Syllabus Notes

- Today we will hopefully get to some of §16. It is worth reading this chapter.
- For those going on in vision, this chapter is a must!
- Book does segmentation, then lines. We will do lines than segmentation.
- Probability reference (may need login “me”, pw, “read4fun”):

  http://www.cs.arizona.edu/people/kobus/teaching/reading/statistical_modeling/ua_cs_only/probability.pdf

Probabilistic Fitting

- Given a model with parameters θ
- Now consider some observations, $\mathbf{x}$
- Suppose that the observations are independent
- So, given the model, the probability of observing the data is given by

$$P(\mathbf{x} \mid \theta) = \prod_i P(x_i \mid \theta)$$

- But what we really want is the probability of the model (parameters) given the data!

Probabilistic Fitting

- Bayes rule: $P(A \mid B) = P(B \mid A)P(A)/P(B)$
- So, $P(\theta \mid \mathbf{x}) = P(\mathbf{x} \mid \theta)P(\theta)/P(\mathbf{x})$
- $P(\theta)$ is the prior probability on the parameters (often taken to be uniform
- $P(\mathbf{x})$ is usually not of interest
- Often use $P(\theta \mid \mathbf{x})$ $P(\mathbf{x} \mid \theta)$

Probabilistic Fitting

- Now the objective is to find the parameters θ such that this *likelihood* is maximum

- Note--this is the same as finding the parameters which minimize the **negative log likelihood** (very convenient if data is independent).

$$\underset{\theta}{\text{minimize}} \quad -\sum_i \log(P(x_i \mid \theta))$$

- Back to lines: au+bv+c=0 where a^2+b^2=1
- Algebraic fact: Distance squared from (x,y) to this line is (ax+by+c)^2
- **Generative model** for lines: Choose point on line, and then, with probability p(d), **normally distributed** (Gaussian) go a distance d from the line.
- Now the probability of an observed (x,y) is given by

$$P((x,y)\,|\,\square) \quad \exp(\square \frac{(ax+by+c)^2}{2\square^2})$$

Now the probability of an observed (x,y) is given by

$$P((x,y)\,|\,\square) \quad \exp(\square \frac{(ax+by+c)^2}{2\square^2})$$

The negative log is

$$\frac{(ax+by+c)^2}{2\square^2}$$

And the negative log likelihood of multiple observations is

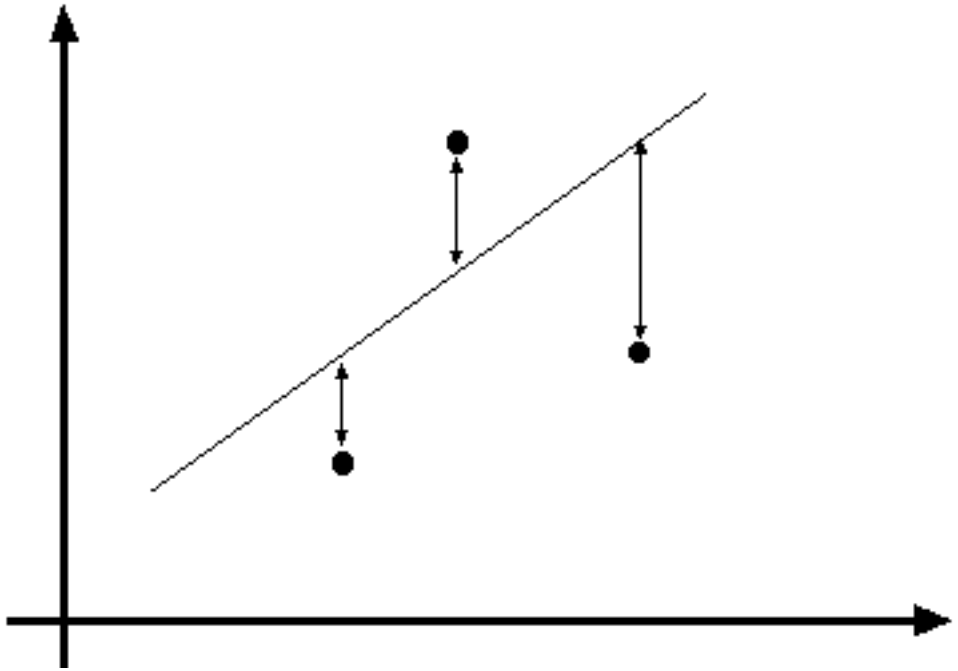
$$\frac{1}{2\square^2}\square_i (ax_i+by_i+c)^2$$

From the previous slide, we had that the negative log likelihood of multiple observations is given by

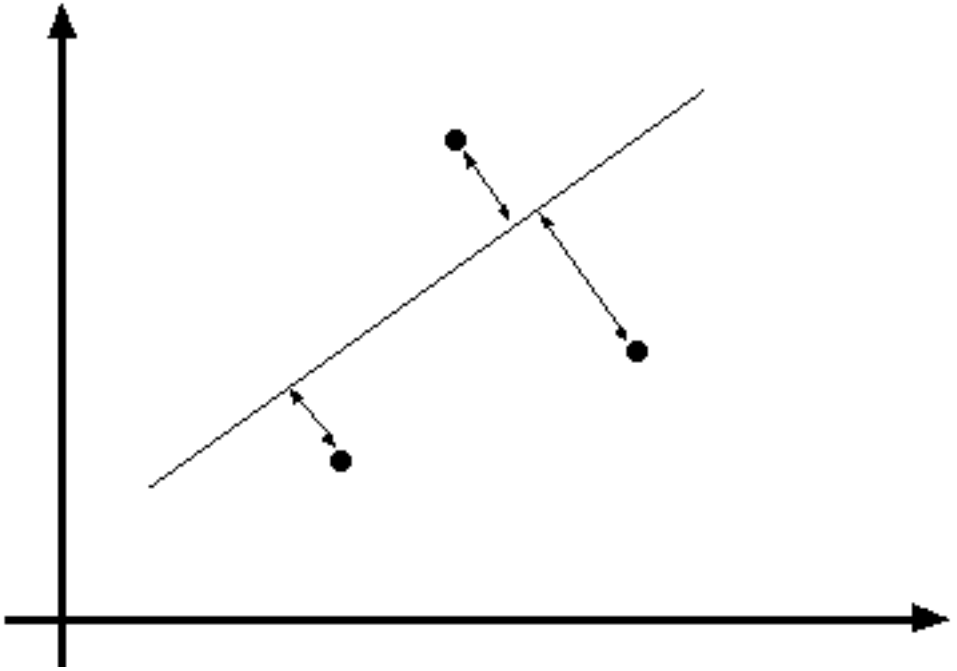
$$\frac{1}{2\sigma^2}\sum_i (ax_i + by_i + c)^2 \qquad \text{(where } a^2 + b^2 = 1\text{)}$$

We could solve this by considering derivatives, but this should be recognizable as homogeneous least squares

Thus we have shown that least squares is maximum likelihood estimation under normality (Gaussian) error statistics!



Line fitting by minimizing error == maximum likelihood estimation



Fitting curves other than lines

- In principle, an easy generalization
  - Assuming Gaussian error statistics, Euclidean distance is a good measure
  - The probability of obtaining a point, given a curve, is given by a negative exponential of distance squared
- In practice, can be hard
  - It can be difficult to compute the distance between a point and a curve
  - Circles, ellipses, and a few others are not too hard
  - Otherwise, craft an approximation
  - §15.3 has more

Now the hard part

- Robustness
  - Squared error grows rapidly as distance increases
  - Since large distance is unlikely given Gaussian assumption, assumption or model is likely incorrect!
- How do we know whether a point is on the line?
  - Incremental line fitting
  - K-means
  - Probabilistic with missing data

Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
   Transfer first few points on the curve to the line point list
   Fit line to line point list
   While fitted line is good enough
      Transfer the next point on the curve
         to the line point list and refit the line
   end
   Transfer last point(s) back to curve
   Refit line
   Attach line to line list
end
```

Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

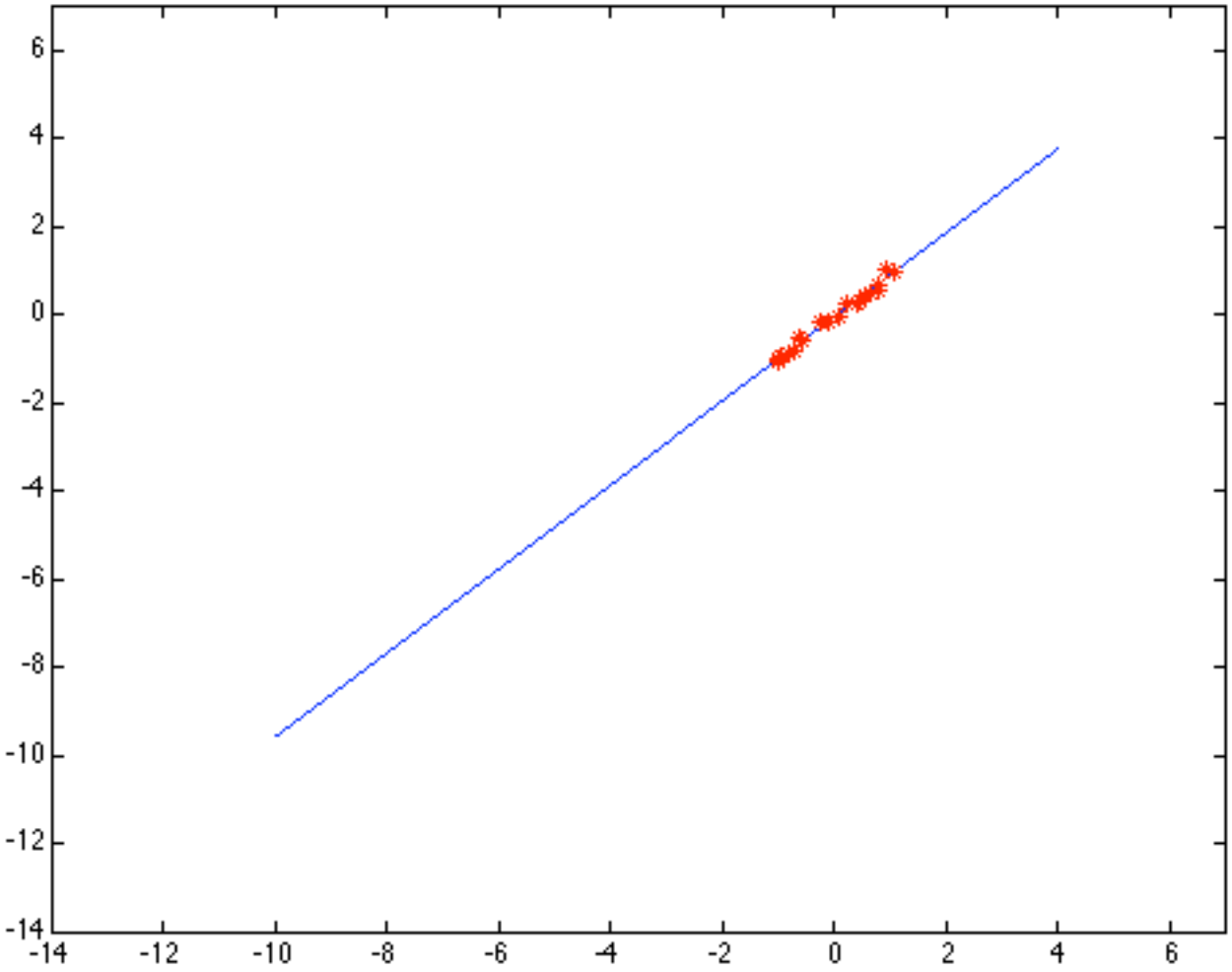
Hypothesize k lines (perhaps uniformly at random)
or
Hypothesize an assignment of lines to points
  and then fit lines using this assignment

Until convergence
  Allocate each point to the closest line
  Refit lines
end

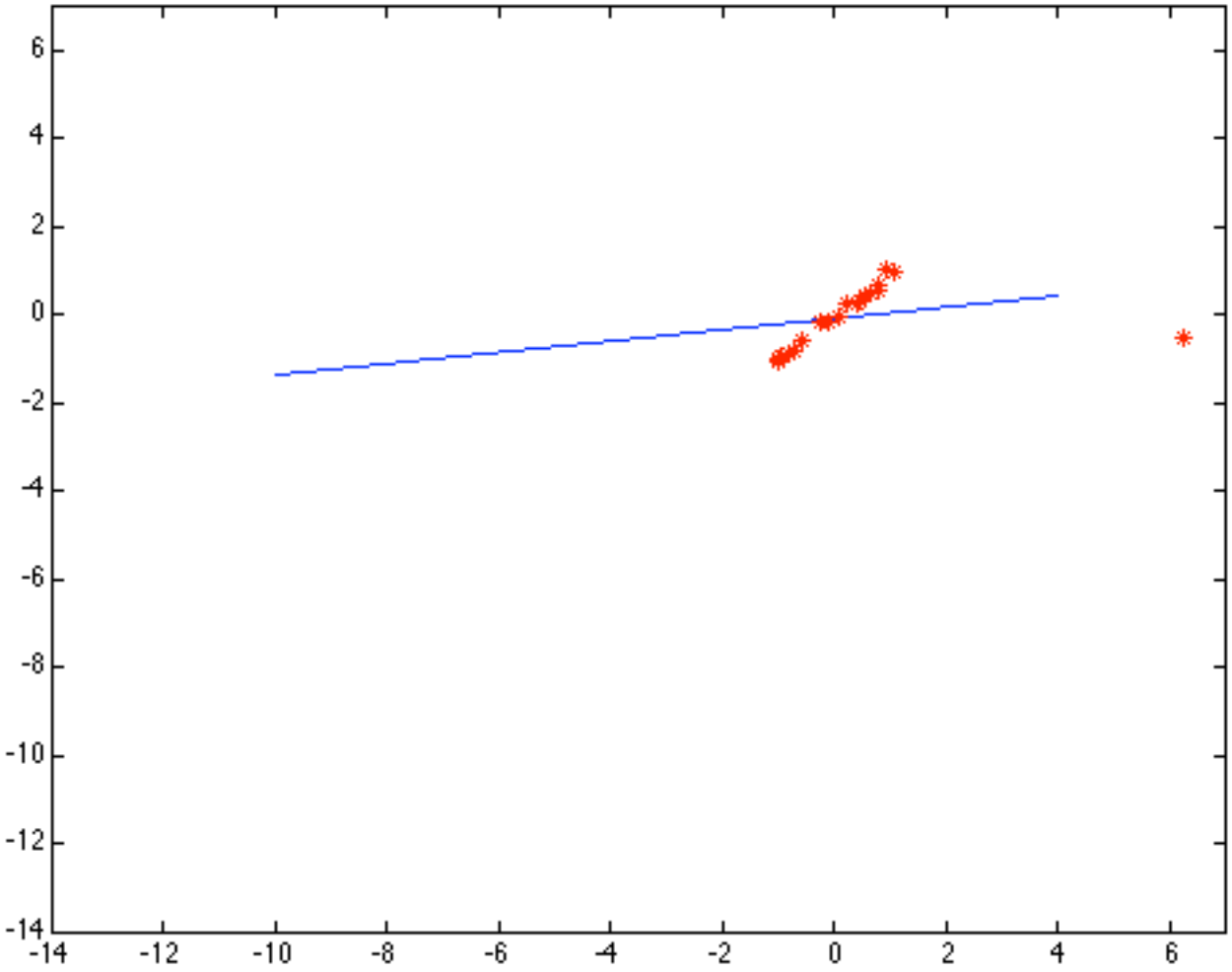
Robustness

- Squared error is a liability when model is wrong
  - One fix is EM - we'll do this shortly
  - Another is an M-estimator
    - Square nearby, threshold far away
  - A third is RANSAC
    - Search for good points

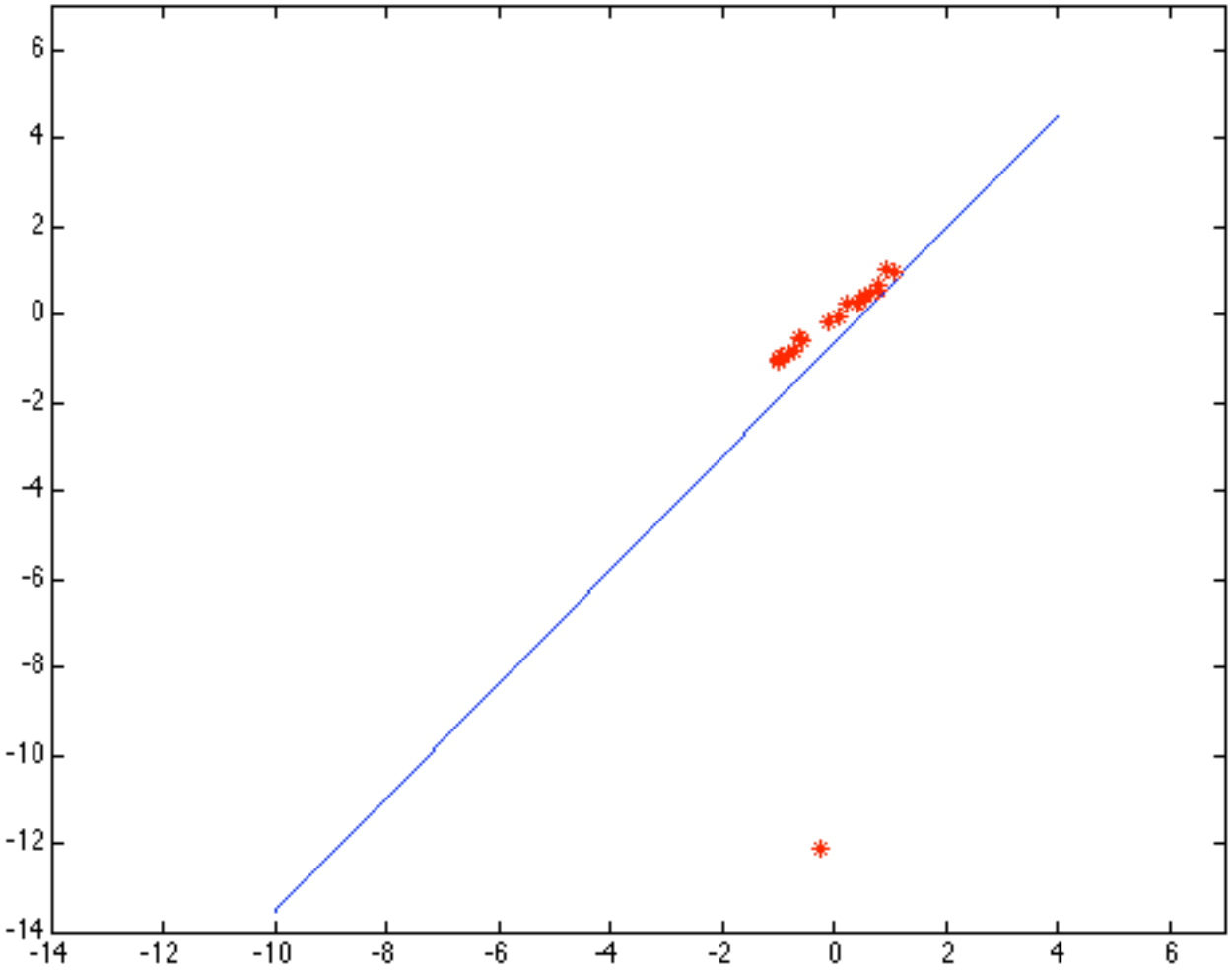
Least squares fit (good example)



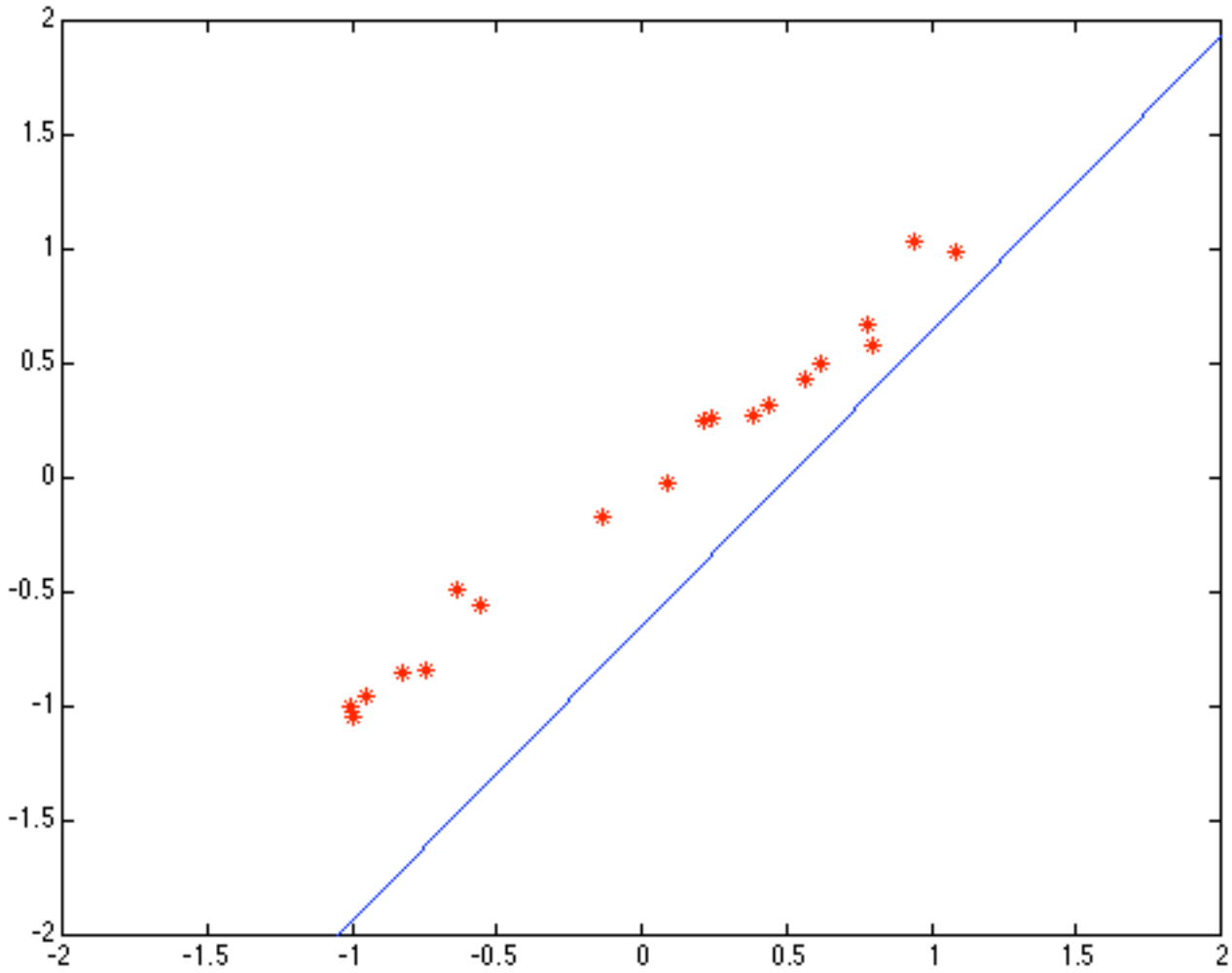
Least squares fit (destroyed by outlier)



Least squares fit (warped by outlier)



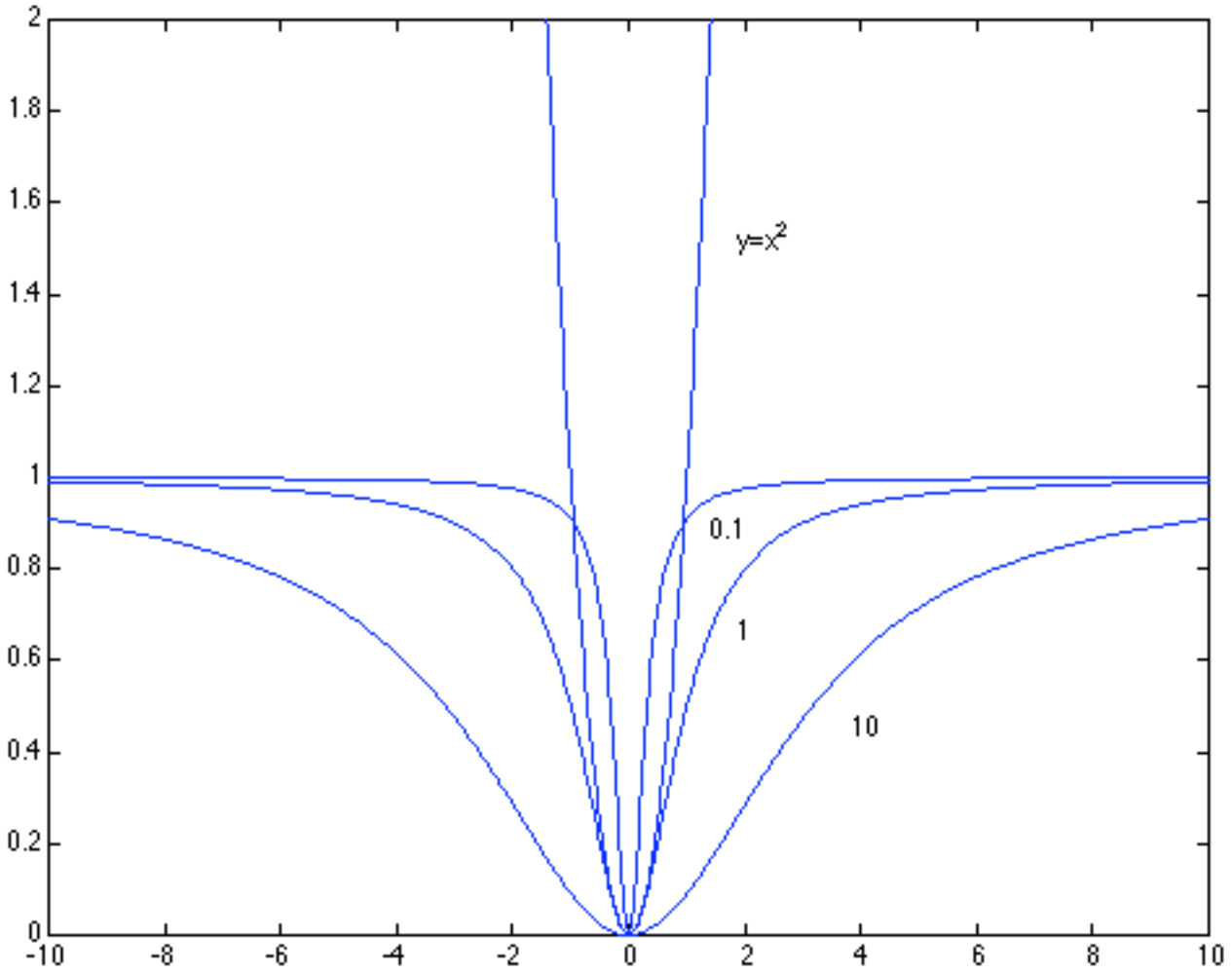
Least squares fit (previous slide details)



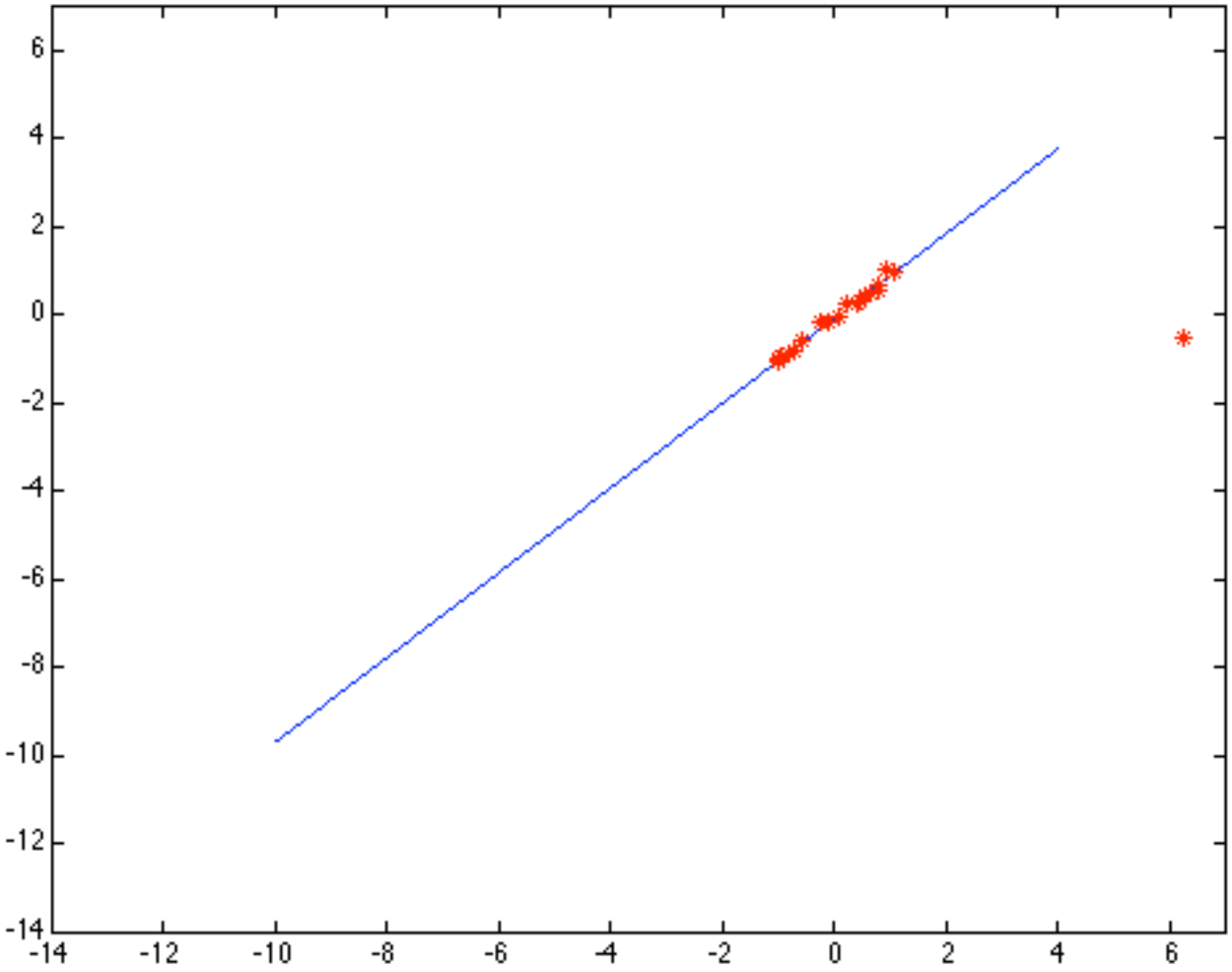
Some robust error measures

$y=x^2/(x^2+s^2)$

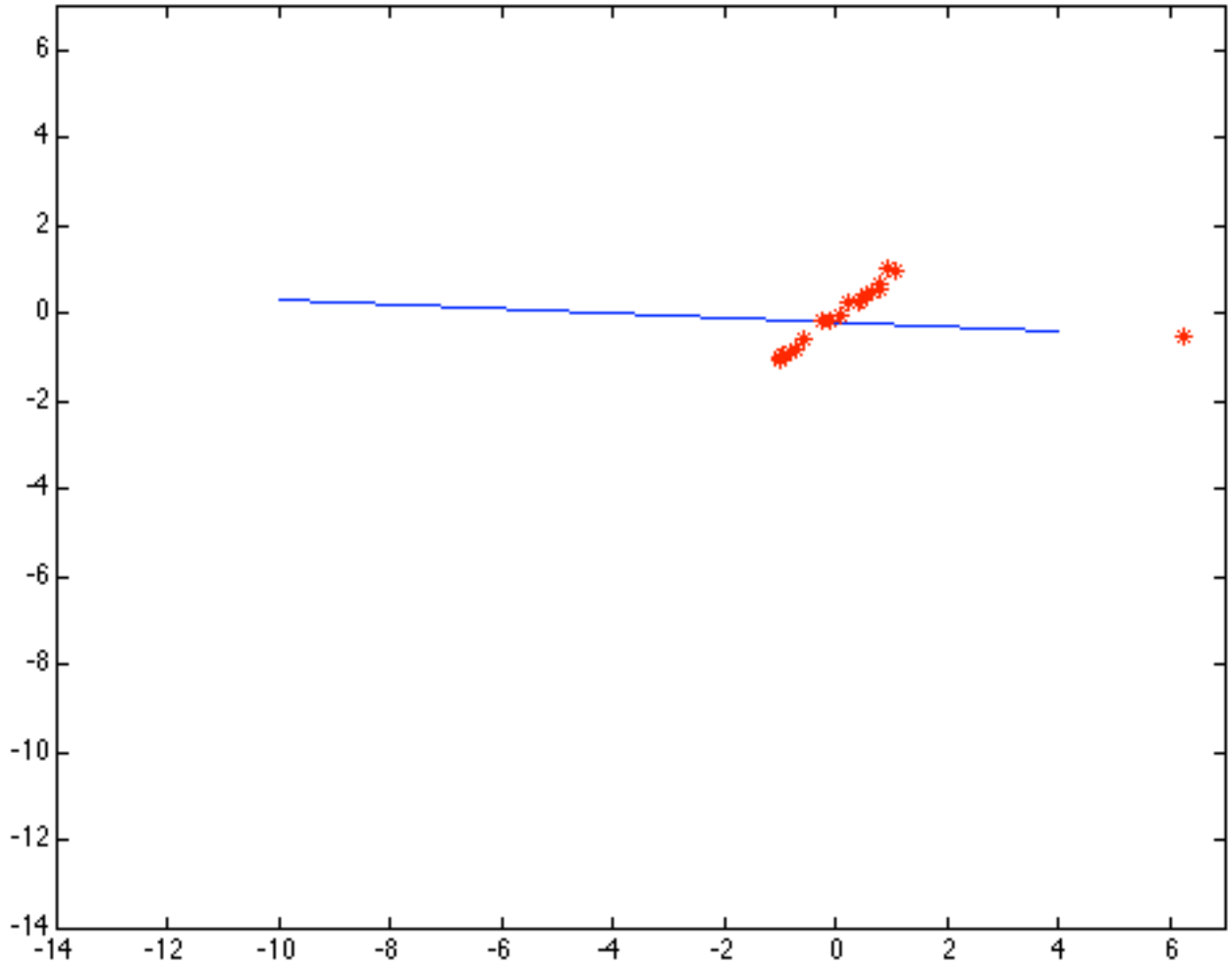
(Results for different values of s shown)



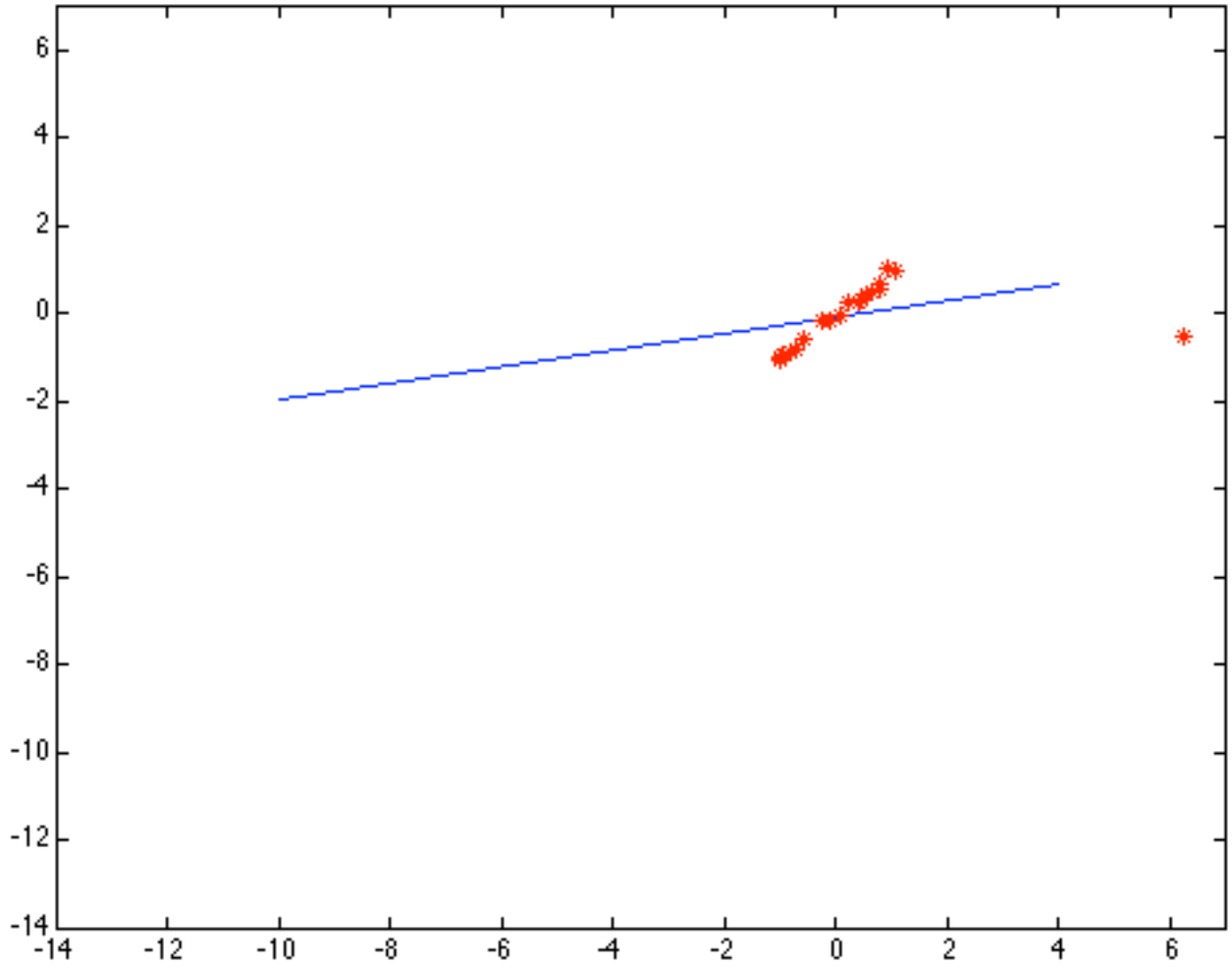
Line fit with estimator with good choice for s



Line fit with estimator with choice for s that is too small



Line fit with estimator with choice for s that is too big



RANSAC

- Choose a small subset uniformly at random
- Fit to that
- Anything that is close to result is signal; all others are noise
- Refit
- Do this many times and choose the best

- Issues
  - How many times?
    - Often enough that we are likely to have a good line
  - How big a subset?
    - Smallest possible
  - What does close mean?
    - Depends on the problem
  - What is a good line?
    - One where the number of nearby points is so big it is unlikely to be all outliers

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
    n — the smallest number of points required
    k — the number of iterations required
    t — the threshold used to identify a point that fits well
    d — the number of nearby points required
      to assert a model fits well
Until k iterations have occurred
    Draw a sample of n points from the data
      uniformly and at random
    Fit to that set of n points
    For each data point outside the sample
      Test the distance from the point to the line
        against t; if the distance from the point to the line
        is less than t, the point is close
    end
    If there are d or more points close to the line
      then there is a good fit. Refit the line using all
      these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion

Missing variable problems

- In many vision problems, if some variables were known the maximum likelihood inference problem would be easy
  - fitting; if we knew which line each token came from, it would be easy to determine line parameters
  - segmentation; if we knew the segment each pixel came from, it would be easy to determine the segment parameters
  - many, many, others!

Missing variable problems

- Strategy
  - estimate appropriate values for the missing variables
  - plug these in and now estimate parameters
  - re-estimate appropriate values for missing variables, continue
- Example with lines
  - guess which line gets which point
  - now fit the lines
  - now reallocate points to lines, using our knowledge of the lines
  - now refit, etc.
- We've seen this line of thought before (k means)

Missing variables - strategy

- In the Expectation-Maximization algorithm, we use the expected values of the missing values as the estimate.
- Thus iterate until convergence
  - replace missing variable with **expected** values, given **fixed** values of parameters
  - fix missing variables, choose parameters to maximize likelihood given fixed values of missing variables
- Line example
  - iterate till convergence
  - allocate each point to a line with a **weight**, which is the probability of the point given the line
  - refit lines to the weighted set of points
- Converges to local extremum
- Unlike K-means, we do not make a hard assignment; rather we the expected value (weight) which is a **soft** assignment.

Iterative Approach

- EM is basically gradient descent on the log likelihood.
- Gradient descent is working towards a local optimum by moving in the direction of maximum change (should know this).
- In the case of our least squares fit to a line, we end up with weights for each points which are indicative of how likely that point is on the line.
- We can easily fit this if the weights are constant.
- However, the weights are function of the line parameters---hence the iterative approach.

Lines and robustness

- We have one line, and n points
- Some come from the line, some from “noise”
- This is a mixture model
- We wish to determine
  - line parameters
  - P(comes from line)
  - Note that P(comes from noise)=(1-P(comes from line))

Lines and robustness

P(point | line and noise parameters)

= P(point | line)P(comes from line)
+ P(point | noise)P(comes from line)

= P(point|line)□ + P(point|noise)(1-□)

Estimating the mixture model

- Introduce a set of hidden variables, α, one for each point. They are one when the point is on the line, and zero when off.
- If these are known, the negative log-likelihood becomes:
- Here K is a normalizing constant, kn is the noise intensity (comments on choosing this later), and the parameters of the line are θ and c.

$$Q_c(x;\theta) = \sum_i \left(\alpha_i \frac{(x_i \cos\theta + y_i \sin\theta + c)^2}{2\sigma^2} + (1-\alpha_i) k_n \right) + K$$

Details optional

Substituting for delta

- Now substitute δ by its expected value (for given $\theta=(\pi, c, \sigma)$ for each point)

$$E(\delta_i) = 1 * P(\delta_i = 1 \mid \theta, \mathbf{x}_i) + 0 * P(\delta_i = 0 \mid \theta, \mathbf{x}_i)$$

$$= P(\delta_i = 1 \mid \theta, \mathbf{x}_i)$$

$$P(\mathbf{x}_i \mid \delta_i = 1, \theta,)$$

Details optional

Substituting for delta

$$P(\delta_i = 1 \mid \theta, \mathbf{x}_i) = \frac{P(\mathbf{x}_i \mid \delta_i = 1, \theta) P(\delta_i = 1)}{P(\mathbf{x}_i \mid \delta_i = 1, \theta) P(\delta_i = 1) + P(\mathbf{x}_i \mid \delta_i = 0, \theta) P(\delta_i = 0)}$$

$$= \frac{\exp\left(-1/2\sigma^2 \left[x_i \cos\theta + y_i \sin\theta + c\right]^2\right)\pi}{\exp\left(-1/2\sigma^2 \left[x_i \cos\theta + y_i \sin\theta + c\right]^2\right)\pi + \exp(-k_n)(1-\pi)}$$

Algorithm for line fitting

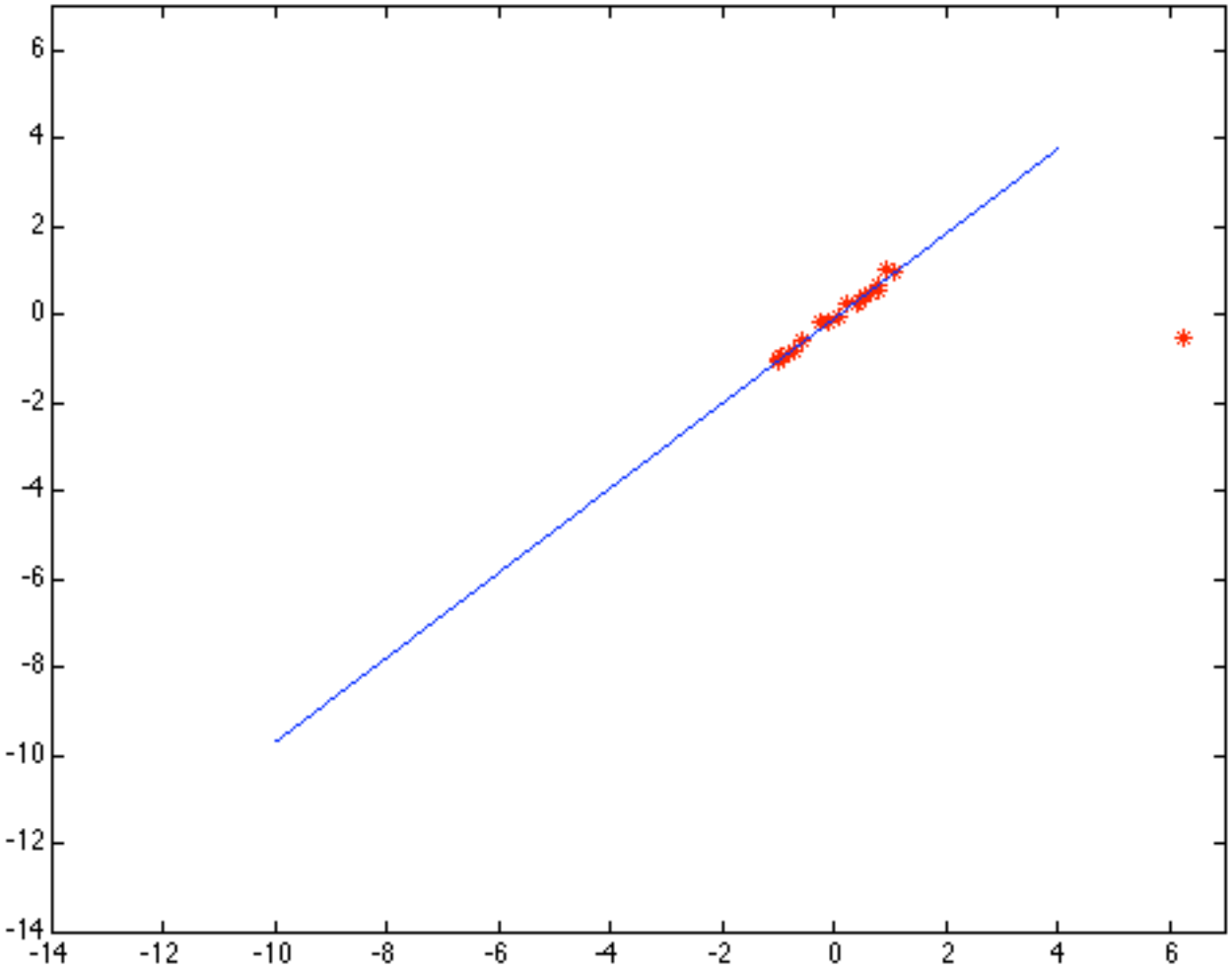
- Obtain some start point

$$\theta^{(0)} = \left(\theta^{(0)}, c^{(0)}, \theta'^{(0)}\right)$$

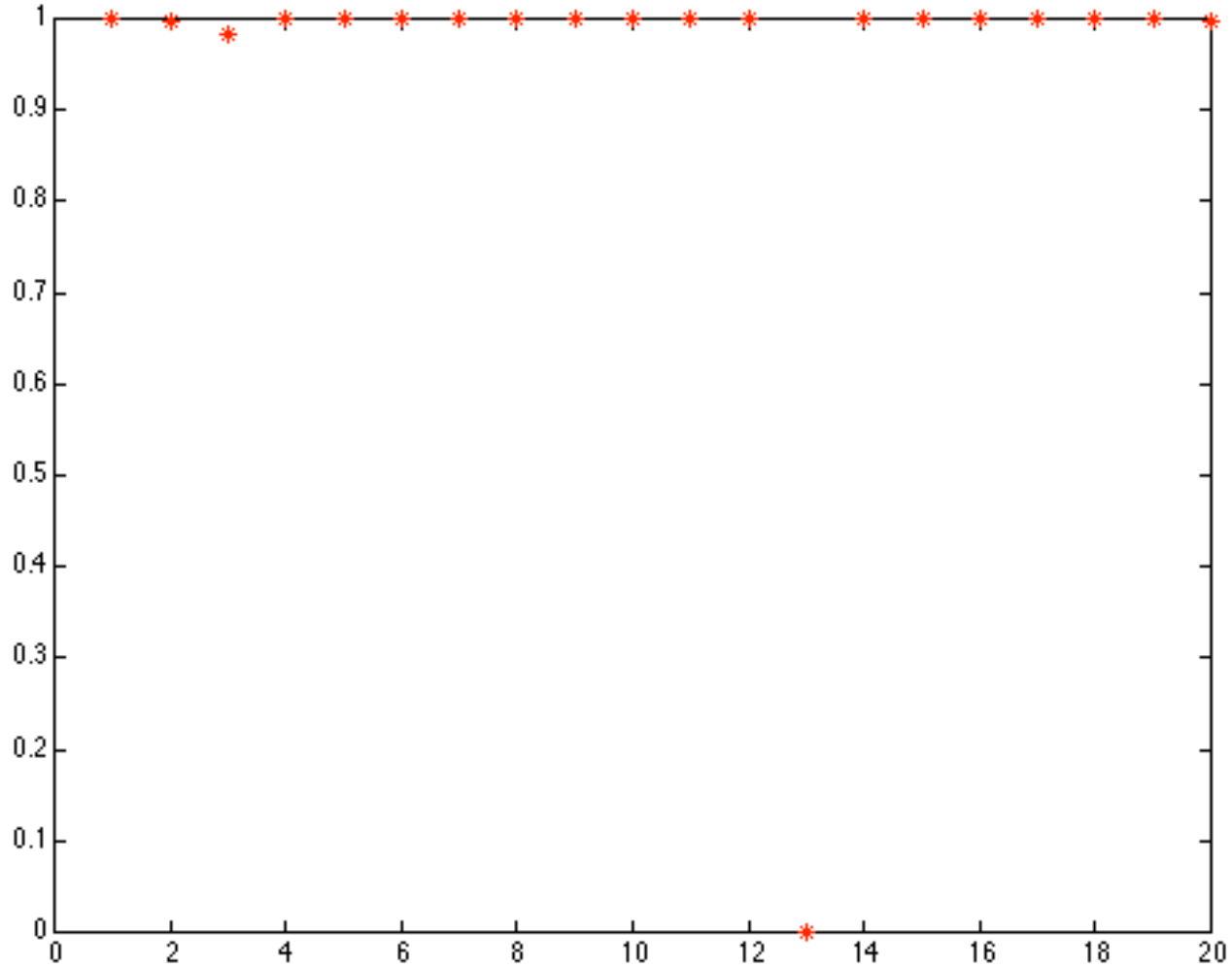
- Now compute ω's using formula above

- Now compute maximum likelihood estimate of $\Pi^{(1)}$
  - θ, c come from fitting to weighted points (ω's are the weights)
  - θ' comes by counting (summing ω's)

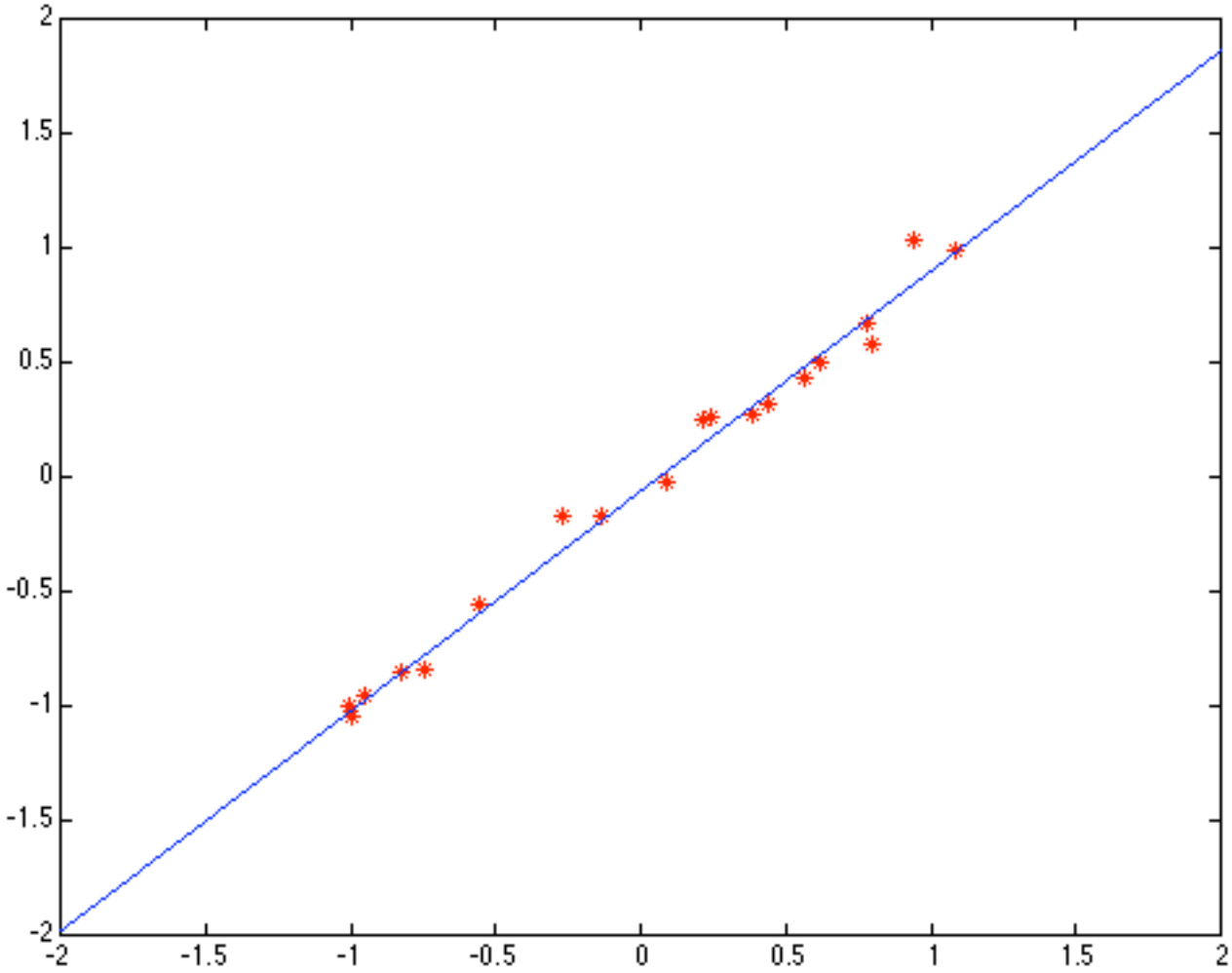
- Iterate to convergence


6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-14
-12
-10
-8
-6
-4
-2
0
2
4
6

The expected values of the deltas at the maximum (notice the one value close to zero).



Closeup of the fit



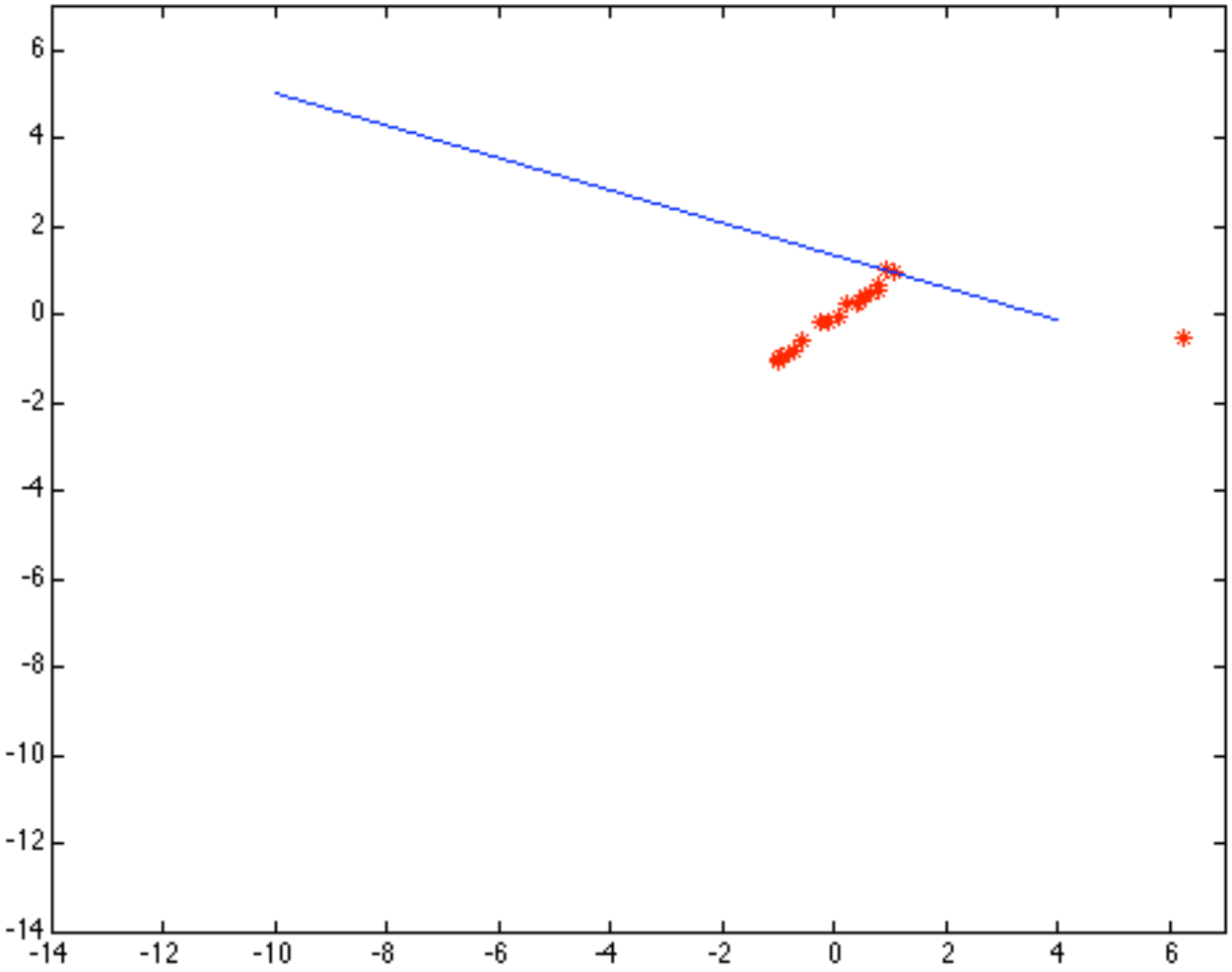
Choosing parameters

- What about the noise parameter, and the sigma for the line?
  - several methods
    - from first principles knowledge of the problem (seldom really possible)
    - play around with a few examples and choose (usually quite effective, as precise choice doesn't matter much)
  - notice that if kn is large, this says that points very seldom come from noise, however far from the line they lie
    - usually biases the fit, by pushing outliers into the line
    - rule of thumb; its better to fit to the better fitting points, within reason; if this is hard to do, then the model could be a problem

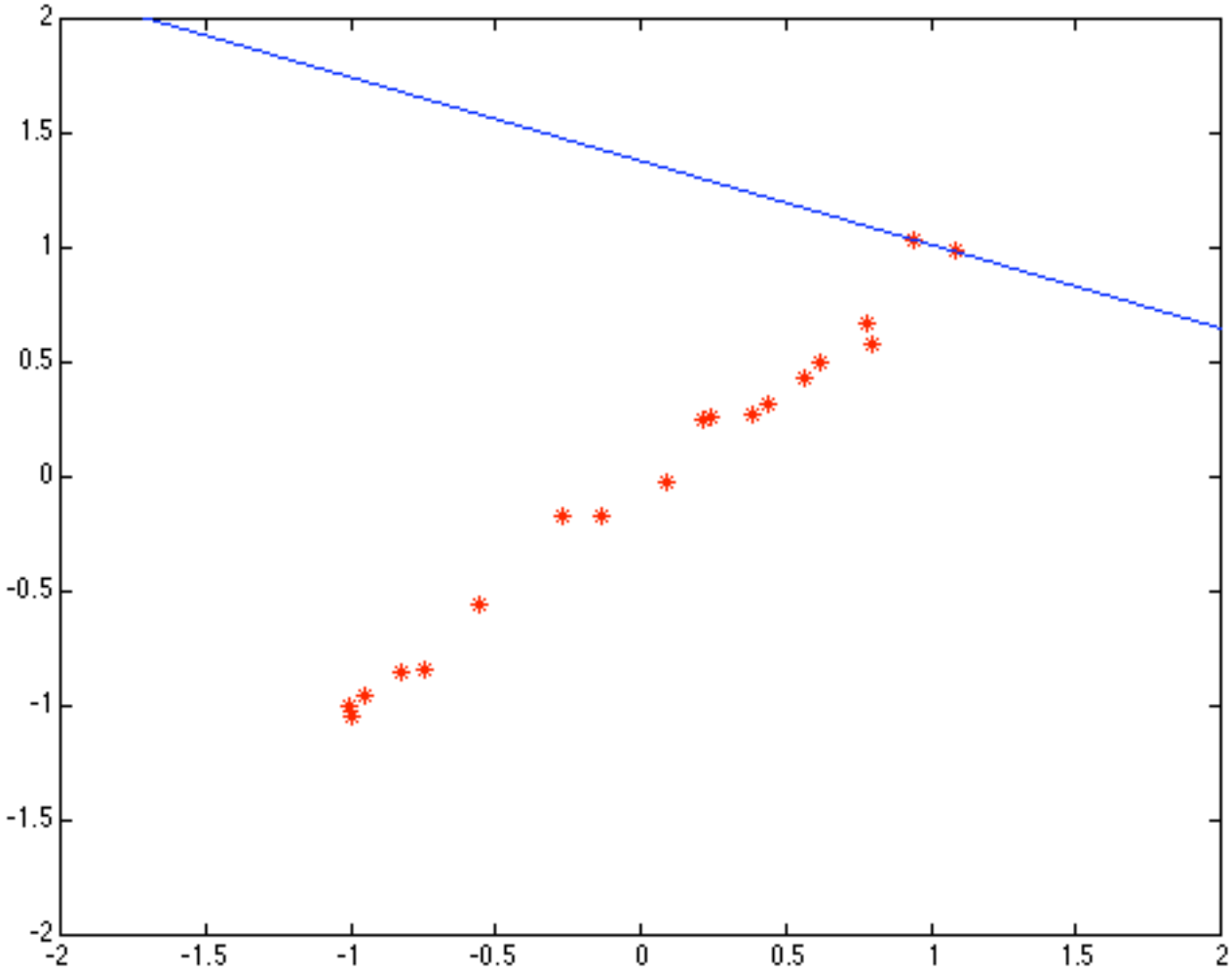
Issues with EM

- Local maxima
  - can be a serious nuisance in some problems
  - no guarantee that we have reached the “right” maximum
- Starting
  - k means to cluster the points is often a good idea

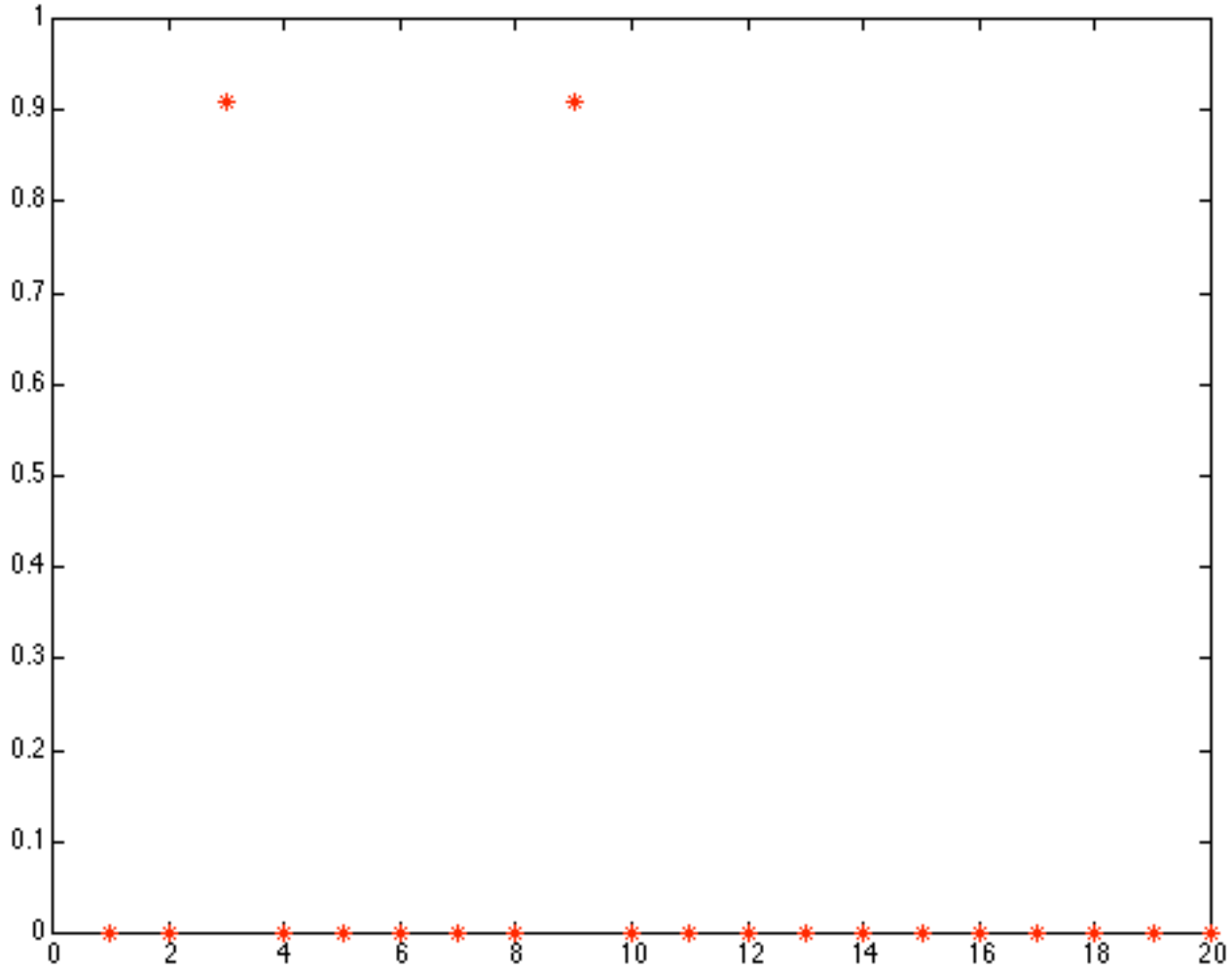
Local maximum



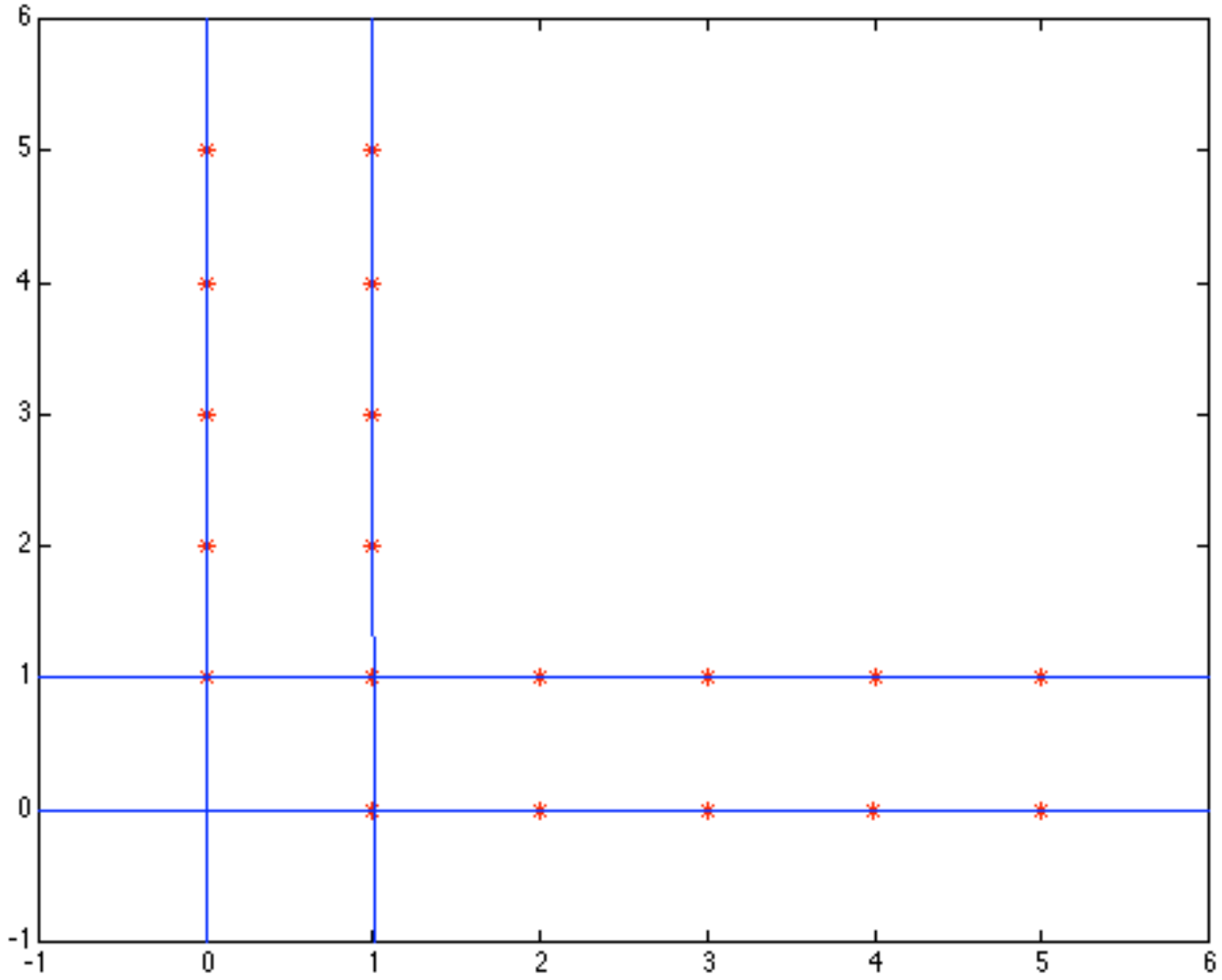
which is an excellent fit to some points



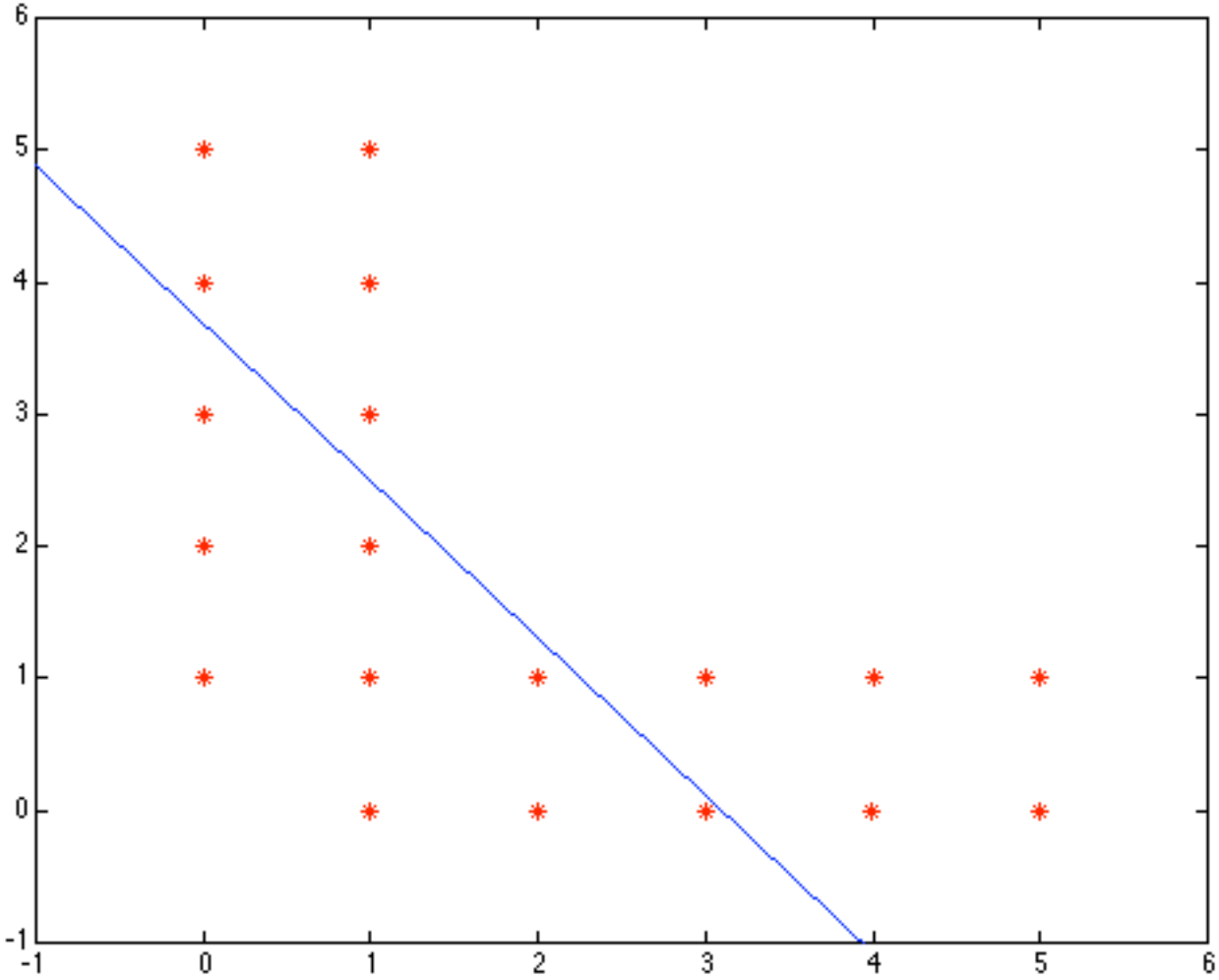
and the deltas for this maximum



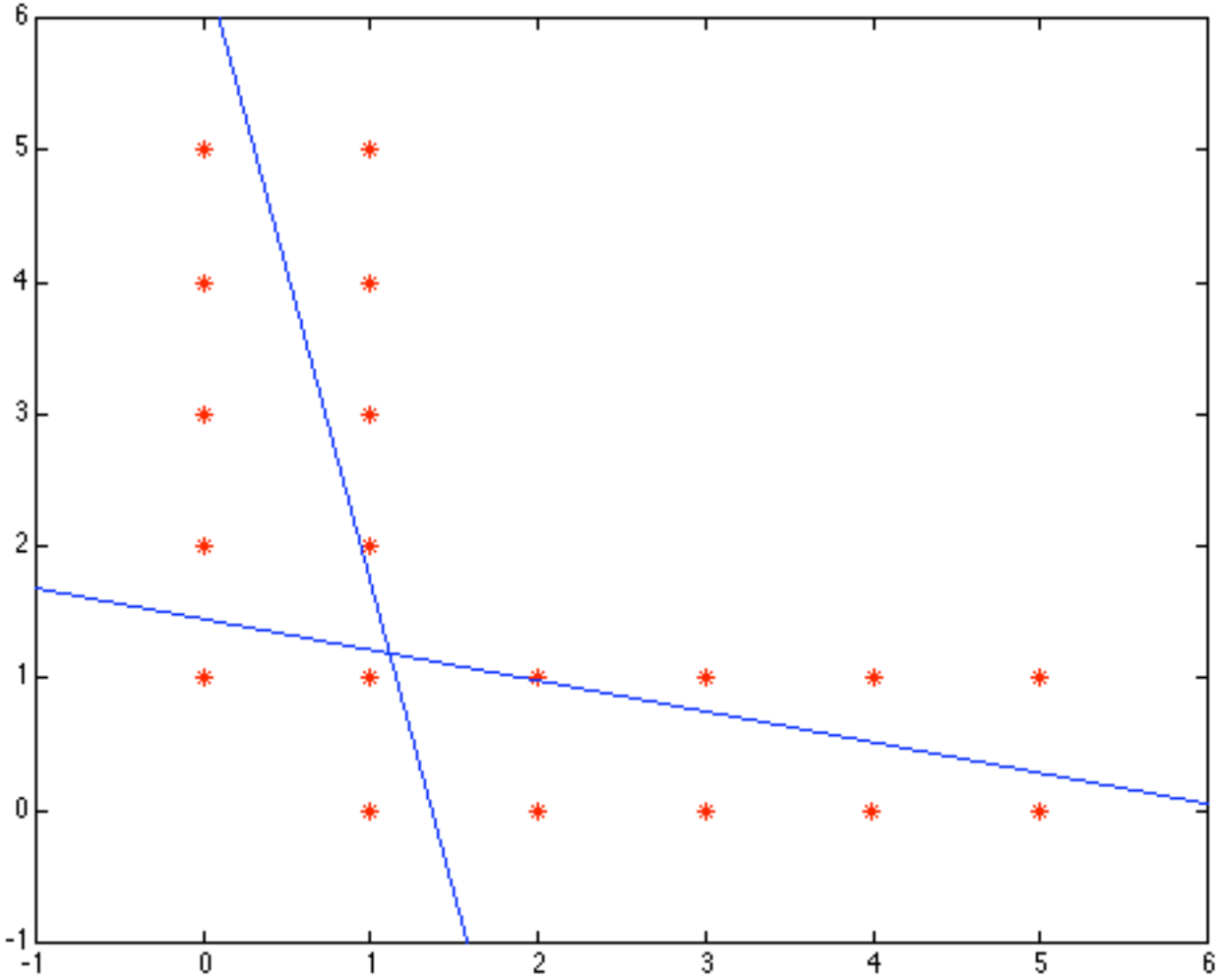
A dataset that is well fitted by four lines



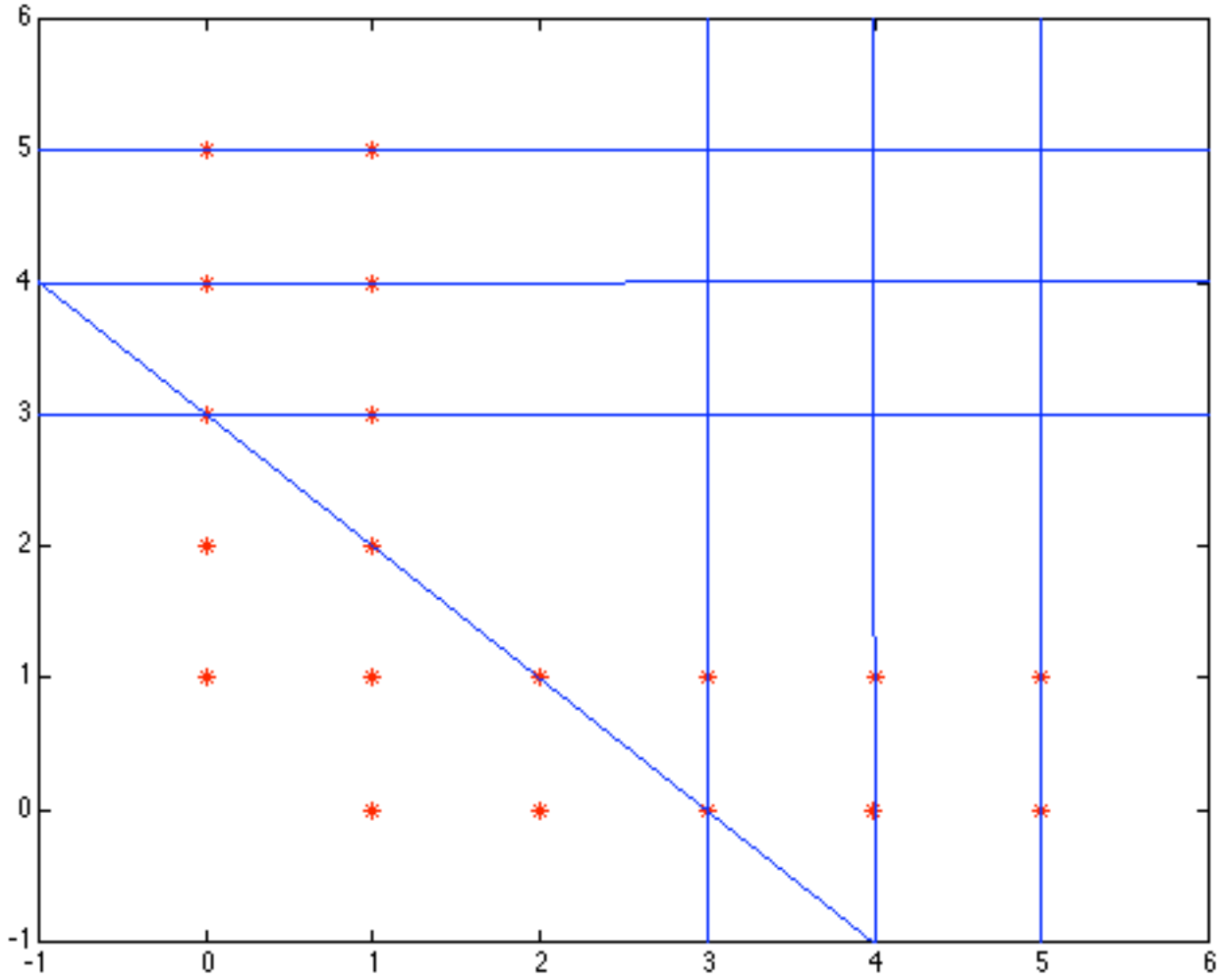
Result of EM fitting, with one line (or at least, one available local maximum).



Result of EM fitting, with two lines (or at least, one available local maximum).



Seven lines can produce a rather logical answer



Fitting multiple lines

- Rather like fitting one line, except there are more hidden variables
- Easiest is to encode as an array of hidden variables, which represent a table with a one where the i'th point comes from the j'th line, zeros otherwise
- Likely want to include a noise source as well
- Rest is similar to single line case

Segmentation/Grouping by EM

- See §16.2.1--note that we assume Gaussian for p().
- A segment is modeled as a Gaussian process that emits feature vectors (which could contain colour; or colour and position; or colour, texture and position).
- Segment parameters are mean and (perhaps) variance or covariance, and prior probability (was λ in the line fitting example).
- If we knew which segment each point belonged to, estimating these parameters would be easy

Segmentation/Grouping by EM

- Instead, we have to use an estimate of the probabilities that a given point belongs to a given segment. We compute means and variances by weighting the standard formulas by these probabilities. For example

$$\mu_m^{(s+1)} = \frac{\sum_{l=1}^{r} \mathbf{x}_l \, p(m \mid \mathbf{x}_l, \theta^{(s)})}{\sum_{l=1}^{r} p(m \mid \mathbf{x}_l, \theta^{(s)})}$$

- Given parameters, the probability that a given point is associated with each cluster is easy to compute. For example

Segmentation/Grouping by EM

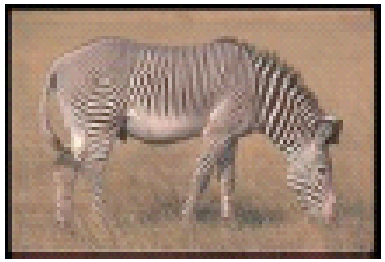
- Given parameters, the probability that a given point is associated with each cluster is easy to compute. For example

$$p(m \mid \mathbf{x}_l, \theta^{(s)}) = \frac{\pi_m^{(s)} p(\mathbf{x}_l \mid \theta_m^{(s)})}{\sum_{k=1}^{M} \pi_k^{(s)} p(\mathbf{x}_l \mid \theta_k^{(s)})}$$

- The book uses $\overline{I_{lm}}$ for $p(m \mid \mathbf{x}_i, \theta^{(s)})$ (I suggests "indicator variable")

- (Also, the books version of the above equation looks wrong to me)

Segmentation with EM



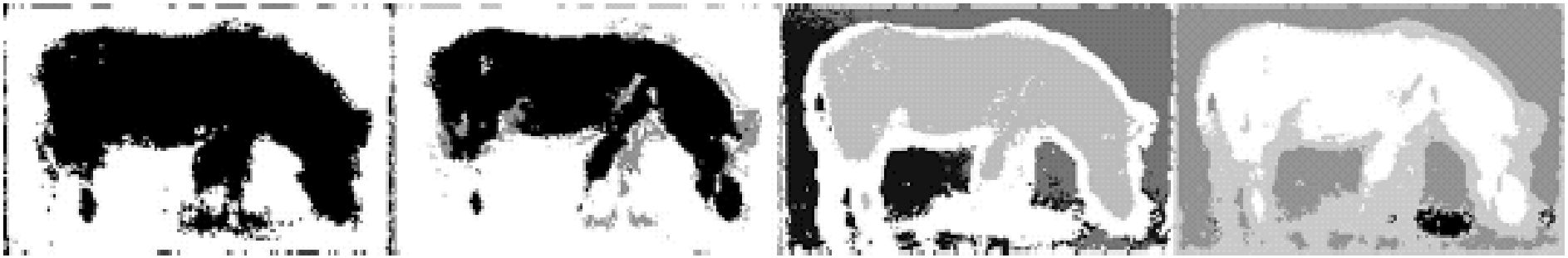


Figure from “Color and Texture Based Image Segmentation Using EM and Its Application to Content Based Image Retrieval”,S.J. Belongie et al., Proc. Int. Conf. Computer Vision, 1998, c1998, IEEE

Skip

Motion segmentation with EM

- Model image pair (or video sequence) as consisting of regions of parametric motion
  - affine motion is popular

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- Now we need to
  - determine which pixels belong to which region
  - estimate parameters

- Likelihood
  - assume

$$I(x, y, t) = I(x + v_x, y + v_y, t + 1) + noise$$

- Straightforward missing variable problem, rest is calculation

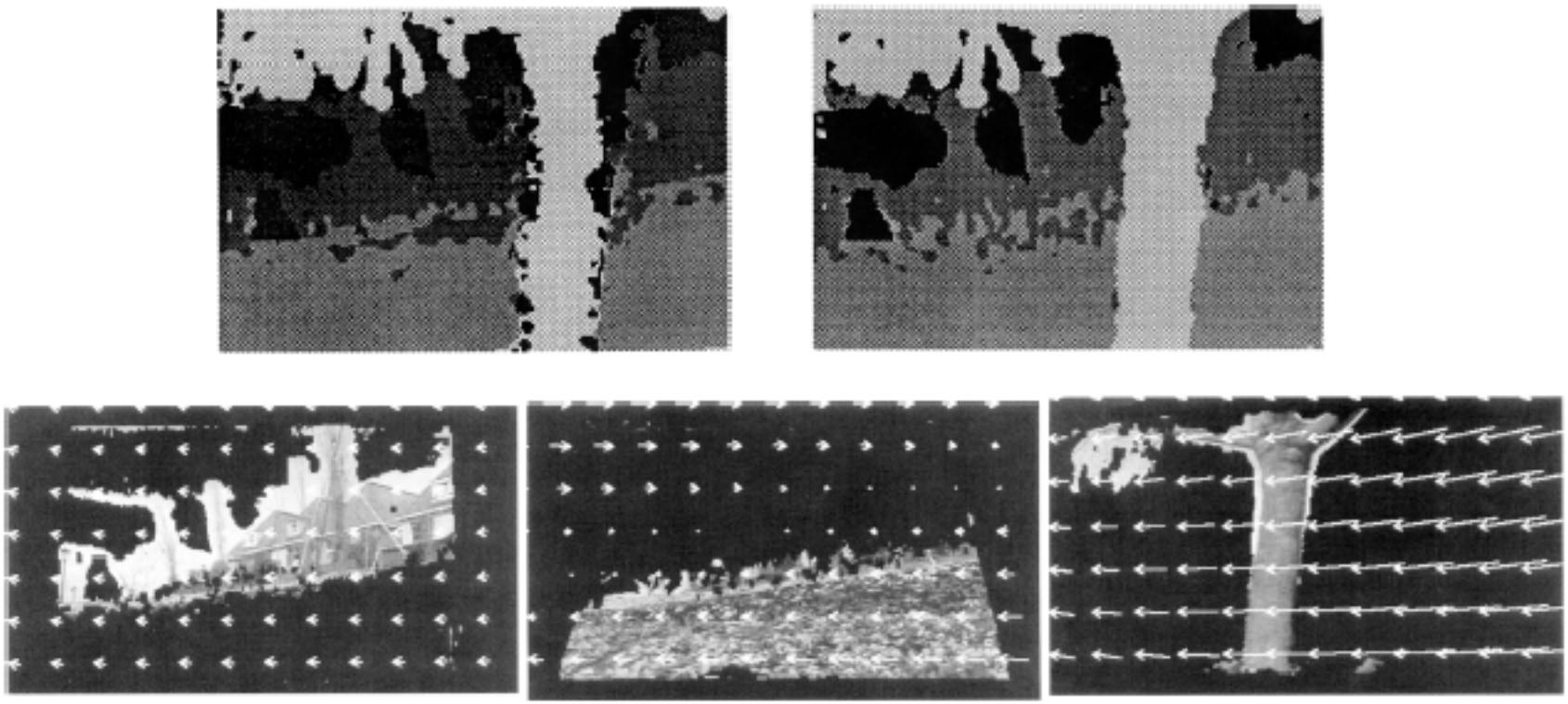
Skip

Three frames from the MPEG “flower garden” sequence

Figure from “Representing Images with layers,”, by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE

Skip

Grey level shows region no. with highest probability



Segments and motion fields associated with them

Figure from “Representing Images with layers,”, by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE

Skip

If we use multiple frames to estimate the appearance of a segment, we can fill in occlusions; so we can re-render the sequence with some segments removed.

Figure from “Representing Images with layers,”, by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE

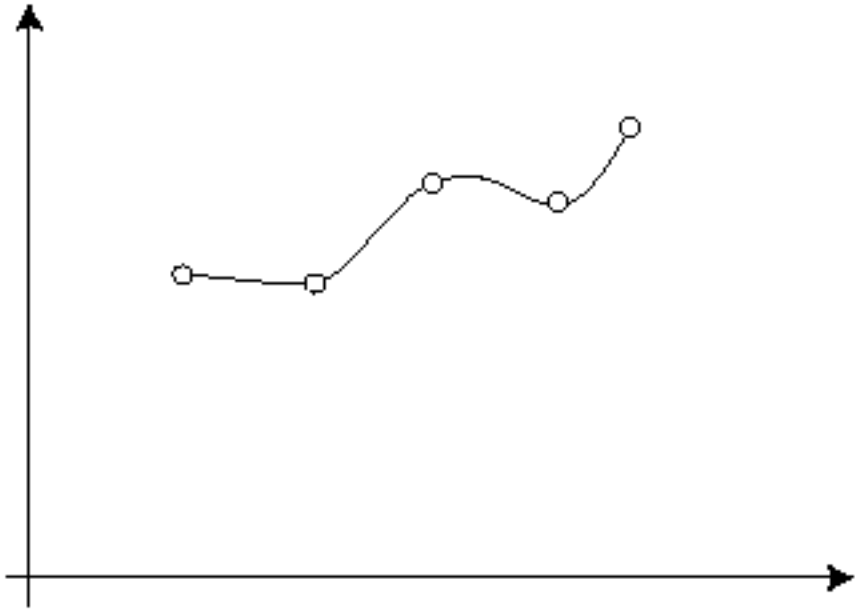
Skip

Some generalities

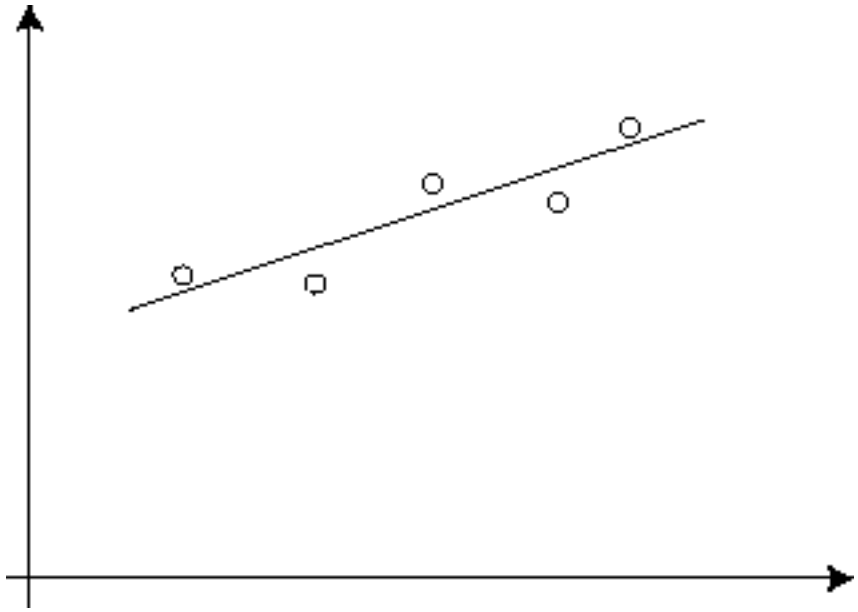
- Many, but not all problems that can be attacked with EM can also be attacked with RANSAC
  - need to be able to get a parameter estimate with a manageably small number of random choices.
  - RANSAC is usually better
- Didn't present in the most general form
  - in the general form, the likelihood may not be a linear function of the missing variables
  - in this case, one takes an expectation of the likelihood, rather than substituting expected values of missing variables
  - Issue doesn't seem to arise in vision applications.

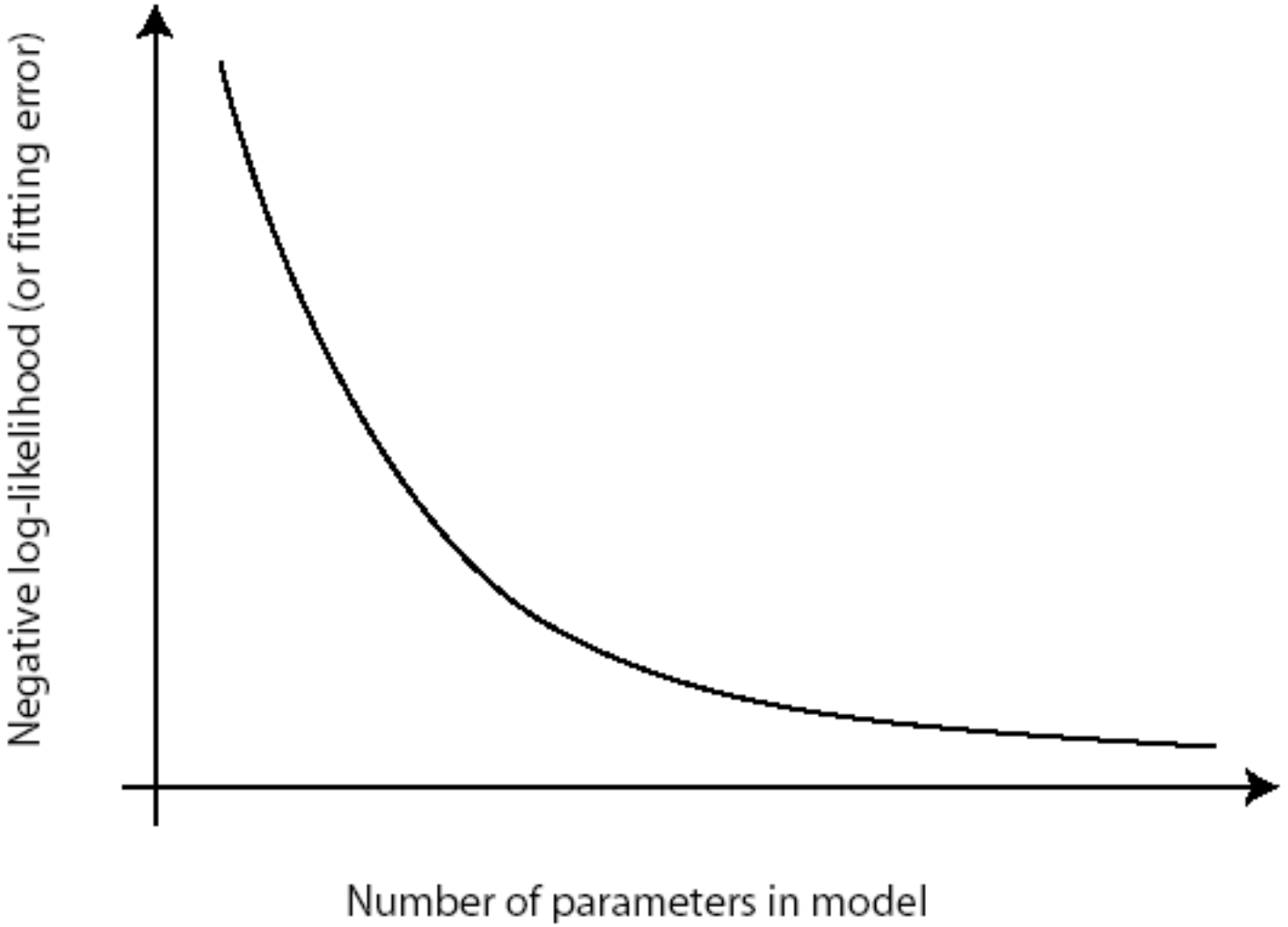
Model Selection

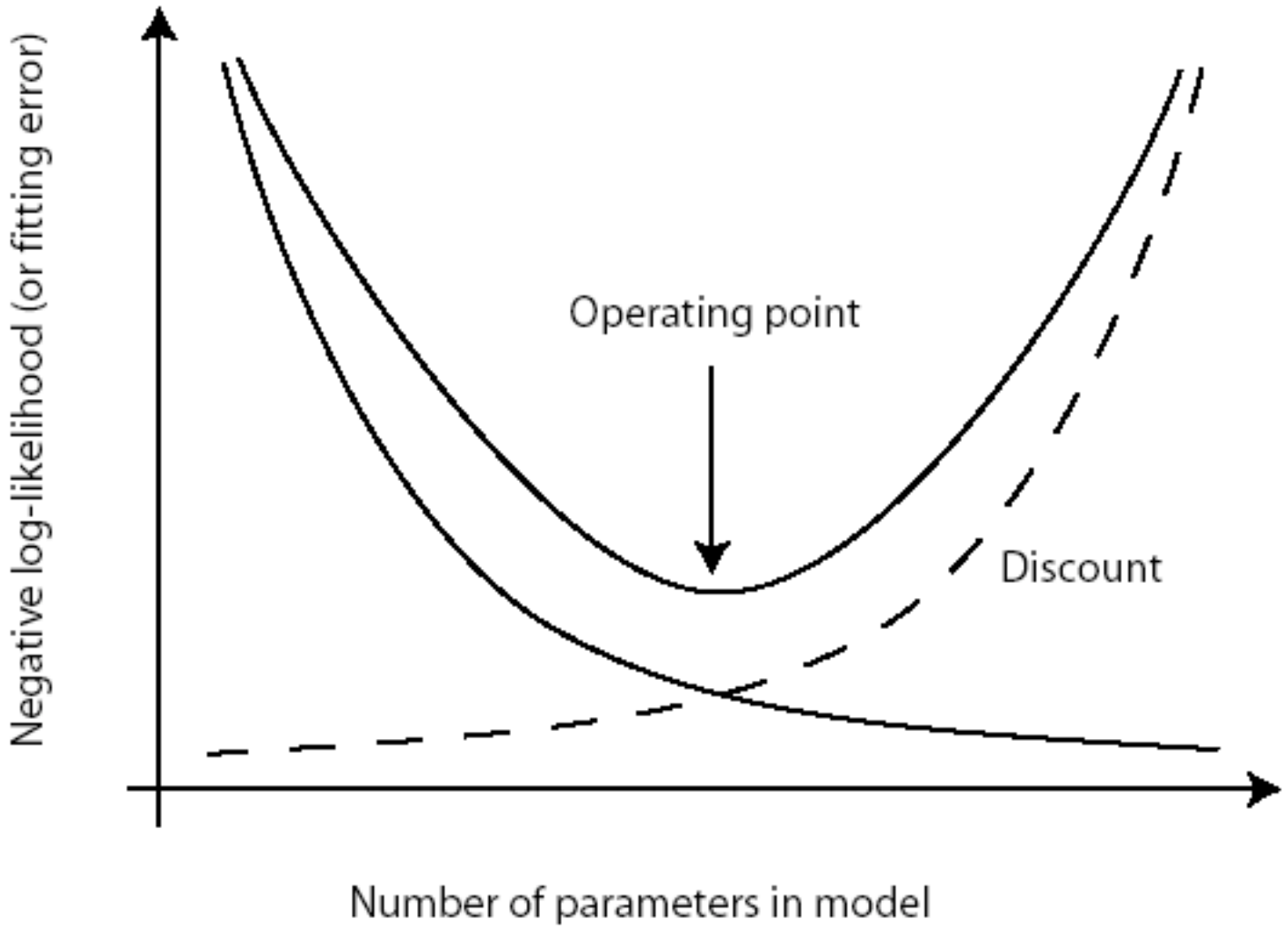
- In general, models with more parameters will fit a dataset better, but are poorer at prediction
- This means we can't simply look at the negative log-likelihood (or fitting error)



Top is not necessarily a better fit than bottom
(actually, almost always worse)




Negative log-likelihood (or fitting error)
Number of parameters in model



We can discount the fitting error with some term in the number of parameters in the model.

Optional

Discounts

- AIC (an information criterion)
  - choose model with smallest value of

$$-2L\left(D;\theta^{*}\right)+2p$$

  - p is the number of parameters
- BIC (Bayes information criterion)
  - choose model with smallest value of

$$-2L\left(D;\theta^{*}\right)+p\log N$$

  - N is the number of data points
- Minimum description length
  - same criterion as BIC, but derived in a completely different way

Cross-validation

- Split data set into two pieces, fit to one, and compute negative log-likelihood on the other
- Set used for fitting is “training data”, other set is “testing data” or “held out data”
- Average over multiple different splits
- Choose the model with the smallest value of this average
- (Optional point) The difference in averages for two different models is an estimate of the difference in KL divergence of the models from the source of the data

Skip

Model averaging

- Very often, it is smarter to use multiple models for prediction than just one
- e.g. motion capture data
  - there are a small number of schemes that are used to put markers on the body
  - given we know the scheme S and the measurements D, we can estimate the configuration of the body X

- We want

$$P(X \mid D) = P(X \mid S_1, D)P(S_1 \mid D) + P(X \mid S_2, D)P(S_2 \mid D) + P(X \mid S_3, D)P(S_3 \mid D)$$

- If it is obvious what the scheme is from the data, then averaging makes little difference
- If it isn't, then not averaging underestimates the variance of X --- we think we have a more precise estimate than we do.