6.801/866

Fitting and Probabilistic Segmentation

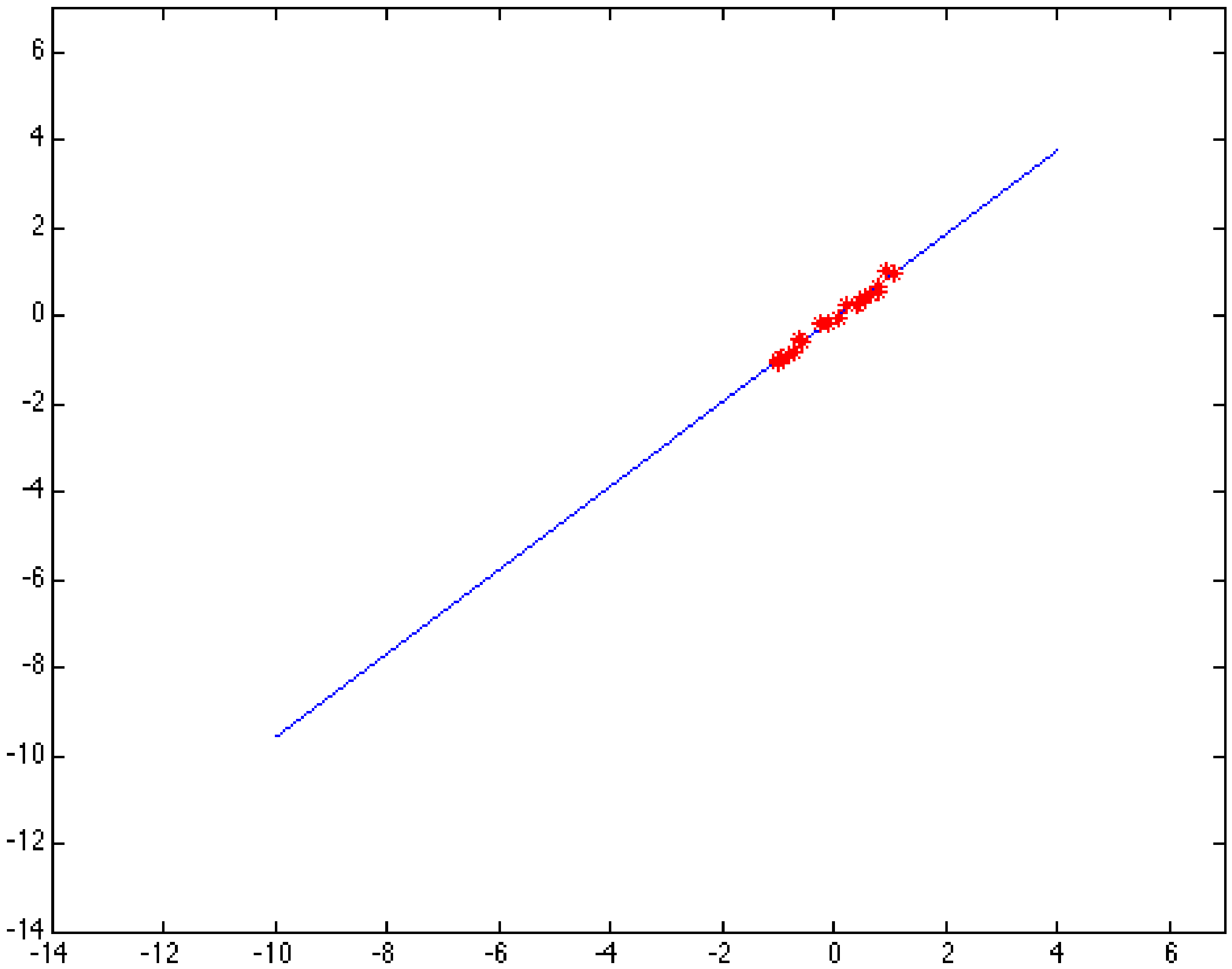
T. Darrell

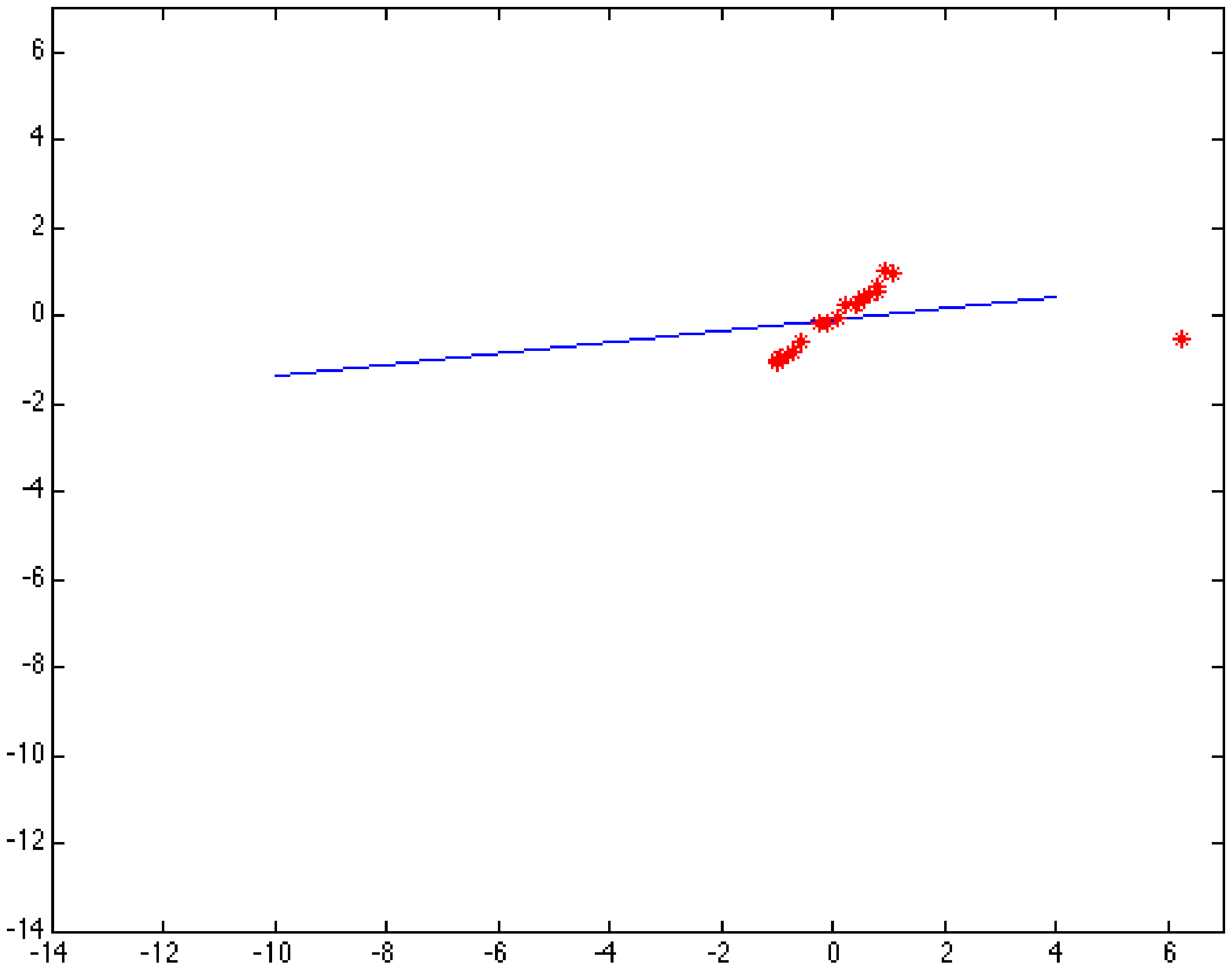
Fitting and Probabilistic Segmentation

- Robust estimation
- RANSAC
- EM
- Model Selection

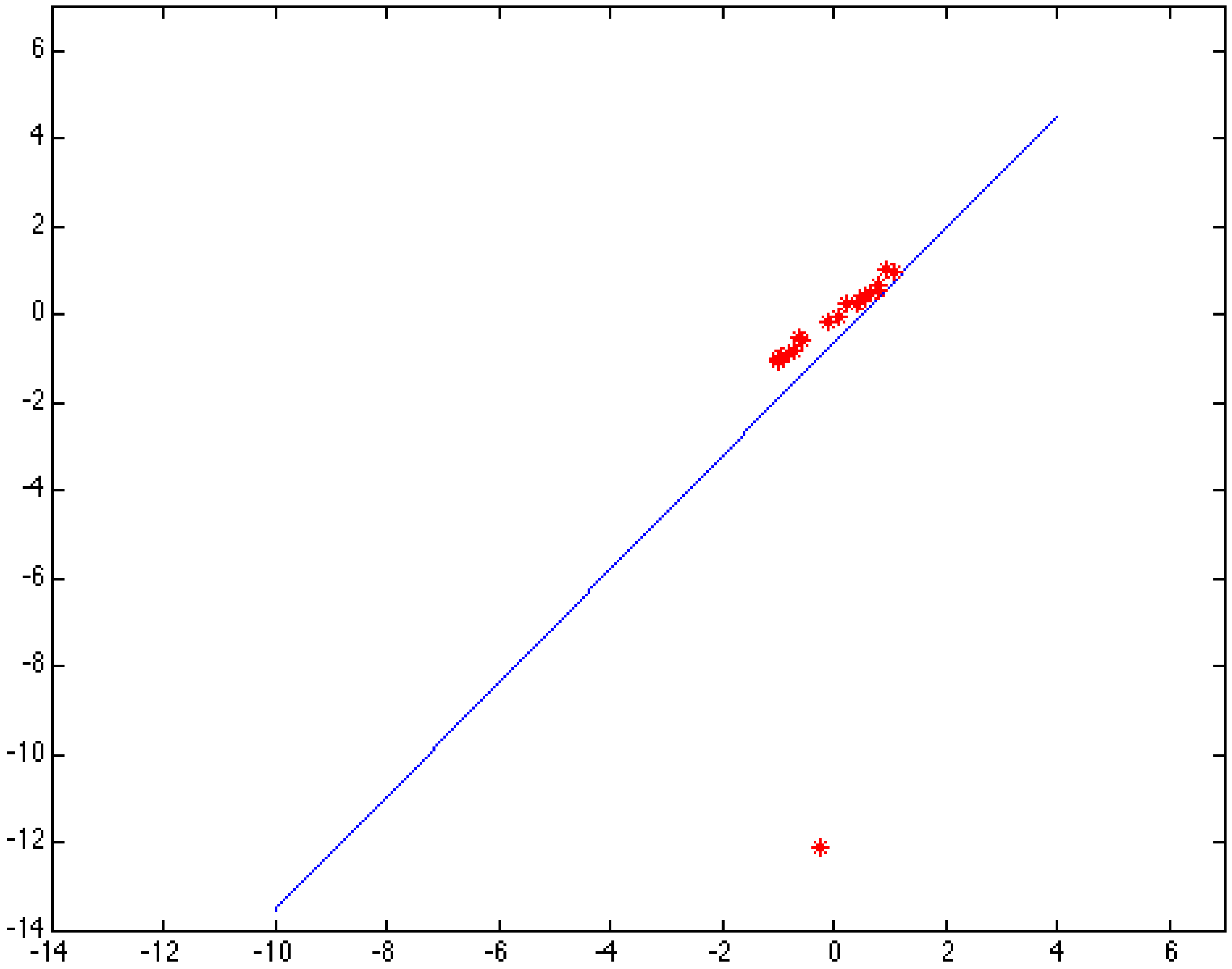
Robustness

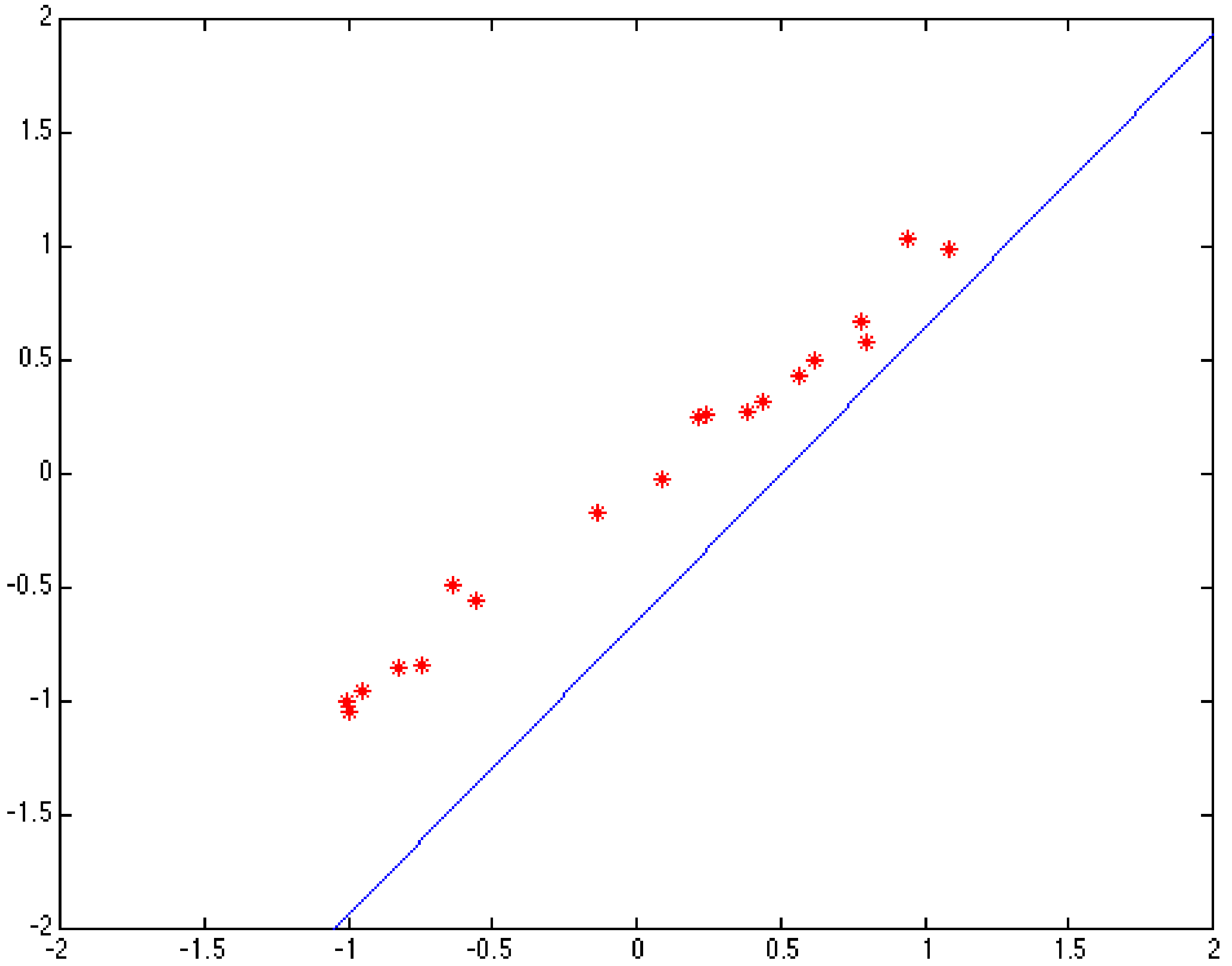
- As we have seen, squared error can be a source of bias in the presence of noise points
  - One fix is EM - we'll do this shortly
  - Another is an M-estimator
    - Square nearby, threshold far away
  - A third is RANSAC
    - Search for good points


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



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


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


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

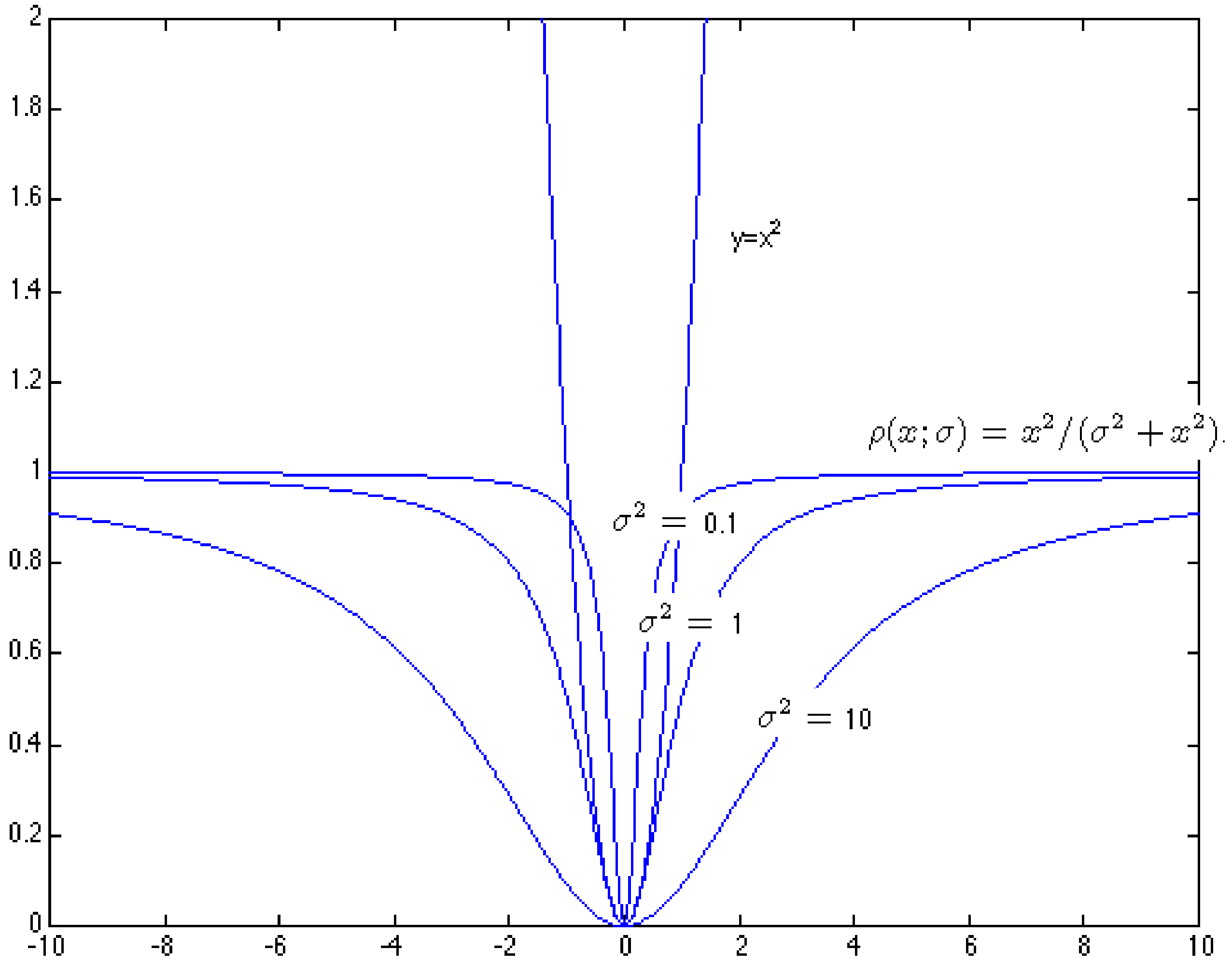
M-Estimators

Generally, minimize

$$\sum_i \rho(r_i(\boldsymbol{x}_i, \theta); \sigma)$$

where

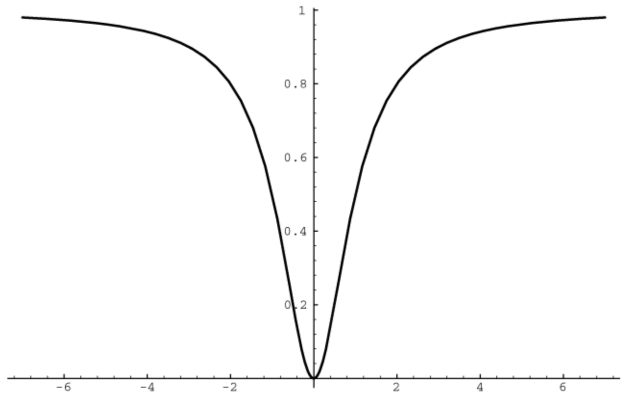
$$r_i(\boldsymbol{x}_i, \theta) \text{ is the residual}$$



y=x²
$\rho(x;\sigma) = x^2/(\sigma^2 + x^2).$
$\sigma^2 = 0.1$
$\sigma^2 = 1$
$\sigma^2 = 10$

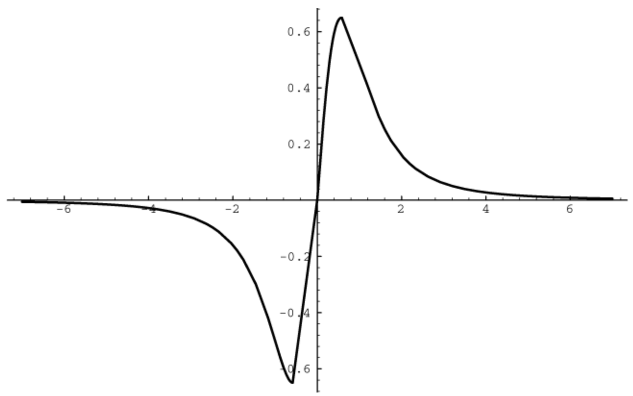
Robust Estimation

A quadratic ρ function gives too much weight to outliers
Instead, use robust norm:

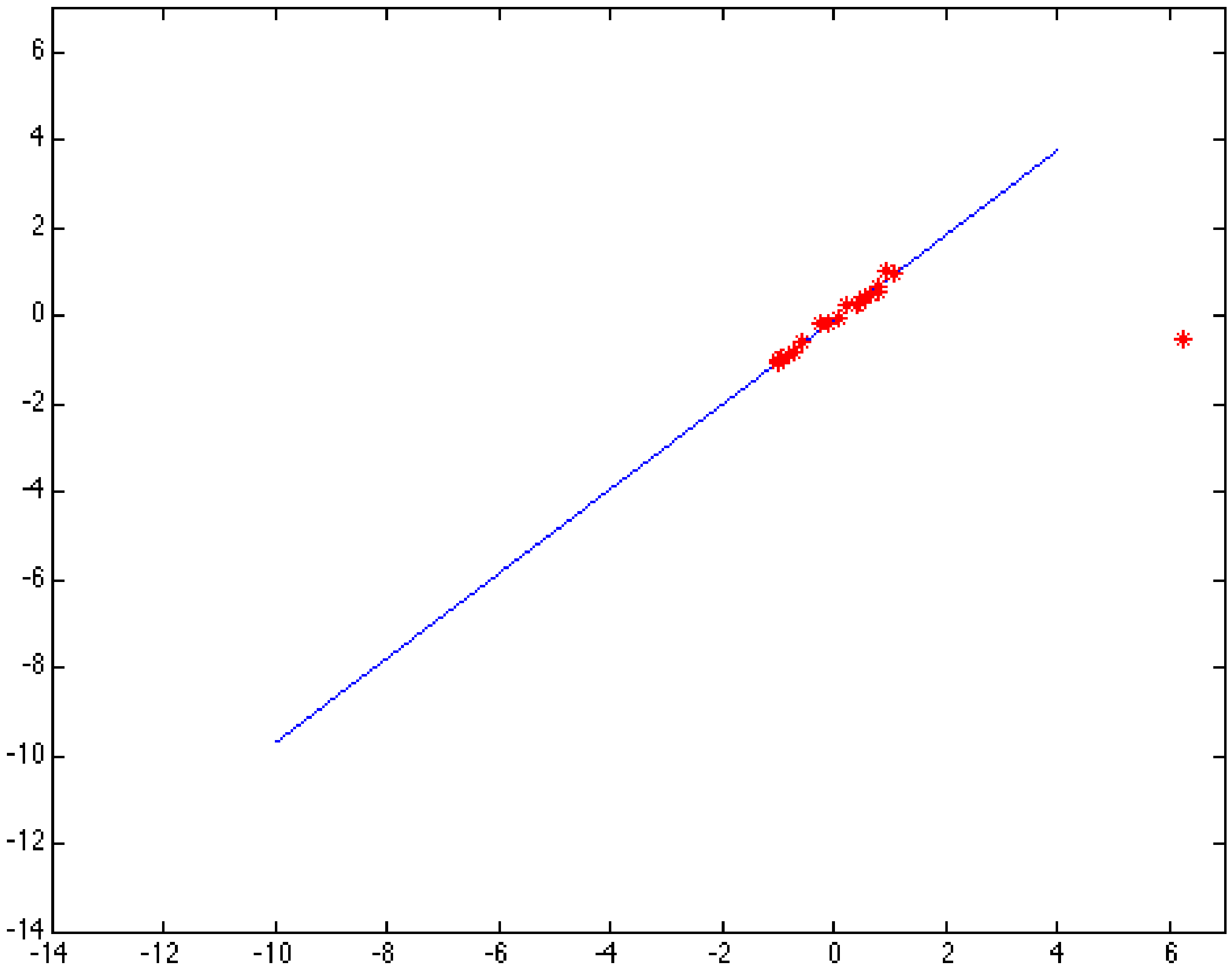


$$\rho(r,\sigma) = \frac{r^2}{\sigma^2 + r^2}$$

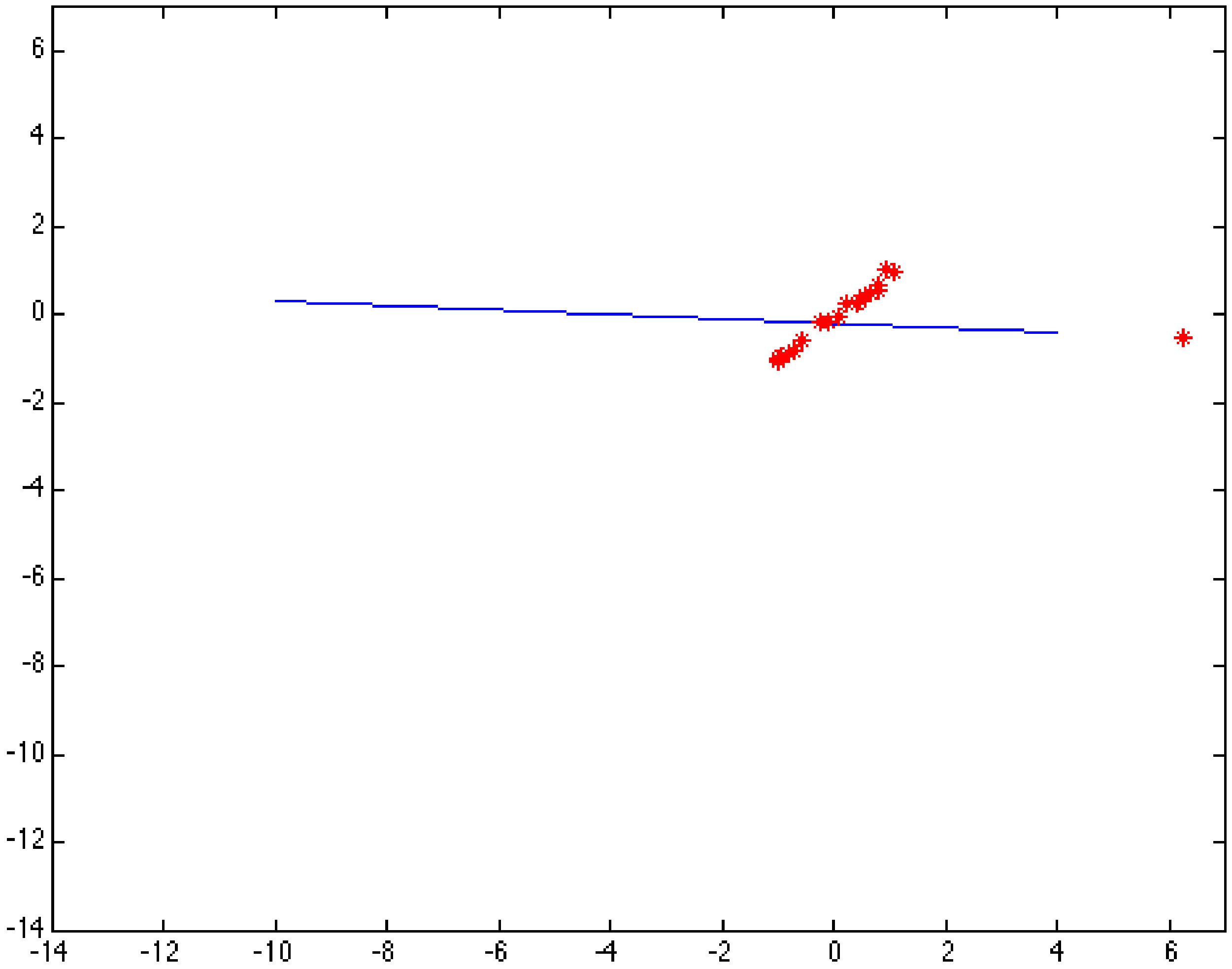
Influence function (d/dr of norm):



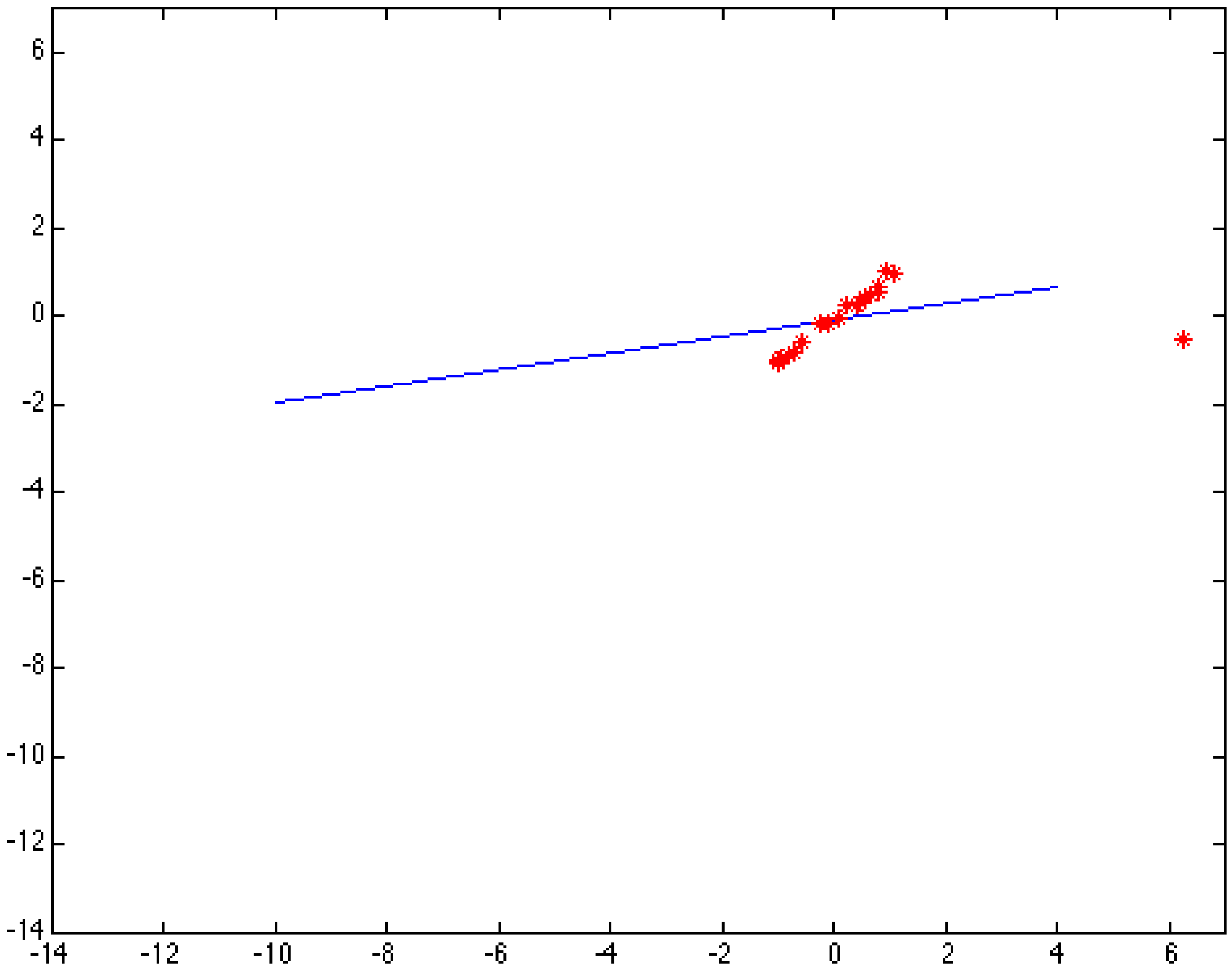
$$\psi(r,\sigma) = \frac{2r\sigma^2}{(\sigma^2 + r^2)^2}$$


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

Too small



Too large



Robust scale

Scale is critical!

Popular choice:

$$\sigma^{(n)} = 1.4826 \text{ median}_i \, |r_i^{(n)}(x_i; \theta^{(n-1)})|$$

Extreme segmentation

What if more than half the points are noise?

RANSAC

- Iterate:
  - Sample
  - Fit
  - Test
- Keep best estimate; refit on inliers

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
```

RANSAC applications

- Fundamental Matricies
  - estimate F from 7 points
  - test agreement with all other points
- Direct motion
  - estimate affine (or rigid motion) from small match
  - see what other parts of image are consistent
- …

General framework

Estimate parameters from segmented data.

Consider segmentation labels to be *missing data*.

Missing variable problems

A missing data problem is a statistical problem where some data is missing

There are two natural contexts in which missing data are important:

- terms in a data vector are missing for some instances and present for other (perhaps someone responding to a survey was embarrassed by a question)
- an inference problem can be made very much simpler by rewriting it using some variables whose values are unknown.

Missing variable problems

A missing data problem is a statistical problem where some data is missing

There are two natural contexts in which missing data are important:

- terms in a data vector are missing for some instances and present for other (perhaps someone responding to a survey was embarrassed by a question)
- an inference problem can be made very much simpler by rewriting it using some variables whose values are unknown.

Missing variable problems

General case:

- Complete space X (e.g., pixel values and labels)
- Incomplete space Y (e.g., pixel values)
- f: X->Y
- Parameters U (e.g., mixing weights cluster mean, covar.)

Complete log-likelihood:

$$
\begin{aligned}
L_c(\boldsymbol{x}; \boldsymbol{u}) &= \log\{\prod_j p_c(\boldsymbol{x}_j; \boldsymbol{u})\} \\
&= \sum_j \log\left(p_c(\boldsymbol{x}_j; \boldsymbol{u})\right)
\end{aligned}
$$

Missing variable problems

- Incomplete density

$$p_i(\boldsymbol{y};\boldsymbol{u}) = \int_{\{\boldsymbol{x}|f(\boldsymbol{x})=\boldsymbol{y}\}} p_c(\boldsymbol{x};\boldsymbol{u})d\eta$$

yields log-likelihood

$$\begin{aligned} L_i(\boldsymbol{y};\boldsymbol{u}) &= \log\left\{\prod_j p_i(\boldsymbol{y}_j;\boldsymbol{u})\right\} \\ &= \sum_j \log\left(p_i(\boldsymbol{y}_j;\boldsymbol{u})\right) \\ &= \sum_j \log\left(\int_{\{\boldsymbol{x}|f(\boldsymbol{x})=\boldsymbol{y}_j\}} p_c(\boldsymbol{x};\boldsymbol{u})d\eta\right) \end{aligned}$$

Hard to deal with: we don’t know which of the many possible x’s that could correspond to the y’s that we observe actually does correspond.

Strategy

For each of our examples, if we knew the missing data we could estimate the parameters effectively.

If we knew the parameters, the missing data would follow. This suggests an iterative algorithm:

1. obtain some estimate of the missing data, using a guess at the parameters;
2. now form a maximum likelihood estimate of the free parameters using the estimate of the missing data.

and we iterate this procedure until (hopefully!) it converges.

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
- fundamental matrix estimation; if we knew which feature corresponded to which, it would be easy to determine the fundamental matrix
- etc.

EM for Mixture models

If log-likelihood is linear in missing variables we can replace missing variables with expectations. E.g.,

$$p(\boldsymbol{y}) = \sum_l \pi_l p(\boldsymbol{y}|\boldsymbol{a}_l)$$

mixture model

$$\sum_{j \in \text{observations}} \left(\sum_{l=1}^{g} z_{lj} \log p(\boldsymbol{y}_j|\boldsymbol{a}_l) \right)$$

complete data log-likelihood

1. (E-step) estimate complete data (e.g, z_j's) using previous parameters
2. (M-step) maximize complete log-likelihood using estimated complete data

$$\begin{aligned} \boldsymbol{u}^{s+1} &= \arg\max_{\boldsymbol{u}} L_c(\overline{\boldsymbol{x}}^s; \boldsymbol{u}) \\ &= \arg\max_{\boldsymbol{u}} L_c([\boldsymbol{y}, \overline{\boldsymbol{z}}^s]; \boldsymbol{u}) \end{aligned}$$

EM in general case

Cant substitute expectations for missing variables.

Take expectation of the complete data log-likelihood with respect to the missing variables conditioned on the current value of the parameter:

$$Q(\boldsymbol{u};\boldsymbol{u}^{(s)}) = \int L_c(\boldsymbol{x};\boldsymbol{u})p(\boldsymbol{x}|\boldsymbol{u}^{(s)},\boldsymbol{y})d\boldsymbol{x}$$

Then maximize w.r.t. parameters

$$\boldsymbol{u}^{(s+1)} = \arg\max_{\boldsymbol{u}} Q(\boldsymbol{u};\boldsymbol{u}^{(s)})$$

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)

$$
\begin{aligned}
P(\text{point} \mid \text{line and noise params}) &= P(\text{point} \mid \text{line})P(\text{comes from line}) + \\
&\quad P(\text{point} \mid \text{noise})P(\text{comes from noise}) \\
&= P(\text{point} \mid \text{line})\lambda + P(\text{point} \mid \text{noise})(1-\lambda)
\end{aligned}
$$

EM for line estimation

- We have a problem with parameters, missing variables

Iterate until convergence:

  - replace missing variable with expected values, given **fixed** values of parameters
  - fix missing variables, choose parameters to maximise likelihood given **fixed** values of missing variables

- e.g.,
  - 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
- Somewhat more general form is available

Estimating the mixture model

- Introduce a set of hidden variables, δ, 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 (the line's parameters are ϕ, c):
- Here K is a normalising constant, k_n is the noise intensity (we'll choose this later).

$$Q_c(x;\theta) = \sum_i \begin{pmatrix} \delta_i \left(\dfrac{(x_i \cos\phi + y_i \sin\phi + c)^2}{2\sigma^2} \right) + \\ (1-\delta_i) k_n \end{pmatrix} + K$$

Substituting for delta

- We shall substitute the expected value of δ, for a given θ
- recall θ=(φ, c, λ)
- $E(\delta_i)=1*P(\delta_i=1|\theta)+0....$

- Notice that if k_n is small and positive, then if distance is small, this value is close to 1 and if it is large, close to zero

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

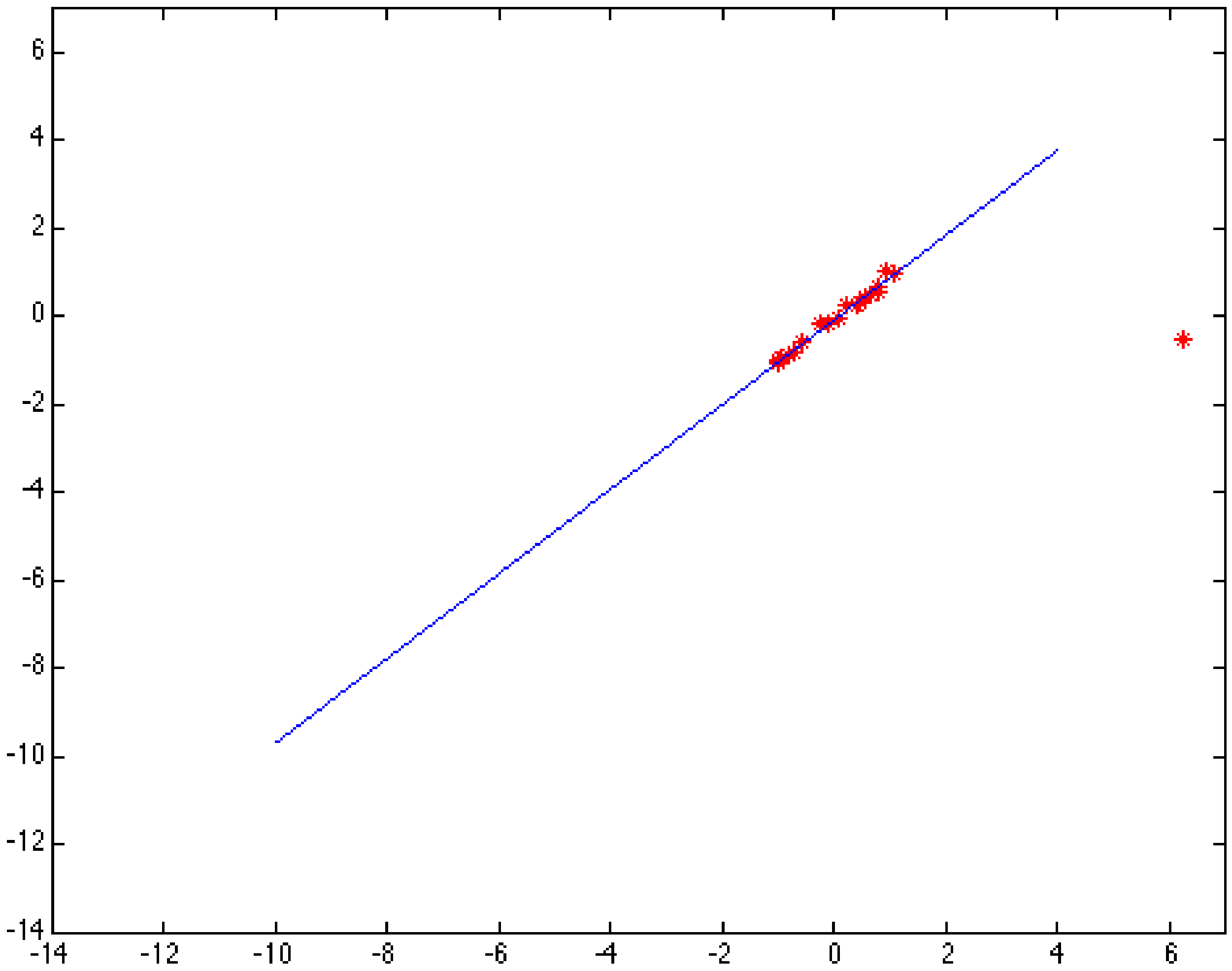
$$= \frac{\exp\left(-1/2\sigma^2 \left[x_i \cos\phi + y_i \sin\varphi + c\right]^2\right)\lambda}{\exp\left(-1/2\sigma^2 \left[x_i \cos\phi + y_i \sin\varphi + c\right]^2\right)\lambda + \exp(-k_n)(1-\lambda)}$$

Algorithm for line fitting

- Obtain some start point

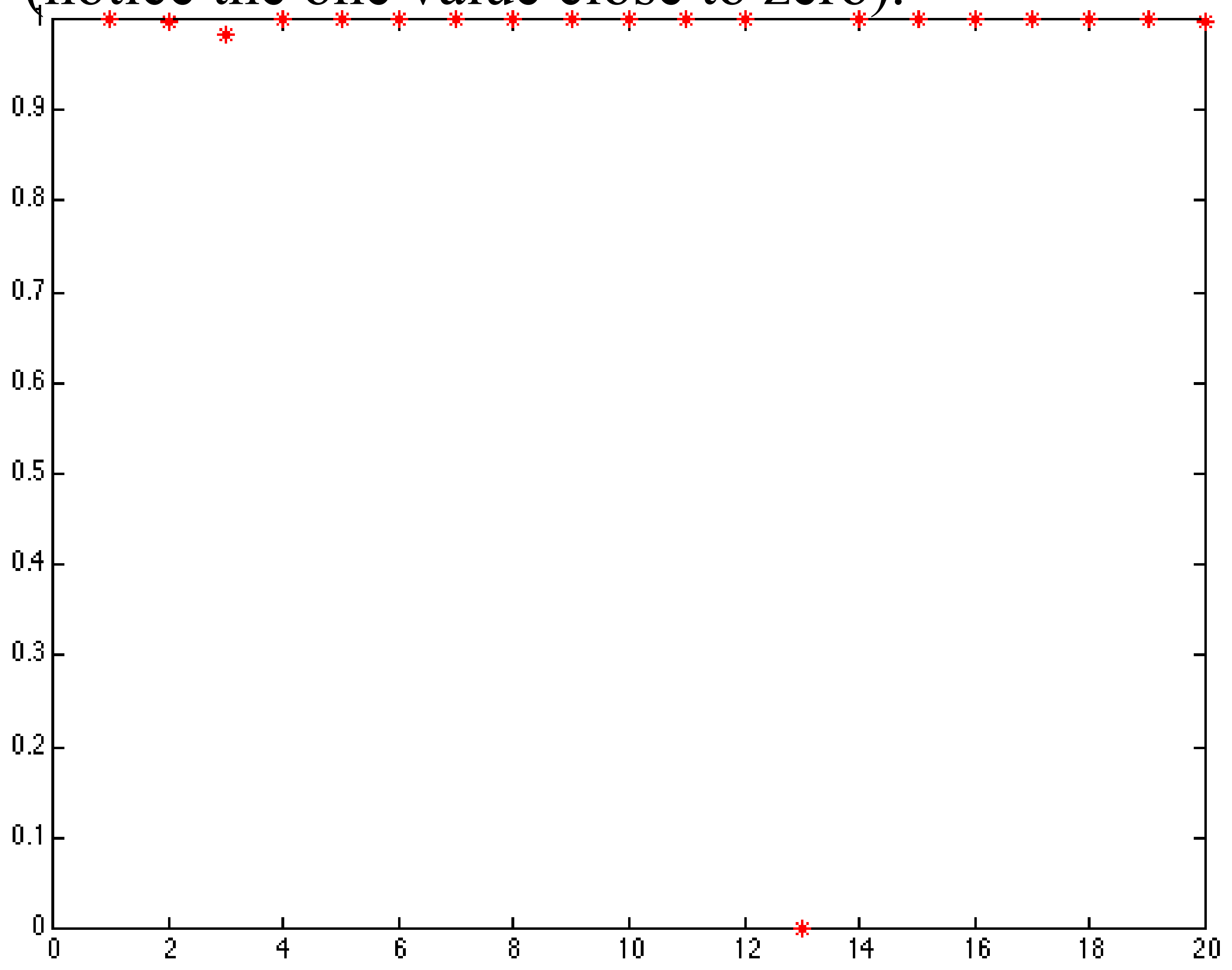
$$\theta^{(0)} = \left(\phi^{(0)}, c^{(0)}, \lambda^{(0)}\right)$$

- Now compute δ's using formula above
- Now compute maximum likelihood estimate of $\theta^{(1)}$
  - ϕ, c come from fitting to weighted points
  - λ comes by counting
- Iterate to convergence

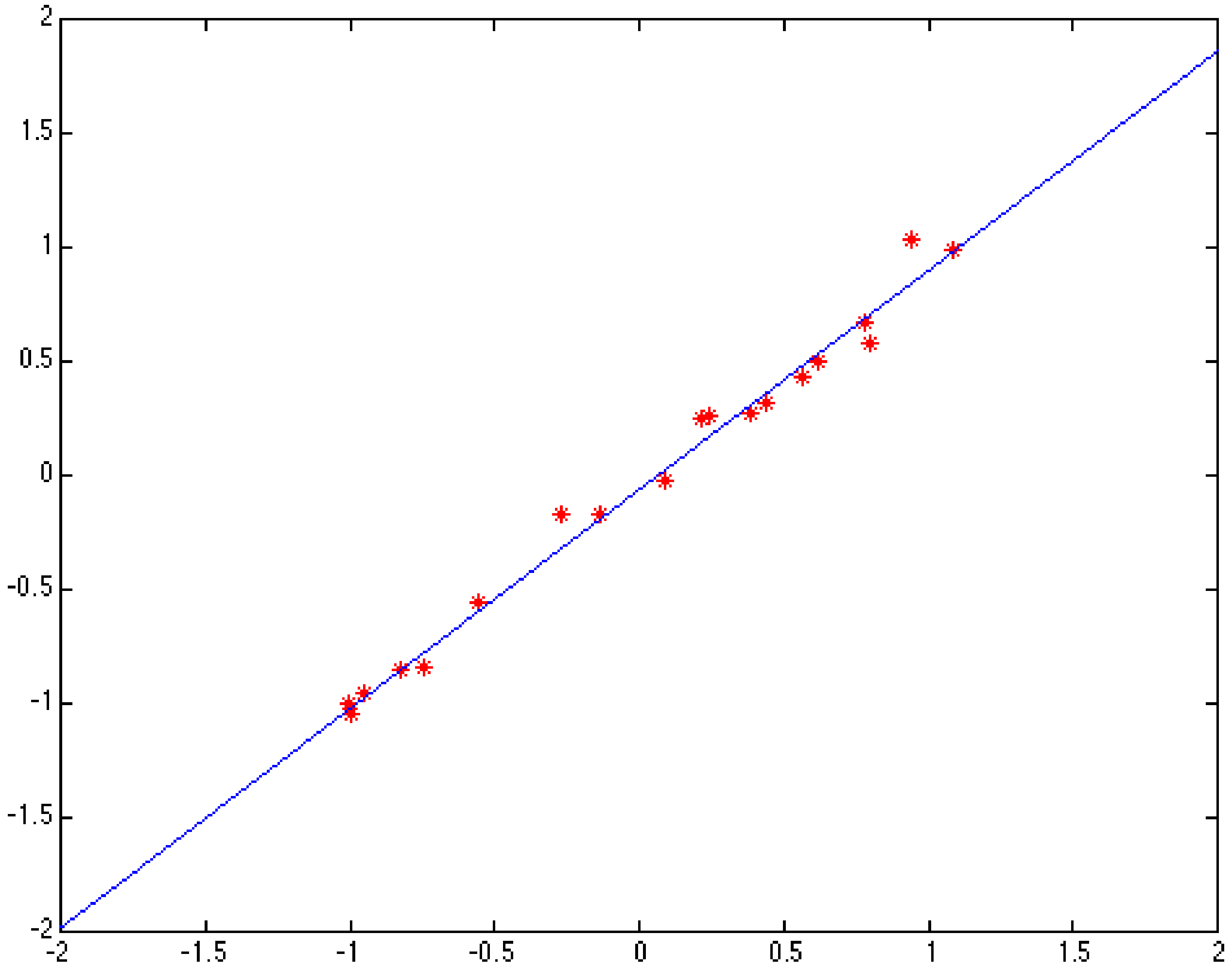


6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-14
-12
-10
-8
-6
-4
-2
0
2
4
6

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 k_n 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

Other examples

- Segmentation
  - a segment is a gaussian that emits feature vectors (which could contain colour; or colour and position; or colour, texture and position).
  - segment parameters are mean and (perhaps) covariance
  - if we knew which segment each point belonged to, estimating these parameters would be easy
  - rest is on same lines as fitting line

- 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
  - rest is on same lines as above

Color segmentation with EM

- At each pixel in an image, we compute a *d*-dimensional feature vector x, which encapsulates position, colour and texture information.
- Pixel is generated by one of G segments, each Gaussian, chosen with probability π:

$$p(\boldsymbol{x}) = \sum_{i} p(\boldsymbol{x}|\theta_l)\pi_l$$

Color segmentation with EM

Parameters include mixing weights and means/covars:

$$\Theta = (\alpha_1, \ldots, \alpha_g, \theta_1, \ldots, \theta_g). \qquad \theta_l = (\boldsymbol{\mu}_l, \Sigma_l)$$

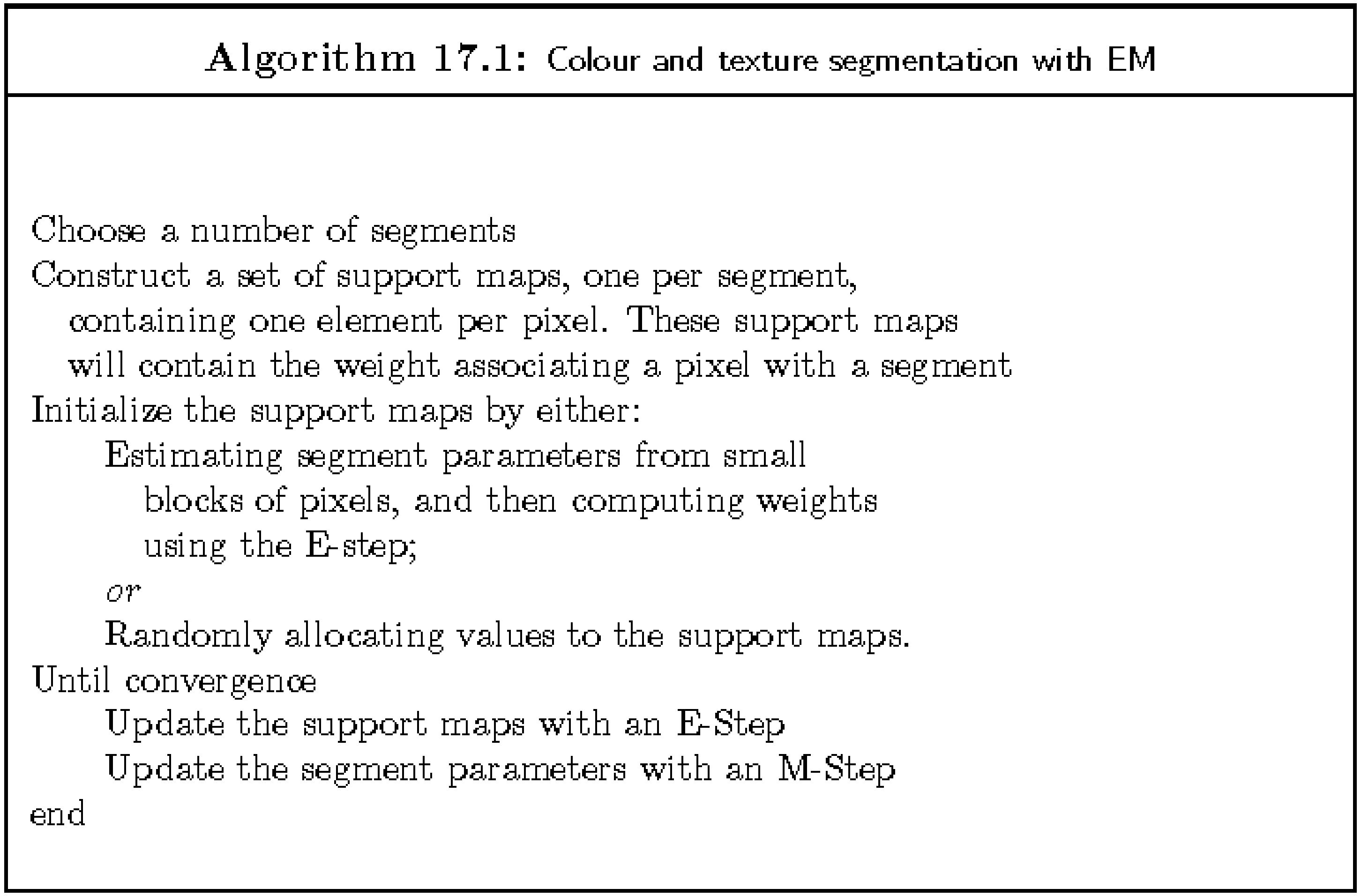
yielding

$$p(\boldsymbol{x}|\Theta) = \sum_{l=1}^{g} \alpha_l p_l(\boldsymbol{x}|\theta_l)$$

with

$$p_l(\boldsymbol{x}|\theta_l) = \frac{1}{(2\pi)^{d/2}\det(\Sigma_i)^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right\}$$

Color segmentation with EM



Algorithm 17.1: Colour and texture segmentation with EM

```
Choose a number of segments
Construct a set of support maps, one per segment,
  containing one element per pixel. These support maps
  will contain the weight associating a pixel with a segment
Initialize the support maps by either:
    Estimating segment parameters from small
      blocks of pixels, and then computing weights
      using the E-step;
    or
    Randomly allocating values to the support maps.
Until convergence
    Update the support maps with an E-Step
    Update the segment parameters with an M-Step
end
```

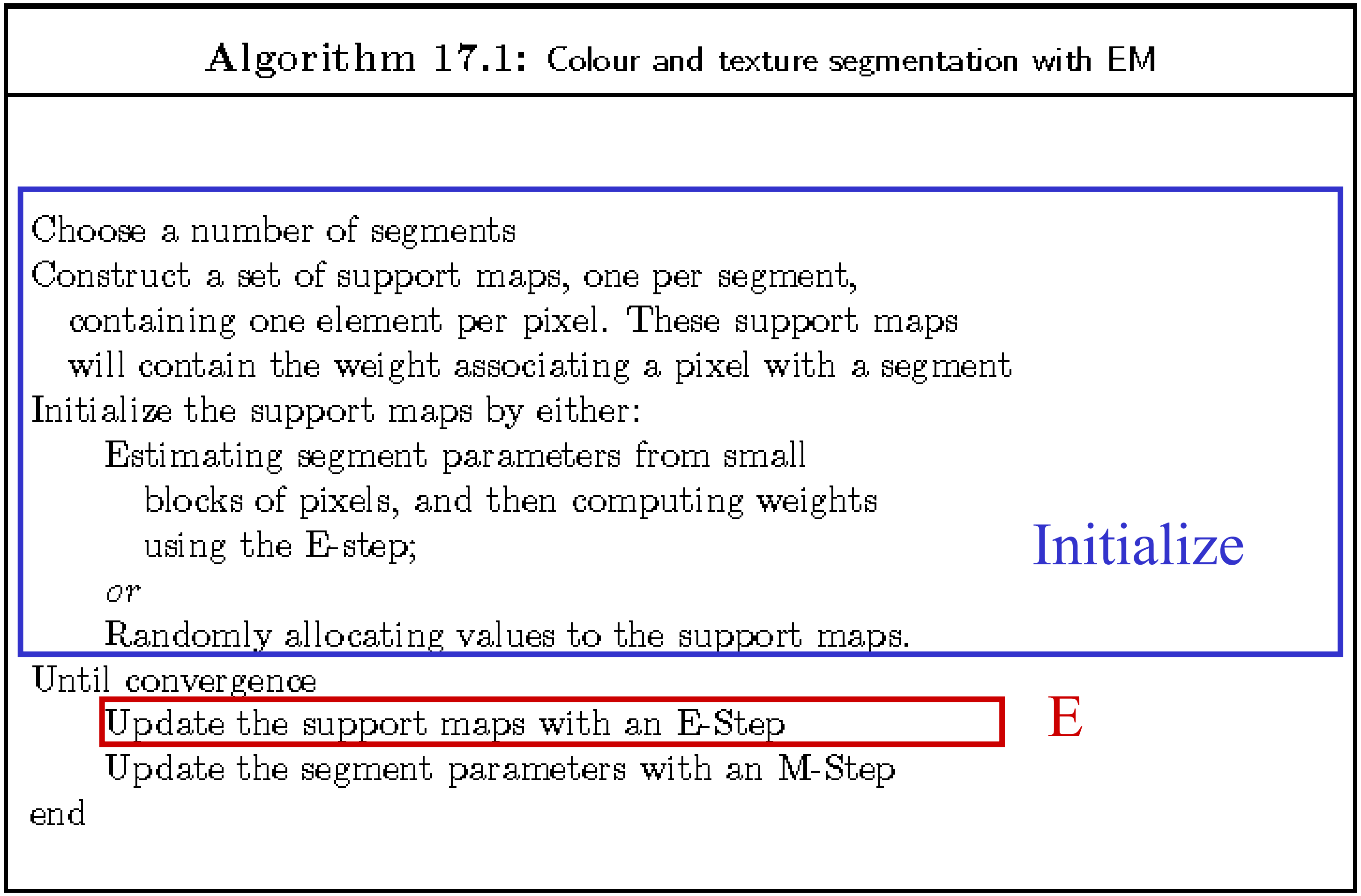
Color segmentation with EM

Algorithm 17.1: Colour and texture segmentation with EM

```
Choose a number of segments
Construct a set of support maps, one per segment,
   containing one element per pixel. These support maps
   will contain the weight associating a pixel with a segment
Initialize the support maps by either:
      Estimating segment parameters from small
         blocks of pixels, and then computing weights
         using the E-step;
      or
      Randomly allocating values to the support maps.
Until convergence
      Update the support maps with an E-Step
      Update the segment parameters with an M-Step
end
```

Initialize

Color segmentation with EM



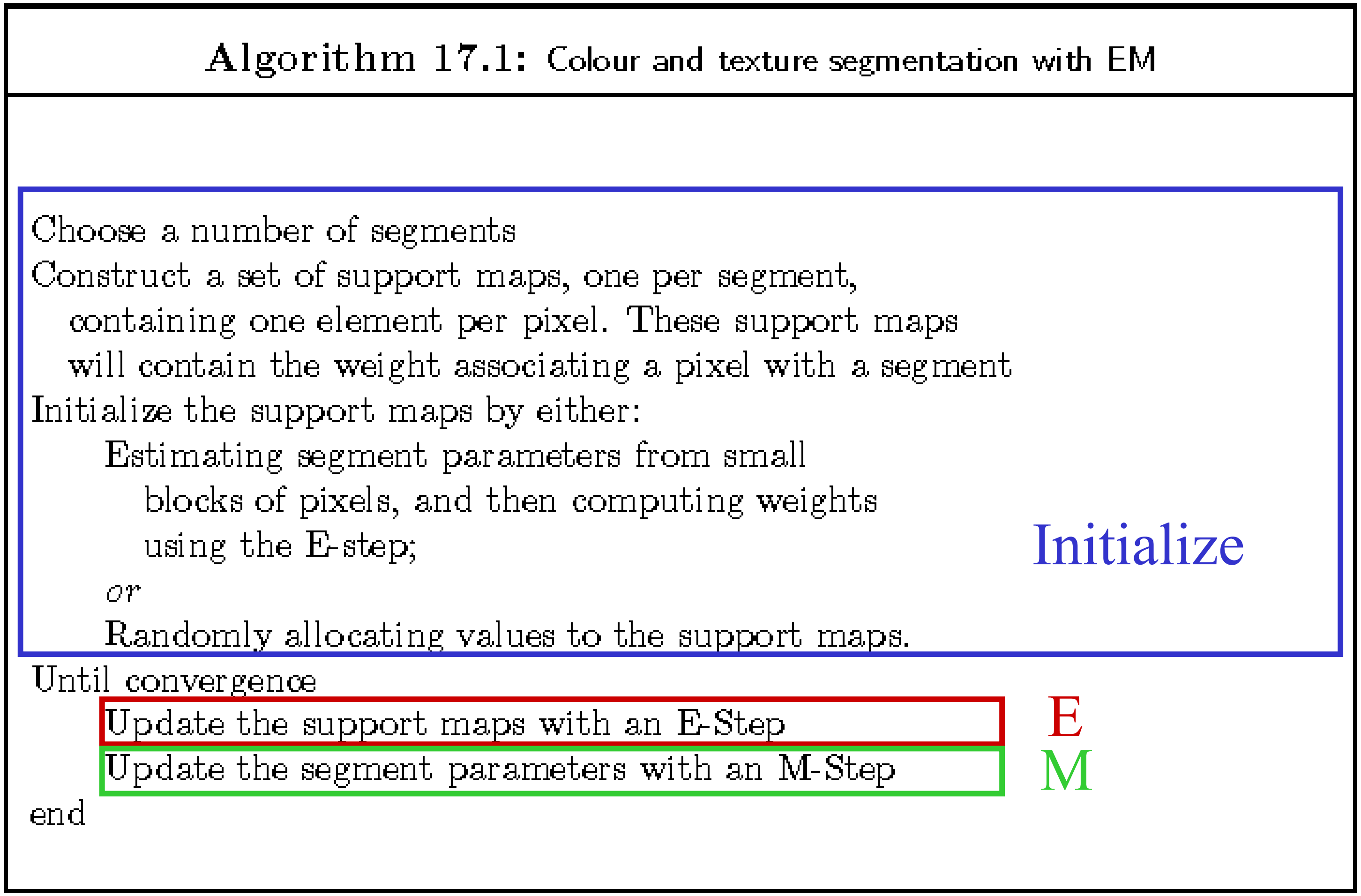
Algorithm 17.1: Colour and texture segmentation with EM

```
Choose a number of segments
Construct a set of support maps, one per segment,
  containing one element per pixel. These support maps
  will contain the weight associating a pixel with a segment
Initialize the support maps by either:
    Estimating segment parameters from small
      blocks of pixels, and then computing weights
      using the E-step;
    or
    Randomly allocating values to the support maps.
Until convergence
    Update the support maps with an E-Step
    Update the segment parameters with an M-Step
end
```

Initialize

E

Color segmentation with EM



Algorithm 17.1: Colour and texture segmentation with EM

```
Choose a number of segments
Construct a set of support maps, one per segment,
  containing one element per pixel. These support maps
  will contain the weight associating a pixel with a segment
Initialize the support maps by either:
    Estimating segment parameters from small
      blocks of pixels, and then computing weights
      using the E-step;
    or
    Randomly allocating values to the support maps.
Until convergence
    Update the support maps with an E-Step
    Update the segment parameters with an M-Step
end
```

Initialize

E

M

The E-step: The l, m'th element of $\mathcal{I}$ is one if the l'th pixel comes from the m'th blob, and zero otherwise. This means that

$$\begin{aligned} \mathrm{E}(I_{lm}) &= 1P(l\text{'th pixel comes from the } m\text{'th blob})+ \\ &\quad 0.P(l\text{'th pixel does not come from the } m\text{'th blob}) \\ &= P(l\text{'th pixel comes from the } m\text{'th blob}) \end{aligned}$$

Assuming that the parameters are for the s'th iteration are $\Theta^{(s)}$, we have:

$$\overline{I}_{lm} = \frac{\alpha_m^{(s)} p_m(x_l|\theta_l^{(s)})}{\sum_{k=1}^{K} \alpha_k^{(s)} p_k(x_l|\theta_l^{(s)})}$$

(keeping in mind that $\alpha_m^{(s)}$ means the value of α_m on the s'th iteration!).

Algorithm 17.2: Colour and texture segmentation with EM: - the E-step

```
For each pixel location $l$
    For each segment $m$
        Insert $\alpha_m^{(s)} p_m(x_l|\theta_l^{(s)})$
            in pixel location $l$ in the support map $m$
    end  Add the support map values to obtain
        $\sum_{k=1}^{K} \alpha_k^{(s)} p_k(x_l|\theta_l^{(s)})$
        and divide the value in location $l$ in each support map by this term
end
```

E-step

The E-step: The l, m'th element of $\mathcal{I}$ is one if the l'th pixel comes from the m'th blob, and zero otherwise. This means that

$$\begin{aligned}\mathrm{E}(I_{lm}) &= 1P(l\text{'th pixel comes from the } m\text{'th blob})+\\ &\quad 0.P(l\text{'th pixel does not come from the } m\text{'th blob})\\ &= P(l\text{'th pixel comes from the } m\text{'th blob})\end{aligned}$$

Assuming that the parameters are for the s'th iteration are $\Theta^{(s)}$, we have:

$$\overline{I}_{lm} = \frac{\alpha_m^{(s)} p_m(x_l|\theta_l^{(s)})}{\sum_{k=1}^{K} \alpha_k^{(s)} p_k(x_l|\theta_l^{(s)})}$$

(keeping in mind that $\alpha_m^{(s)}$ means the value of α_m on the s'th iteration!).

E-step

Estimate support maps:

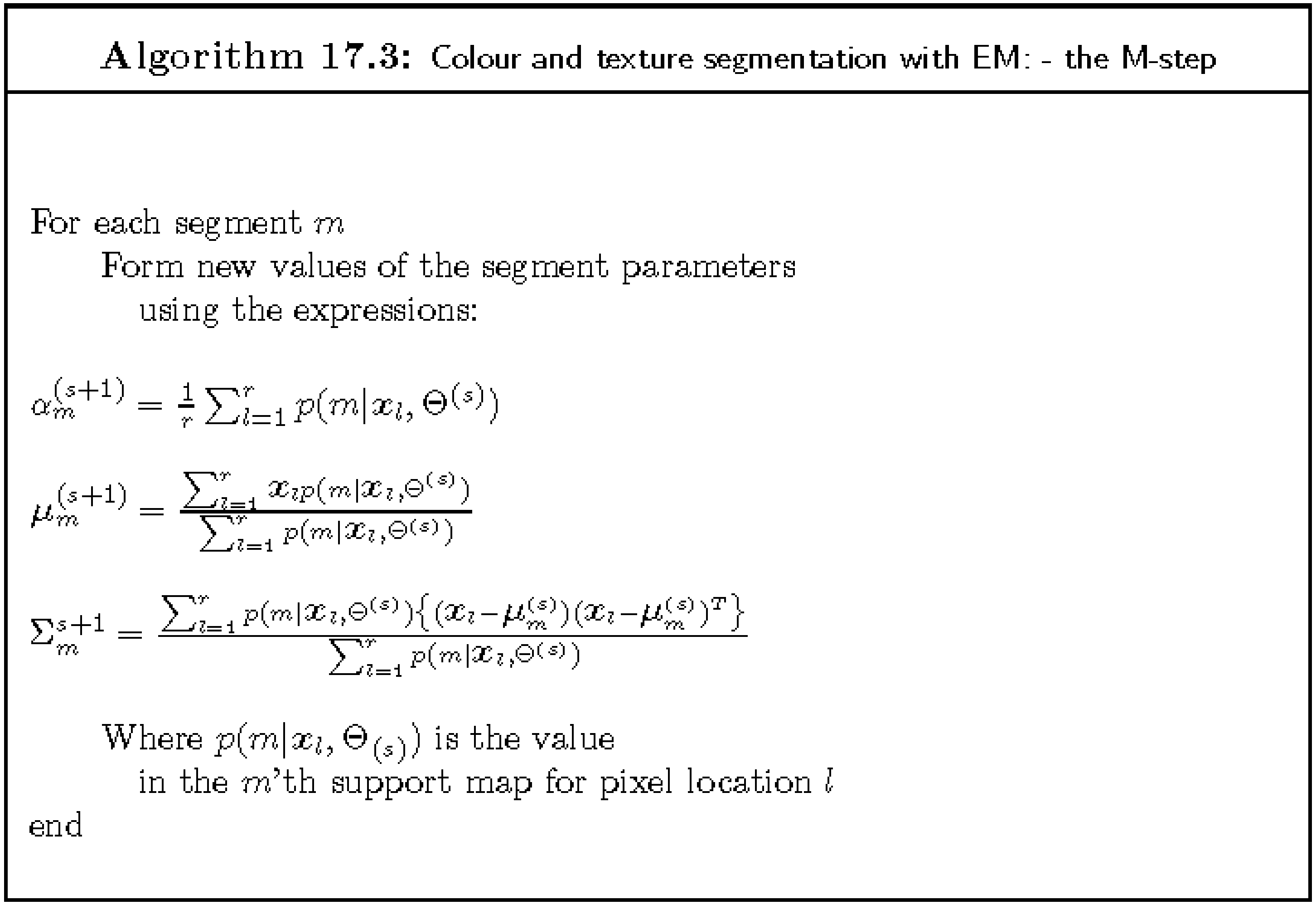
$$p(m|\boldsymbol{x}_l, \Theta_{(s)}) = \frac{\alpha_m^{(s)} p_m(\boldsymbol{x}_l|\theta_l^{(s)})}{\sum_{k=1}^{K} \alpha_k^{(s)} p_k(\boldsymbol{x}_l|\theta_l^{(s)})}$$

Algorithm 17.2: Colour and texture segmentation with EM: - the E-step

```
For each pixel location $l$
    For each segment $m$
      Insert $\alpha_m^{(s)} p_m(x_l|\theta_l^{(s)})$
        in pixel location $l$ in the support map $m$
    end  Add the support map values to obtain
      $\sum_{k=1}^{K} \alpha_k^{(s)} p_k(x_l|\theta_l^{(s)})$
      and divide the value in location $l$ in each support map by this term
end
```

M-step

Update mean’s, covar’s, and mixing coef.’s using support map:



Algorithm 17.3: Colour and texture segmentation with EM: - the M-step

```
For each segment $m$
    Form new values of the segment parameters
      using the expressions:

$\alpha_m^{(s+1)} = \frac{1}{r}\sum_{l=1}^{r} p(m|\boldsymbol{x}_l, \Theta^{(s)})$

$\mu_m^{(s+1)} = \frac{\sum_{l=1}^{r} \boldsymbol{x}_l p(m|\boldsymbol{x}_l, \Theta^{(s)})}{\sum_{l=1}^{r} p(m|\boldsymbol{x}_l, \Theta^{(s)})}$

$\Sigma_m^{s+1} = \frac{\sum_{l=1}^{r} p(m|\boldsymbol{x}_l, \Theta^{(s)})\{(\boldsymbol{x}_l - \mu_m^{(s)})(\boldsymbol{x}_l - \mu_m^{(s)})^T\}}{\sum_{l=1}^{r} p(m|\boldsymbol{x}_l, \Theta^{(s)})}$

    Where $p(m|\boldsymbol{x}_l, \Theta_{(s)})$ is the value
      in the $m$'th support map for pixel location $l$
end
```

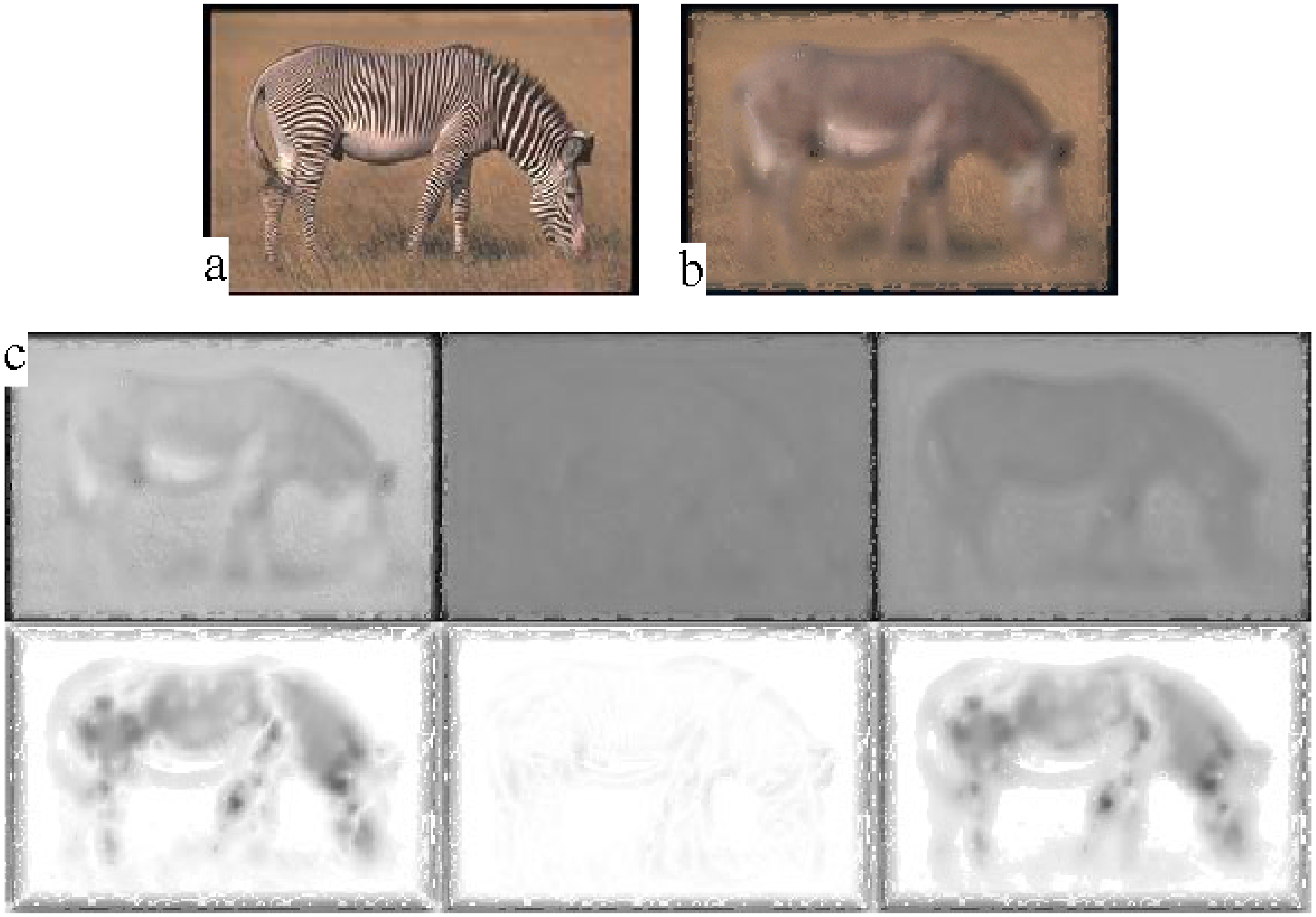
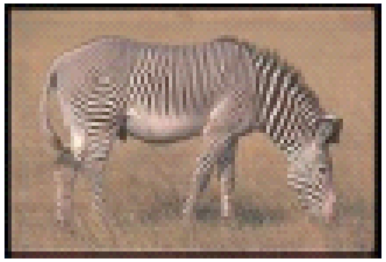


FIGURE 17.1: The image of the zebra in (a) is smoothed at varying scales to yield (b). This smoothing is done using local estimates of scale. These scale measurements essentially measure the scale of the change around a pixel; at edges, the scale is narrow, and in stripey regions it is broad, for example. The features that result are shown in (c); the top three images show the smoothed colour coordinates and the bottom three show the texture features (ac, pc and c — the scale and anisotropy features are weighted by contrast).

Segmentation with EM



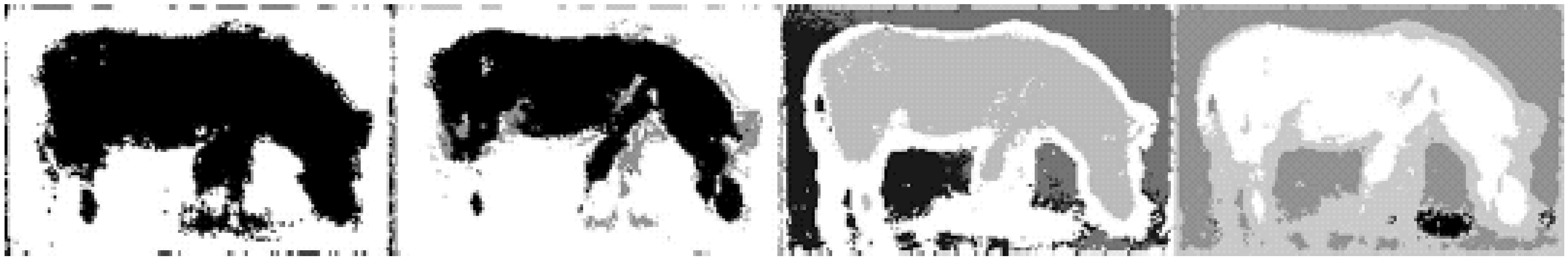


FIGURE 17.2: Each pixel of the zebra image (which is the same as that in figure 17.1) is labelled with the value of m for which $p(m|x_l, \Theta^s)$ is a maximum, to yield a segmentation. The images in show the result of this process for $K = 2, 3, 4, 5$. Each image has K grey-level values corresponding to the segment indexes. *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 © 1998 IEEE*

Motion segmentation with EM

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

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

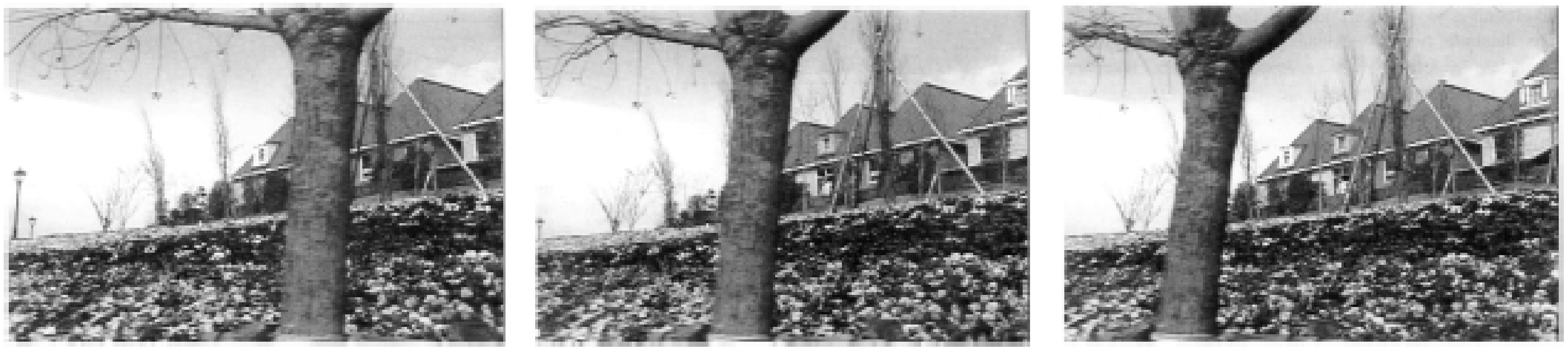
- Now we need to
  - determine which pixels belong to which region
  - estimate parameters

- Likelihood
  - assume

$$I(x, y, t) = I\left(x + v_x, y + v_y, t + 1\right)$$

$$+noise$$

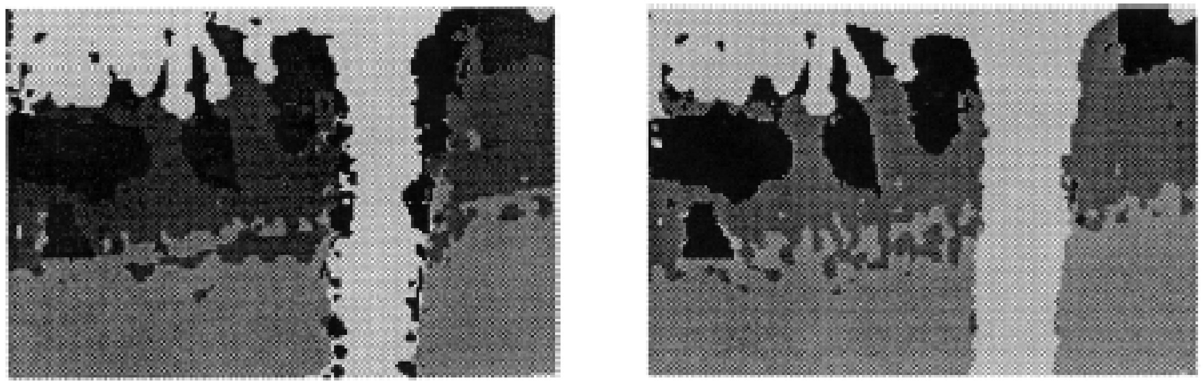
- Straightforward missing variable problem, rest is calculation

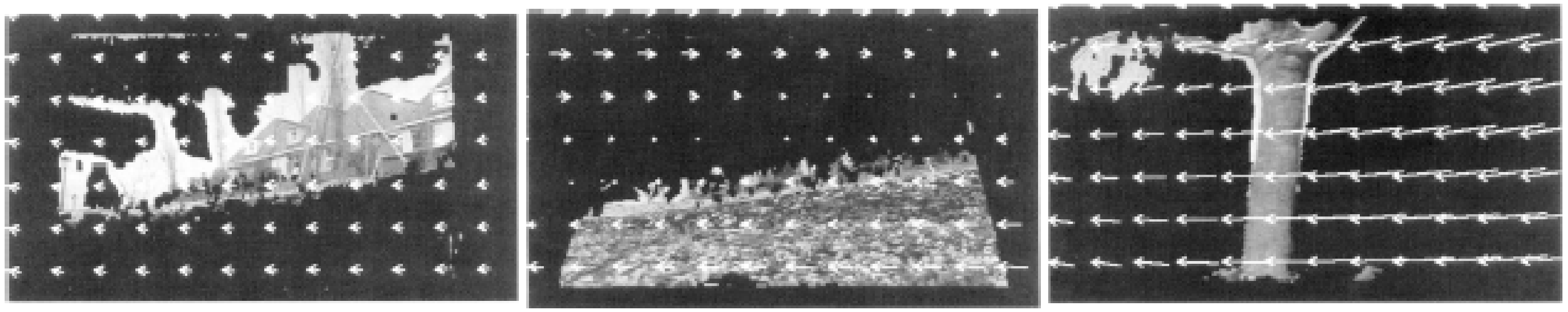


Three frames from the MPEG “flower garden” sequence

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

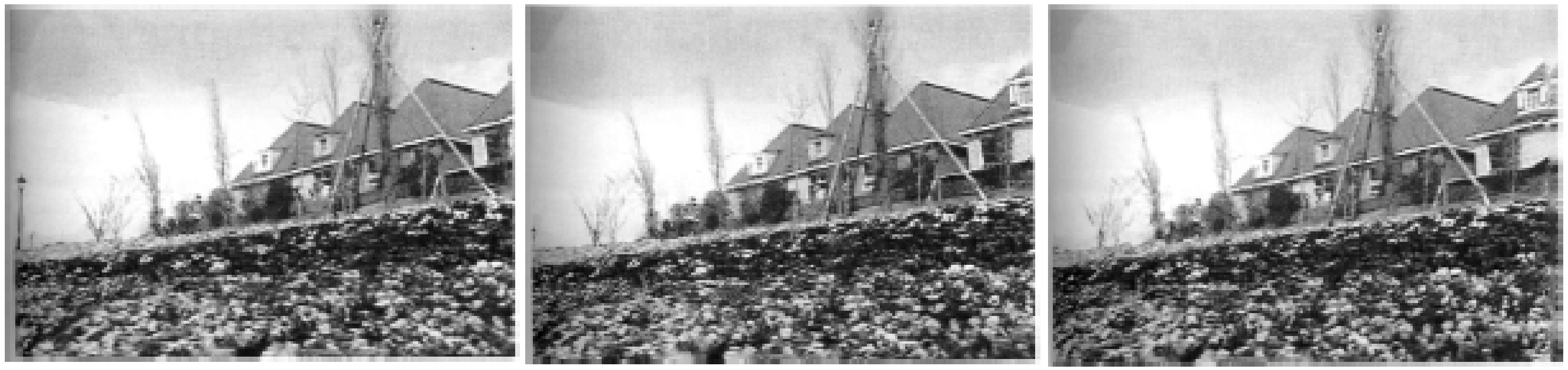
Grey level shows region no. with highest probability





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



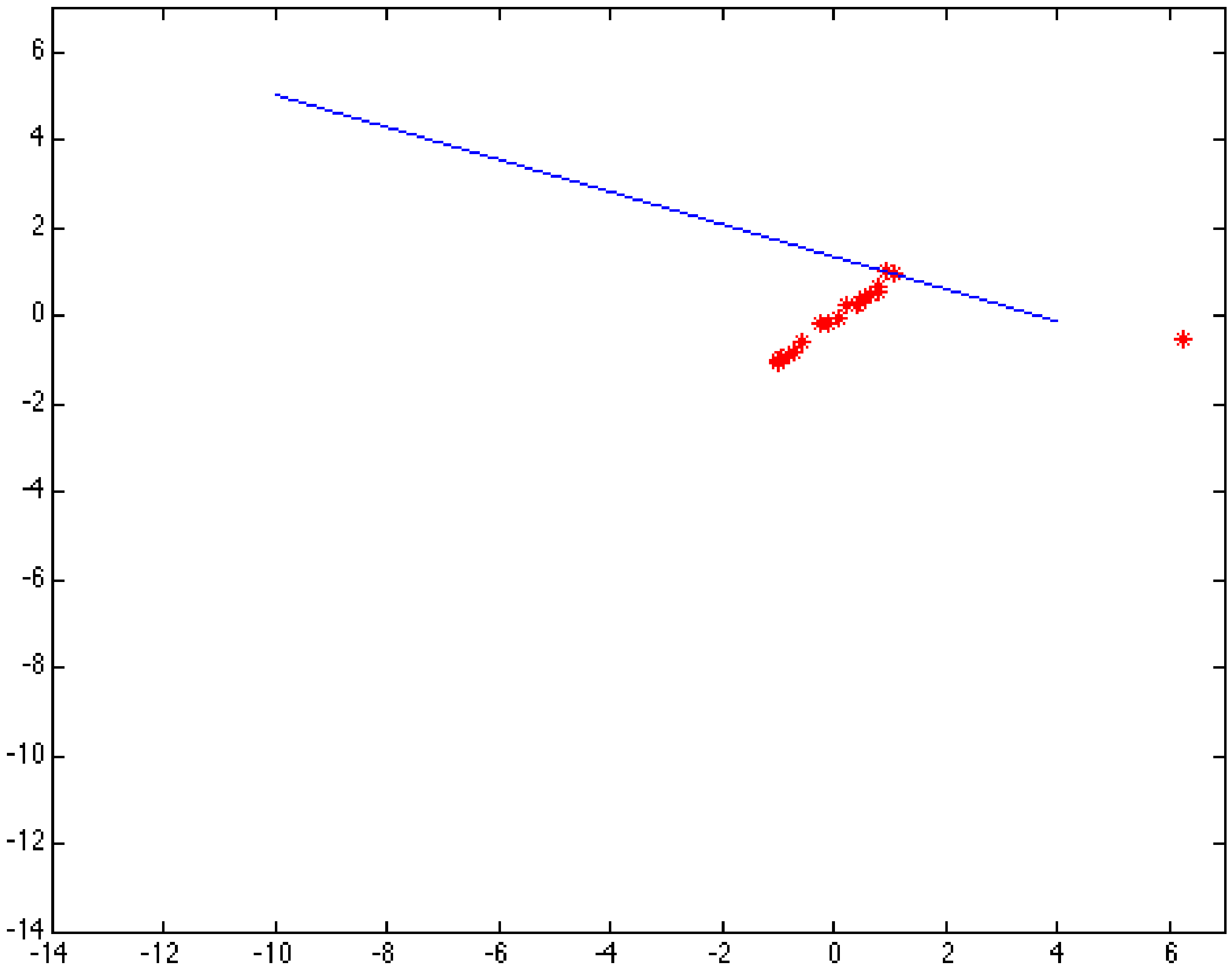
If we use multiple frames to estimate the appearance of a segment, we can fill in occlusions; so we can re-render the sequence with some segments removed.

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

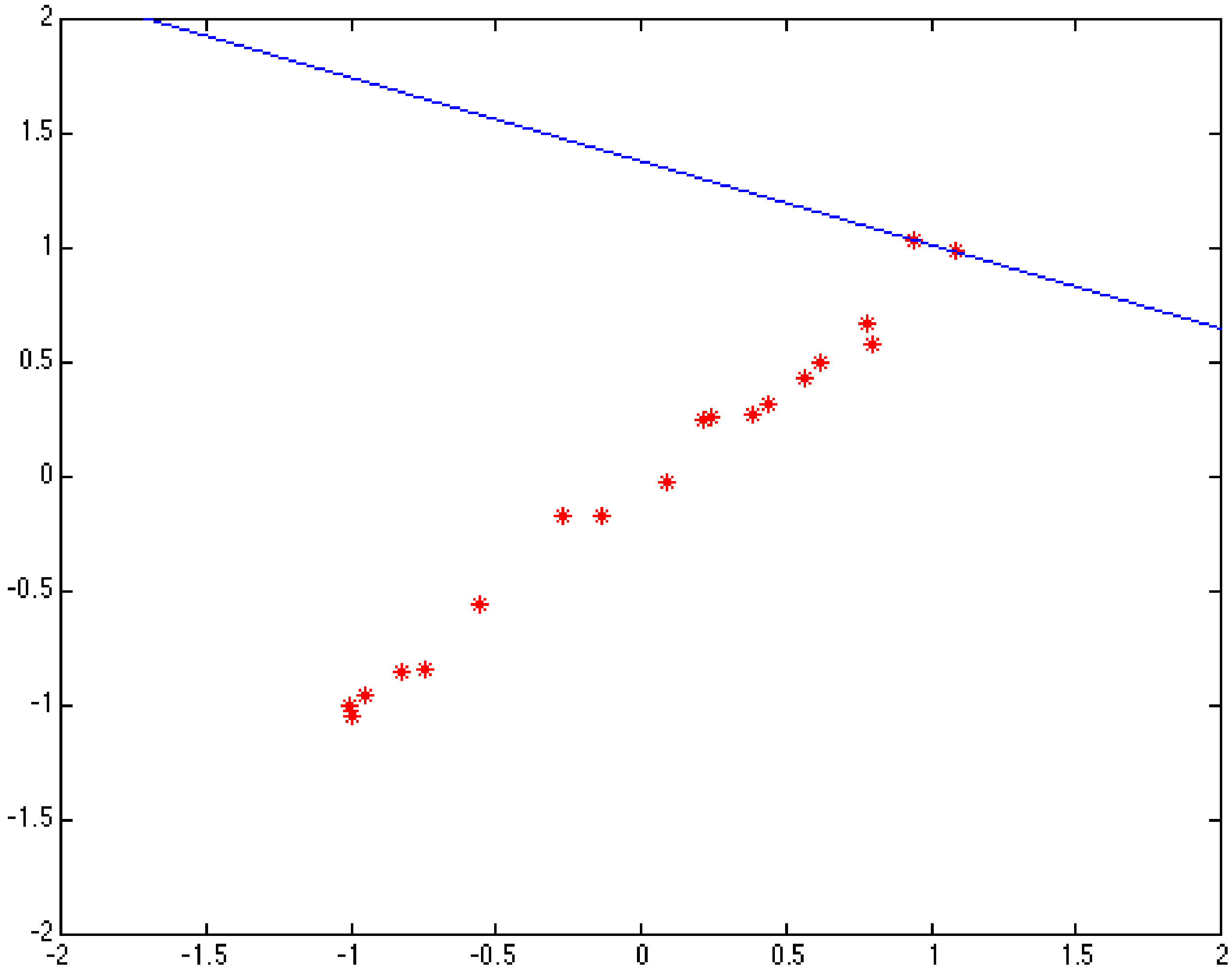
Issues with EM

- 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

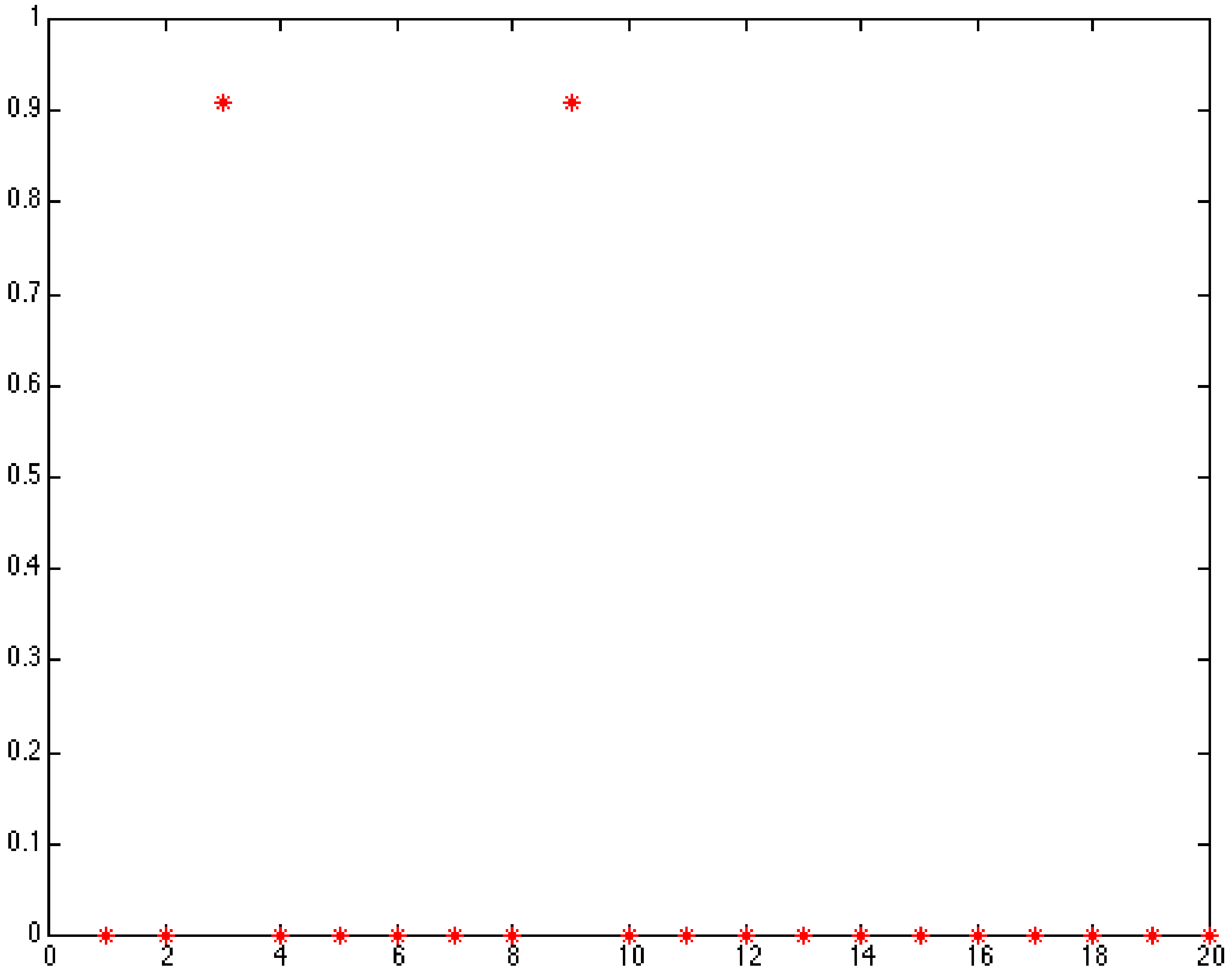
Local maximum



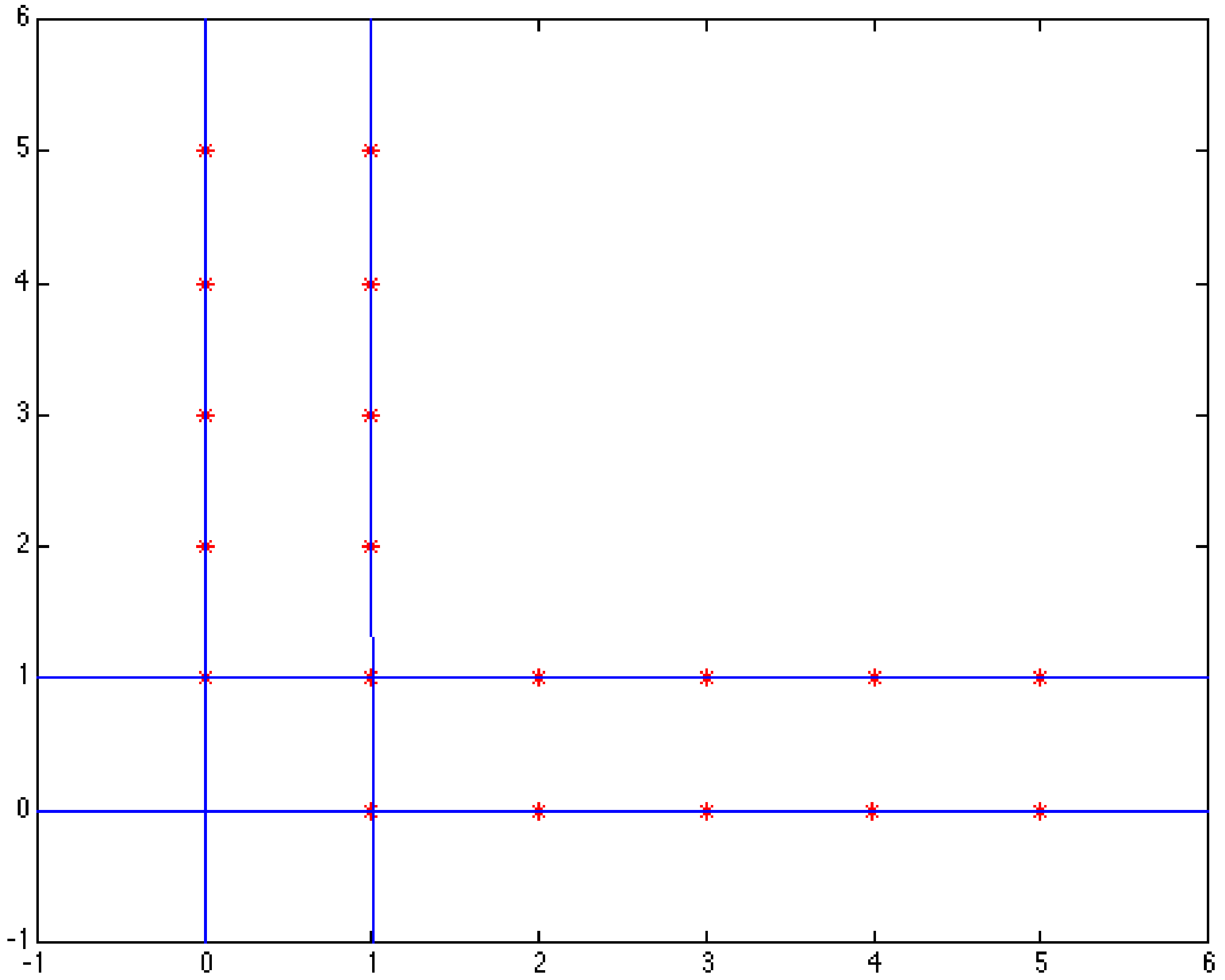
which is an excellent fit to some points



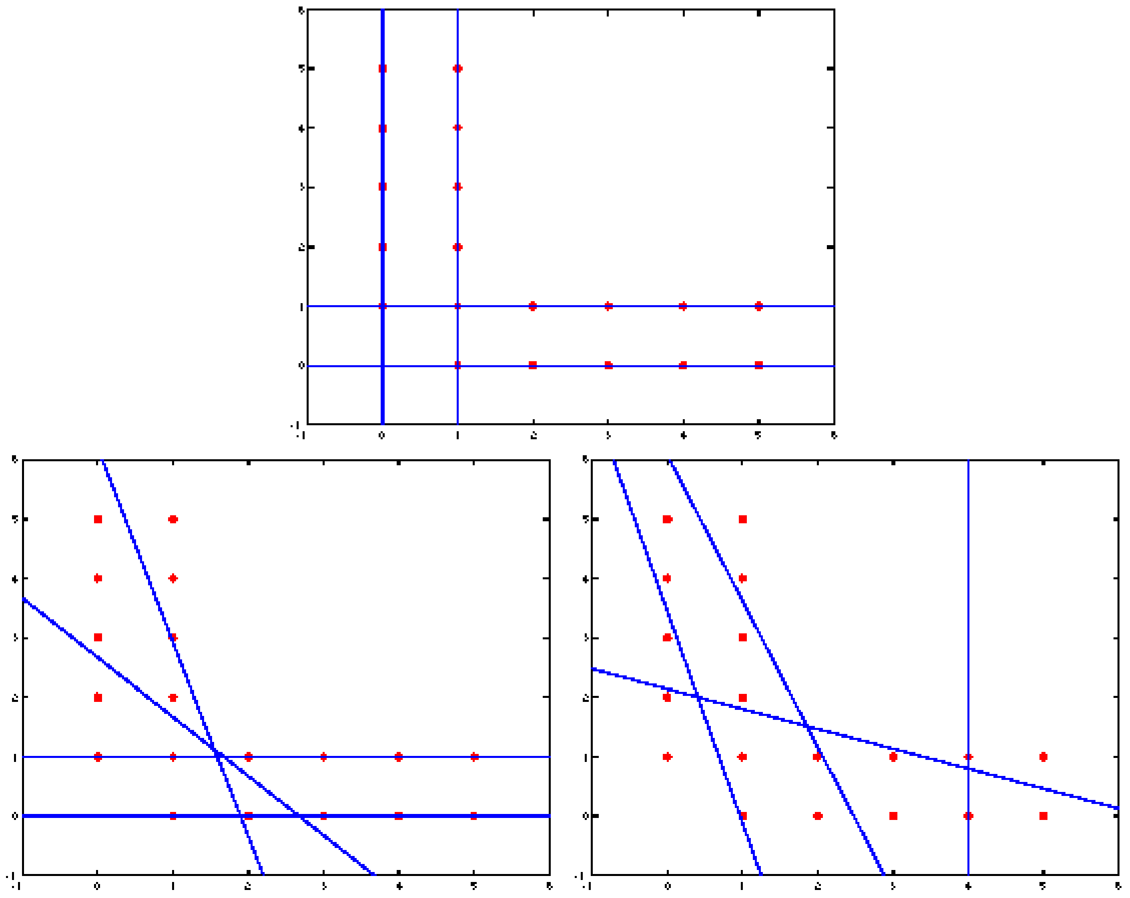
and the deltas for this maximum



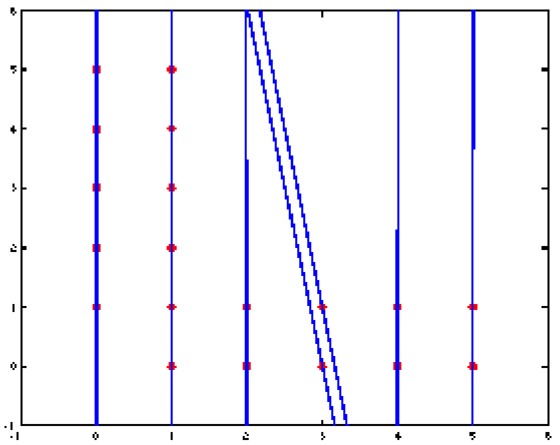
A dataset that is well fitted by four lines

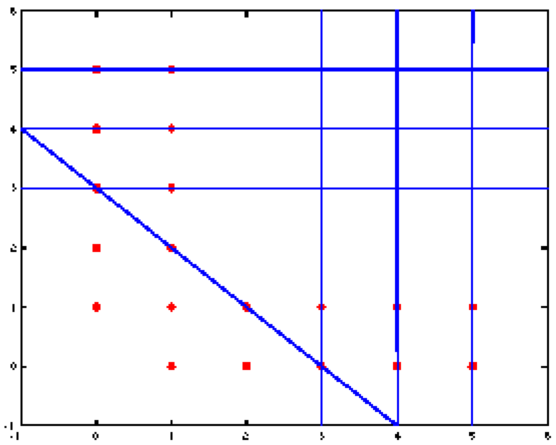


but with several local minima:



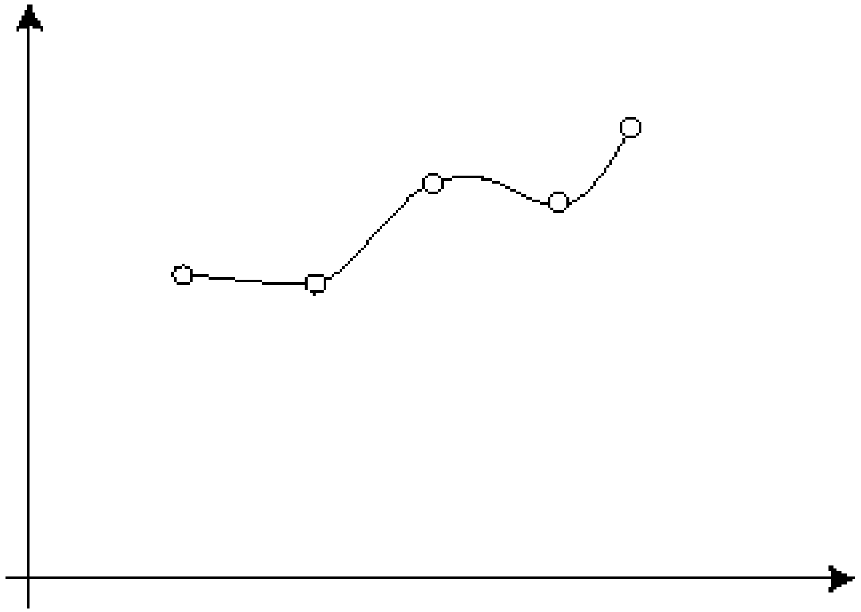
Also appealing:



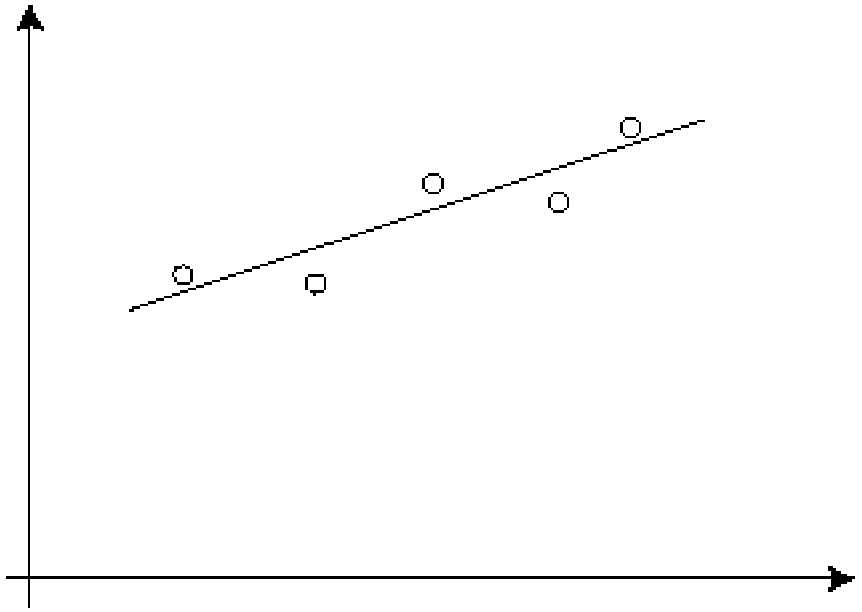


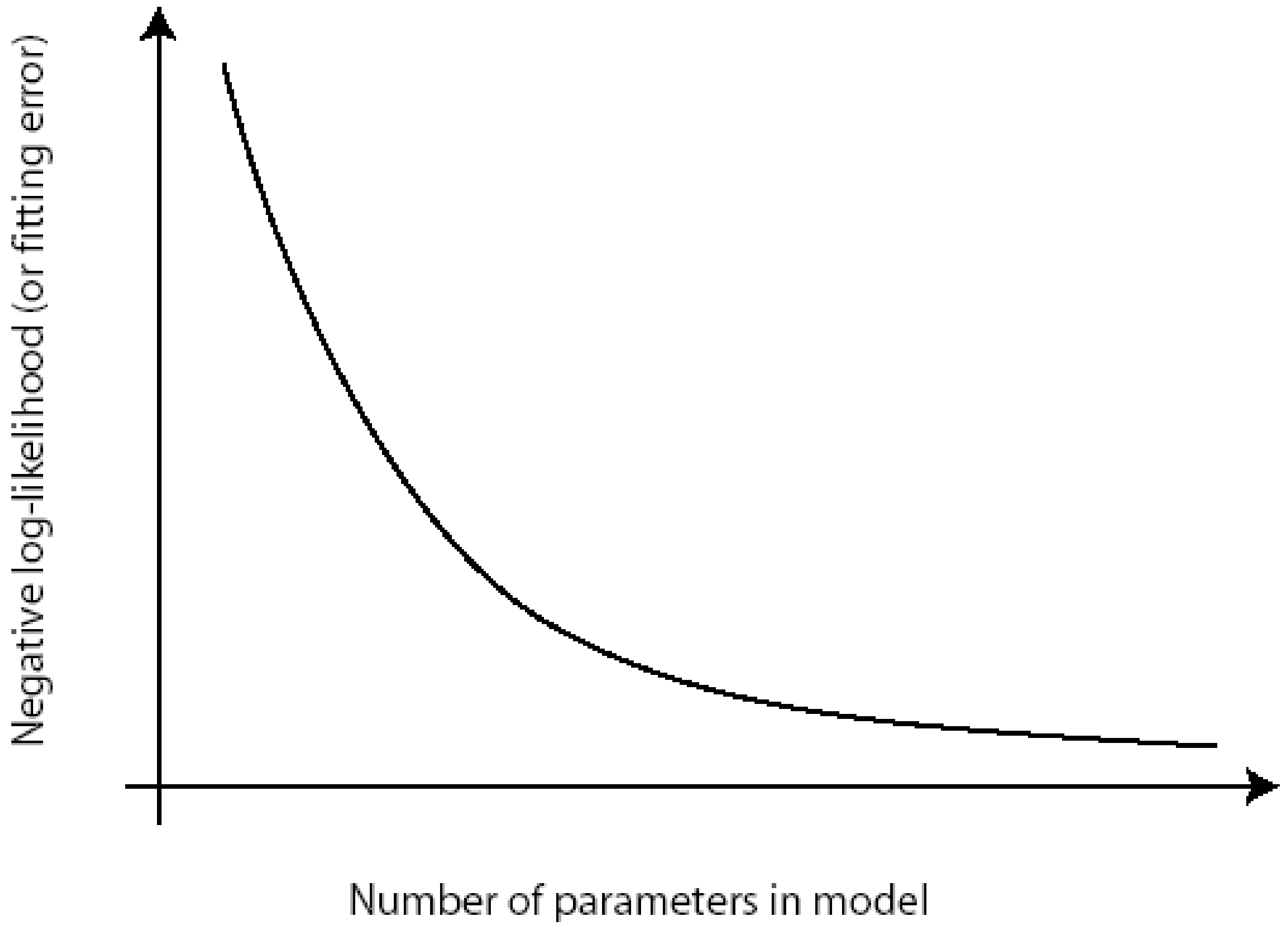
Model Selection

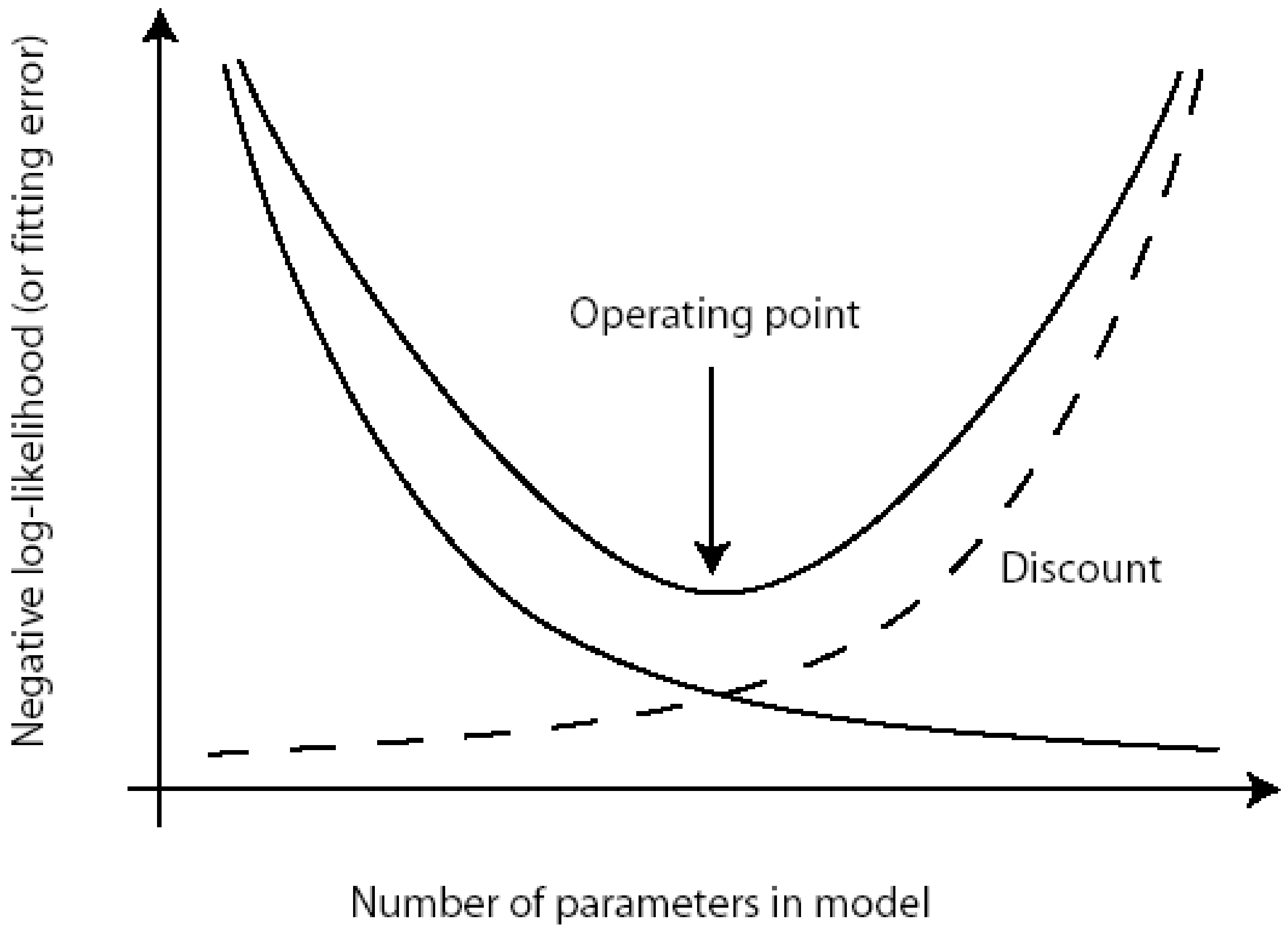
- We wish to choose a model to fit to data
  - e.g. is it a line or a circle?
  - e.g is this a perspective or orthographic camera?
  - e.g. is there an aeroplane there or is it noise?
- Issue
  - In general, models with more parameters will fit a dataset better, but are poorer at prediction
  - This 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.

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
- Average over multiple different splits
- Choose the model with the smallest value of this average

- 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

Fitting and Probabilistic Segmentation

- Robust estimation
- RANSAC
- EM
- Model Selection

[Slides from F&P]