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

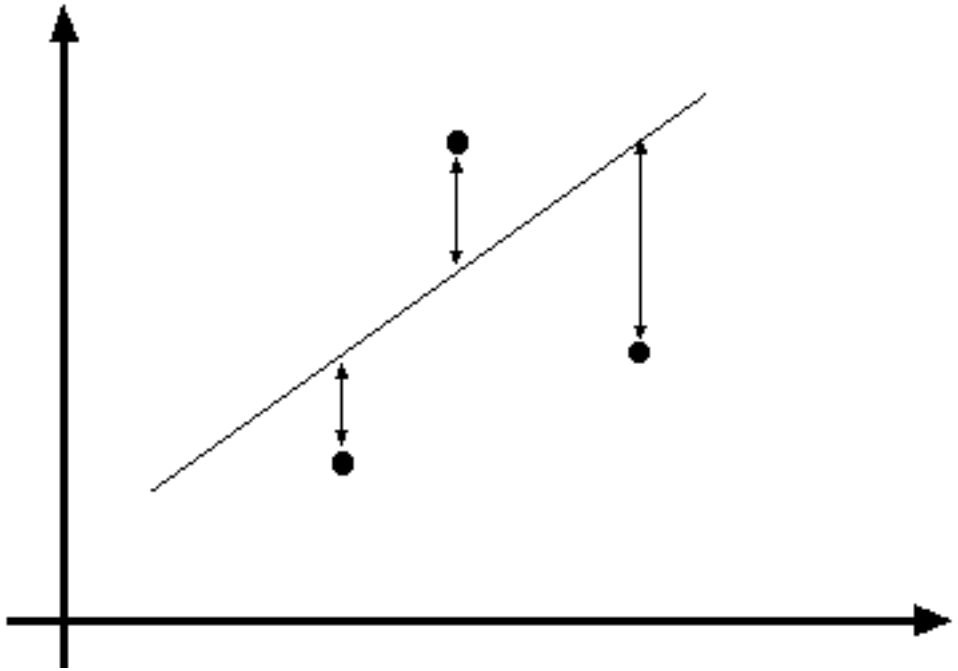
$$\frac{1}{2\square^2}\square_i (ax_i+by_i+c)^2$$

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

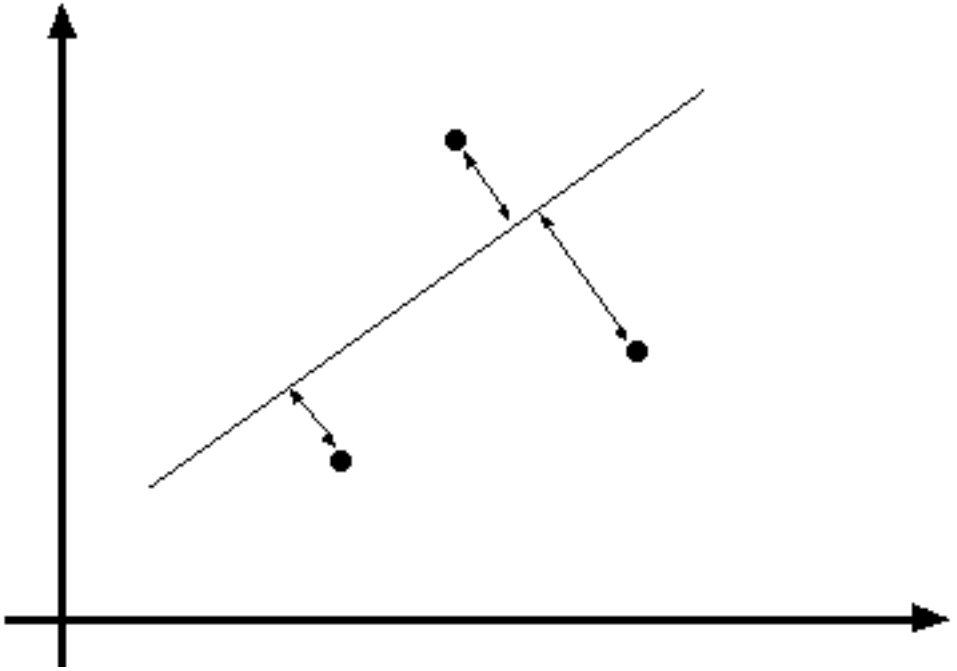
$$\frac{1}{2\sigma^2}\sum_i (ax_i + by_i + c)^2 \qquad \text{(where } a^2 + b^2 = 1\text{)}$$

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.

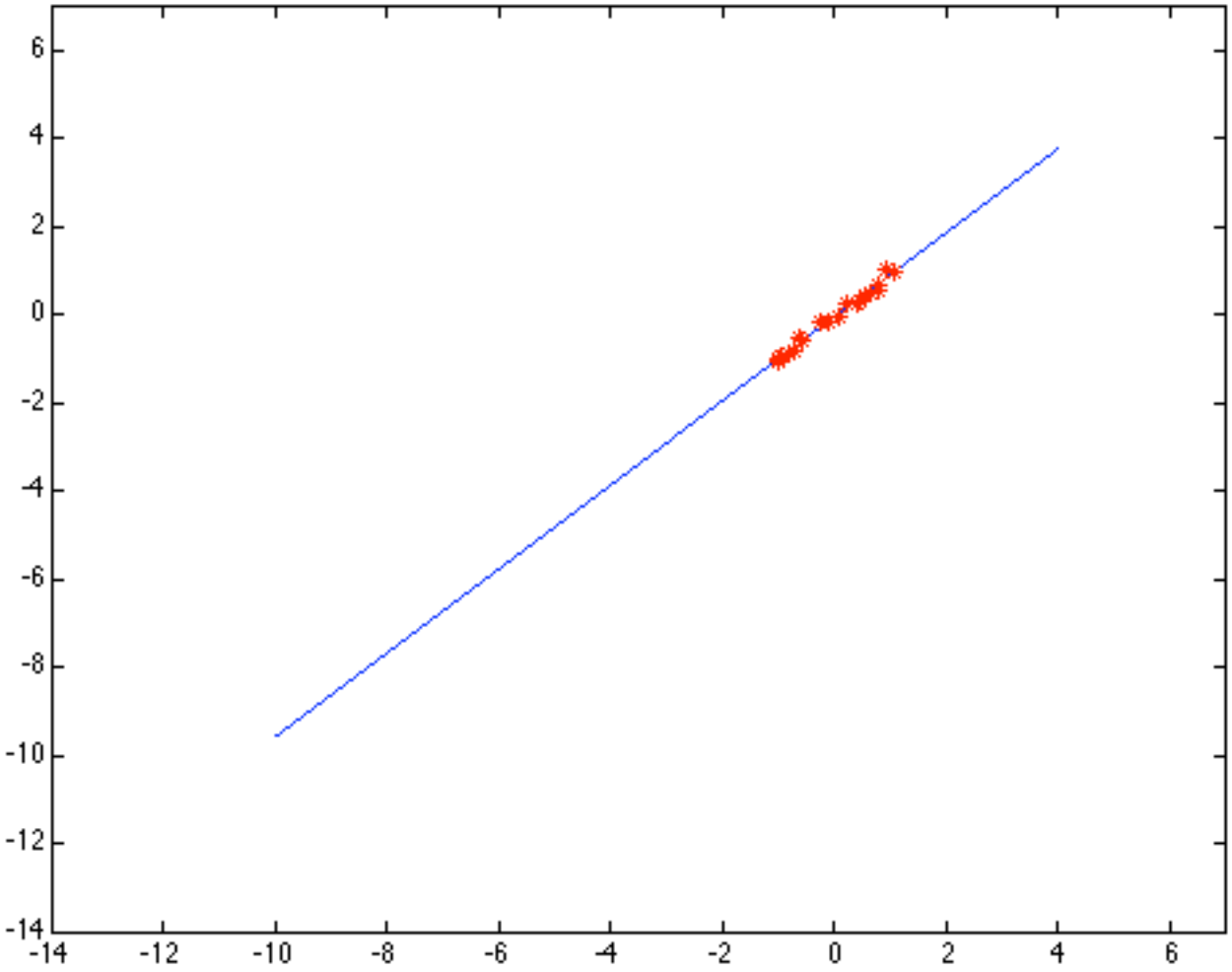
Hypothesize k lines (perhaps uniformly at random)
or
Hypothesize an assignment of lines to points
  and then fit lines using this assignment

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

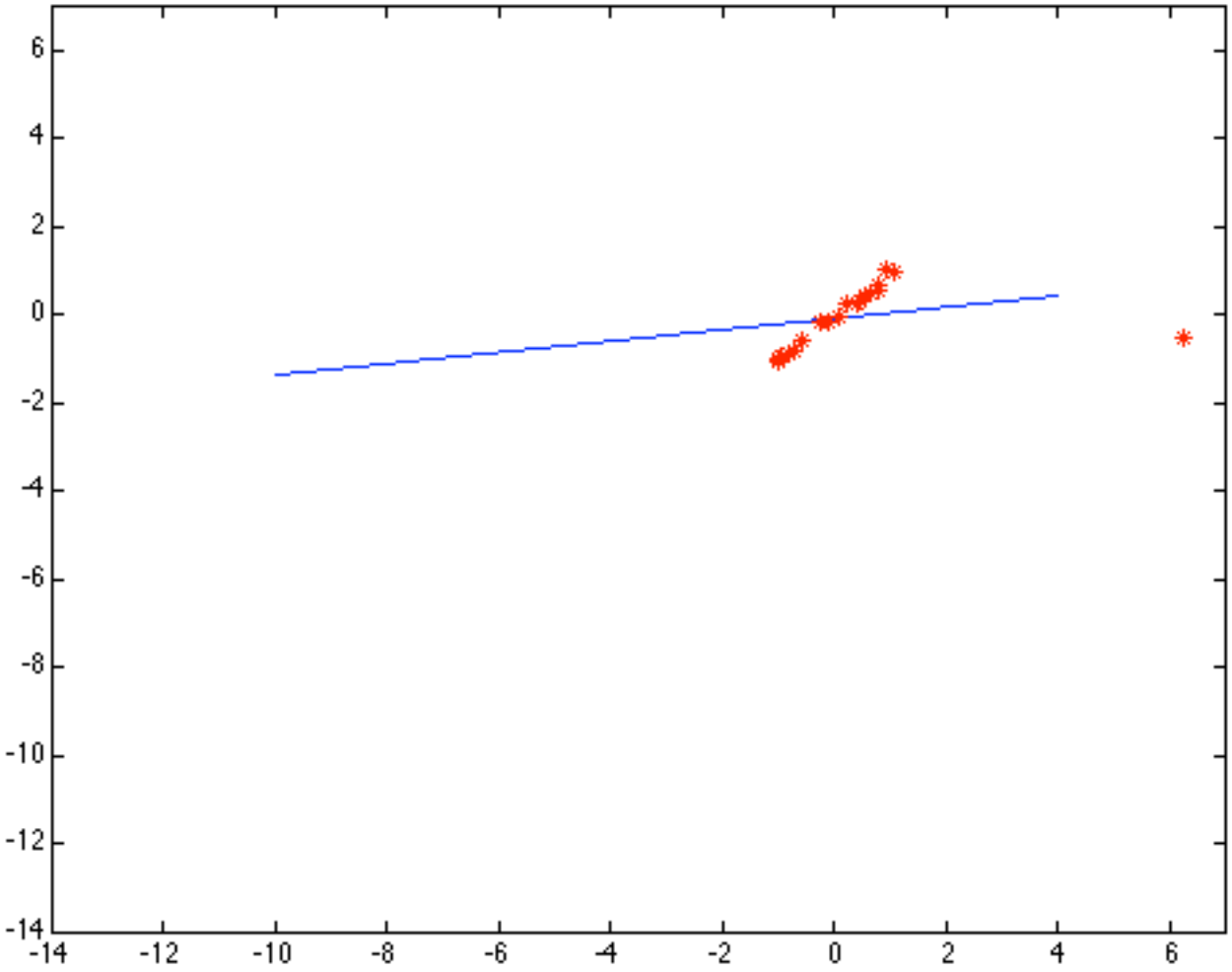
Robustness

- 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

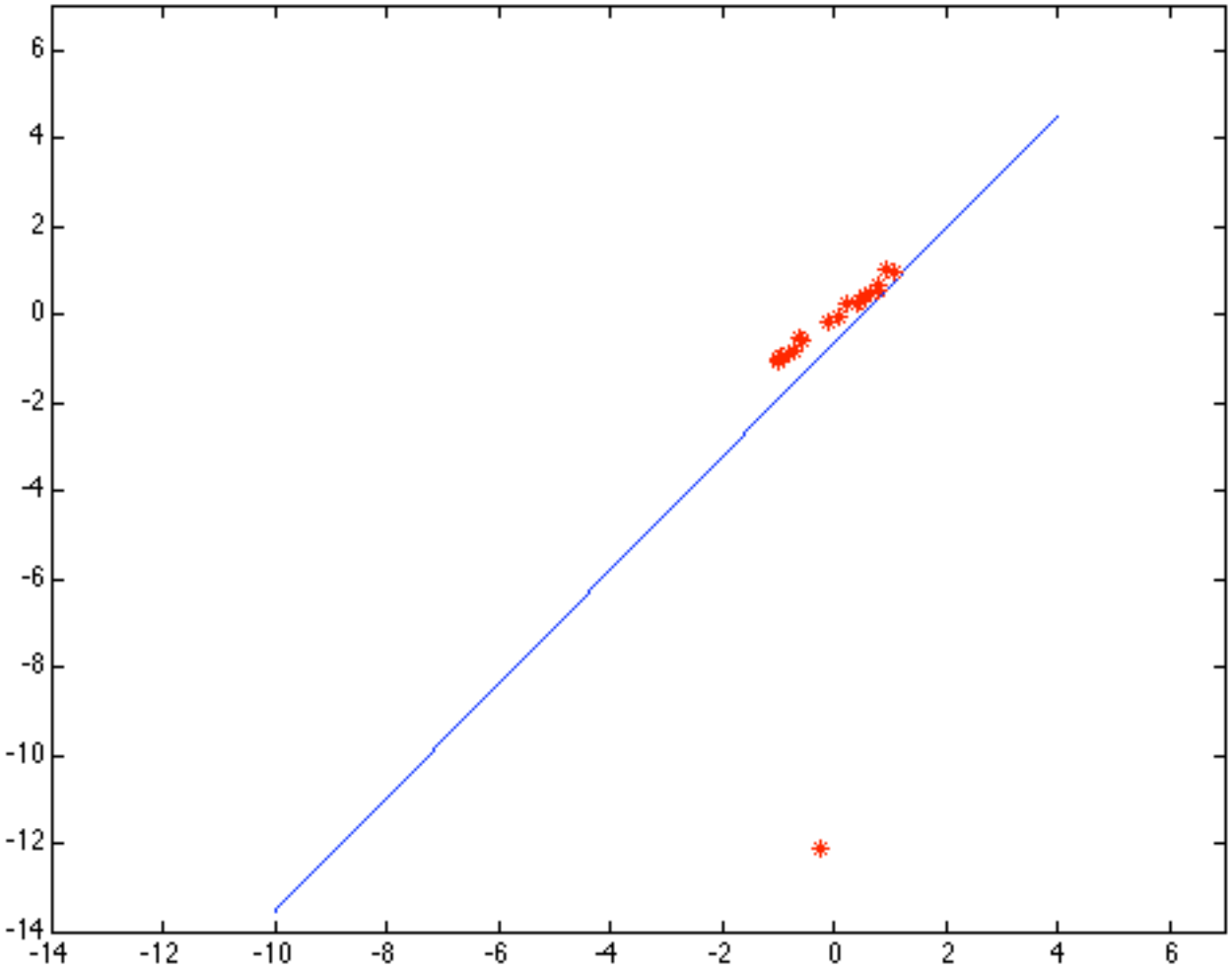
Least squares fit (good example)



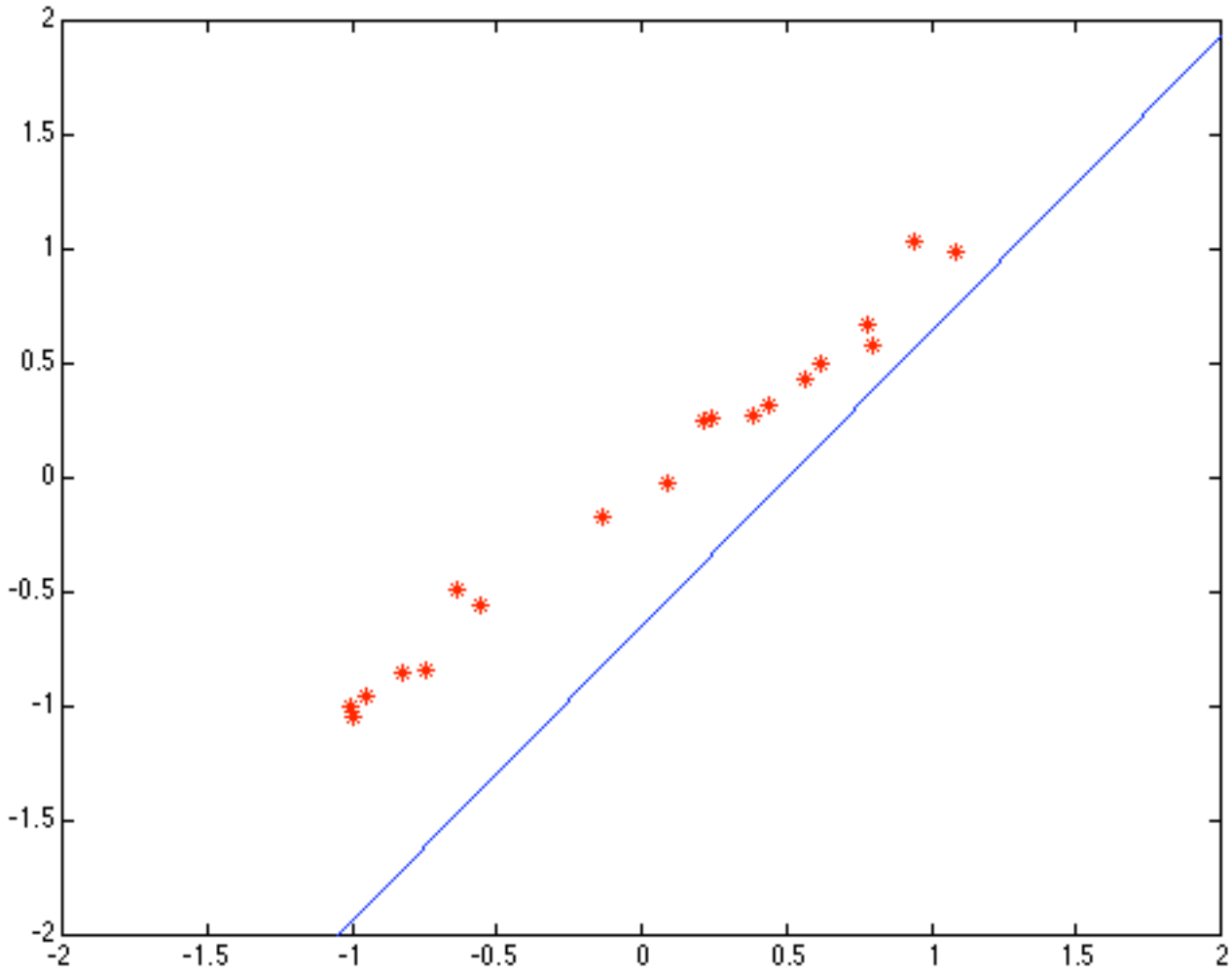
Least squares fit (destroyed by outlier)



Least squares fit (warped by outlier)



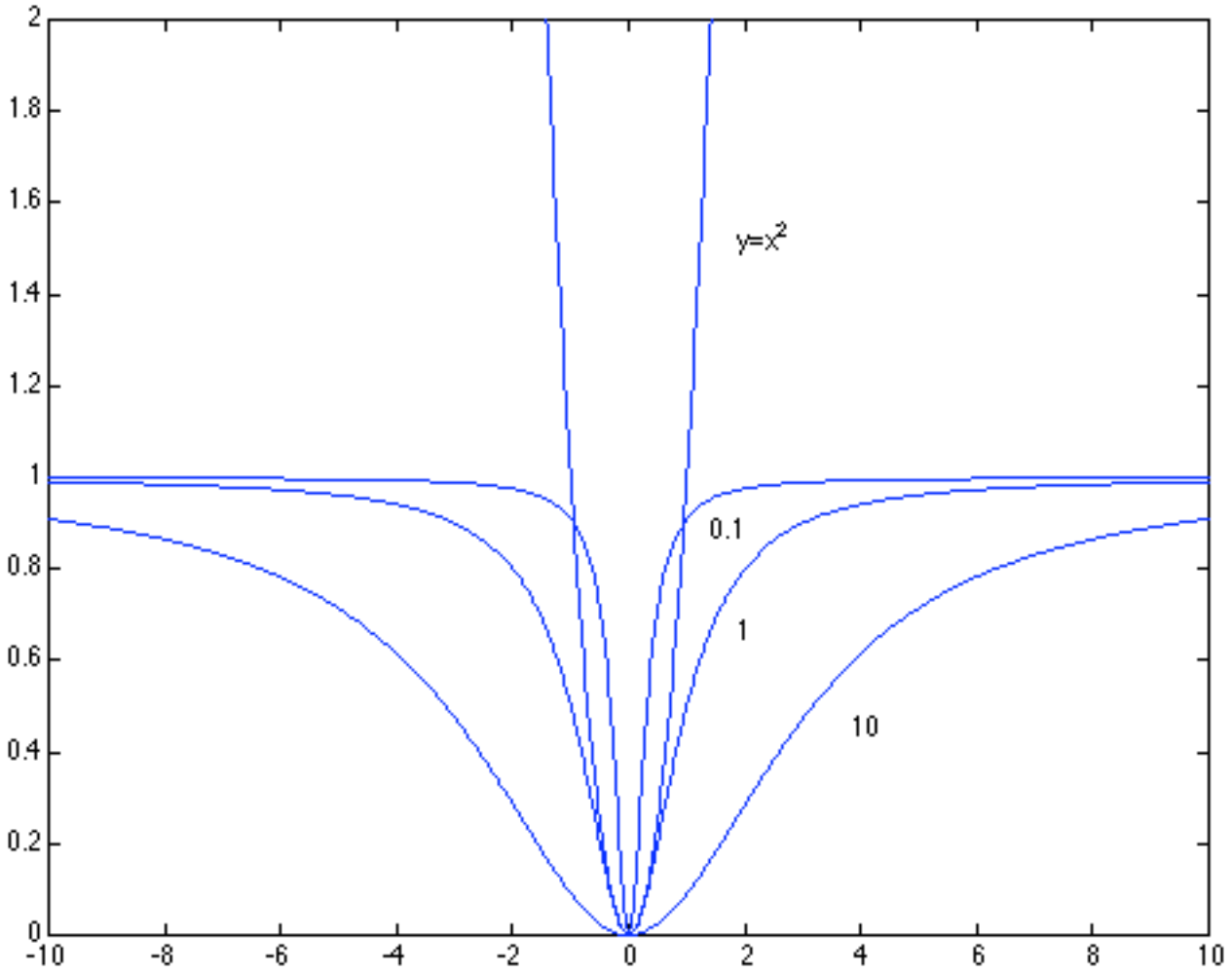
Least squares fit (previous slide details)



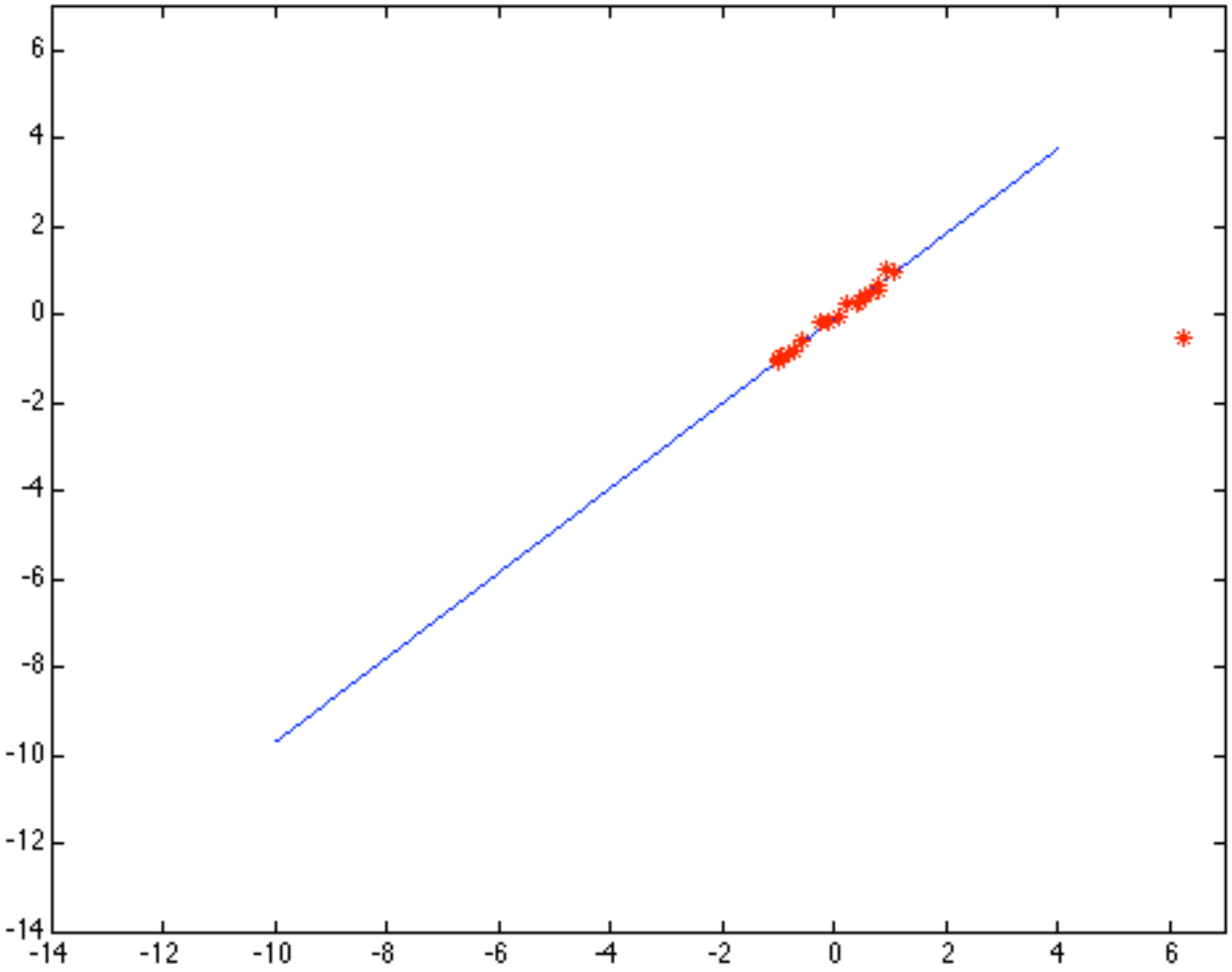
Some robust error measures

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

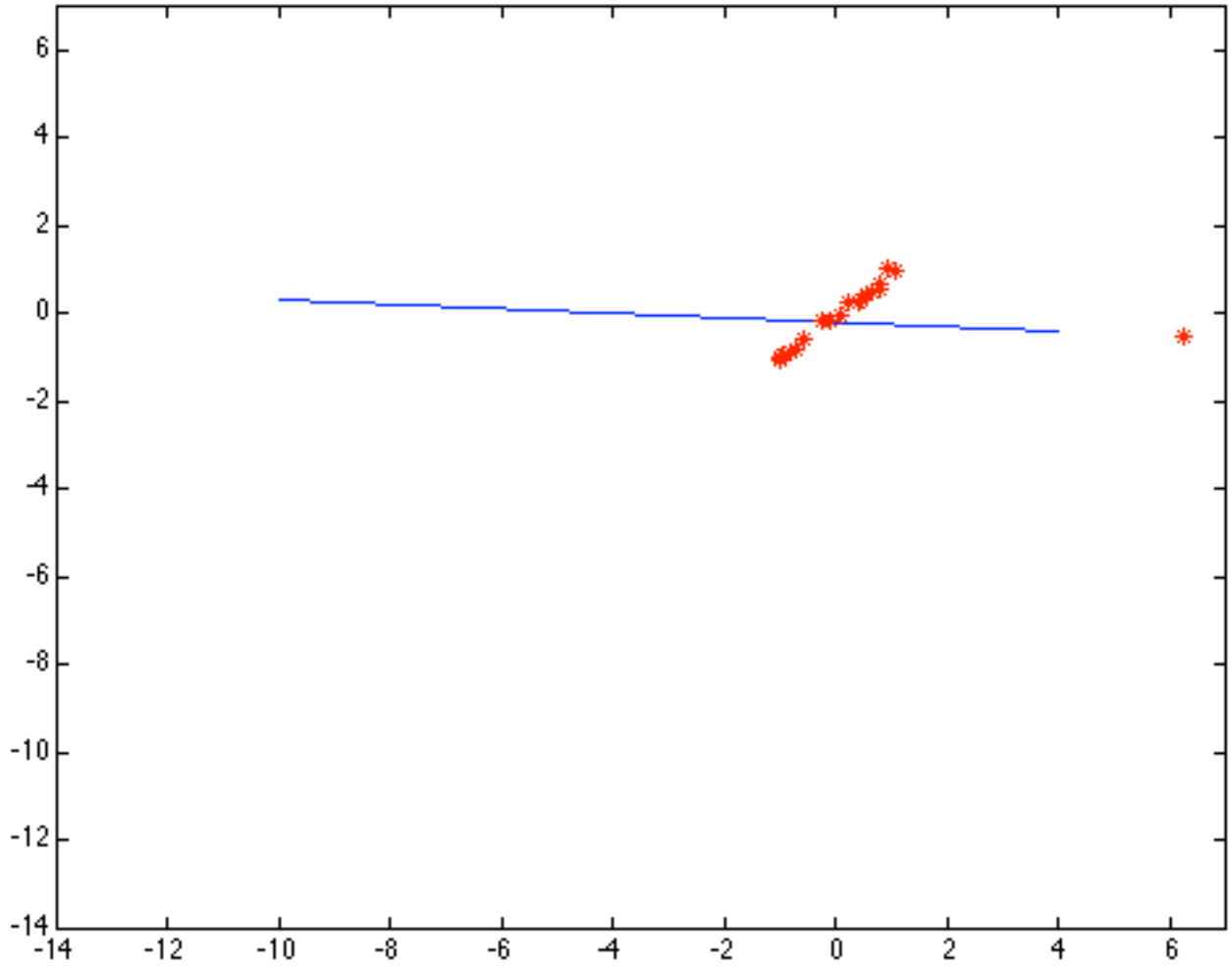
(Results for different values of s shown)



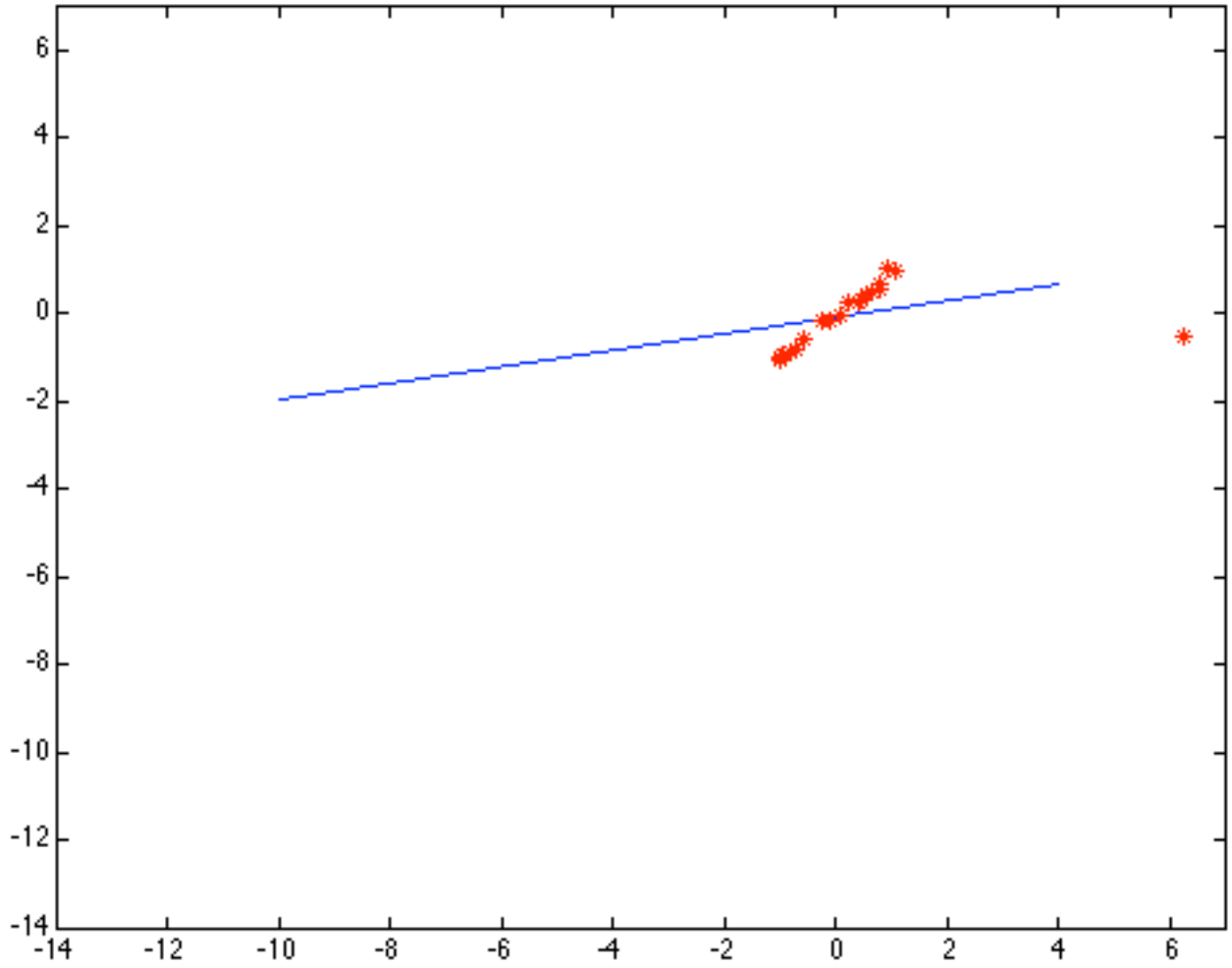
Line fit with estimator with good choice for s



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.