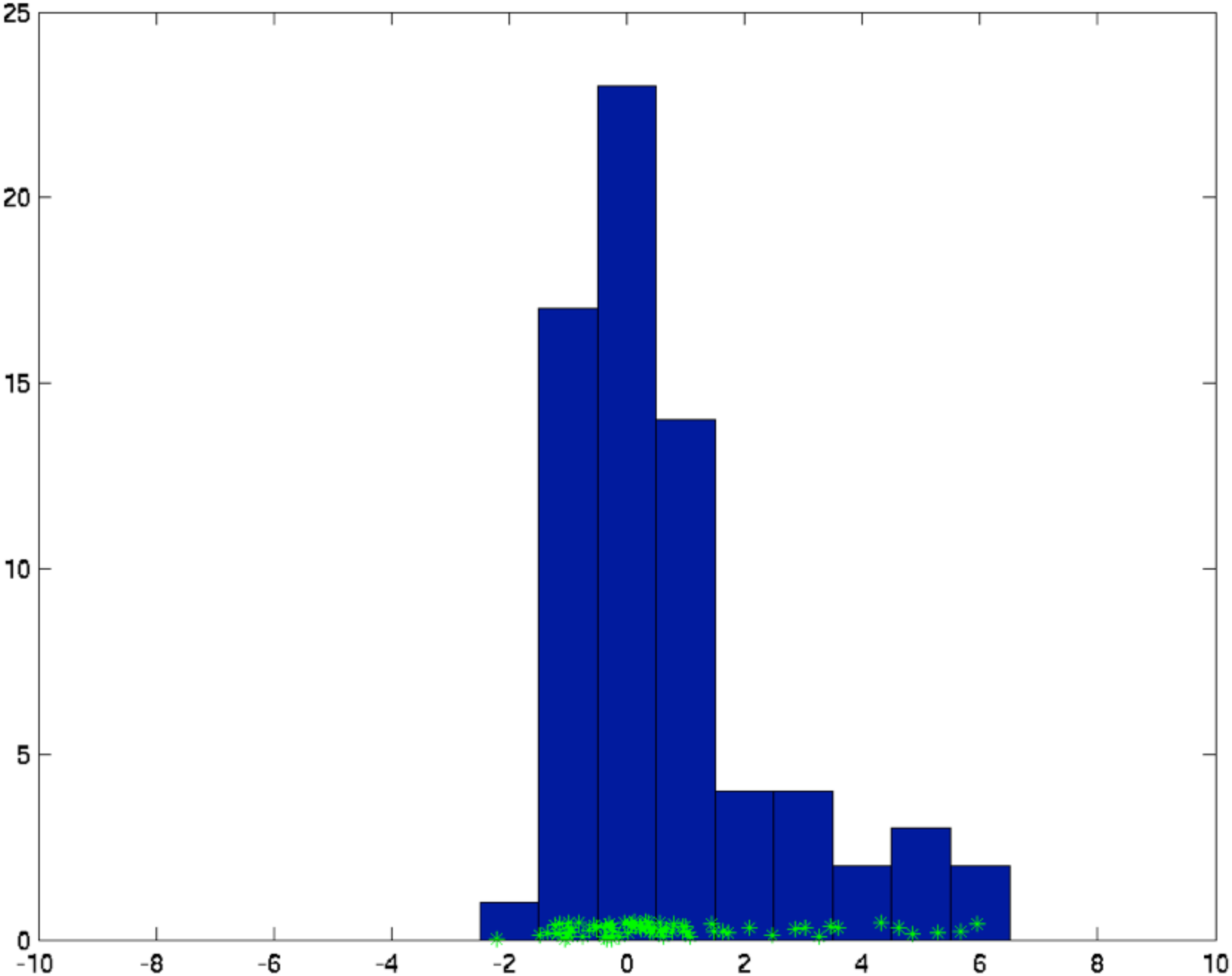



iversity

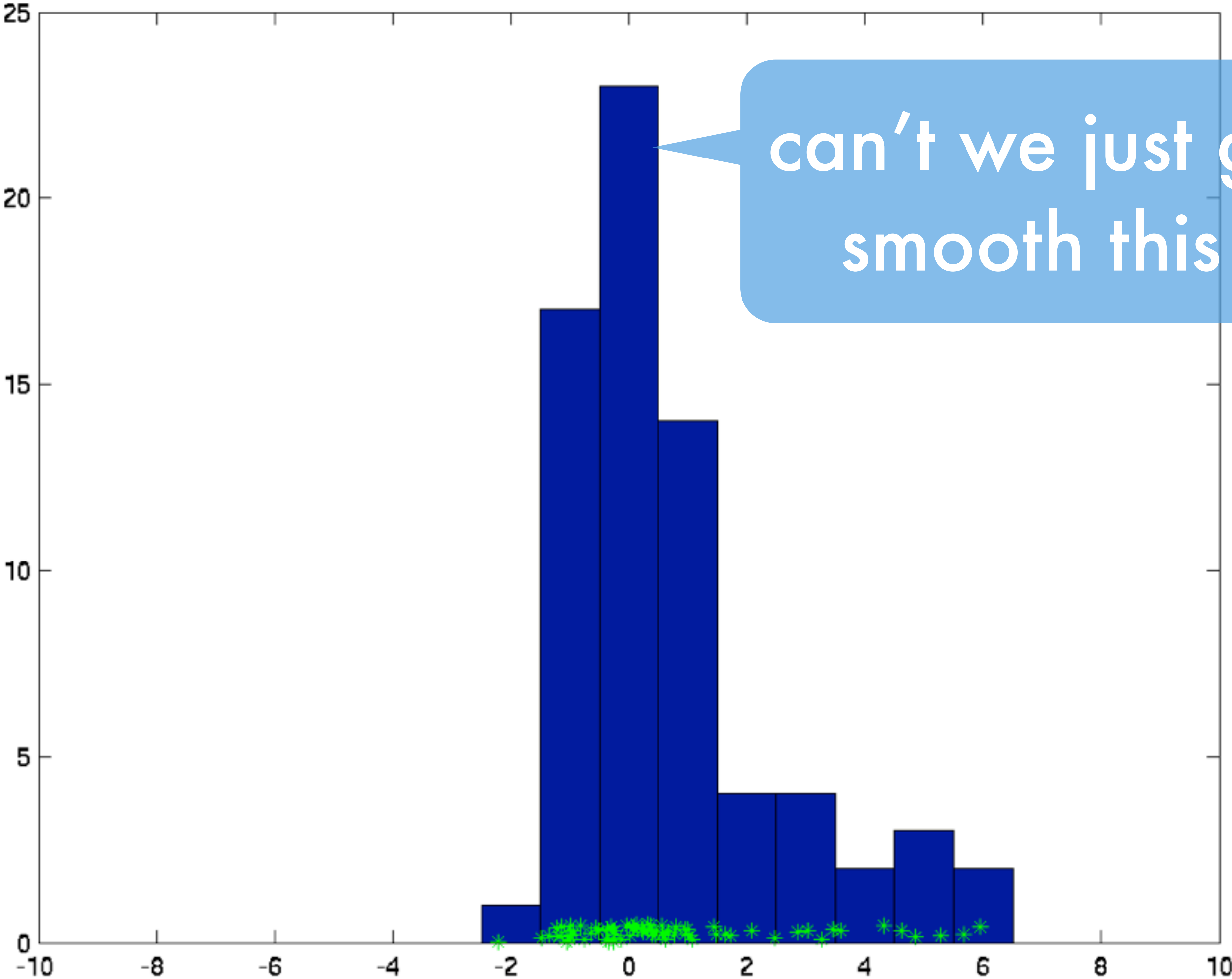
# Bin Counting



# Bin Counting



# Bin Counting




can't we just go and smooth this out?

iversity

# Parzen Windows

- Naive approach
  Use empirical density (delta distributions)

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

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

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

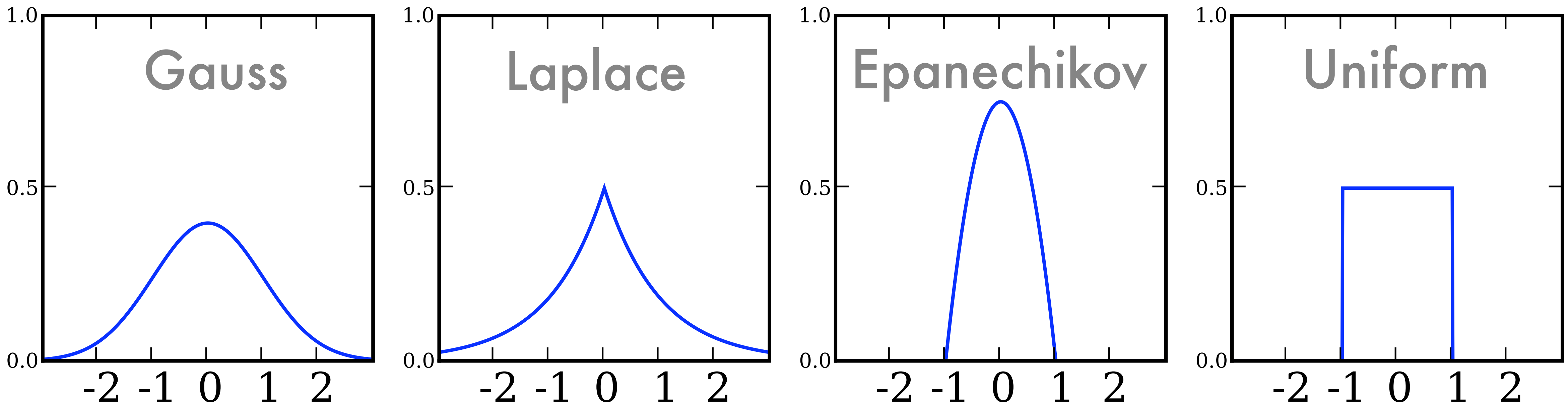
# Parzen Windows

- **Density estimate**

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

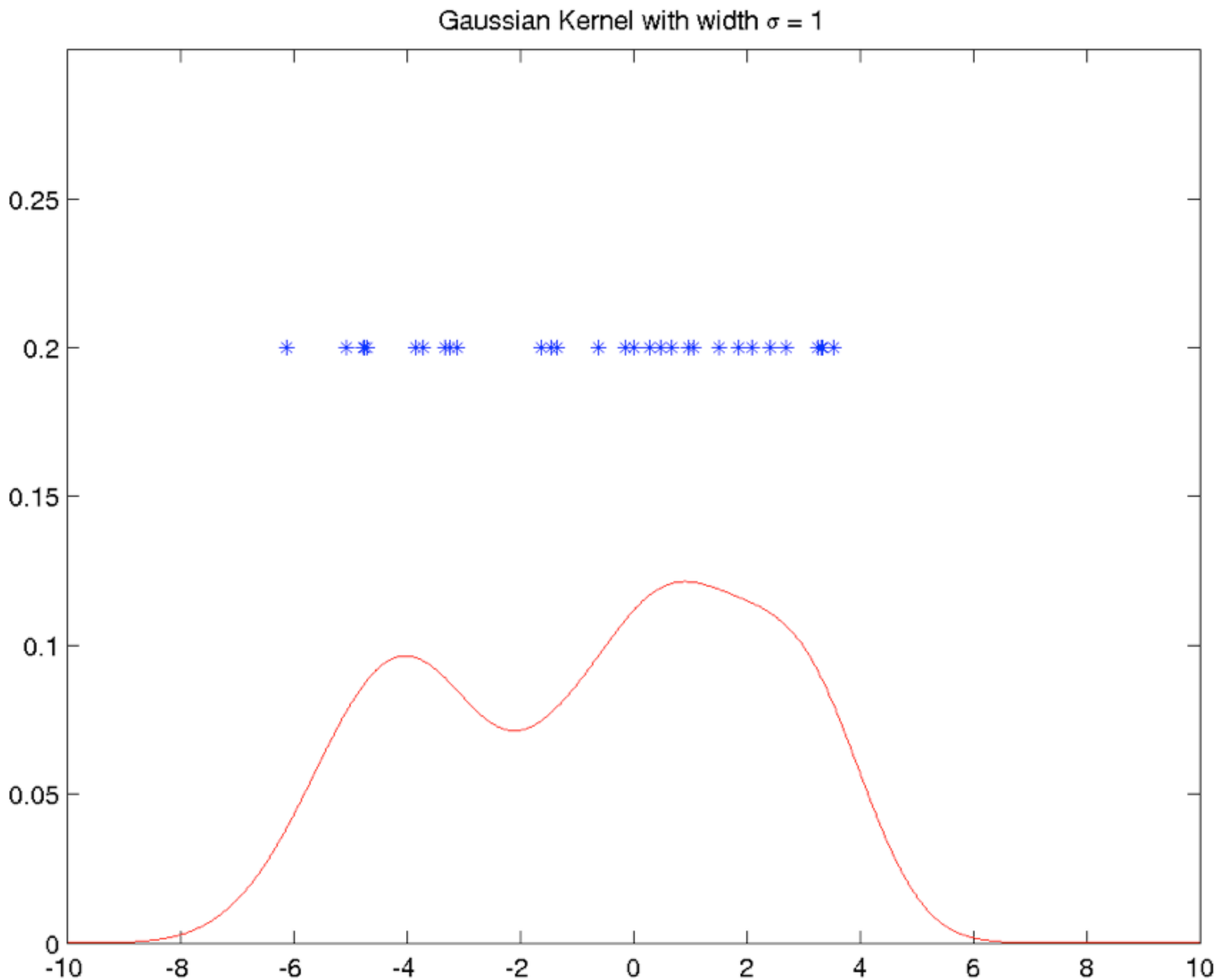
$$\hat{p}(x) = \frac{1}{m}\sum_{i=1}^{m} k_{x_i}(x)$$

- **Smoothing kernels**




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

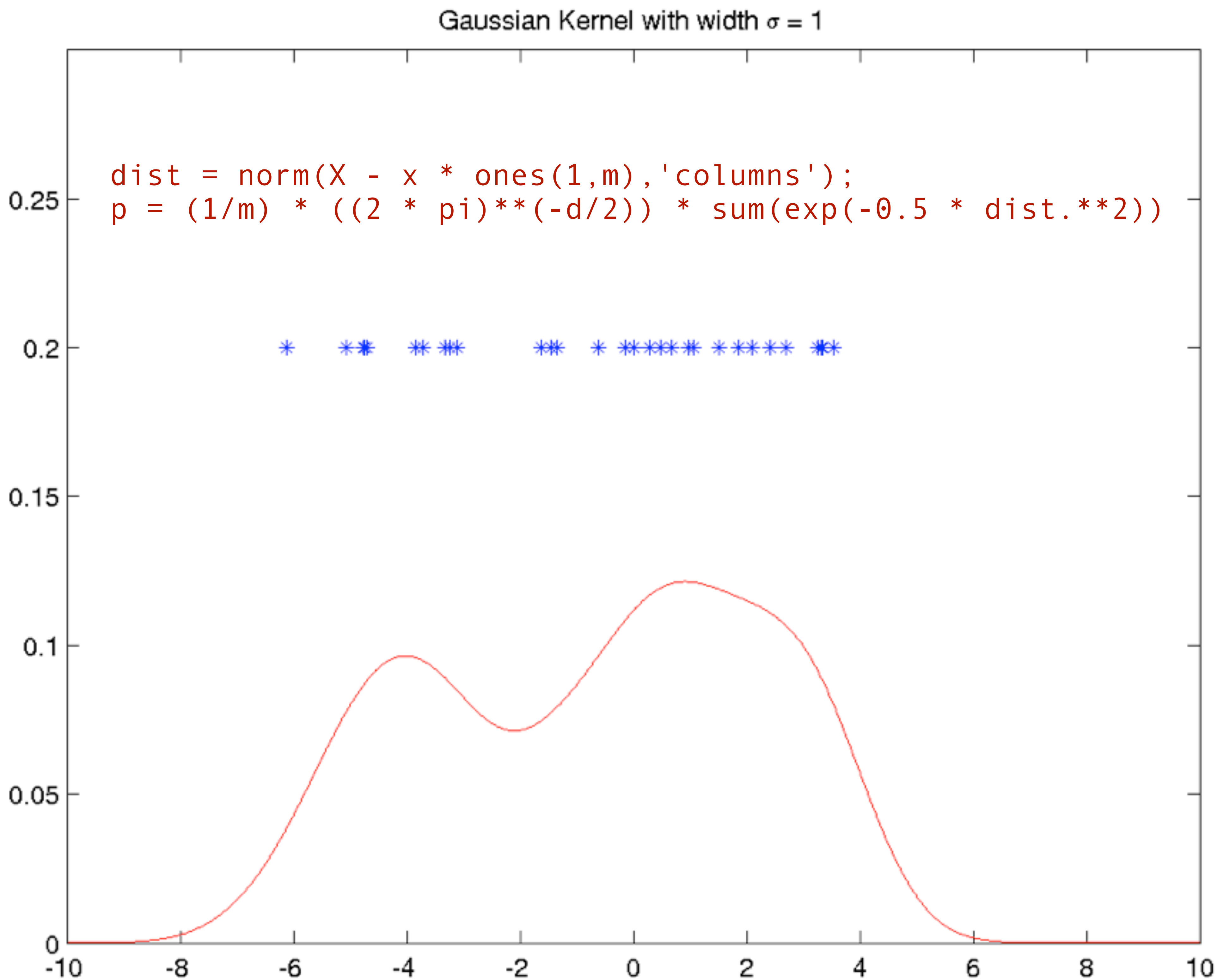
# Smoothing



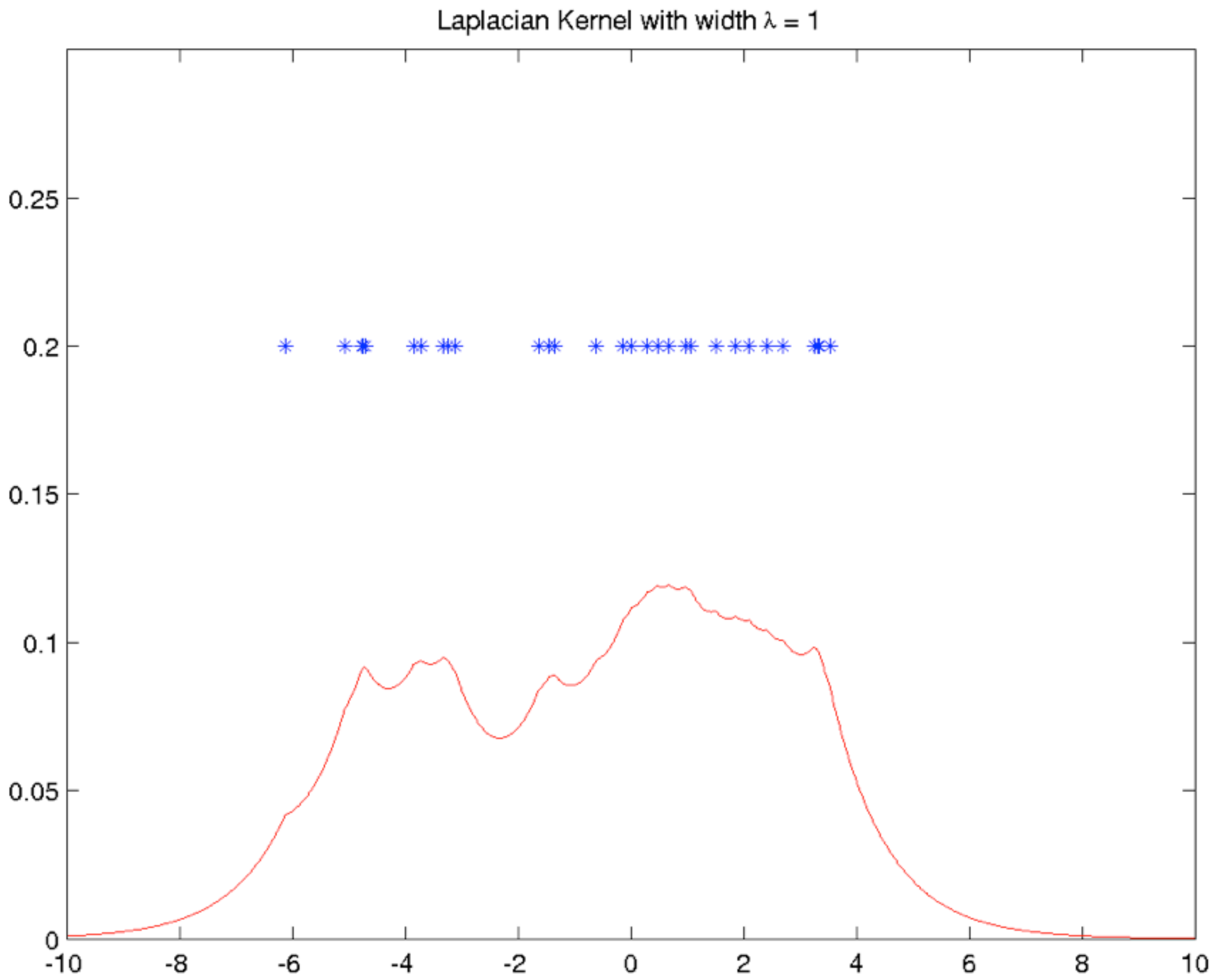

iversity

# Smoothing

```
dist = norm(X - x * ones(1,m),'columns');
p = (1/m) * ((2 * pi)**(-d/2)) * sum(exp(-0.5 * dist.**2))
```

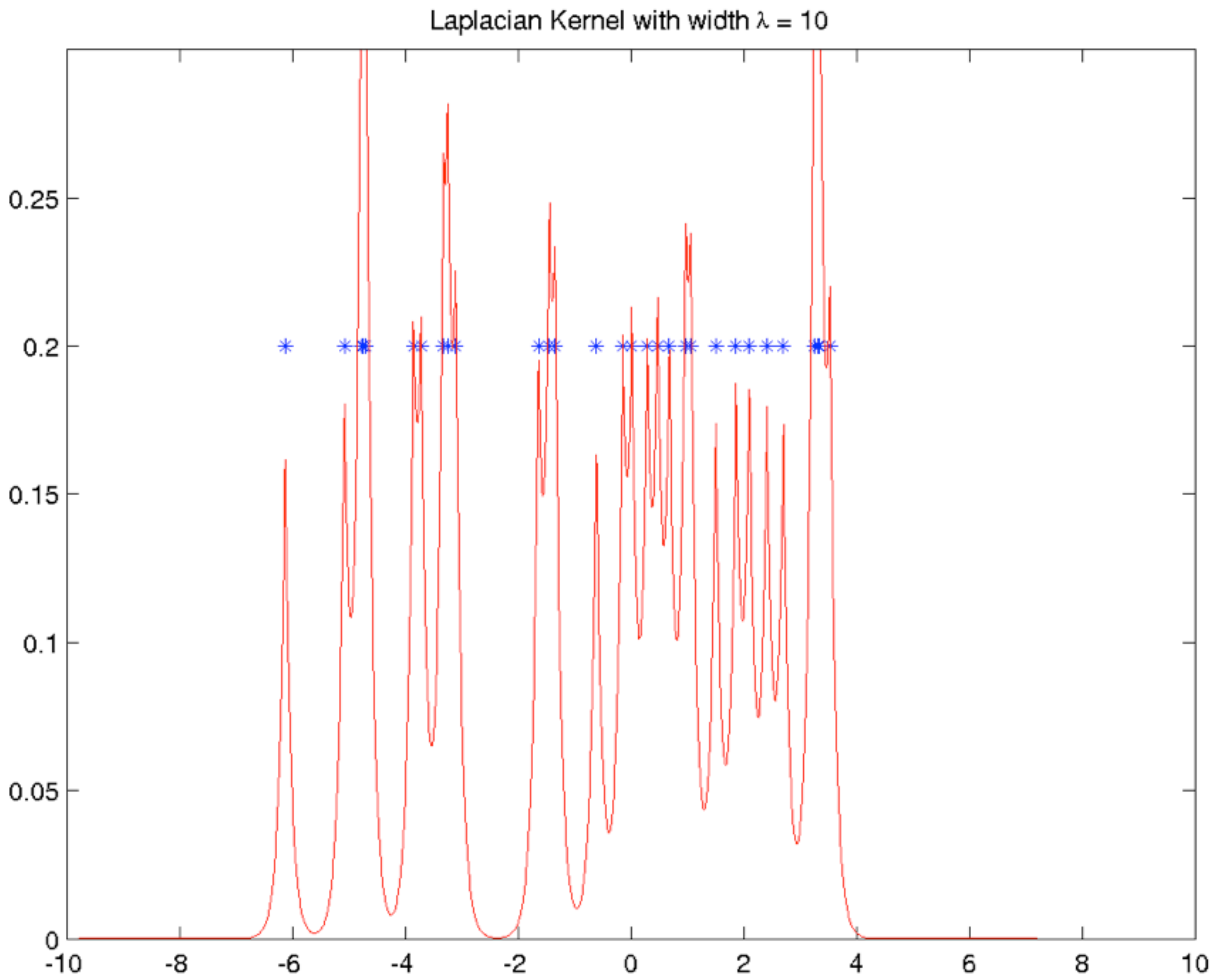


# Smoothing



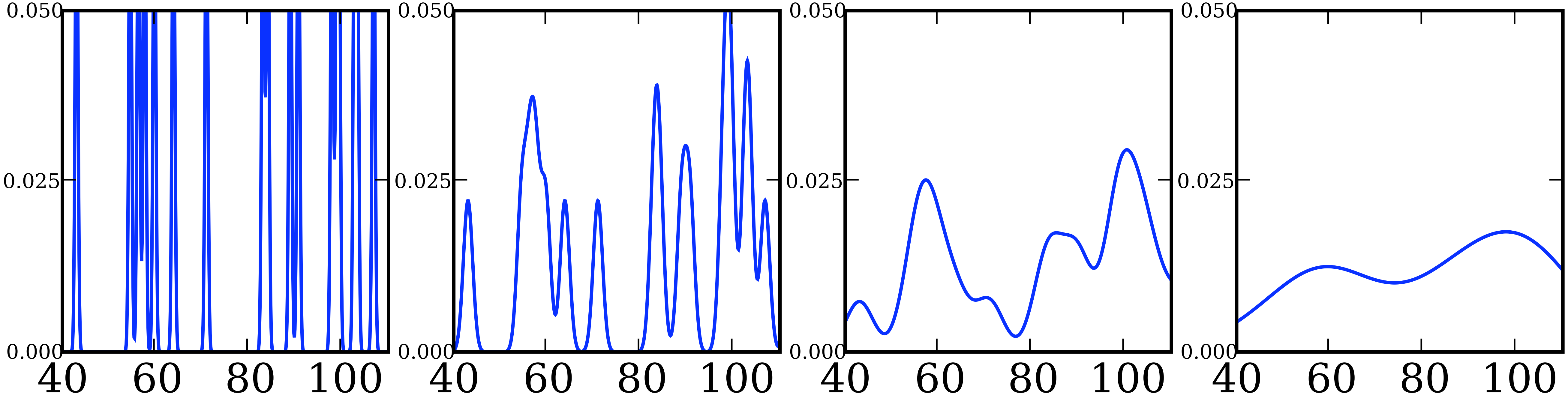

iversity

# Smoothing

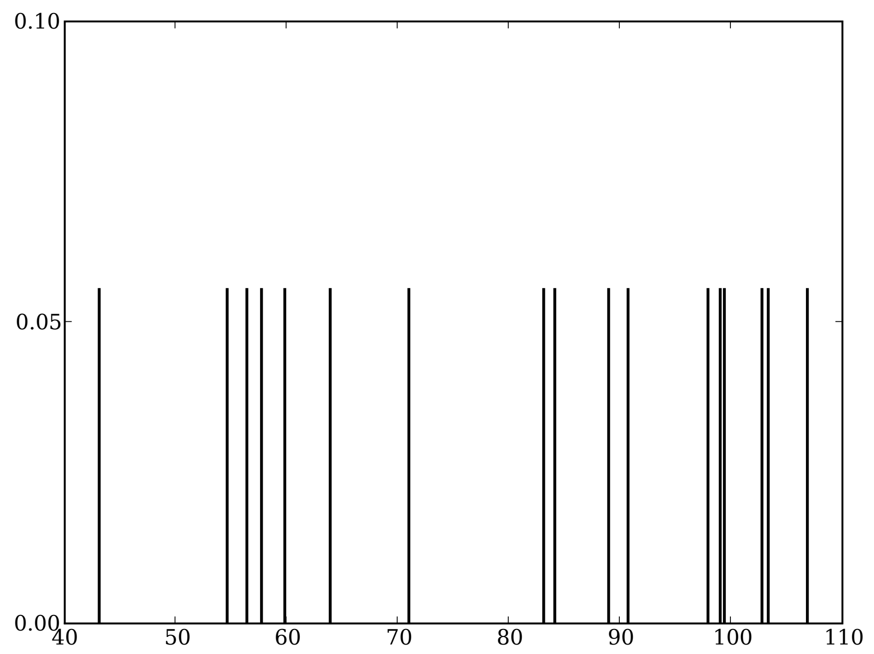


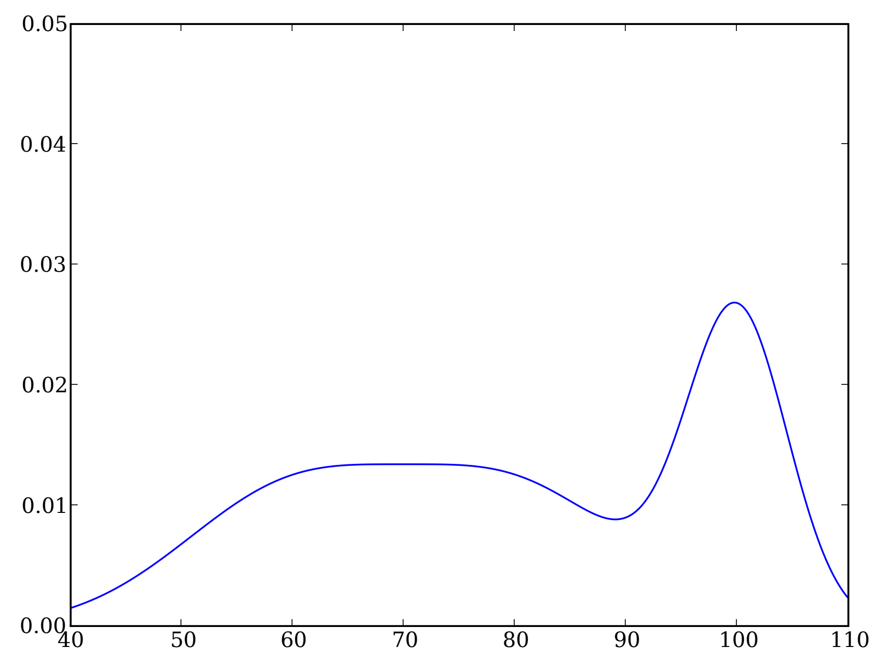
Laplacian Kernel with width λ = 10
0.25
0.2
0.15
0.1
0.05
0
-10
-8
-6
-4
-2
0
2
4
6
8
10

# Size matters



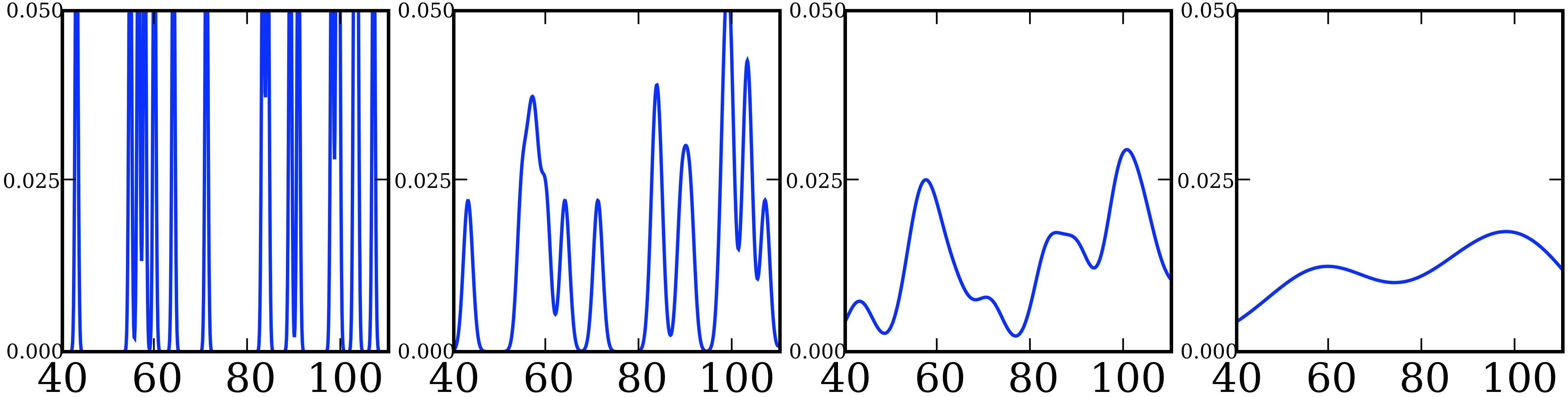

0.3 1 3 10




Carnegie Mellon University

# Size matters
# Shape matters mostly in theory




- Kernel width $$k_{x_i}(x) = r^{-d} h\left(\frac{x - x_i}{r}\right)$$
    - Too narrow overfits
    - Too wide smoothes with constant distribution
- How to choose?

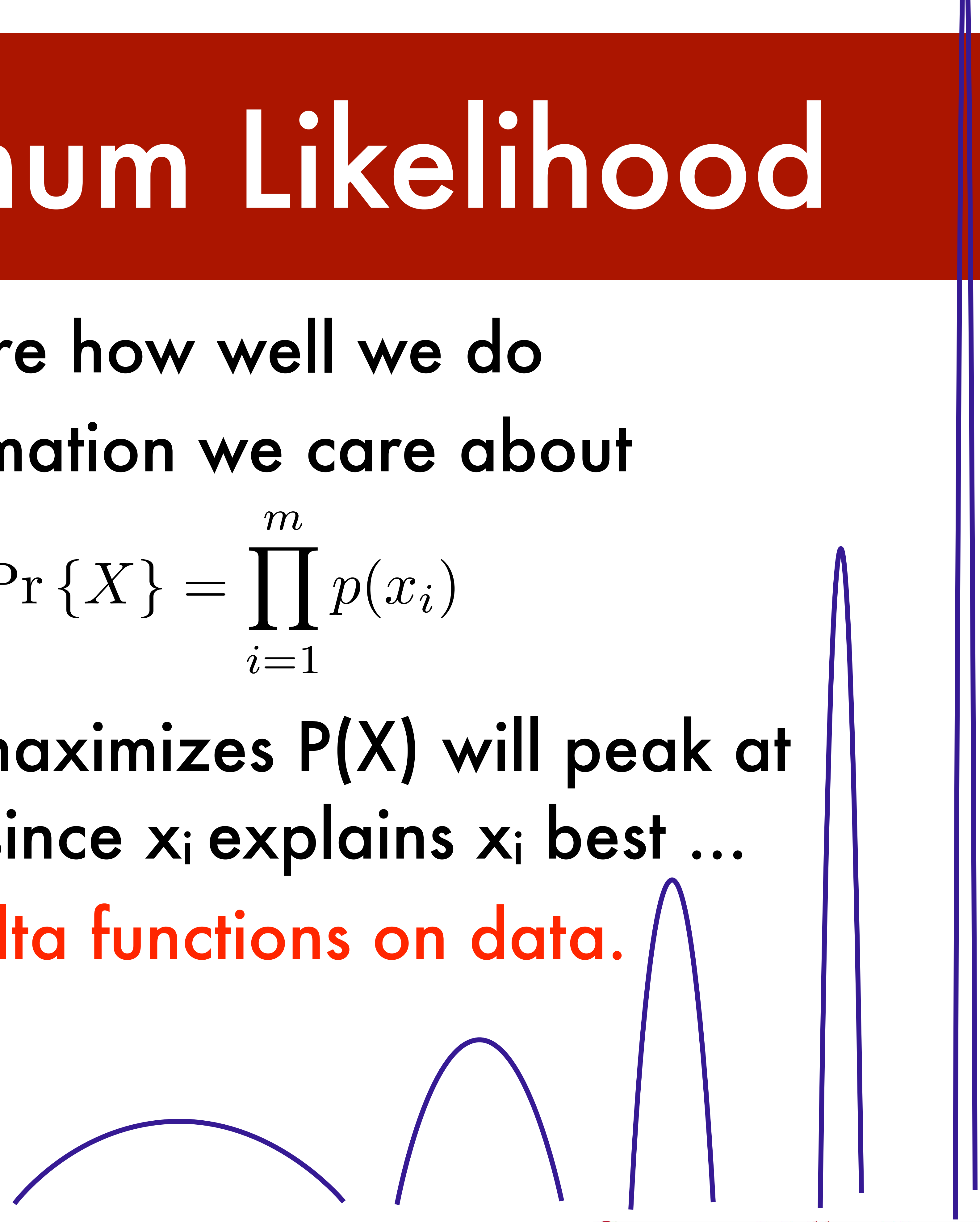
# Model Selection

# Maximum Likelihood

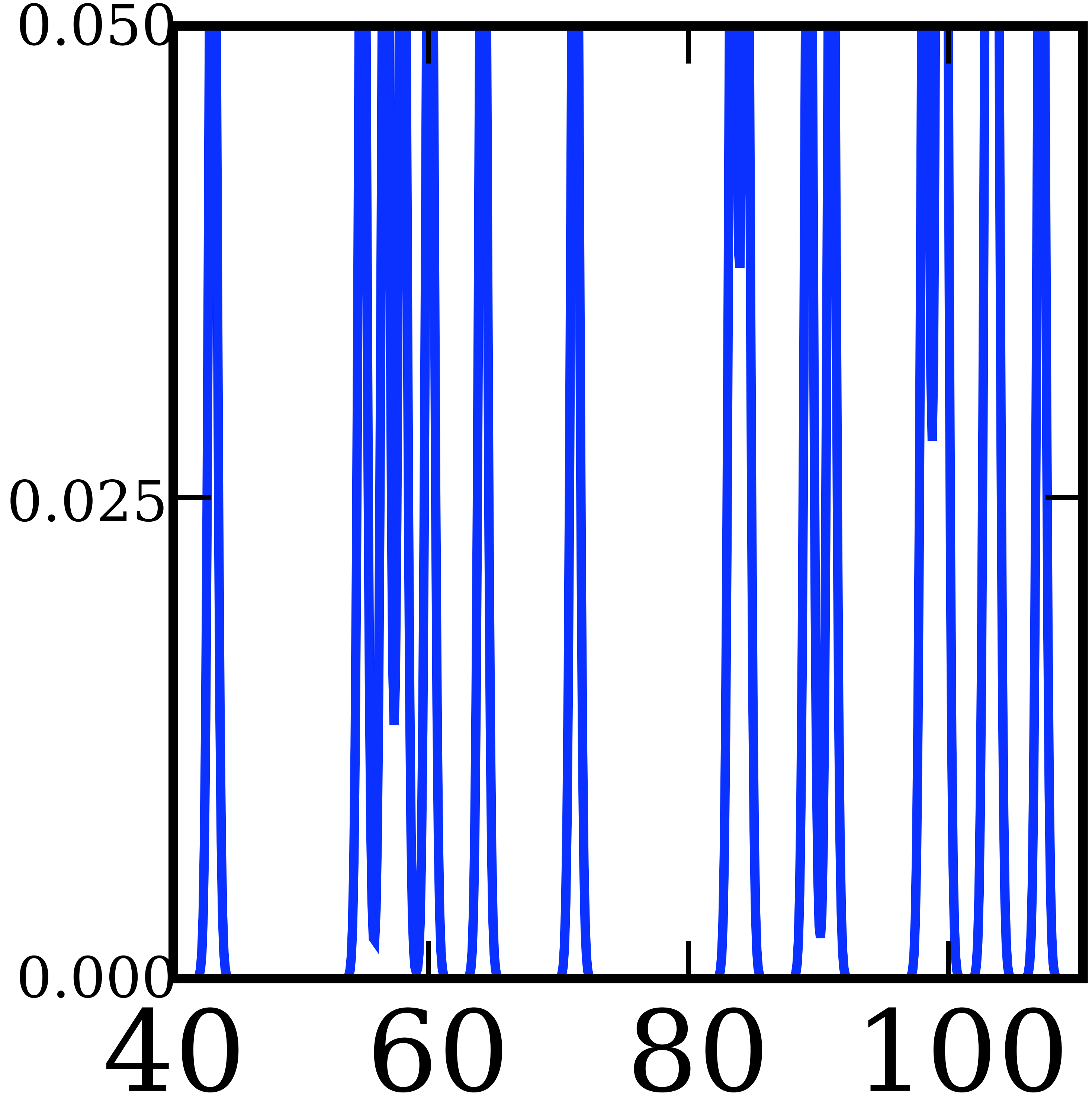
- Need to measure how well we do
- For density estimation we care about

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

- Finding a that maximizes P(X) will peak at all data points since $x_i$ explains $x_i$ best ...
- Maxima are delta functions on data.
- Overfitting!

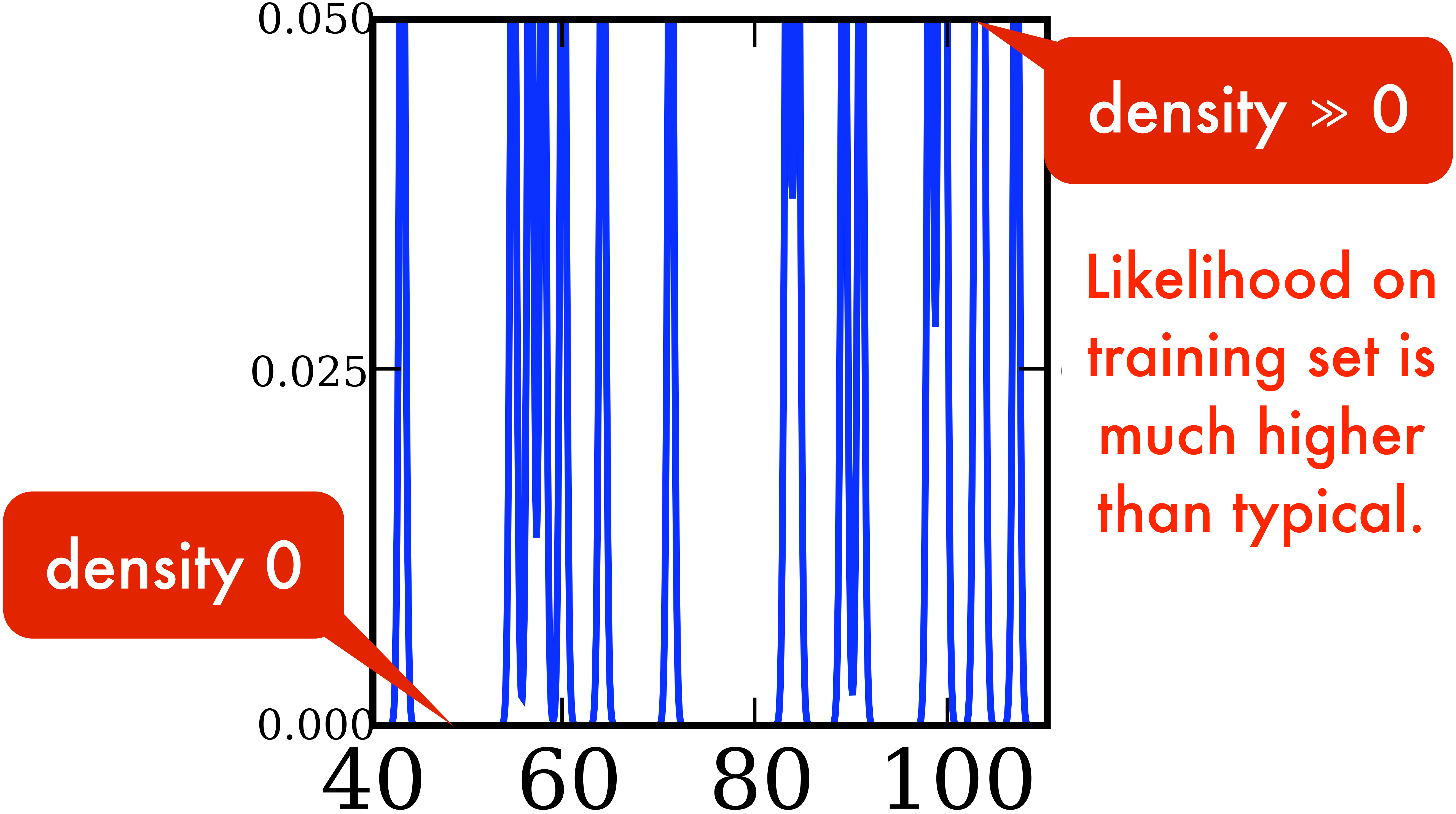


# Overfitting



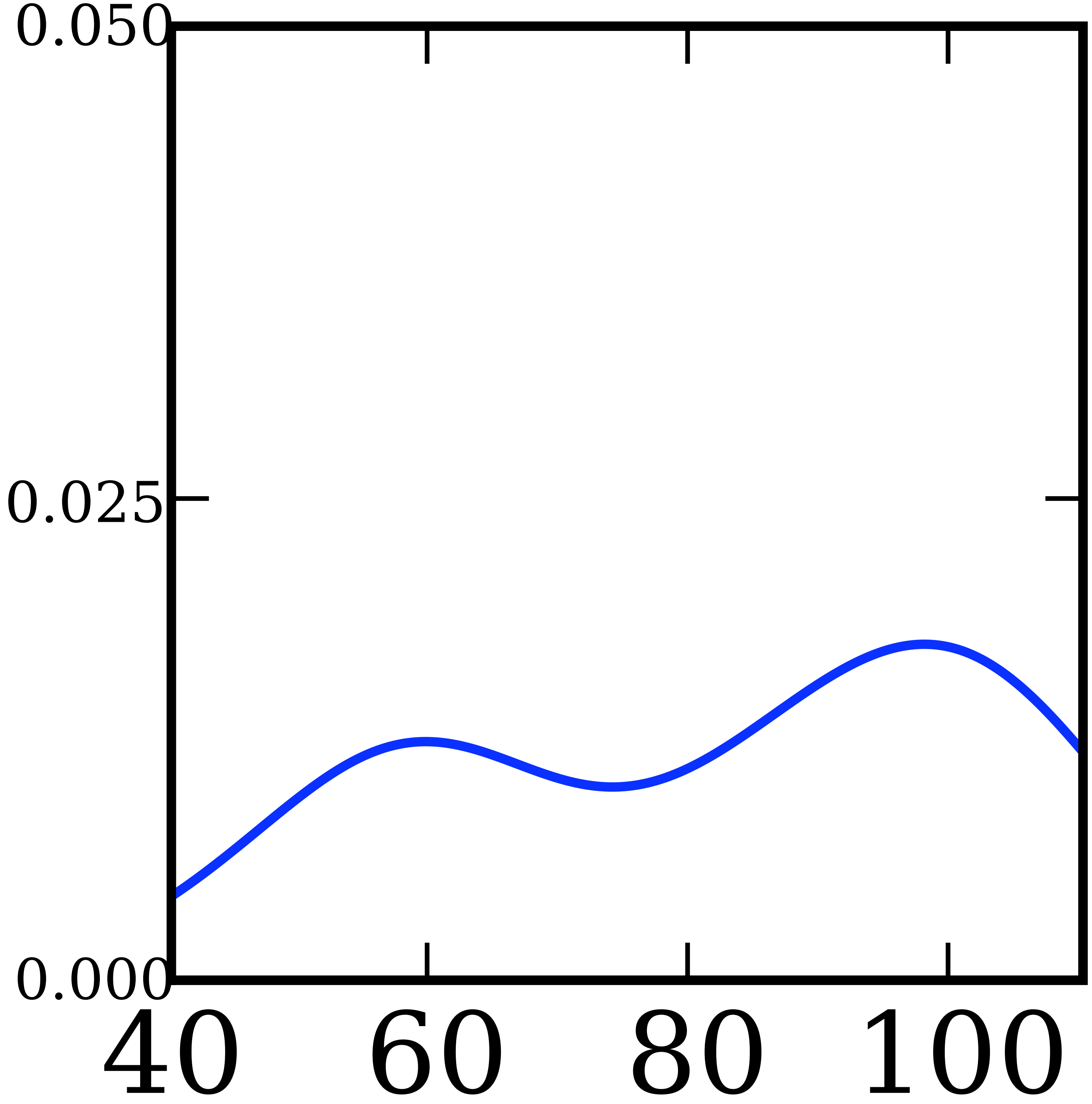

Likelihood on training set is much higher than typical.

# Overfitting




Likelihood on training set is much higher than typical.

# Underfitting




Likelihood on training set is very similar to typical one.

Too simple.

# Model Selection

- Validation
  - Use some of the data to estimate density.
  - Use other part to evaluate how well it works
  - Pick the parameter that works best

$$\mathcal{L}(X'|X) := \frac{1}{n'} \sum_{i=1}^{n'} \log \hat{p}(x_i')$$

- Learning Theory
  - Use data to build model
  - Measure complexity and use this to bound

$$\frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(x_i) - \mathbf{E}_x \left[\log \hat{p}(x)\right]$$

# Model Selection

- Validation
  - Use some of the data to estimate density.
  - Use other part to evaluate how well it works
  - Pick the parameter that works best

easy

$$\mathcal{L}(X'|X) := \frac{1}{n'} \sum_{i=1}^{n'} \log \hat{p}(x_i')$$

- Learning Theory
  - Use data to build model
  - Measure complexity and use this to bound

$$\frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(x_i) - \mathbf{E}_x \left[\log \hat{p}(x)\right]$$

# Model Selection

- Validation
  - Use some of the data to estimate density.
  - Use other part to evaluate how well it works
  - Pick the parameter that works best

easy

$$\mathcal{L}(X'|X) := \frac{1}{n'} \sum_{i=1}^{n'} \log \hat{p}(x_i')$$

wasteful

- Learning Theory
  - Use data to build model
  - Measure complexity and use this to bound

$$\frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(x_i) - \mathbf{E}_x \left[\log \hat{p}(x)\right]$$

# Model Selection

- Validation
  - Use some of the data to estimate density.
  - Use other part to evaluate how well it works
  - Pick the parameter that works best

easy

$$\mathcal{L}(X'|X) := \frac{1}{n'} \sum_{i=1}^{n'} \log \hat{p}(x_i')$$

wasteful

- Learning Theory
  - Use data to build model
  - Measure complexity and use this to bound

difficult

$$\frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(x_i) - \mathbf{E}_x \left[\log \hat{p}(x)\right]$$

# Model Selection

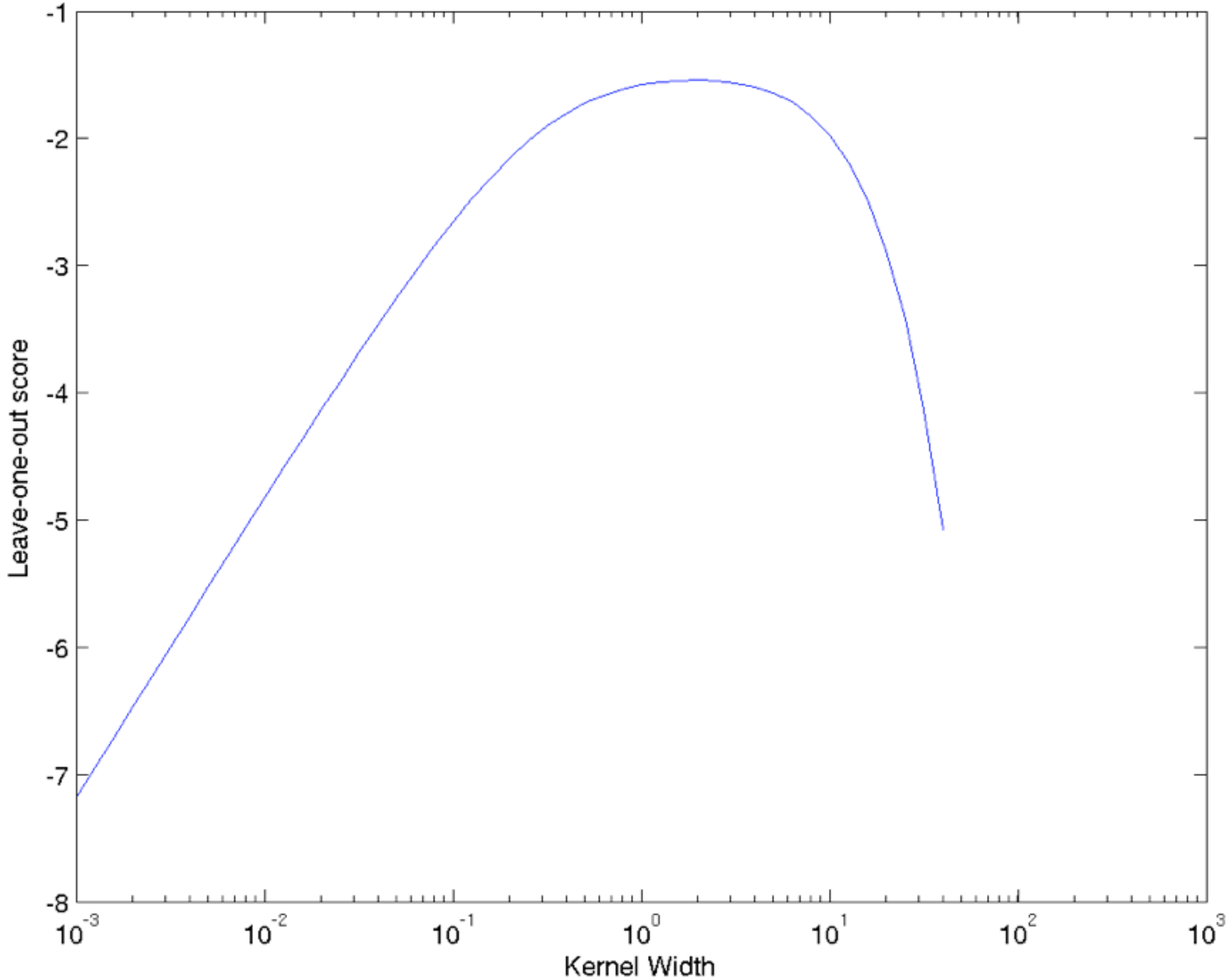
- Leave-one-out Crossvalidation
  - Use almost all data to estimate density.
  - Use single instance to estimate how well it works

$$\log p(x_i | X \setminus \{x_i\}) = \log \frac{1}{n-1} \sum_{j \neq i} k(x_i, x_j)$$

  - This has huge variance
  - Average over estimates for all training data
  - Pick the parameter that works best
- Simple implementation

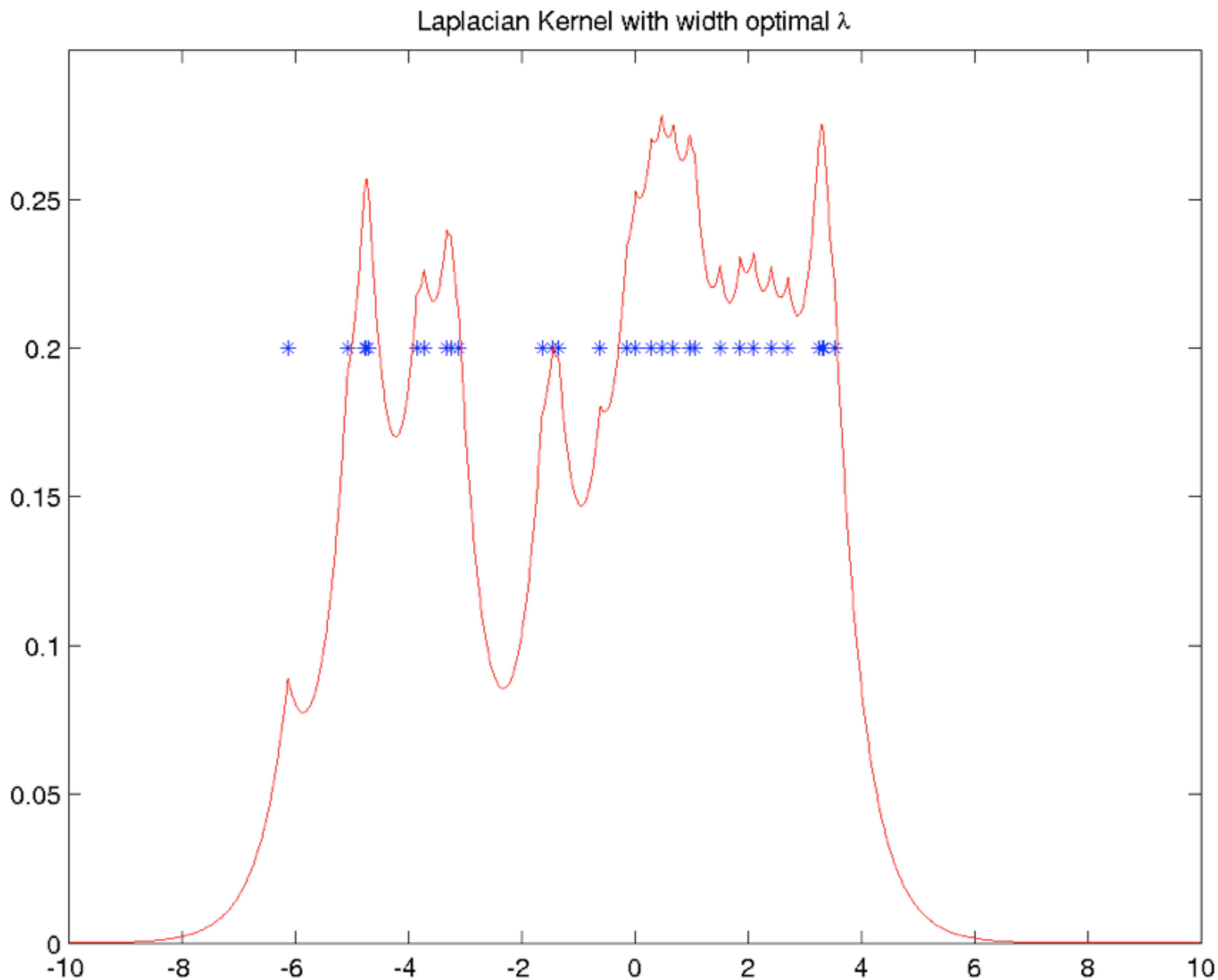
$$\frac{1}{n} \sum_{i=1}^{n} \log \left[ \frac{n}{n-1} p(x_i) - \frac{1}{n-1} k(x_i, x_i) \right] \text{ where } p(x) = \frac{1}{n} \sum_{i=1}^{n} k(x_i, x)$$

# Leave-one out estimate




niversity

# Optimal estimate




iversity

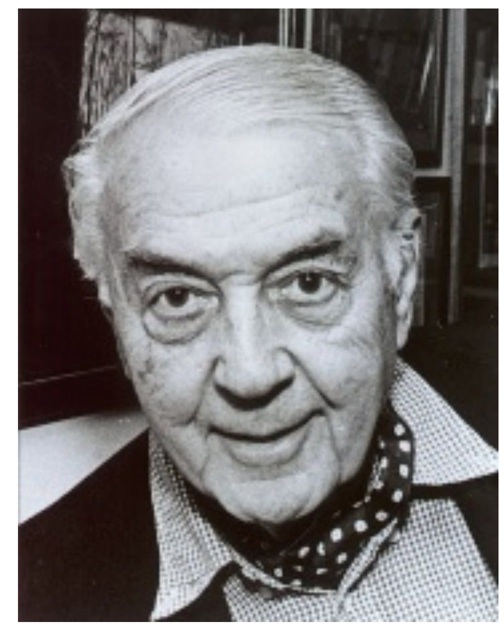
# Model Selection

- k-fold Crossvalidation
  - Partition data into k blocks (typically 10)
  - Use all but one block to compute estimate
  - Use remaining block as validation set
  - Average over all validation estimates

$$\frac{1}{k}\sum_{i=1}^{k} l(p(X_i|X\backslash X_i))$$

- Almost unbiased (e.g. via Luntz and Brailovski, 1969) (error is for (k-1)/k sized set)
- Pick best parameter (why must we not check too many?)

# Watson Nadaraya Estimator



Geoff Watson

# From density estimation to classification

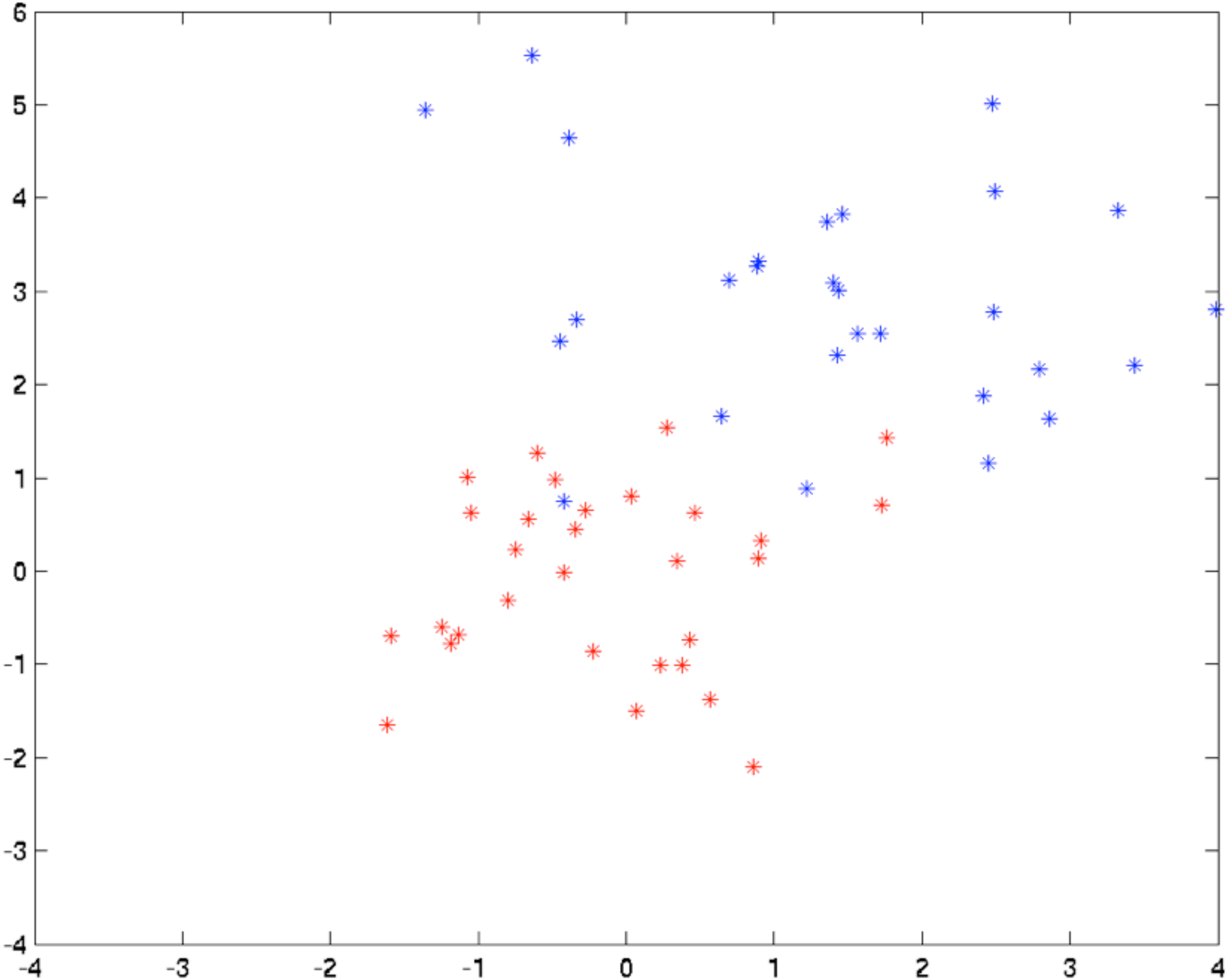
- Binary classification
  - Estimate $p(x|y=1)$ and $p(x|y=-1)$
  - Use Bayes rule

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{\frac{1}{m_y}\sum_{y_i=y} k(x_i,x)\cdot\frac{m_y}{m}}{\frac{1}{m}\sum_i k(x_i,x)}$$

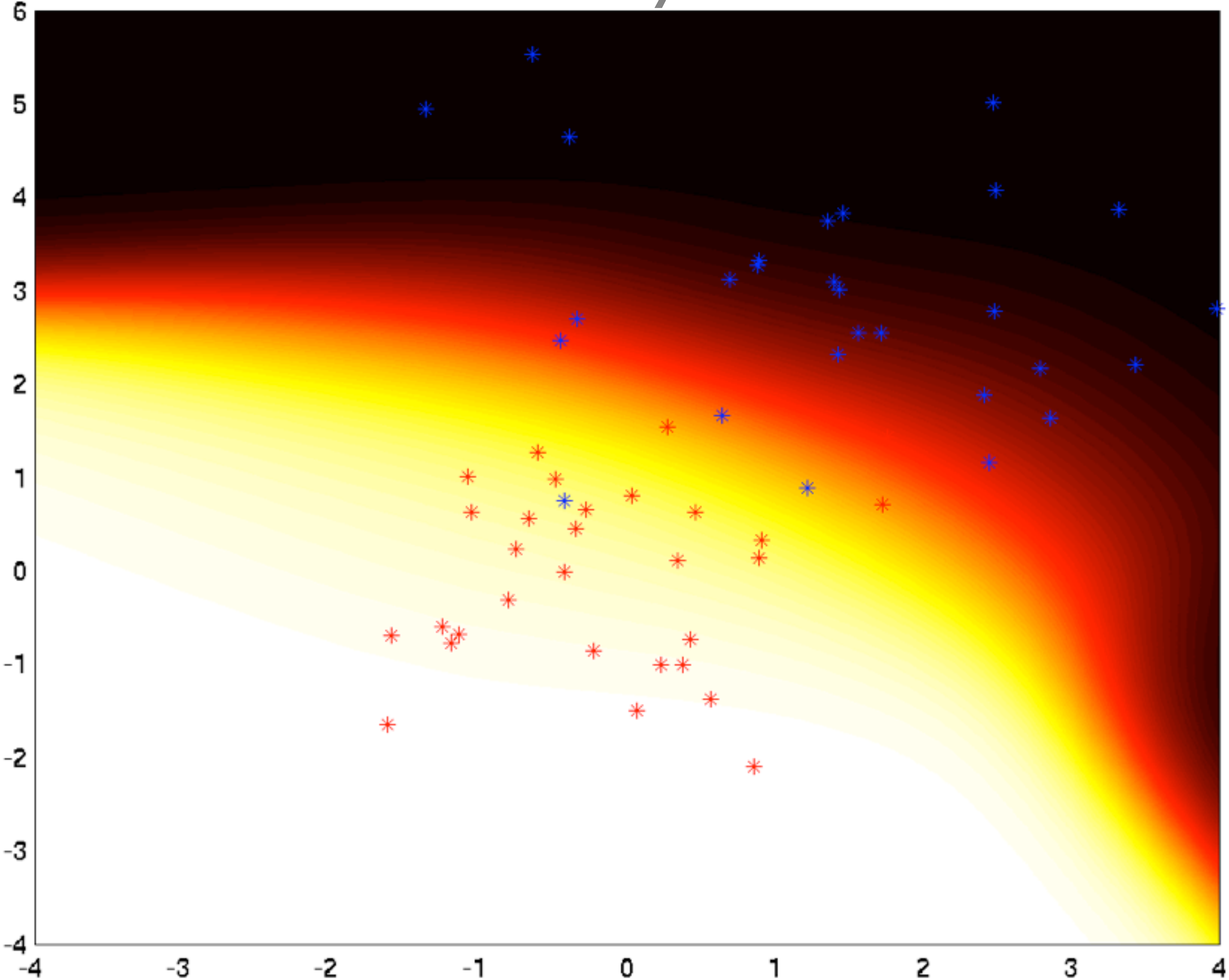
  - Decision boundary

$$p(y=1|x) - p(y=-1|x) = \frac{\sum_j y_j k(x_j,x)}{\sum_i k(x_i,x)} = \sum_j y_j \frac{k(x_j,x)}{\sum_i k(x_i,x)}$$

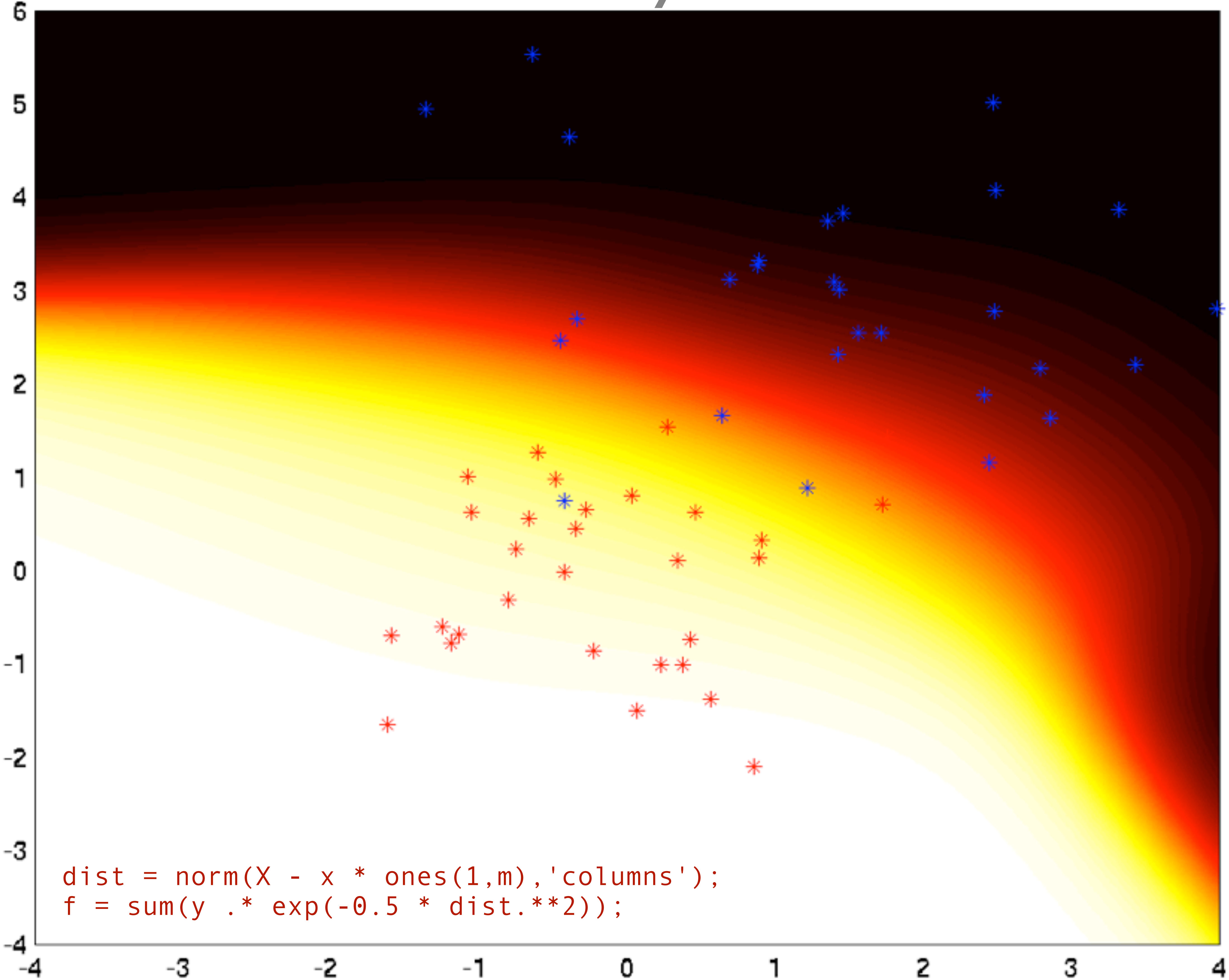
local weights


6
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4

# Watson-Nadaraya Classifier



# Watson-Nadaraya Classifier



# Watson Nadaraya Regression

- Binary classification

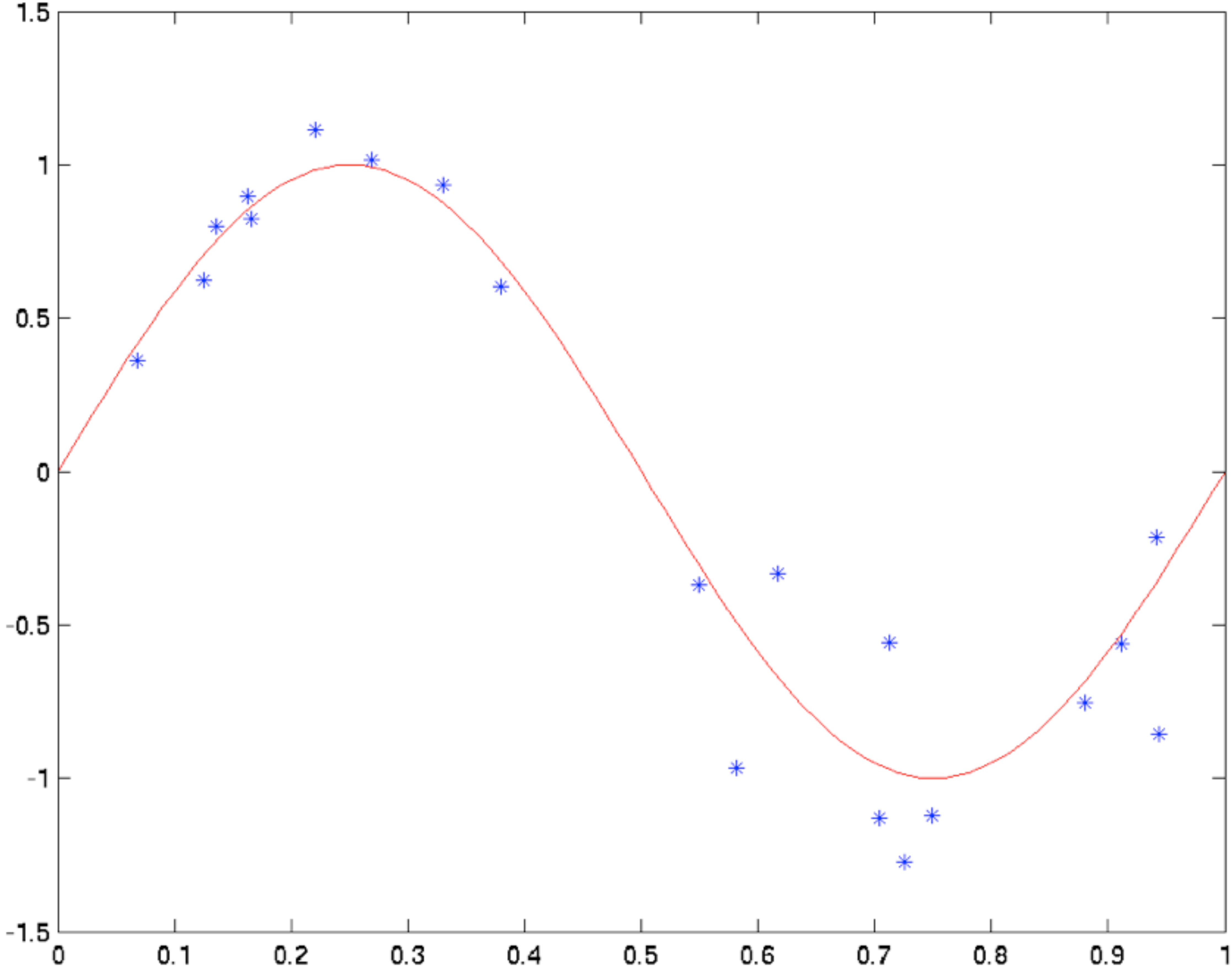
$$p(y=1|x) - p(y=-1|x) = \frac{\sum_j y_j k(x_j, x)}{\sum_i k(x_i, x)} = \sum_j y_j \frac{k(x_j, x)}{\sum_i k(x_i, x)}$$

- Regression - use same weighted expansion

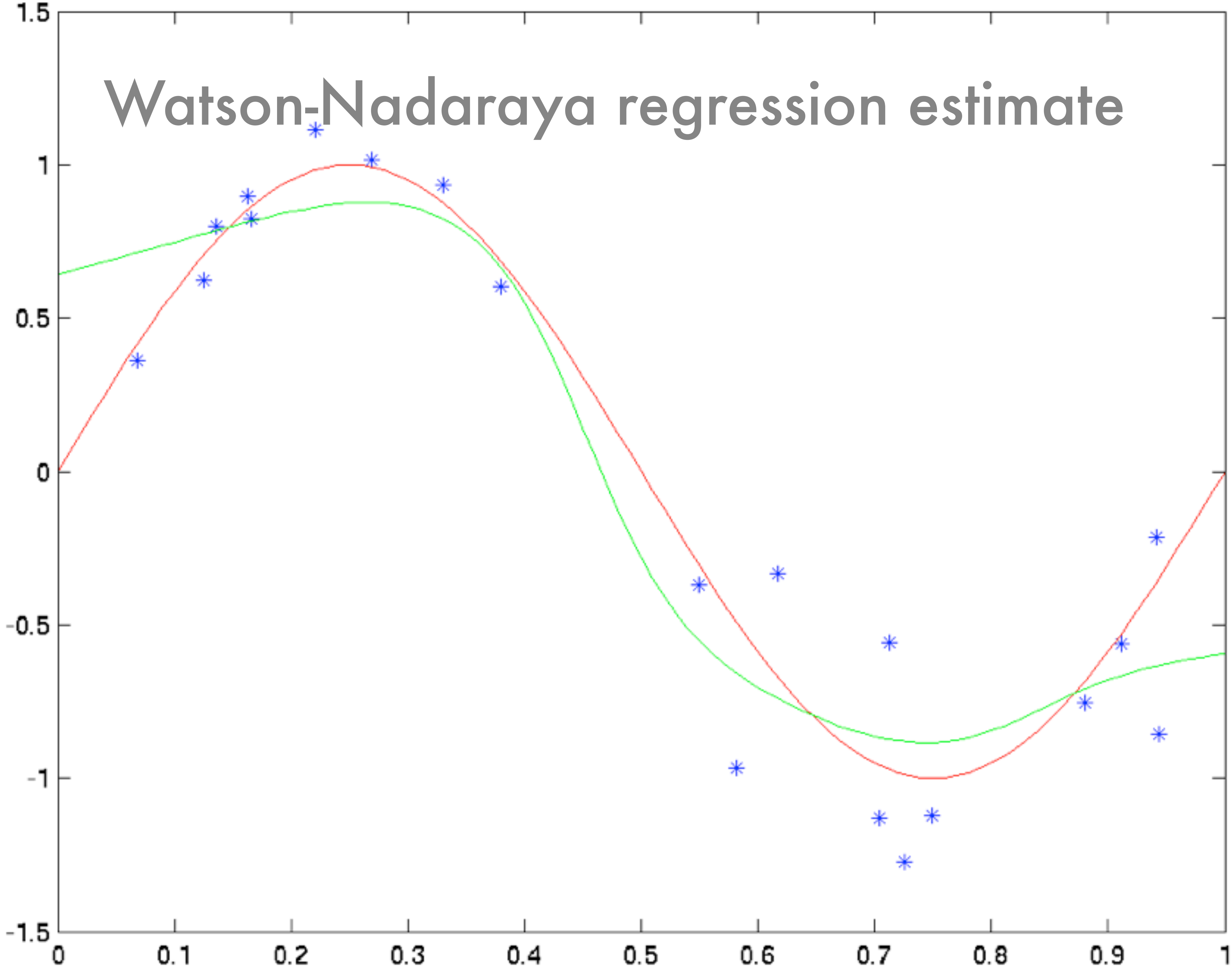
$$\hat{y}(x) = \sum_j y_j \frac{k(x_j, x)}{\sum_i k(x_i, x)}$$

labels

local weights


1.5
1
0.5
0
-0.5
-1
-1.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

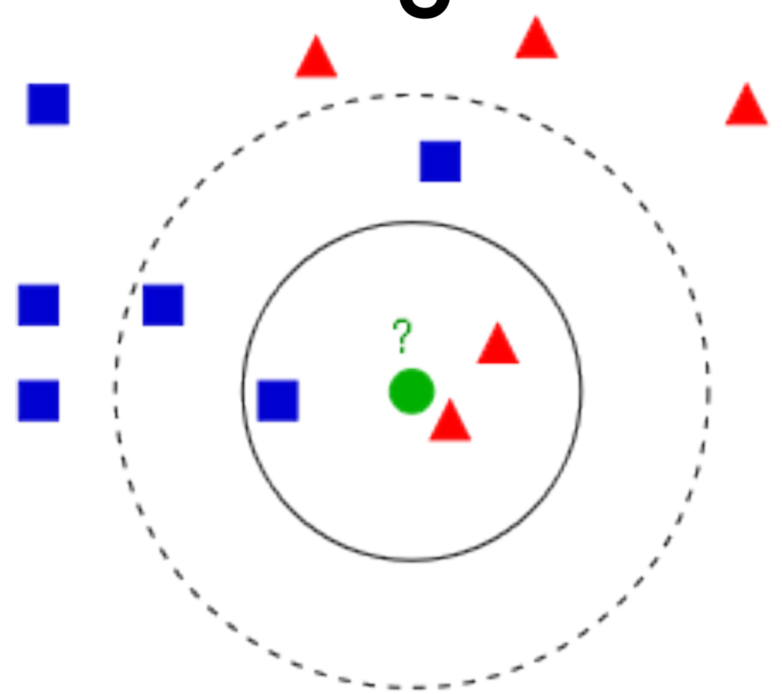
# Watson-Nadaraya regression estimate



# Nearest Neighbor

# Nearest Neighbors

- Table lookup
  For previously seen instance remember label
- Nearest neighbor
  - Pick label of most similar neighbor
  - Slight improvement - use k-nearest neighbors
  - For regression average
  - Really useful baseline!
  - Easy to implement for small amounts of data.



# Relation to Watson Nadaraya

- Watson Nadaraya estimator

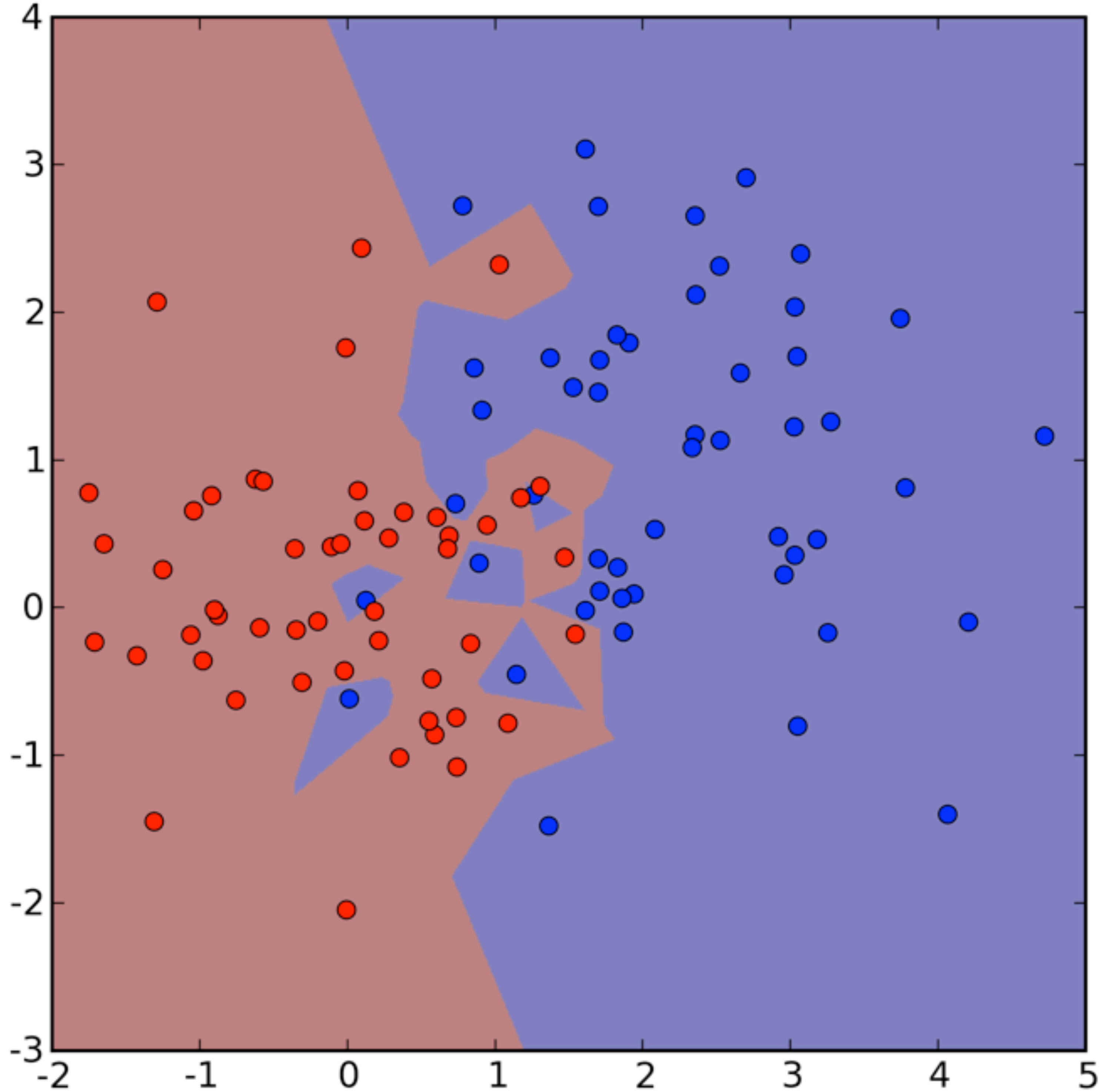
$$\hat{y}(x) = \sum_j y_j \frac{k(x_j, x)}{\sum_i k(x_i, x)} = \sum_j y_j w_j(x)$$

- Nearest neighbor estimator

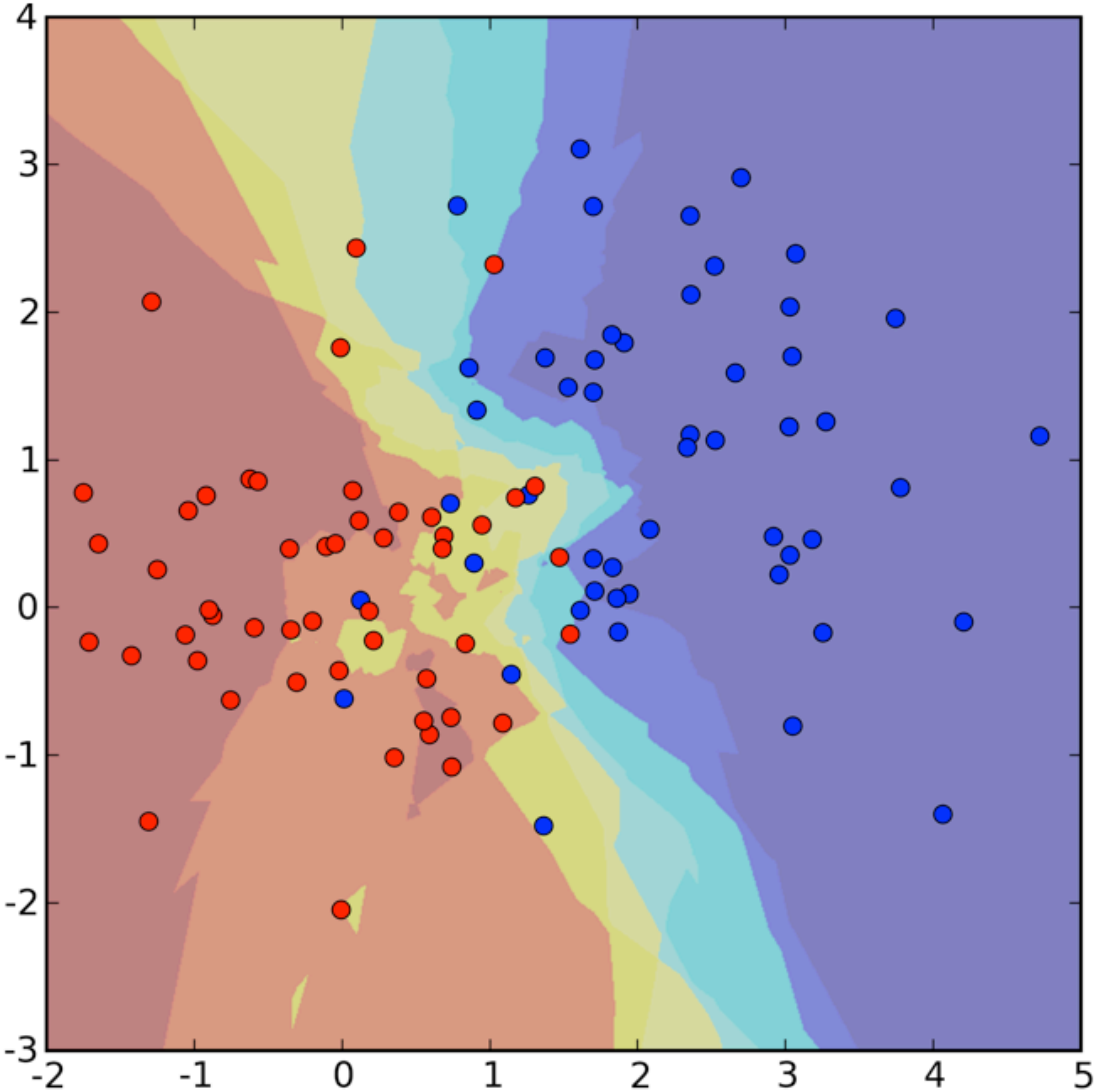
$$\hat{y}(x) = \sum_j y_j \frac{k(x_j, x)}{\sum_i k(x_i, x)} = \sum_j y_j w_j(x)$$

Neighborhood function is hard threshold.

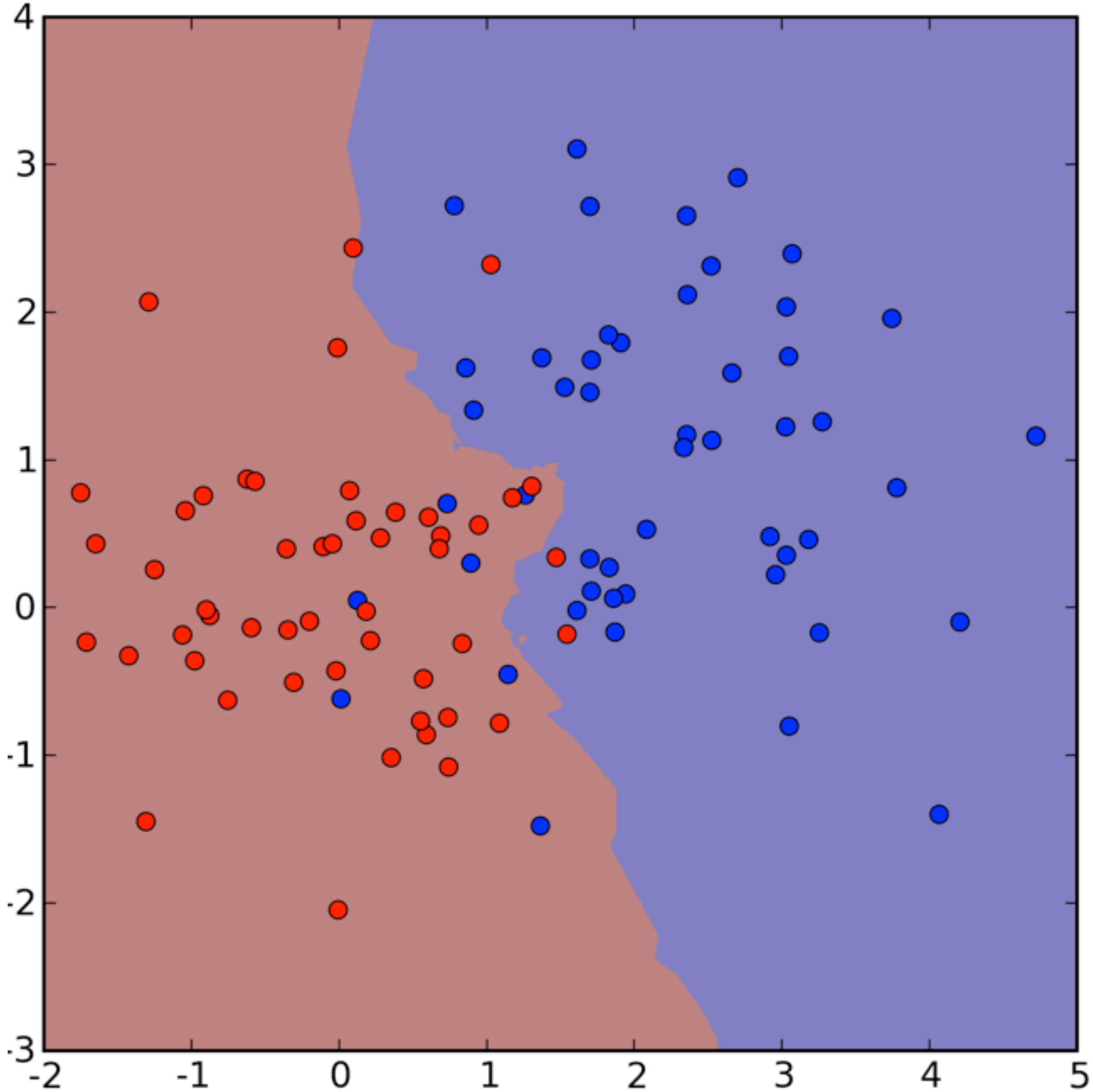
# 1-Nearest Neighbor



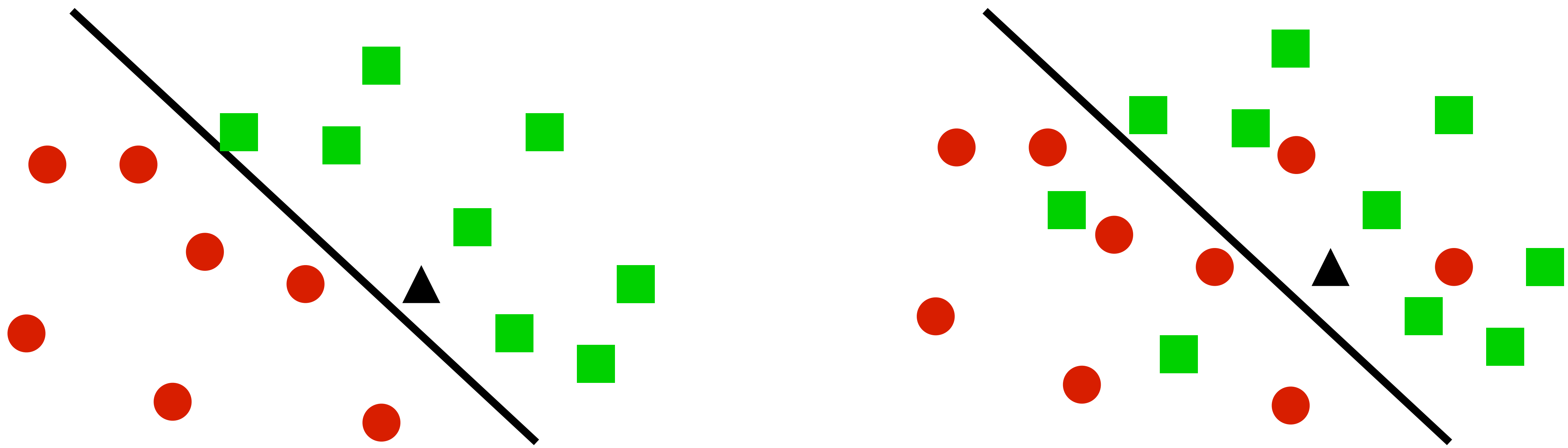
# 4-Nearest Neighbors



# 4-Nearest Neighbors Sign

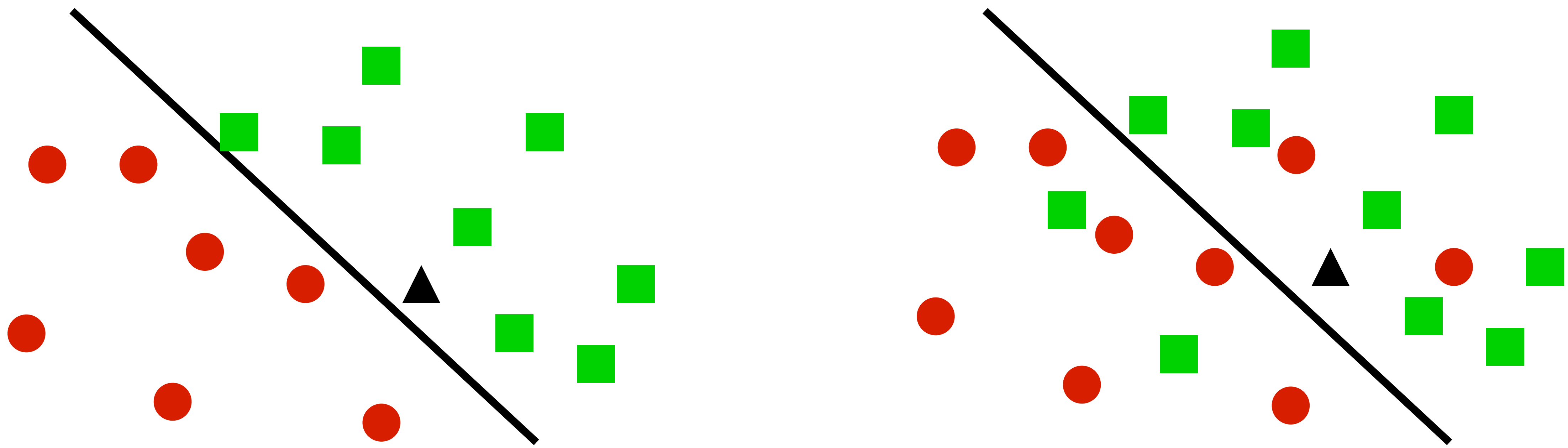


# If we get more data



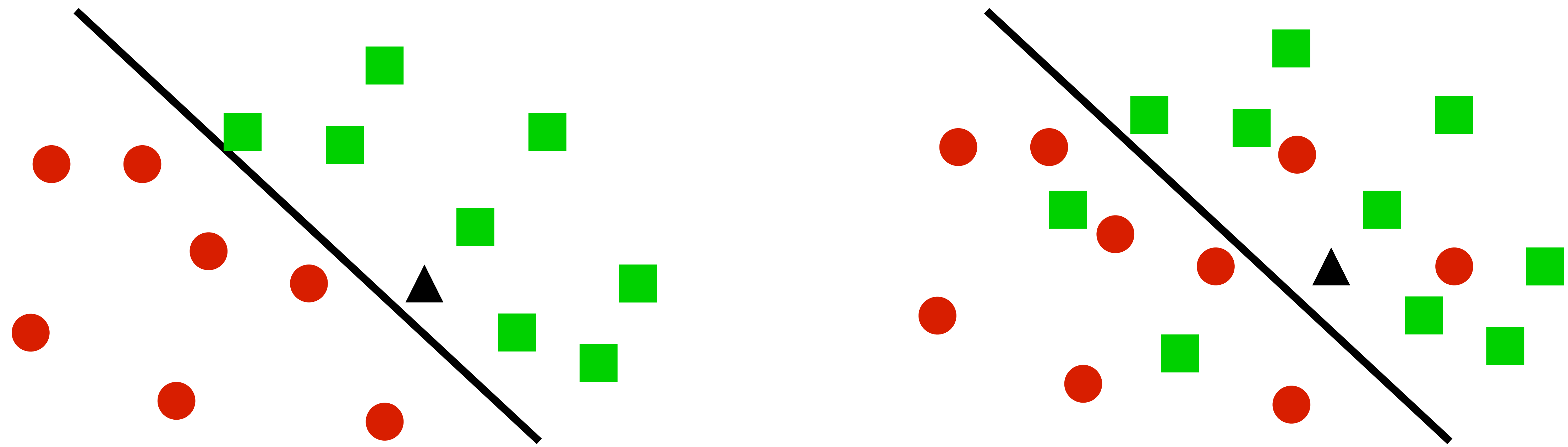
- 1 Nearest Neighbor
  - Converges to perfect solution if separation
  - Twice the minimal error rate 2p(1-p) for noisy problems
- k-Nearest Neighbor
  - Converges to perfect solution if separation (but needs more data)
  - Converges to minimal error min(p,1-p) for noisy problems (use increasing k)

# 1 Nearest Neighbor



- For given point x take ϵ neighborhood N with probability mass > d/n
- Probability that at least one point of n is in this neighborhood is $1-e^{-d}$ so we can make this small
- Assume that probability mass doesn't change much in neighborhood
- Probability that labels of query and point do not match is 2p(1-p) (up to some approximation error in neighborhood)

# k Nearest Neighbor



- For given point x take ϵ neighborhood N with probability mass > dk/n
- Small probability that we don't have at least k points in neighborhood.
- Assume that probability mass doesn't change much in neighborhood
- Bound probability that majority of points doesn't match majority for p (e.g. via Hoeffding's theorem for tail). Show that it vanishes
- Error is therefore min(p, 1-p), i.e. Bayes optimal error.

# Fast lookup

- KD trees (Moore et al.)
  - Partition space (one dimension at a time)
  - Only search for subset that contains point
- Cover trees (Beygelzimer et al.)
  - Hierarchically partition space with distance guarantees
  - No need for nonoverlapping sets
  - Bounded number of paths to follow (logarithmic time lookup)

# Silverman's Rule

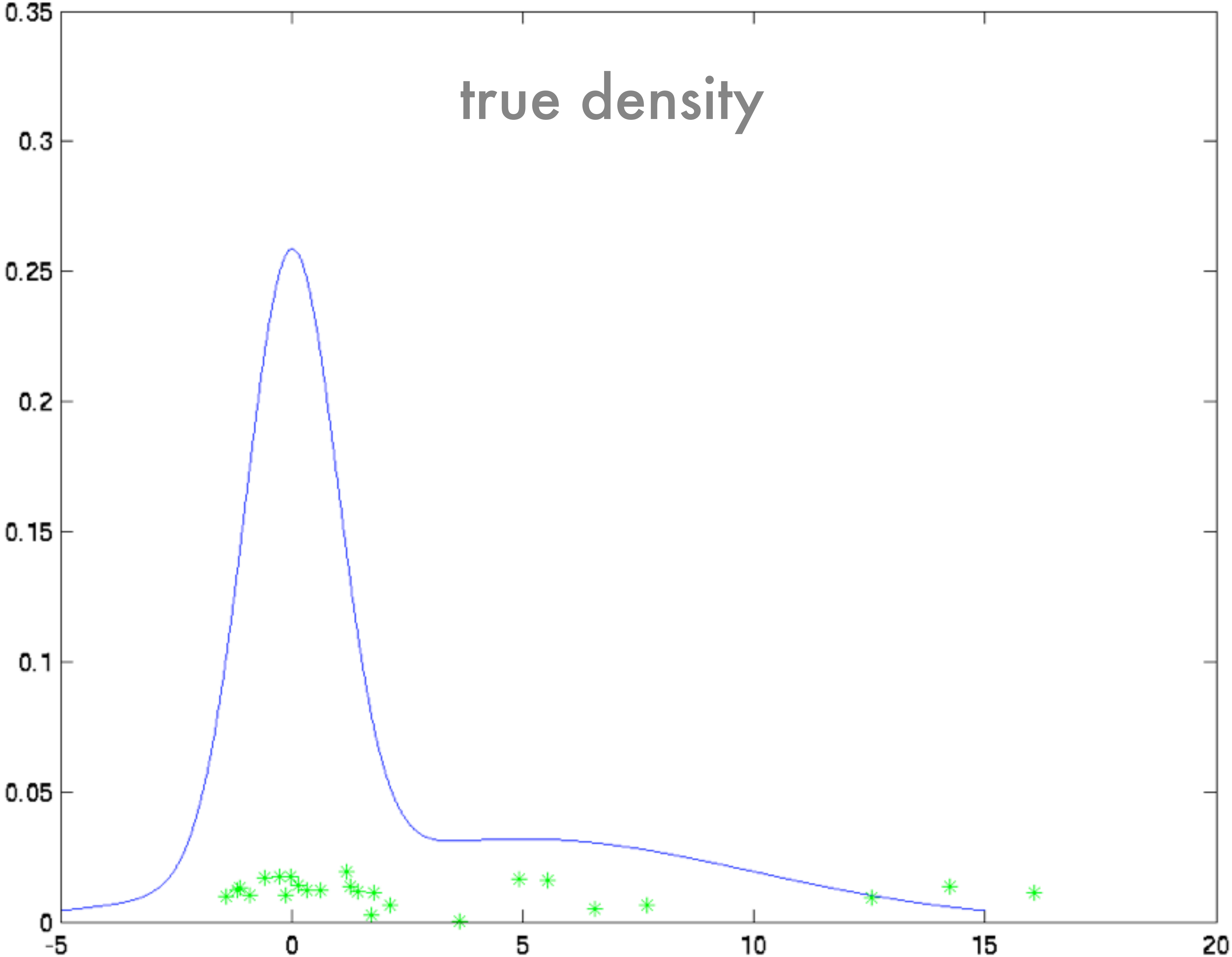
Bernard Silverman

# Silverman's rule

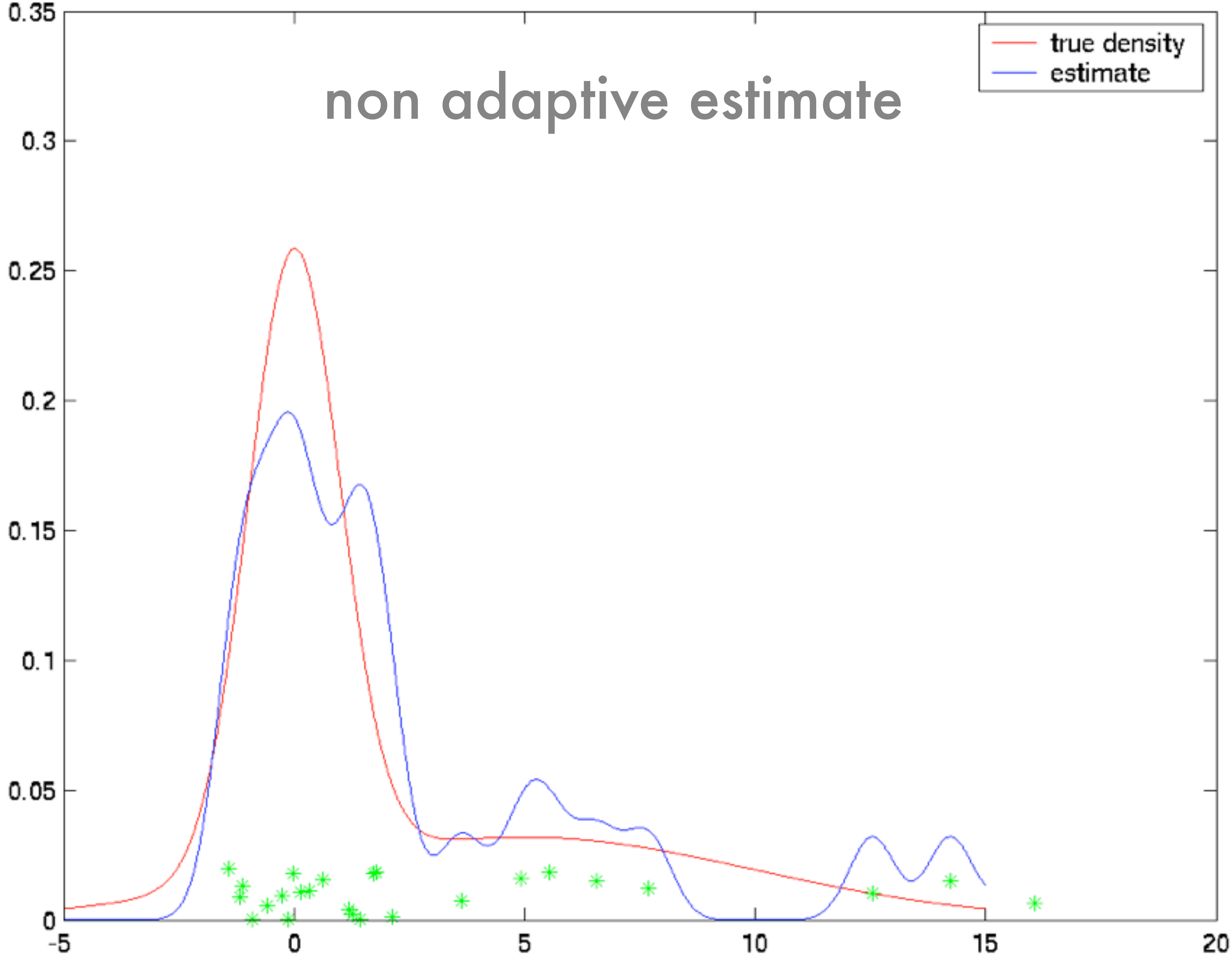
- Chicken and egg problem
  - Want wide kernel for low density region
  - Want narrow kernel where we have much data
  - Need density estimate to estimate density
- Simple hack
  Use average distance from k nearest neighbors

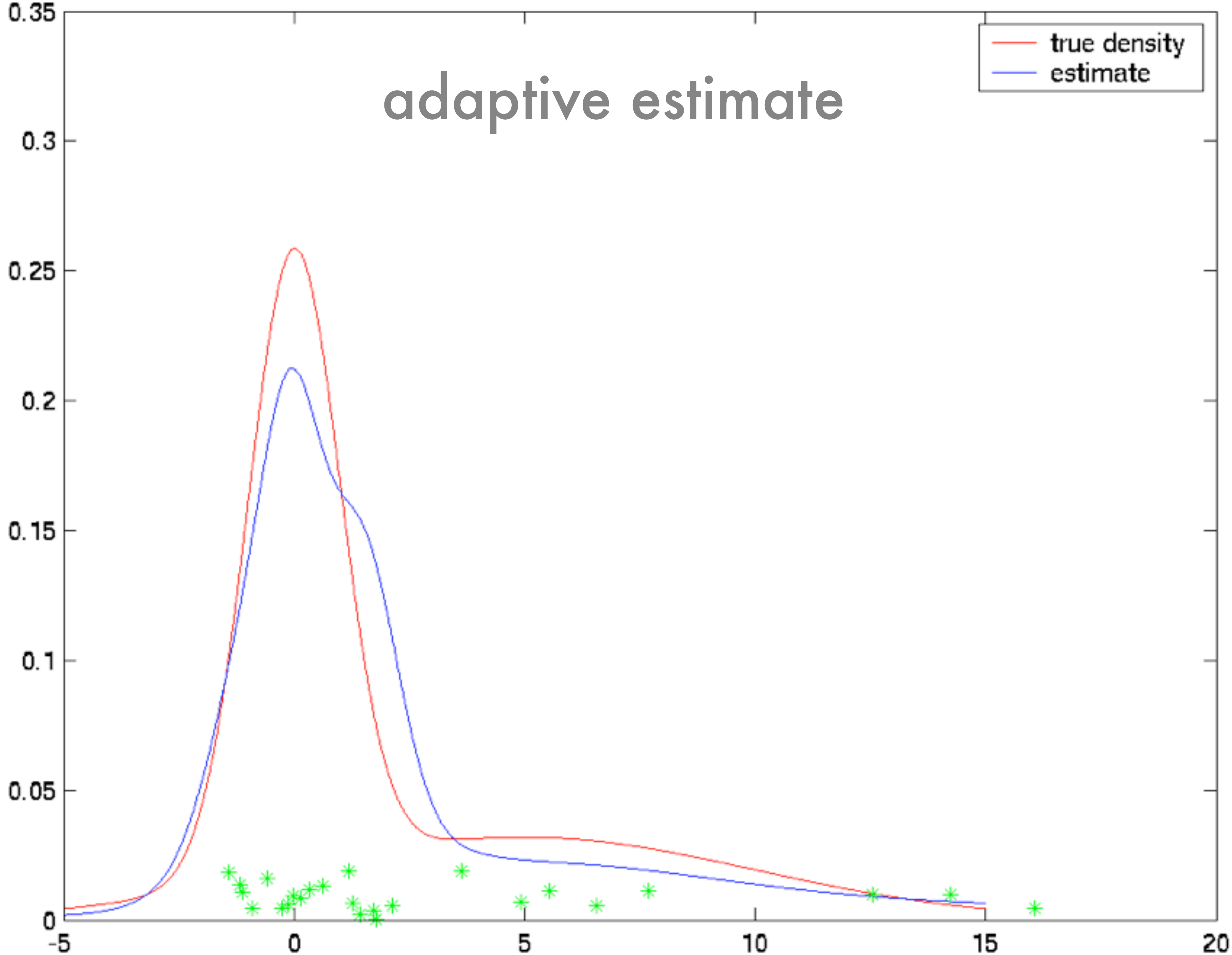
$$r_i = \frac{r}{k} \sum_{x \in \mathrm{NN}(x_i, k)} \|x_i - x\|$$

- Nonuniform bandwidth for smoother.

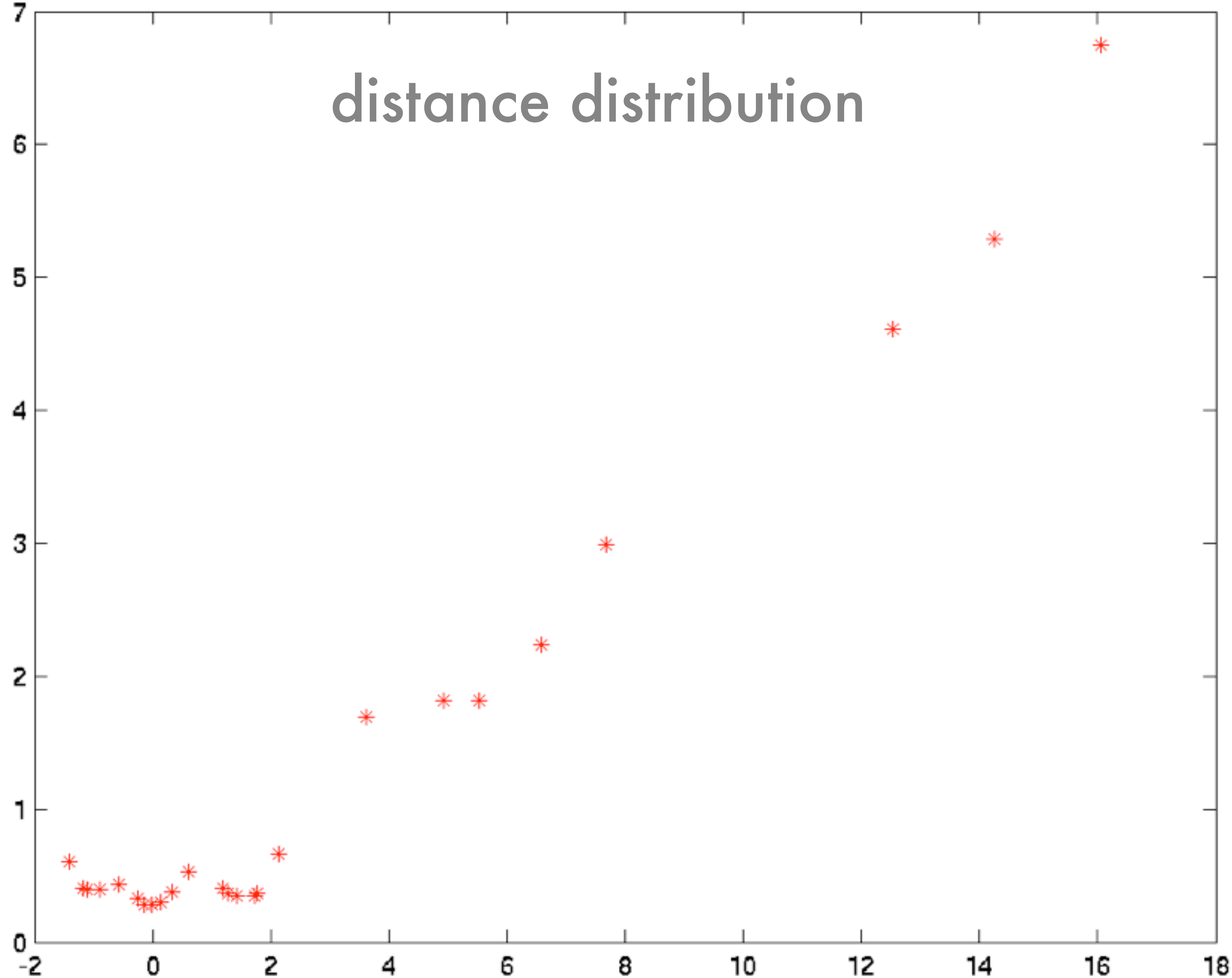


true density
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-5
0
5
10
15
20


non adaptive estimate
true density
estimate
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-5
0
5
10
15
20



adaptive estimate
true density
estimate
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-5
0
5
10
15
20


distance distribution
0
1
2
3
4
5
6
7
-2
0
2
4
6
8
10
12
14
16
18

# Summary

- Parzen Windows
  Kernels, algorithm
- Model selection
  Crossvalidation, leave one out, bias variance
- Watson-Nadaraya estimator
  Classification, regression, novelty detection

# Further Reading

- Cover tree homepage (paper & code)
  http://hunch.net/~jl/projects/cover_tree/cover_tree.html
- http://doi.acm.org/10.1145/361002.361007 (kd trees, original paper)
- http://www.autonlab.org/autonweb/14665/version/2/part/5/data/moore-tutorial.pdf
  (Andrew Moore's tutorial from his PhD thesis)
- Nadaraya's regression estimator (1964)
  http://dx.doi.org/10.1137/1109020
- Watson's regression estimator (1964)
  http://www.jstor.org/stable/25049340
- Watson-Nadaraya regression package in R
  http://cran.r-project.org/web/packages/np/index.html
- Stone's k-NN regression consistency proof
  http://projecteuclid.org/euclid.aos/1176343886
- Cover and Hart's k-NN classification consistency proof
  http://www-isl.stanford.edu/people/cover/papers/transIT/0021cove.pdf
- Tom Cover's rate analysis for k-NN
  Rates of Convergence for Nearest Neighbor Procedures.
- Sanjoy Dasgupta's analysis for k-NN estimation with selective sampling
  http://cseweb.ucsd.edu/~dasgupta/papers/nnactive.pdf
- Multiedit & Condense (Dasarathy, Sanchez, Townsend)
  http://cgm.cs.mcgill.ca/~godfried/teaching/pr-notes/dasarathy.pdf
- Geometric approximation via core sets
  http://valis.cs.uiuc.edu/~sariel/papers/04/survey/survey.pdf