
Univ.AI

Regression

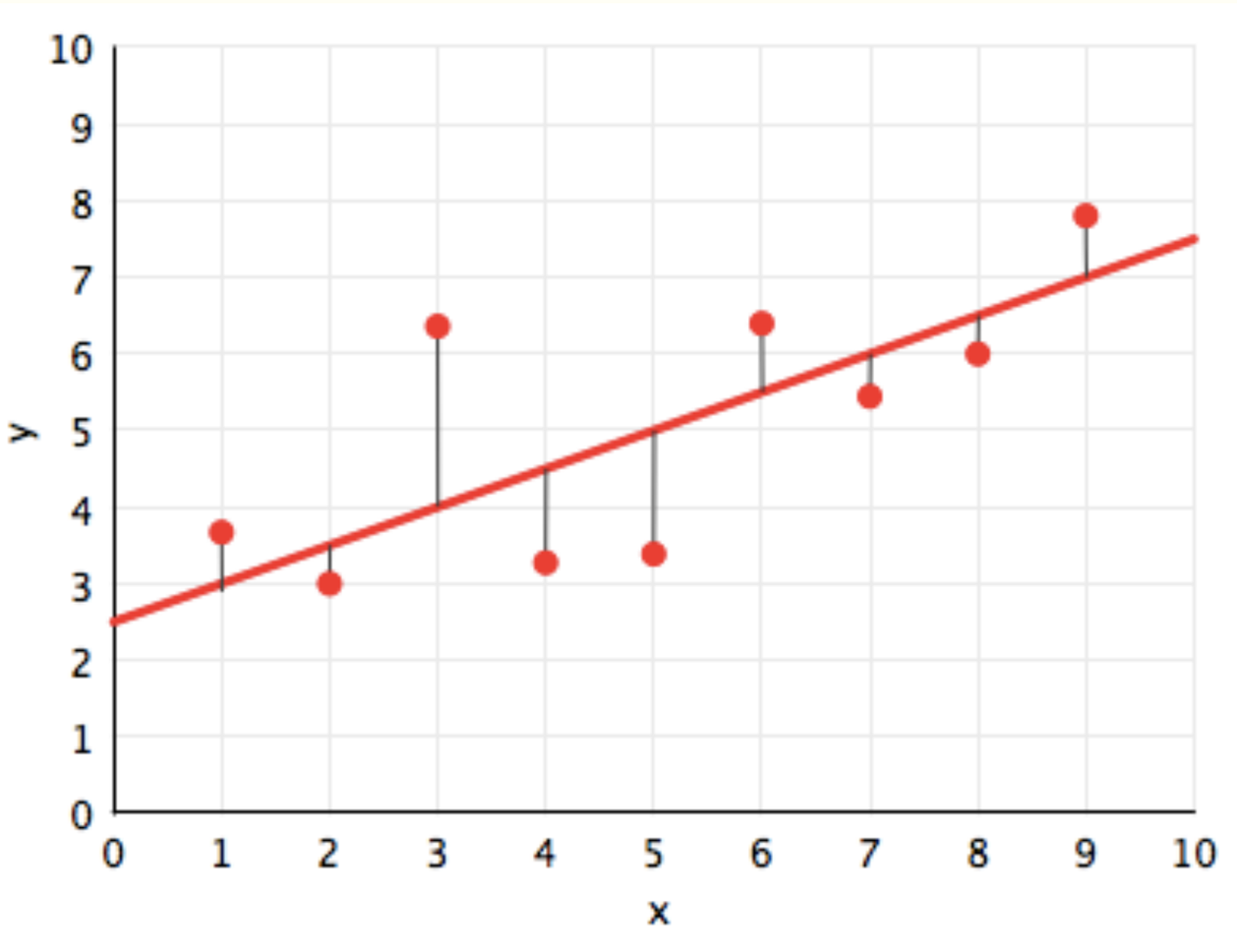
Topics we will cover

1. What is Regression
2. Loss and Gradient Descent
3. Generation Story
4. Noise and Sampling
5. Prediction is more uncertain than the mean

What is Regression?

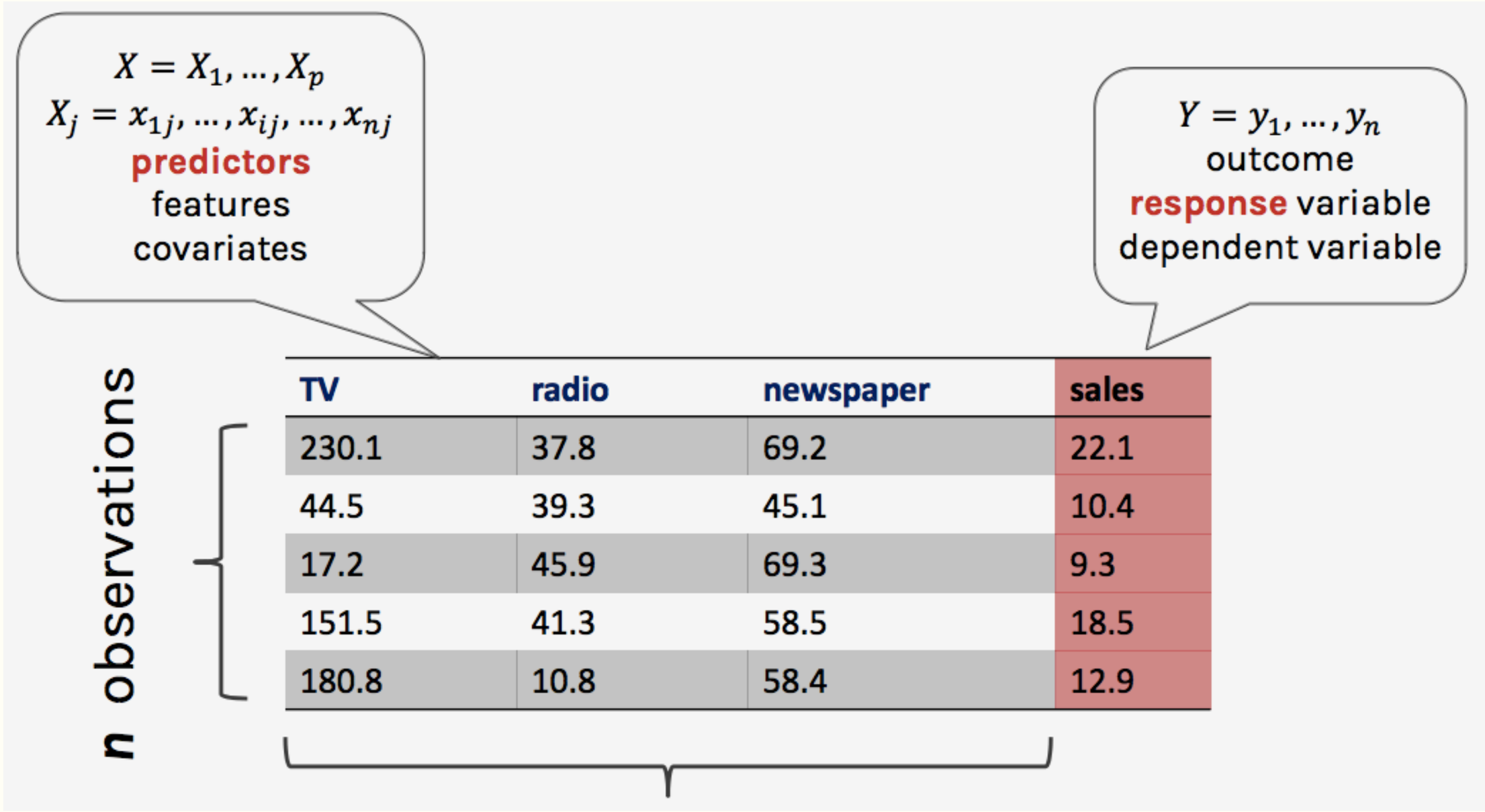
Regression

- how many dollars will you spend?
- what is your creditworthiness
- how many people will vote for Bernie t days before election
- use to predict probabilities for classification
- causal modeling in econometrics



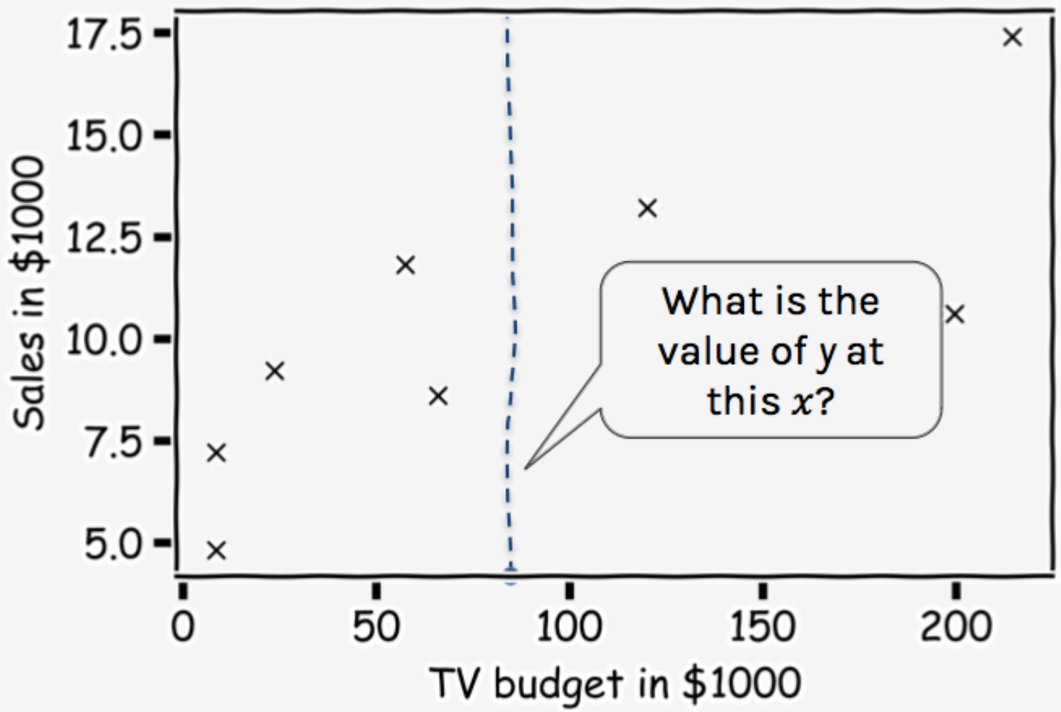
Dataset: Sales and Ad spending

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9


$X = X_1, \ldots, X_p$
$X_j = x_{1j}, \ldots, x_{ij}, \ldots, x_{nj}$
predictors
features
covariates
$Y = y_1, \ldots, y_n$
outcome
response variable
dependent variable
n observations
TV
radio
newspaper
sales
230.1
37.8
69.2
22.1
44.5
39.3
45.1
10.4
17.2
45.9
69.3
9.3
151.5
41.3
58.5
18.5
180.8
10.8
58.4
12.9

Two Questions

How do we find $\hat{f}(x)$?



True vs Statistical Model

We will assume that the measured response variable, y, relates to the predictors, x, through some unknown function expressed generally as:

$$y = f(x) + \epsilon$$

Here, f is the unknown function expressing an underlying rule for relating y to x, and ϵ is a random amount (unrelated to x) that y differs from the rule $f(x)$.

In real life we never know the true generating model $f(x)$

The best we can do..is to estimate $f(x)$.

A statistical model is any algorithm that estimates f. We denote the estimated function as $\hat{f}$.

There are two reasons for this:

1. We have no idea about the true generating process. So the function we use (we'll sometimes call this g) may have no relation to the real function that generated the data.

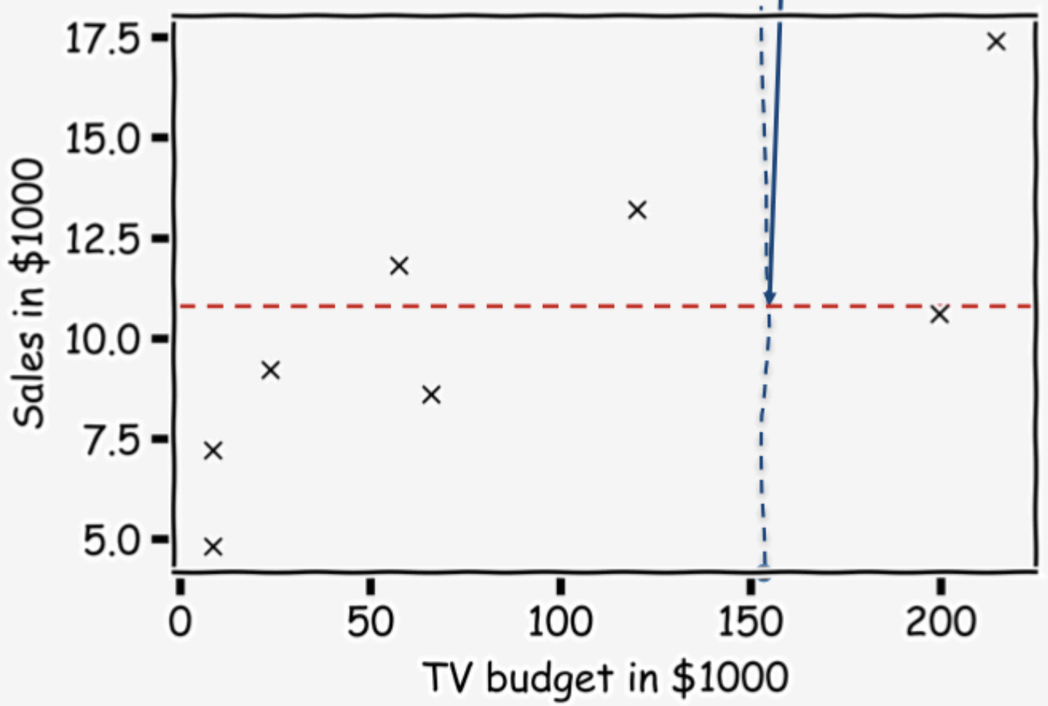
2. We only have access to a sample, with reality denying us access to the population.

We shall use the notation $\hat{f}$ or g to incorporate both of these considerations.

2. Fitting Models by minimizing Loss

Possibly the simplest model: the mean

Simple idea is to take the mean of all y's, $\hat{f}(x) = \frac{1}{n}\sum_1^n y_i$

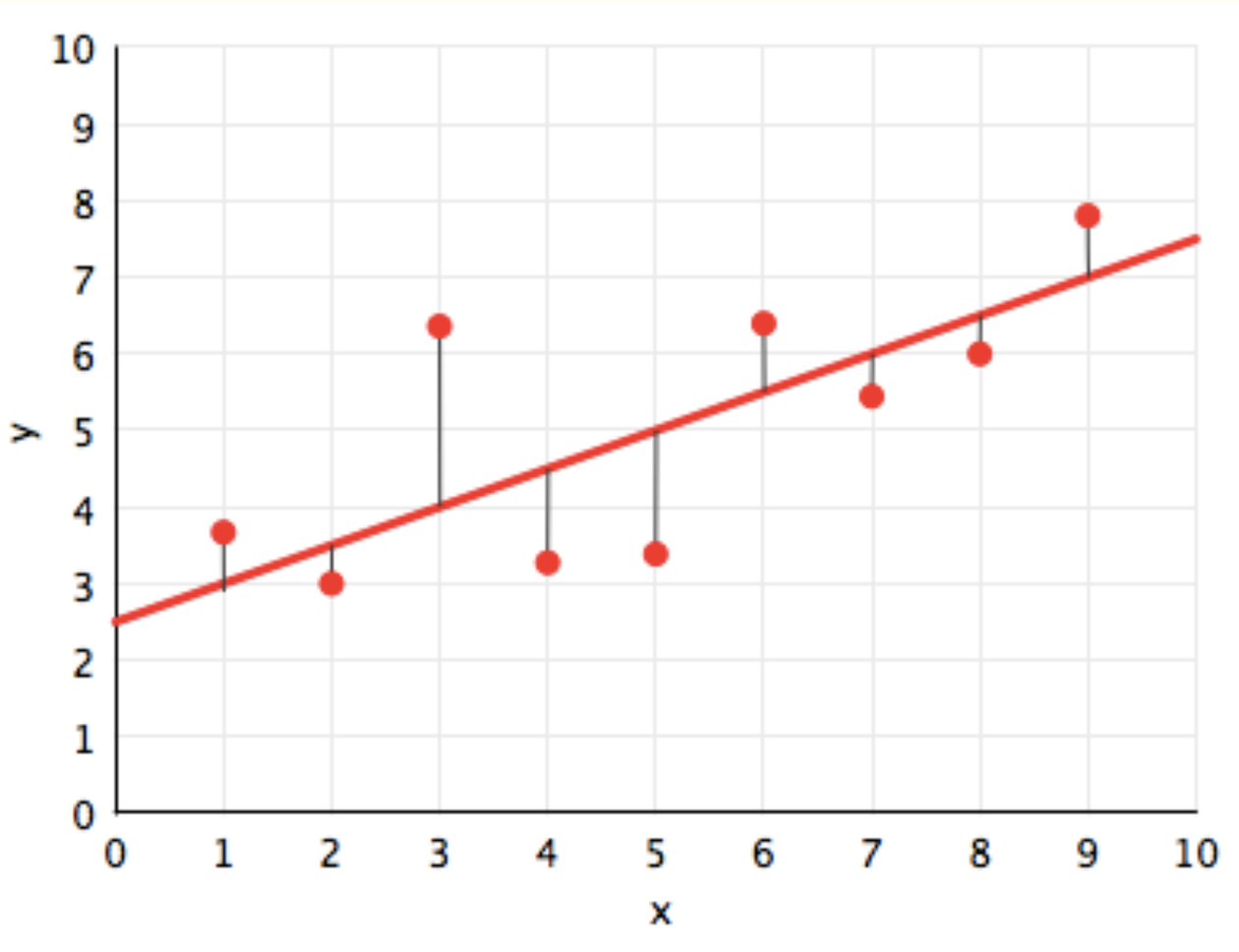


The next simplest: fit a straight line

How? Use the **Mean Squared Error**:

$$MSE = \frac{1}{N}\sum_i (\hat{y}_i - y_i)^2$$

Minimize this with respect to the *parameters*. (Here the intercept and slope)



What kind of loss is this?

For $\hat{y} = a + bx$, the loss is :

$$MSE = \frac{1}{N}\sum_{i}(a + bx_i - y_i)^2$$

This is quadratic and thus convex (bowl shaped).

So you are guaranteed to get to the bottom of the bowl! How?

Gradient Descent

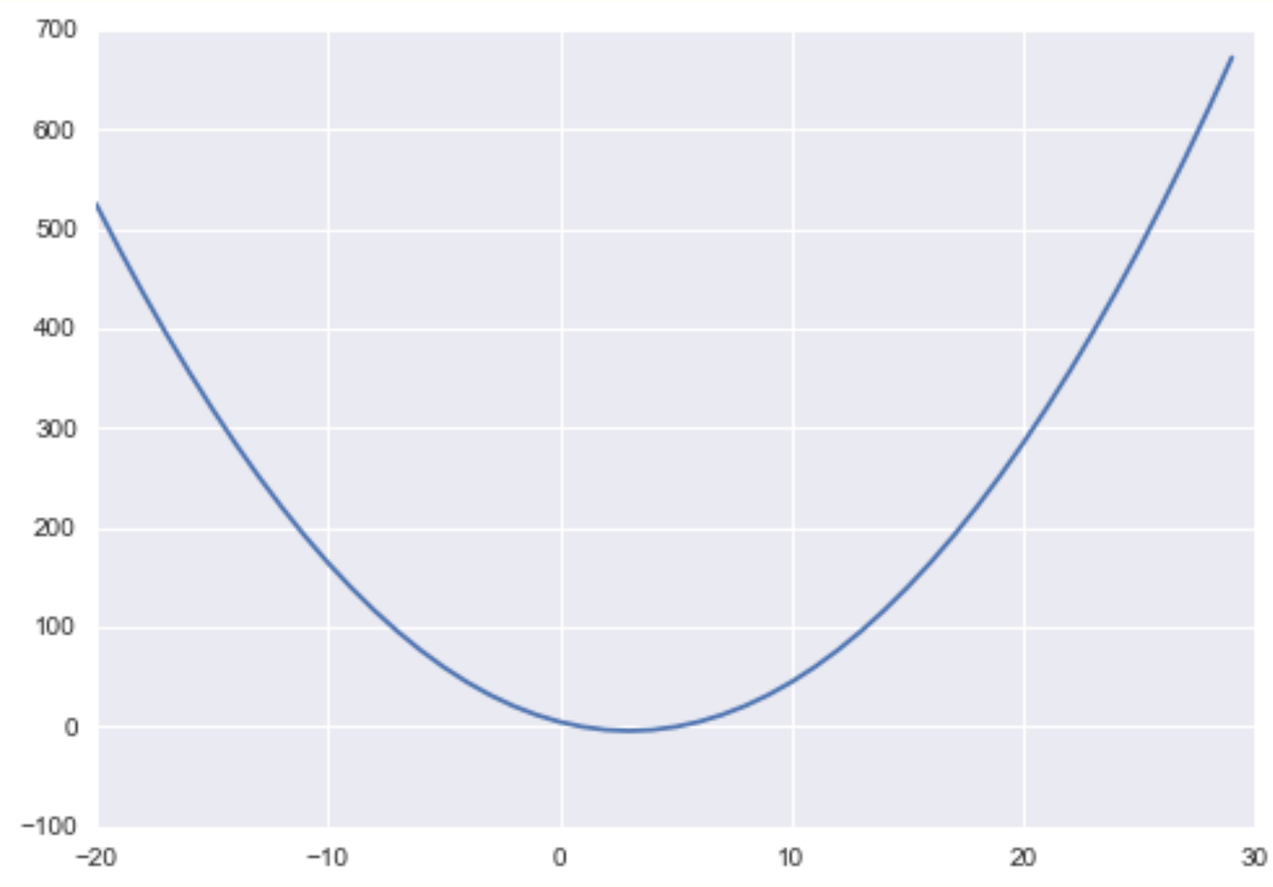
basically go opposite the direction of the derivative.

Consider the objective function:
$J(x) = x^2 - 6x + 5$

Its derivative or **gradient** is $2x - 6$. This is positive for $x > 3$. Negative below.

Now do:

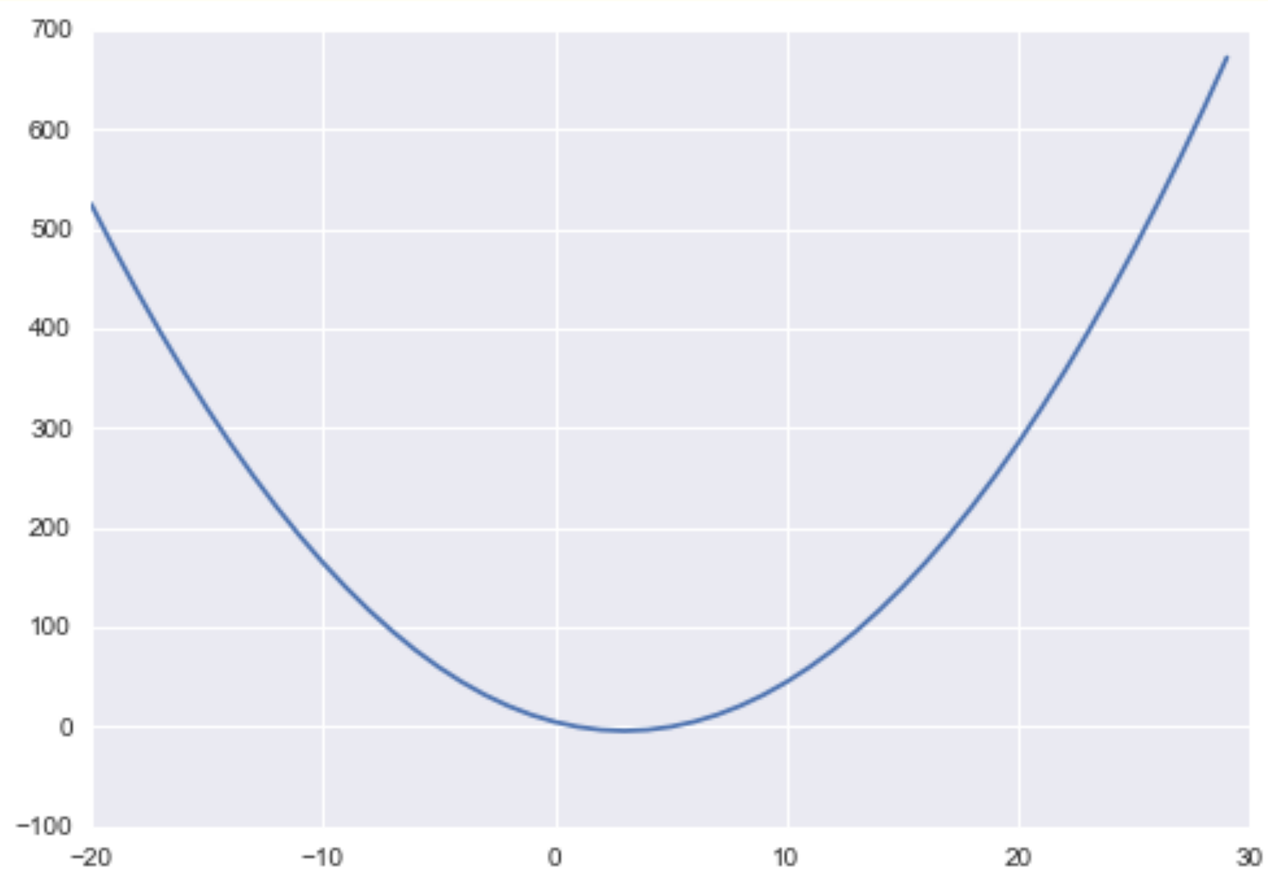
```
gradient = fprime(old_x)
move = gradient * step
current_x = old_x - move
```



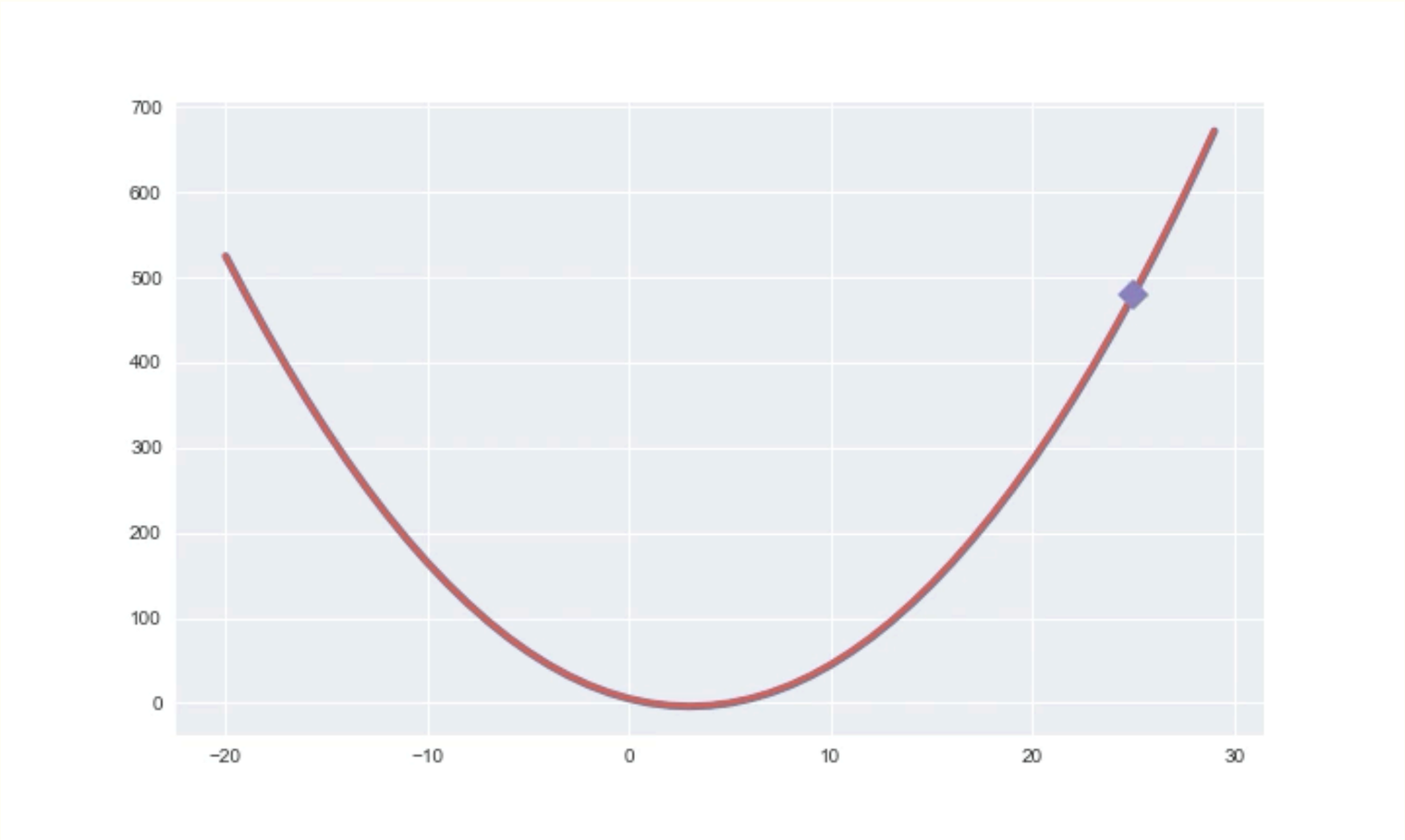
How do you go?

- For $x > 3$, $step \times (2x - 6)$ is positive, and so you go to smaller (leftward) x by $step \times (2x - 6)$.
- The moment $x < 3$, you are on the other side of the parabola, the derivative is negative, and so you go rightward by $step \times (2x - 6)$.
- How much do you go? Depends on step size.

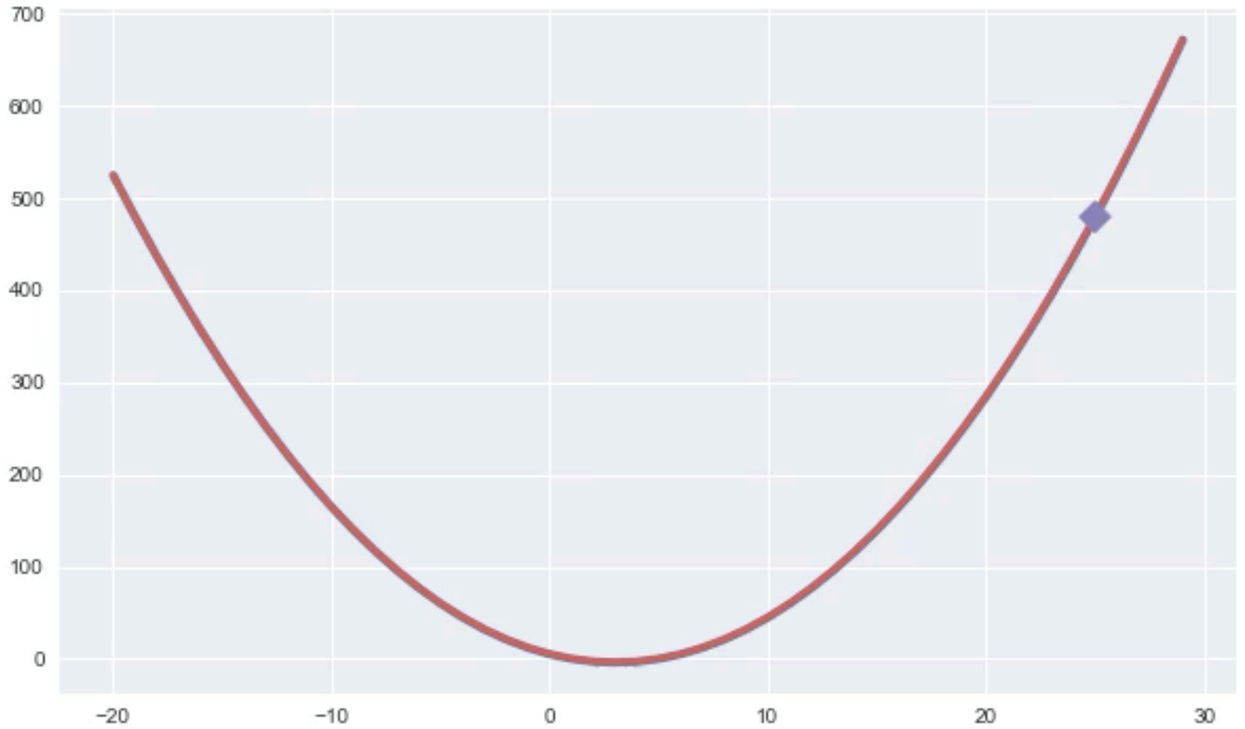
```
move = (2x - 6) * step
current_x = old_x - move
```



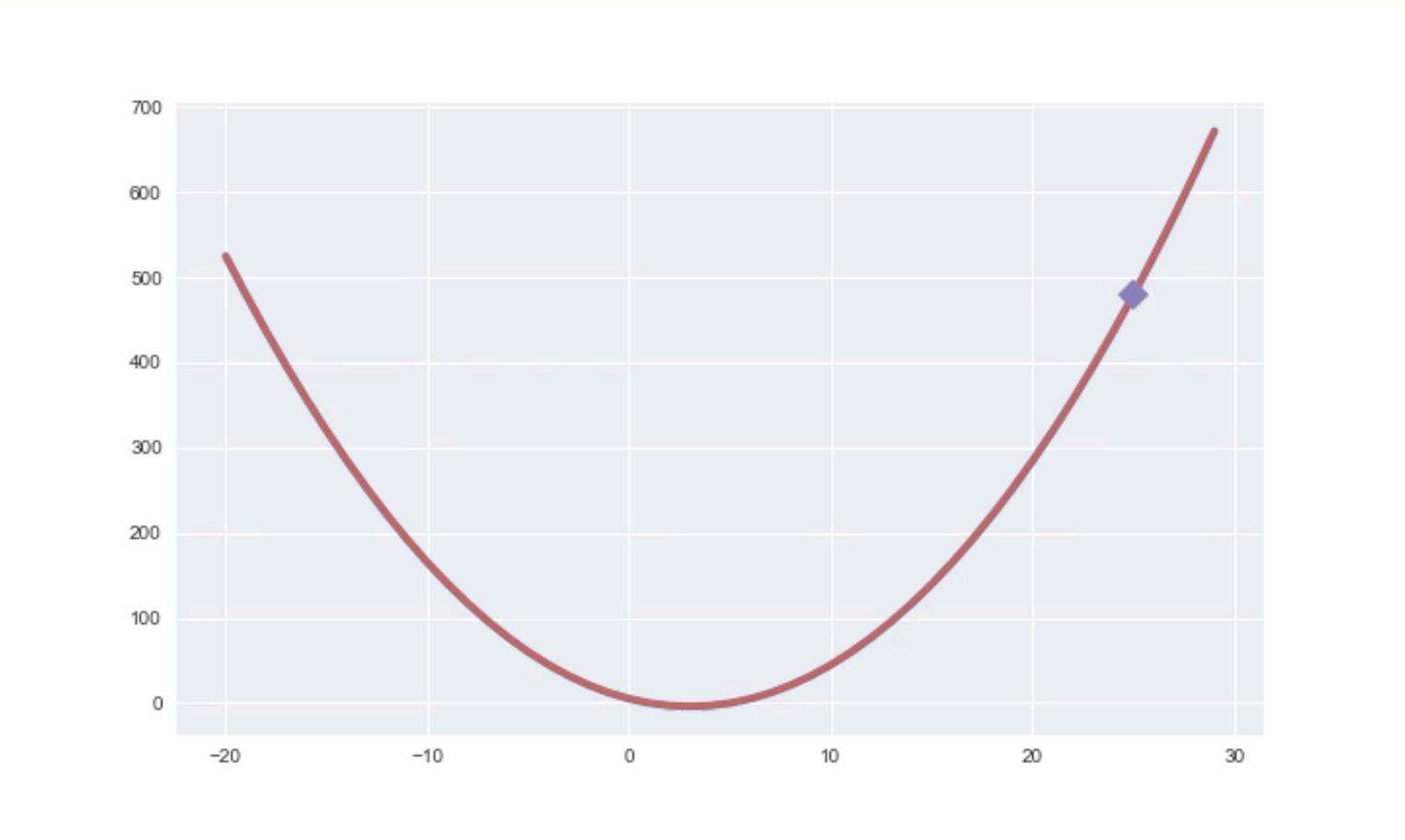
too big step size



too small step size



good step size



Gradient Descent

$$\theta := \theta - \eta \nabla_\theta J(\theta) = \theta - \eta \sum_{i=1}^{m} \nabla J_i(\theta)$$

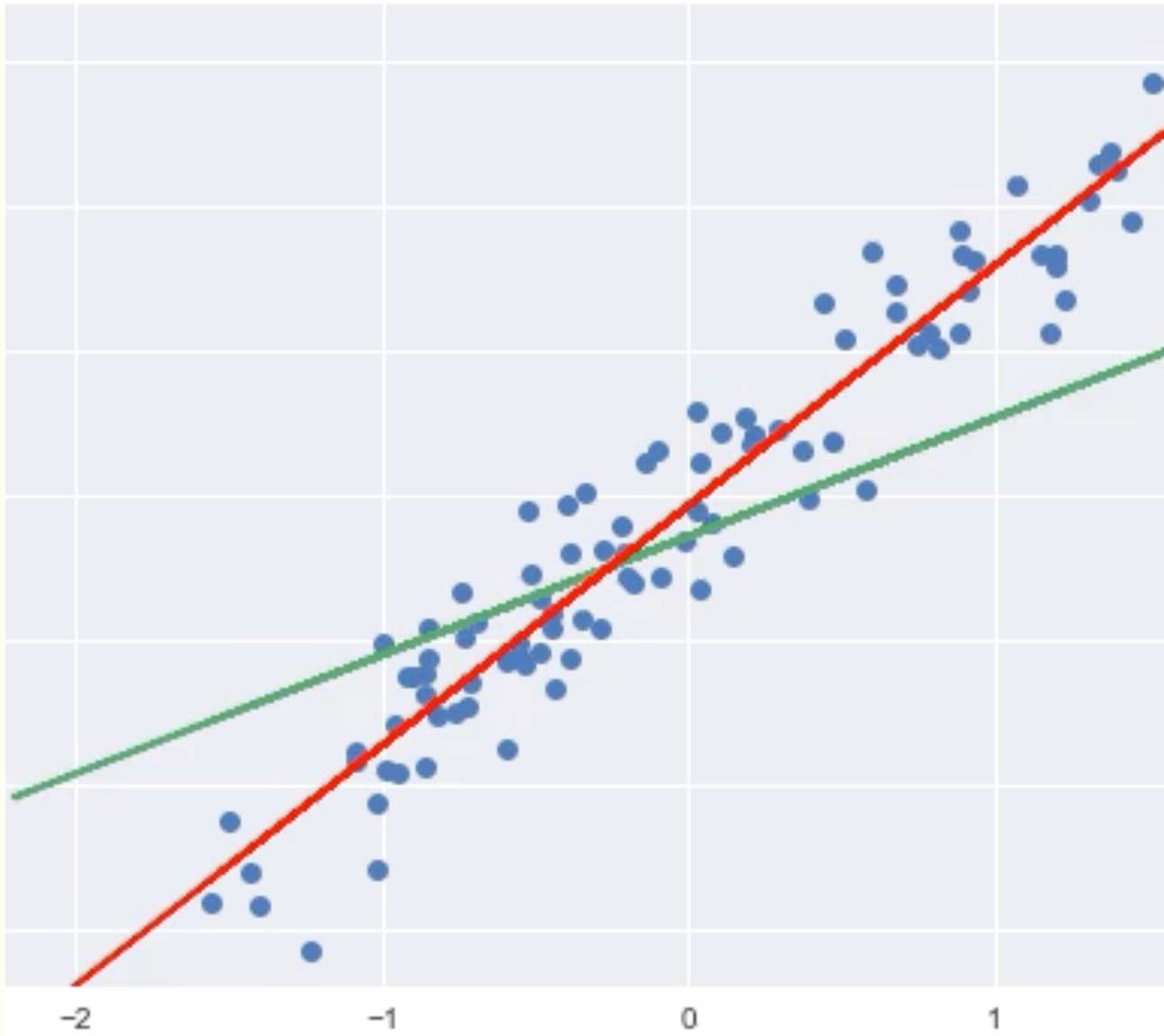
where η is the learning rate.

ENTIRE DATASET NEEDED

```
for i in range(n_epochs):
  params_grad = evaluate_gradient(loss_function, data, params)
  params = params - learning_rate * params_grad`
```

Linear Regression: Gradient Descent

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - f_\theta(x^{(i)}))x_j^{(i)}$$



Stochastic Gradient Descent

$$\theta := \theta - \alpha \nabla_\theta J_i(\theta)$$

ONE POINT AT A TIME

For Linear Regression:

$$\theta_j := \theta_j + \alpha(y^{(i)} - f_\theta(x^{(i)}))x_j^{(i)}$$

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```

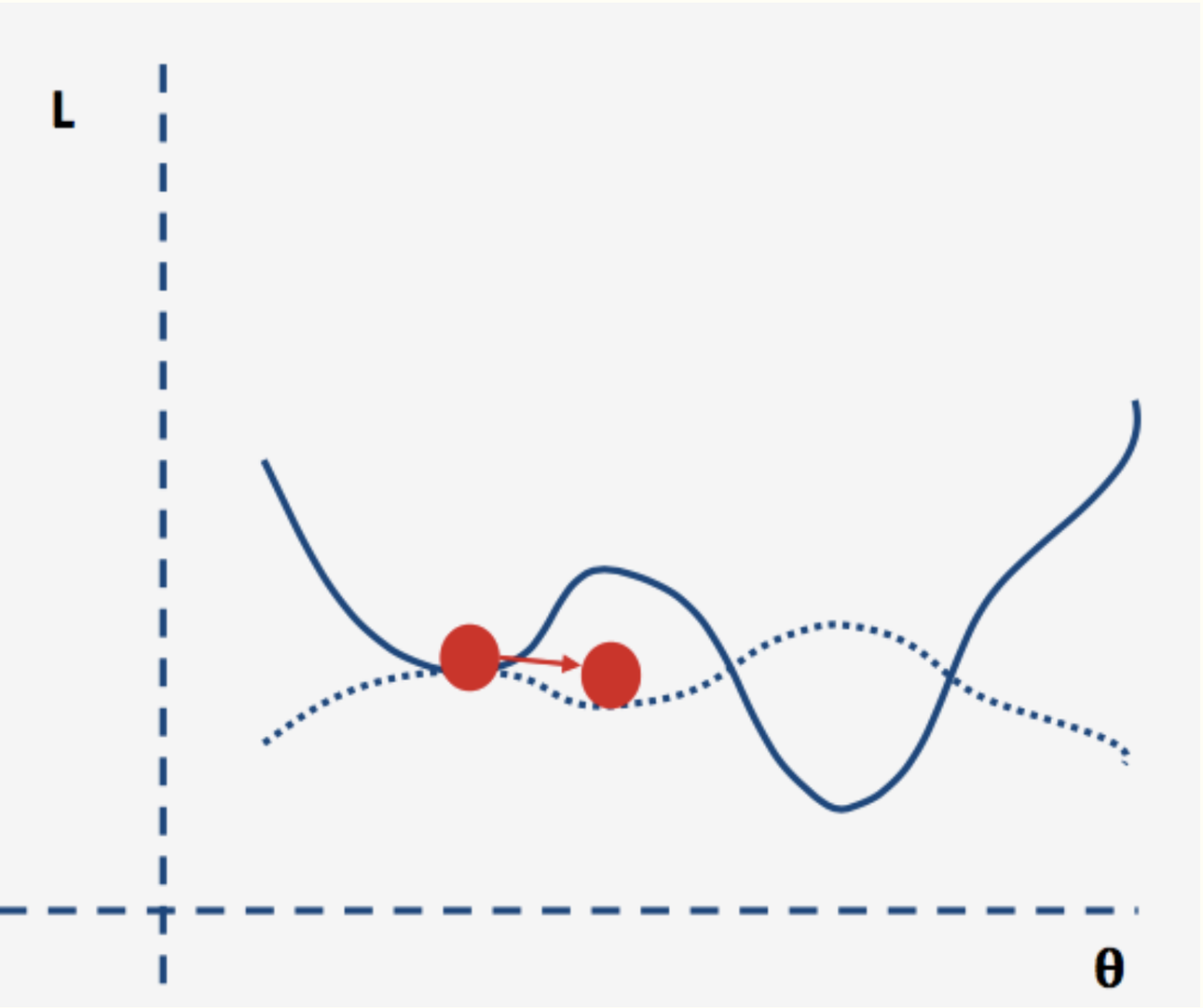
Mini-Batch SGD (the most used)

$$\theta := \theta - \eta \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

```
for i in range(mb_epochs):
  np.random.shuffle(data)
  for batch in get_batches(data, batch_size=50):
    params_grad = evaluate_gradient(loss_function, batch, params)
    params = params - learning_rate * params_grad
```

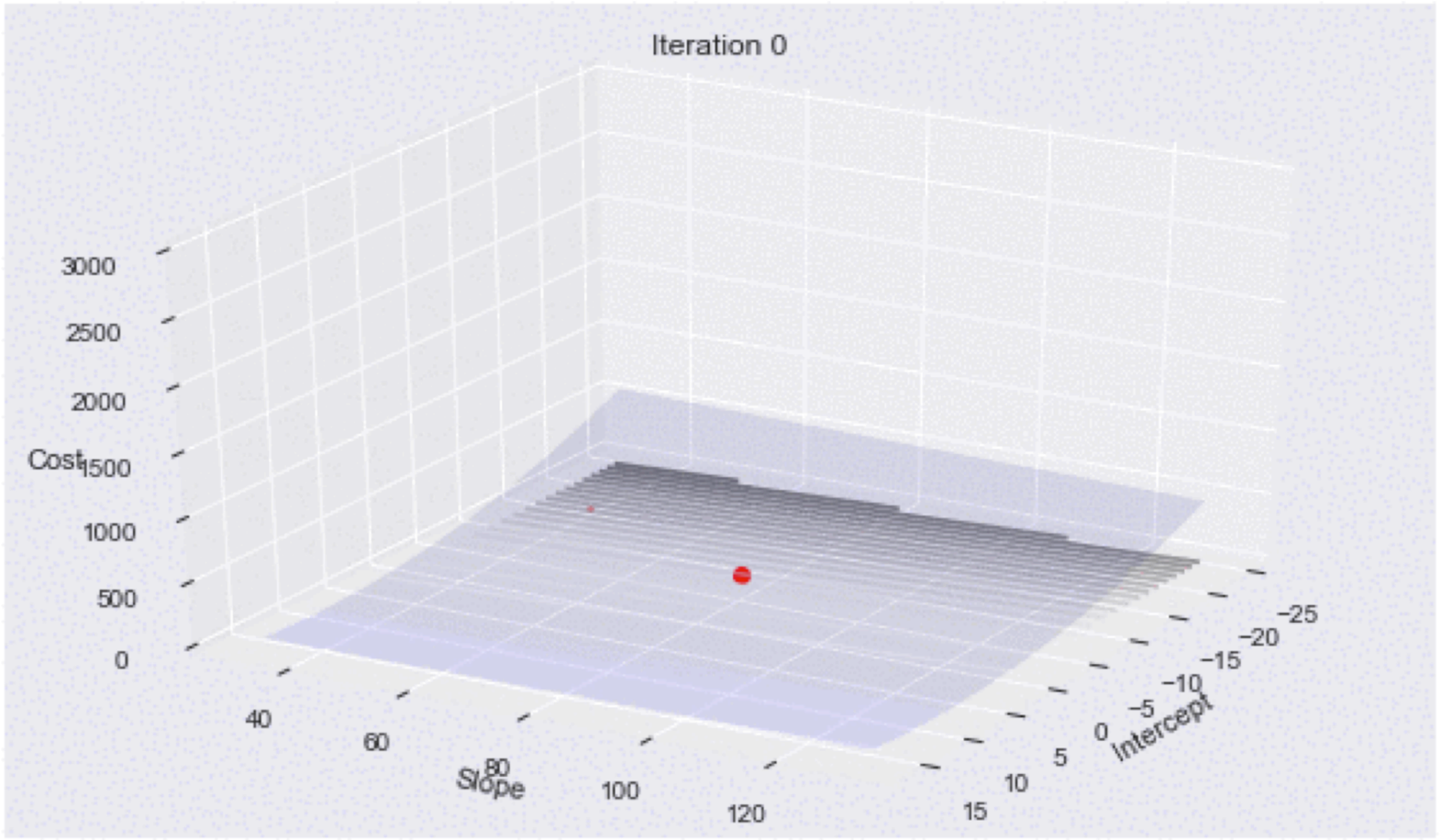
Mini-Batch: do some at a time

- the risk surface changes at each gradient calculation
- thus things are noisy
- cumulated risk is smoother, can be used to compare to SGD
- epochs are now the number of times you revisit the full dataset
- shuffle in-between to provide even more stochasticity



What does SGD and mini-batch SGD do?

- unlike for linear regression, loss functions may not be convex
- in this case gradient descent can get trapped
- by using only some data you chage the surface, and this may release you from the trap
- thus you can avoid shallow local minima


Iteration 0
Cost
3000
2500
2000
1500
1000
500
0
40
60
80
100
120
Slope
15
10
5
0
−5
−10
−15
−20
−25
Intercept

What value of the mean squared error is a good one?

The value of the MSE depends on the units of y.

So eliminate dependence on units of y.

$$R^2 = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (\bar{y} - y_i)^2}$$

Evaluation

$$R^2 = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (\bar{y} - y_i)^2}$$

- If our model is as good as the mean value $\bar{y}$, then $R^2 = 0$.
- If our model is perfect then $R^2 = 1$.
- R^2 can be negative if the model is worst than the average. This can happen when we evaluate the model on the test set.

3. Generation Story

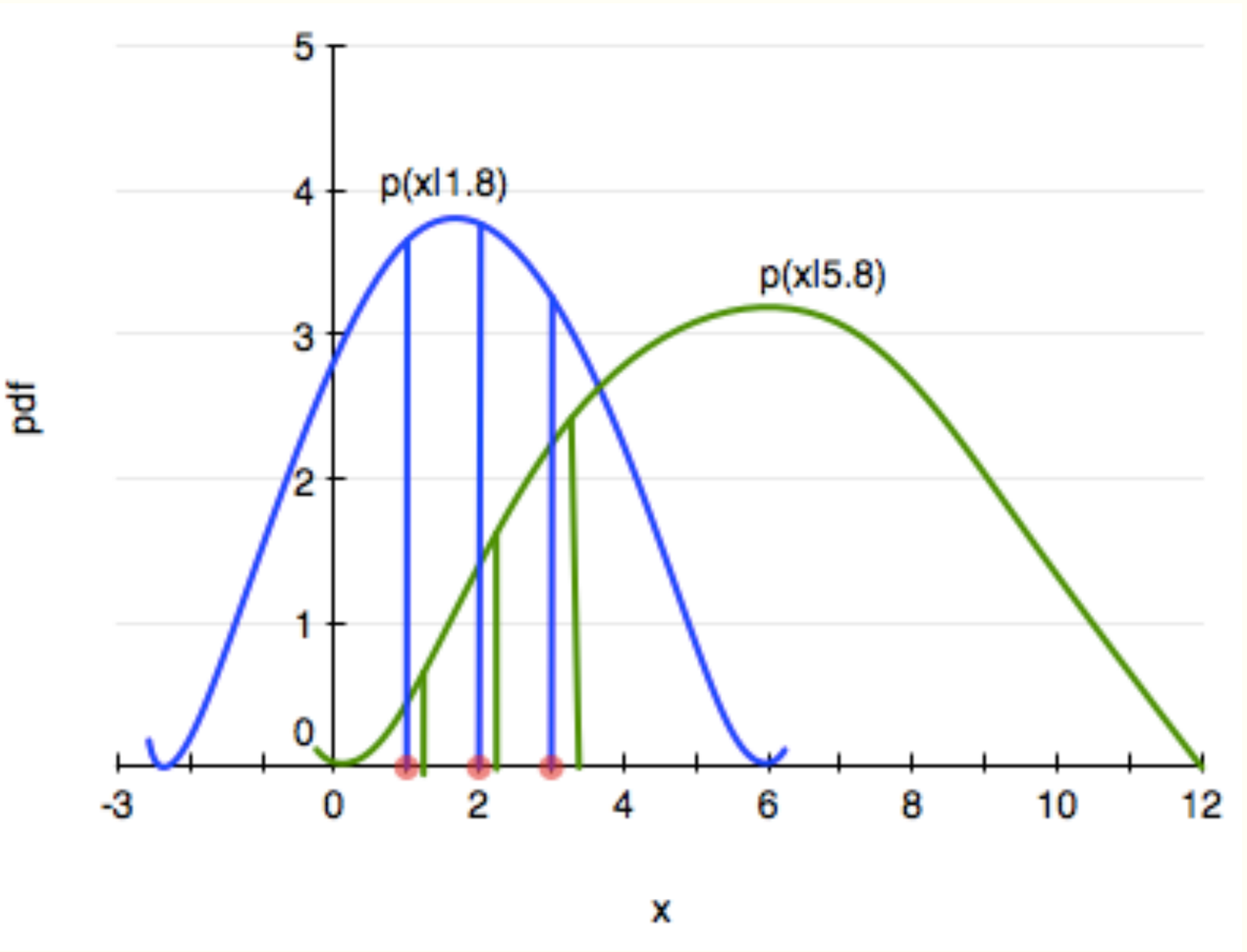
Likelihood

How likely it is to observe values $x_1, \ldots, x_n$ given the parameters λ?

$$L(\lambda) = \prod_{i=1}^{n} P(x_i|\lambda)$$

How likely are the observations if the model is true?

Maximum Likelihood estimation



We have 3 data points indicated in red.

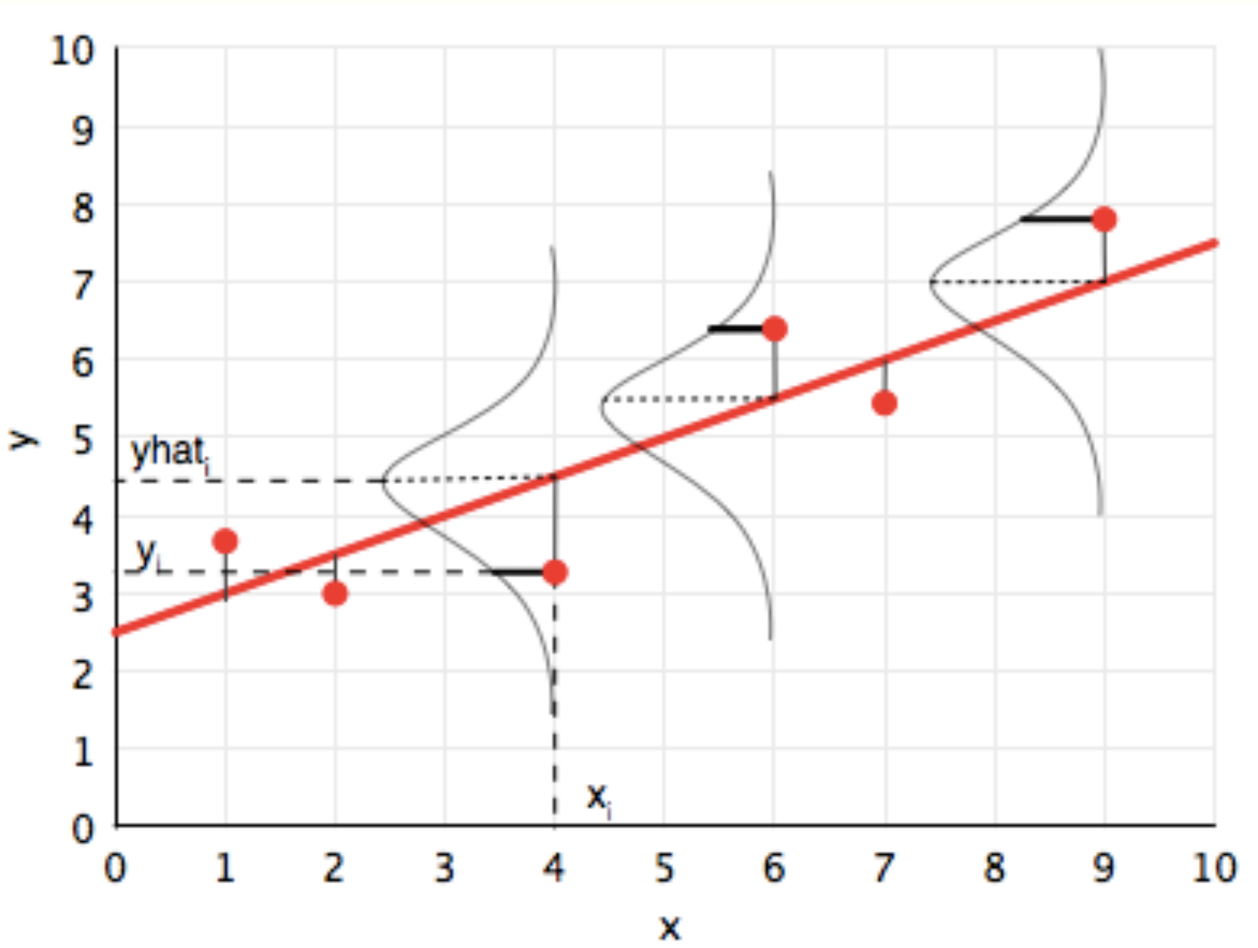
The blue and green "distributions" tell us the probability of a data point x ocurring under the blue and green models.

Which model is more likely? The blue? Or the green?

The Gaussian (bell curve) Assumption

Minimizing the mean squared error is equivalent to maximum-likelihood under a gaussian distribution.

Our generating story is this: the data are generated by taking the red line. Pick a x and it will give you a $\hat{f}(x) = a + bx$. Now imagine a gaussian distribution centered at that $\hat{f}$ and which gives you probabilities for different y at that x. Our data comes from this probability distribution, by making *draws* from it in proportion to the probability.



Gaussian Distribution assumption

Each y_i is gaussian distributed with mean $\mathbf{w} \cdot \mathbf{x}_i$ (the y predicted by the regression line) and variance σ^2:

$$y_i \sim N(\mathbf{w} \cdot \mathbf{x}_i, \sigma^2).$$

$$N(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(y-\mu)^2/2\sigma^2},$$

We can then write the likelihood:

$$\mathcal{L} = p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \sigma) = \prod_i p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}, \sigma)$$

$$\mathcal{L} = (2\pi\sigma^2)^{(-n/2)} e^{\frac{-1}{2\sigma^2}\sum_i (y_i - \mathbf{w}\cdot\mathbf{x}_i)^2}.$$

The log likelihood ℓ then is given by:

$$\ell = \frac{-n}{2} log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_i (y_i - \mathbf{w}\cdot\mathbf{x}_i)^2.$$

MINIMIZE

The negative log likelihood using gradient descent

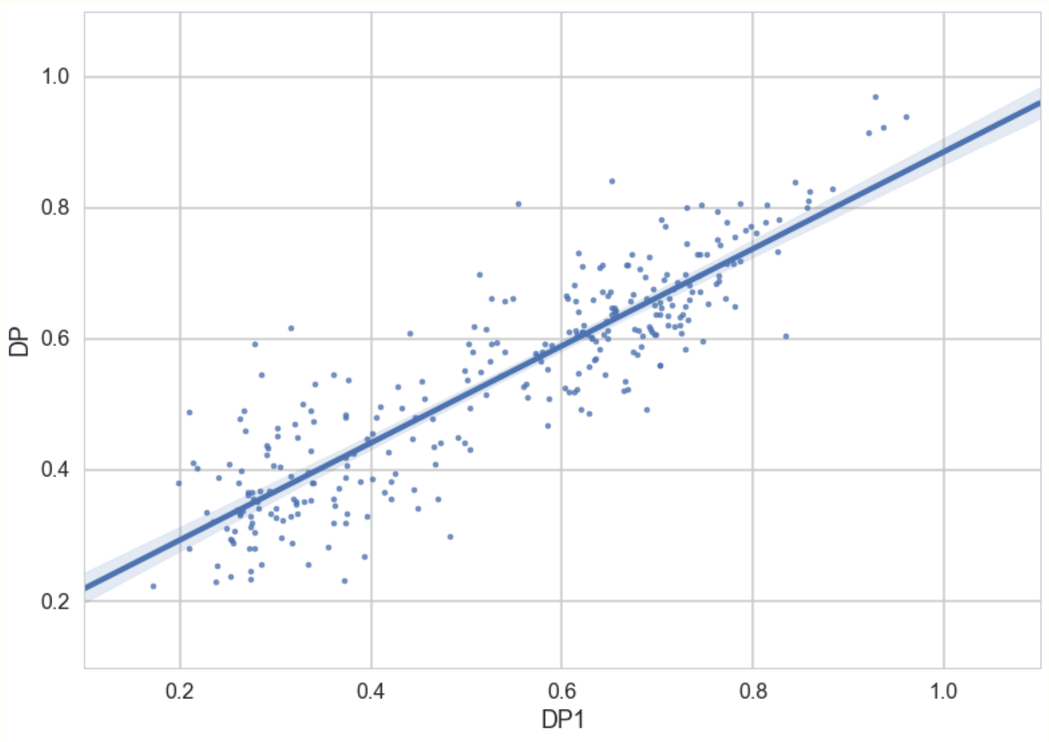
Minimize the Cost, Risk, Loss

Example: House Elections

Dep. Variable:	DP	**R-squared:**	0.806
Model:	OLS	**Adj. R-squared:**	0.804
Method:	Least Squares	**F-statistic:**	612.0
Date:	Tue, 13 Oct 2015	**Prob (F-statistic):**	1.04e-105
Time:	16:33:01	**Log-Likelihood:**	368.81
No. Observations:	298	**AIC:**	-731.6
Df Residuals:	295	**BIC:**	-720.5
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
Intercept	0.2326	0.020	11.503	0.000	0.193 0.272
DP1	0.5622	0.040	14.220	0.000	0.484 0.640
I	0.0429	0.008	5.333	0.000	0.027 0.059

Omnibus:	7.465	**Durbin-Watson:**	1.728
Prob(Omnibus):	0.024	**Jarque-Bera (JB):**	7.316
Skew:	0.374	**Prob(JB):**	0.0258
Kurtosis:	3.174	**Cond. No.**	13.1

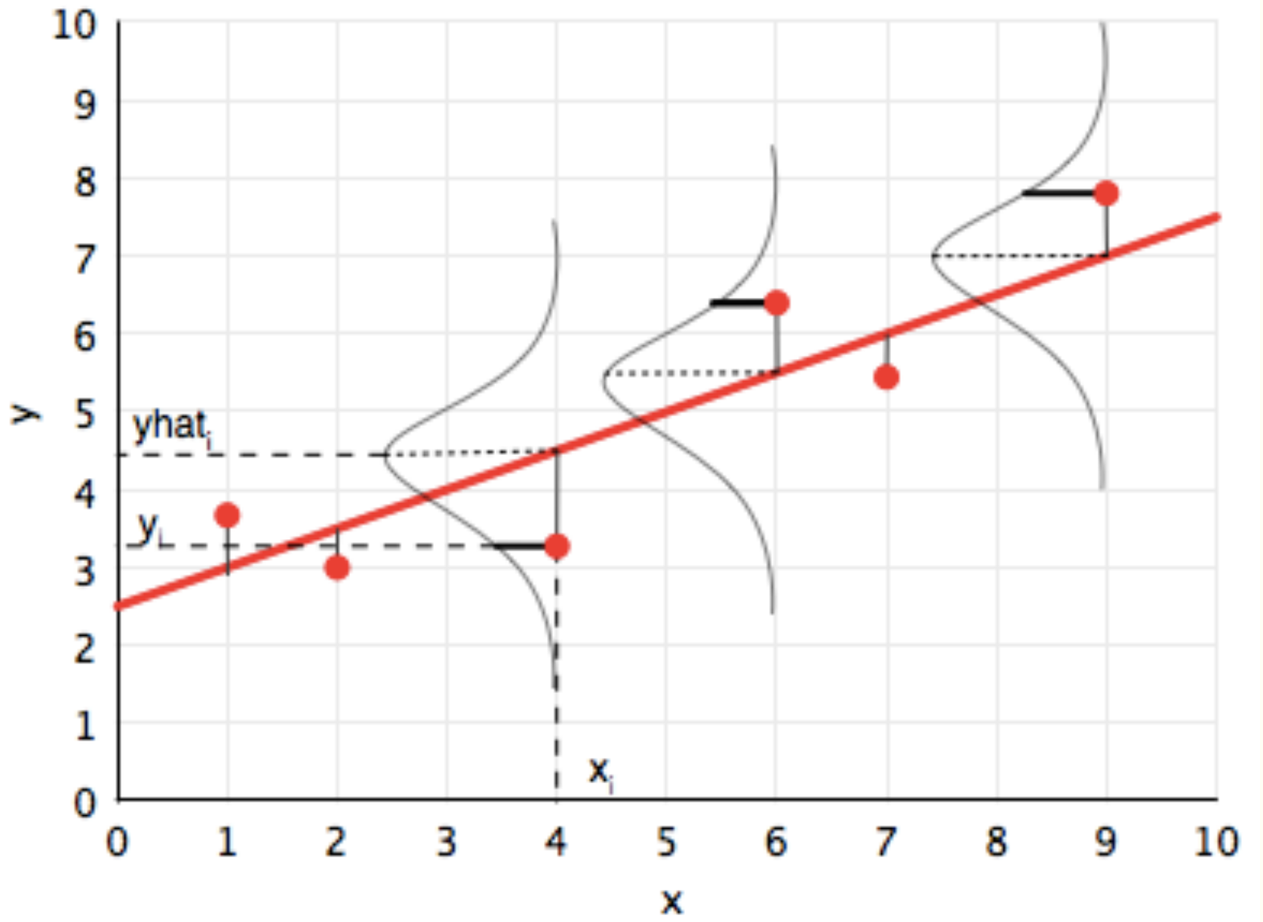


`Dem_Perc(t) ~ Dem_Perc(t-2) + I`, done in statsmodels

4. Noise and Sampling

Regression Noise Sources

- lack of knowledge of the true generating process
- sampling
- measurement error or irreducible error ϵ.
- lack of knowledge of x



We will first address ϵ

Predictions made with the "true" function f will not match observed values of y.

Because of ϵ, every time we measure the response y for a fixed value of x we will obtain a different observation, and hence a different estimate of the weights.

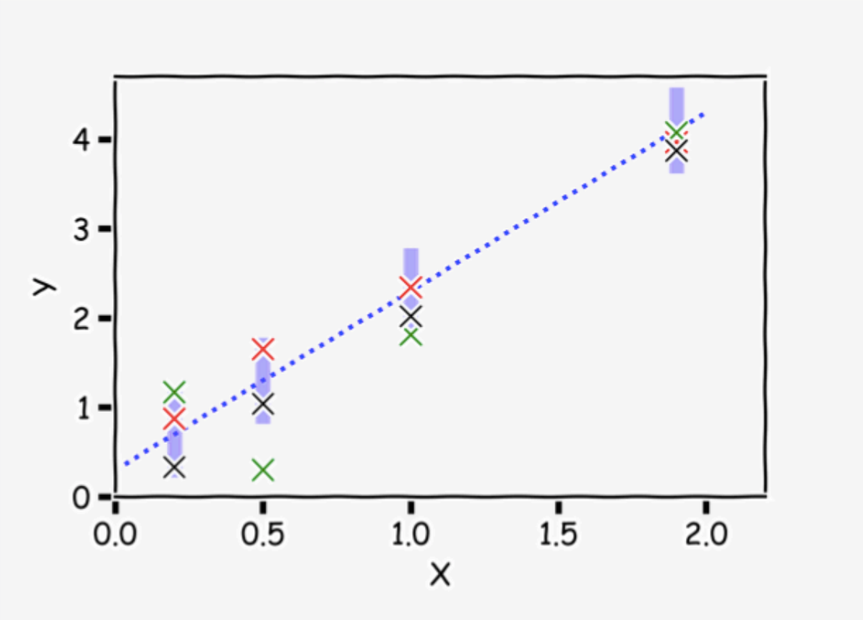
Indeed the MLE idea is that ϵ at any point x, comes from the gaussian in y-space at that point x.

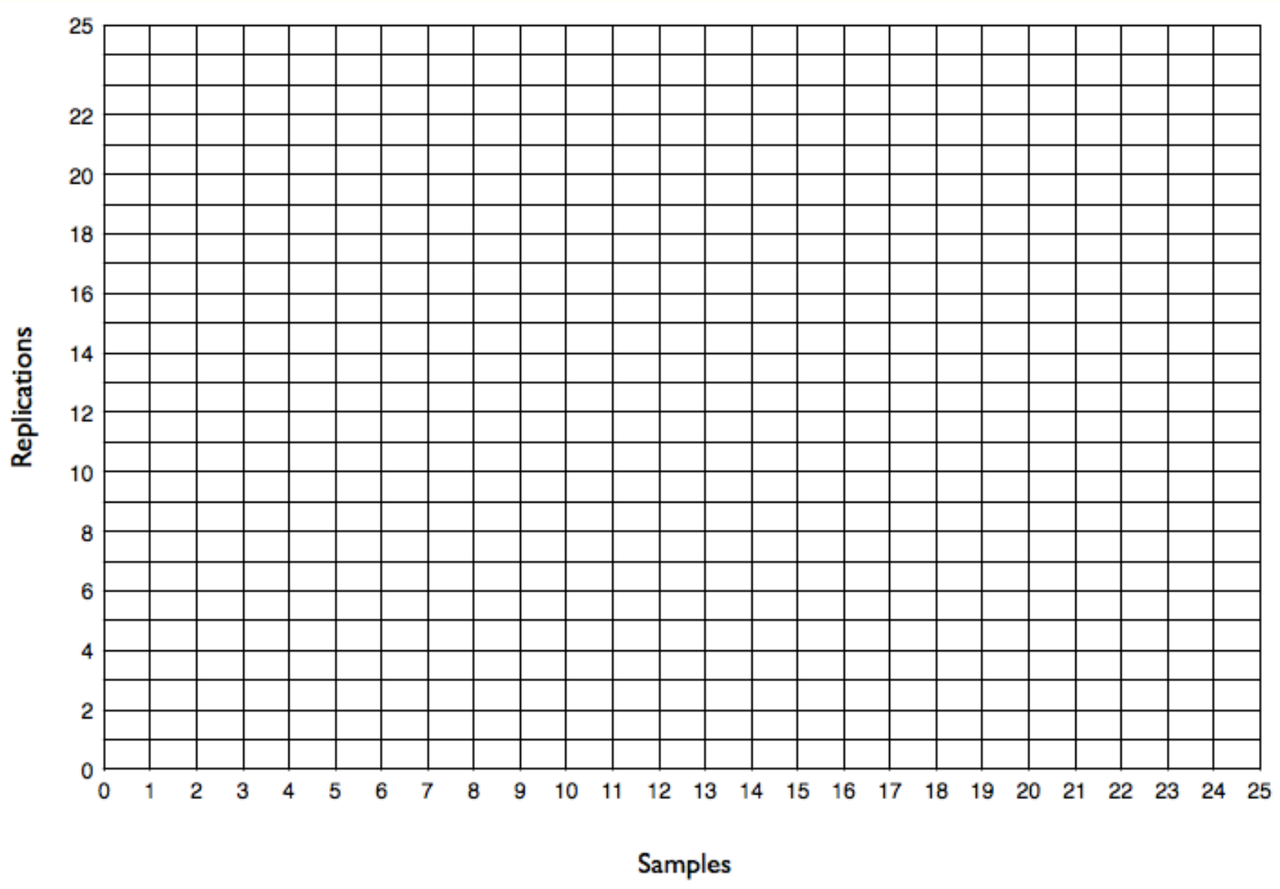
Samples from a population

Now, imagine that God gives you some M data sets or samples **drawn** from the population. This is a hallucination, a magic realism..

THIS USUALLY DOES NOT HAPPEN EXCEPT IN DREAMS!

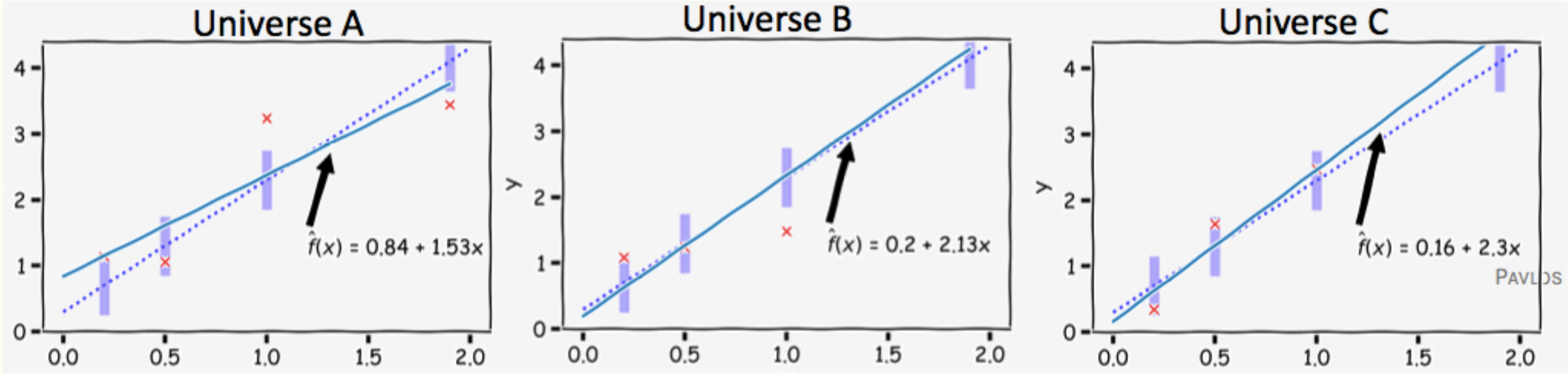
In real life we only get one sample!



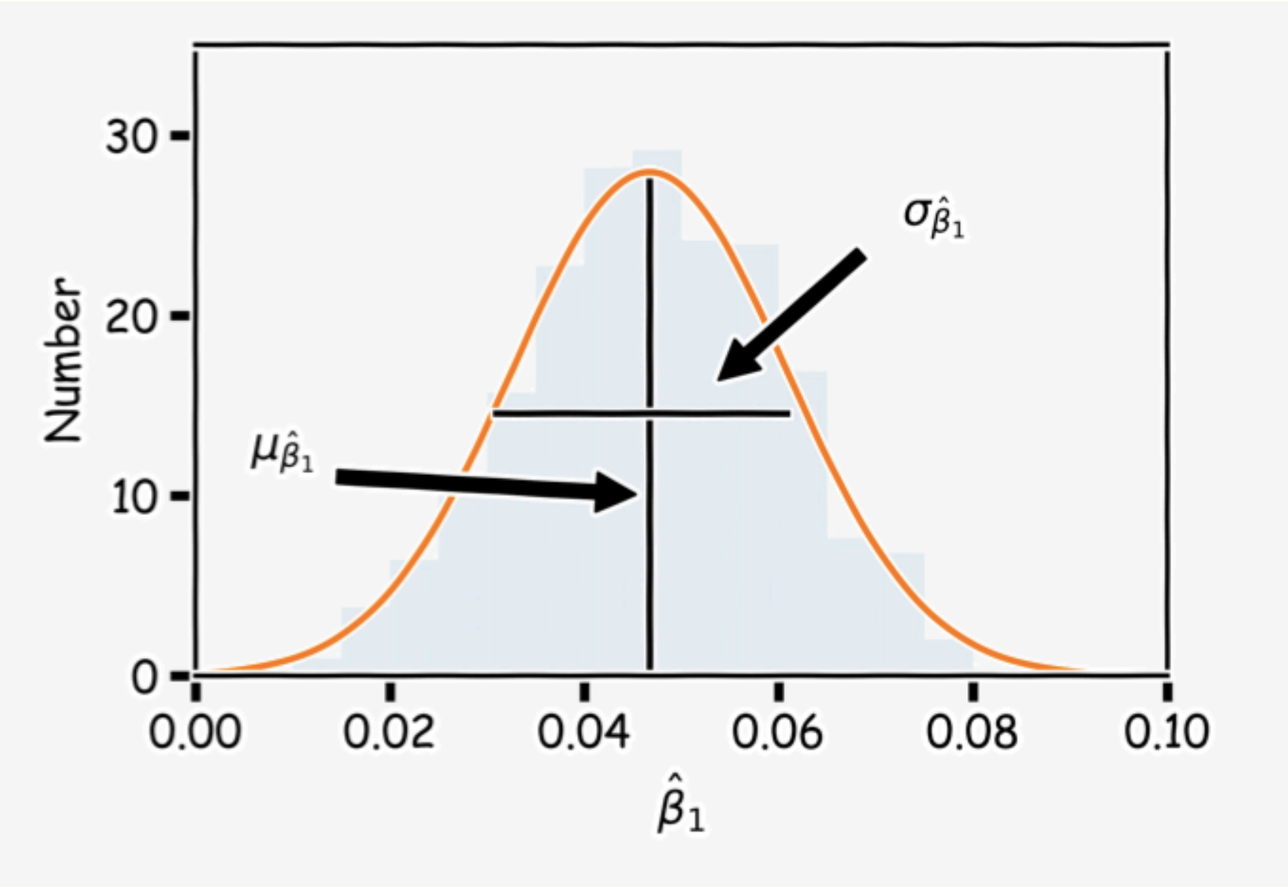


Multiple Fits

..and you can now find the regression on each such dataset (because you fit for the slope and intercept). So, we'd have M estimates of the slope and intercept.



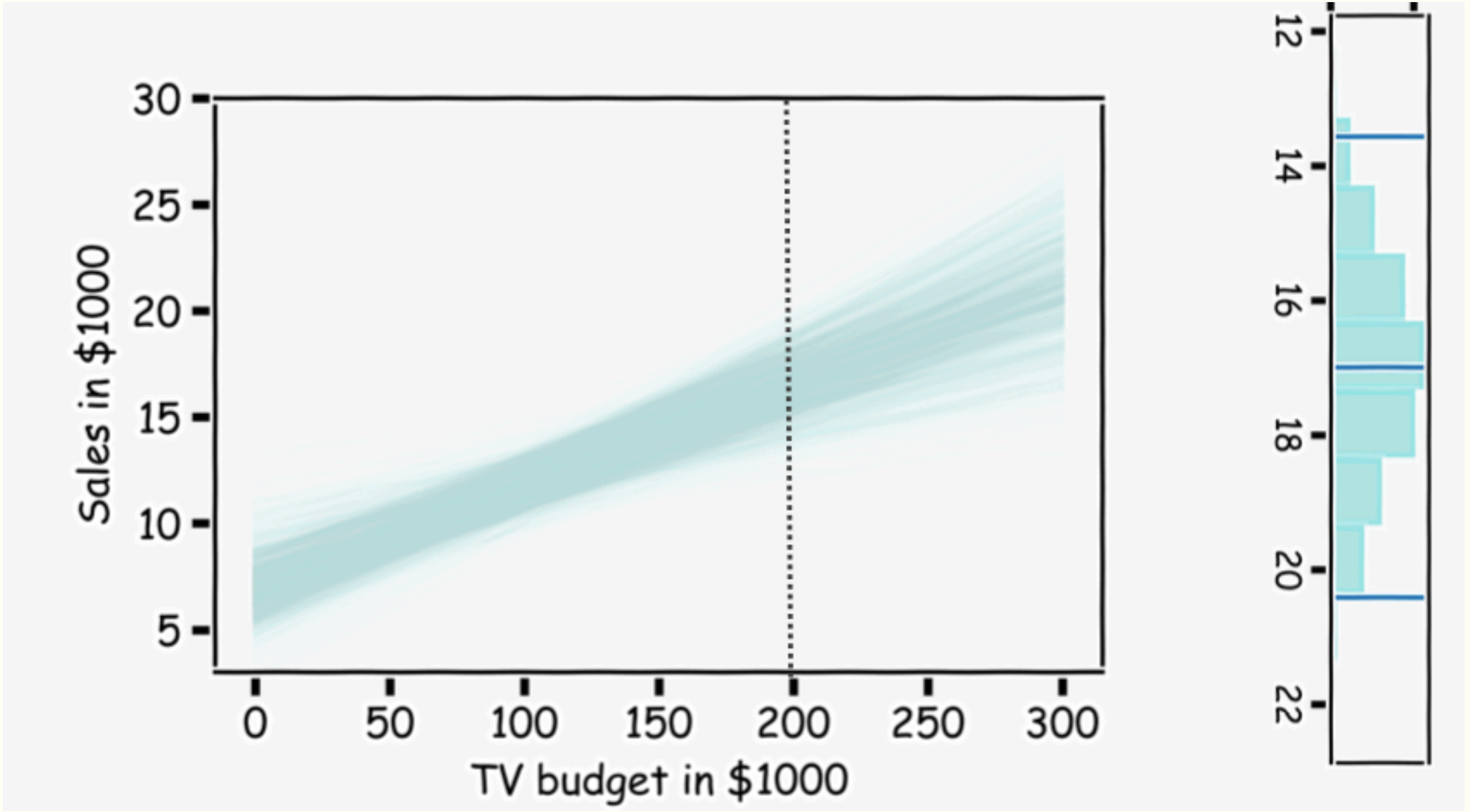
Sampling Distributions of parameters



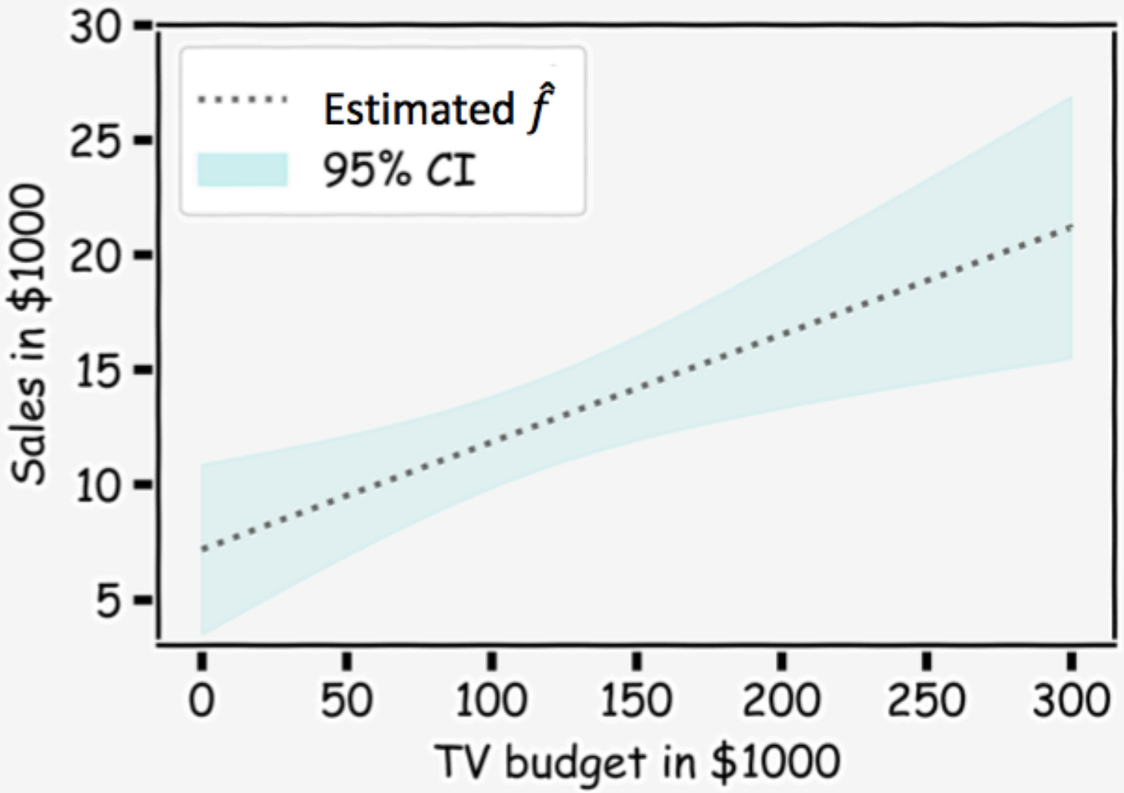
As we let $M \to \infty$, the distributions induced on the slope and intercept are the empirical **sampling distribution of the parameters**.

We can use these sampling distribution to get confidence intervals on the intercept and slope, and thus the range of allowed lines.

The variance of these distributions is called the **standard error**.

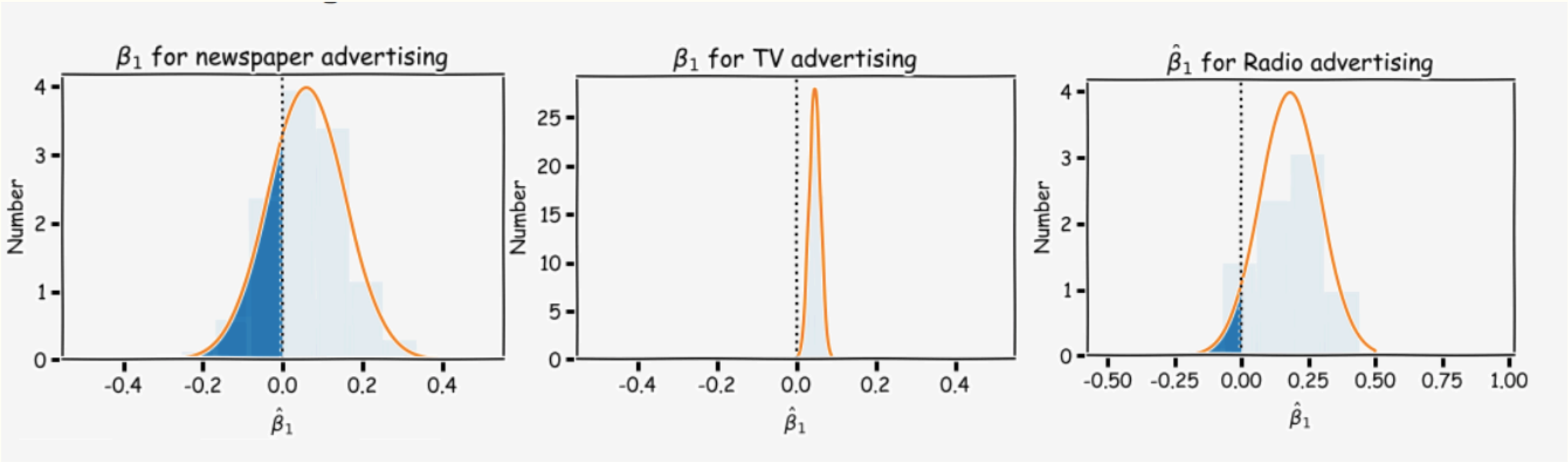

Sales in $1000
30
25
20
15
10
5
0
50
100
150
200
250
300
TV budget in $1000
12
14
16
18
20
22

Confidence Intervals on the "line"



- Each line is a rendering of $\mu = a + bx$, the mean value of the MLE Gaussian at each point x
- Thus the sampling distributions on the slope and intercept induce a sampling distribution on the lines
- And then the estimated $\hat{f}$ is taken to be the line with the mean parameters

Sampling Distributions and Significance



Which parameters are important?

You dont want the parameters in a regression to be 0.

So, in a sense, you want parameters to have their sampling distributions as far away from 0 as possible.

Is this enough? Its certainly evocative.

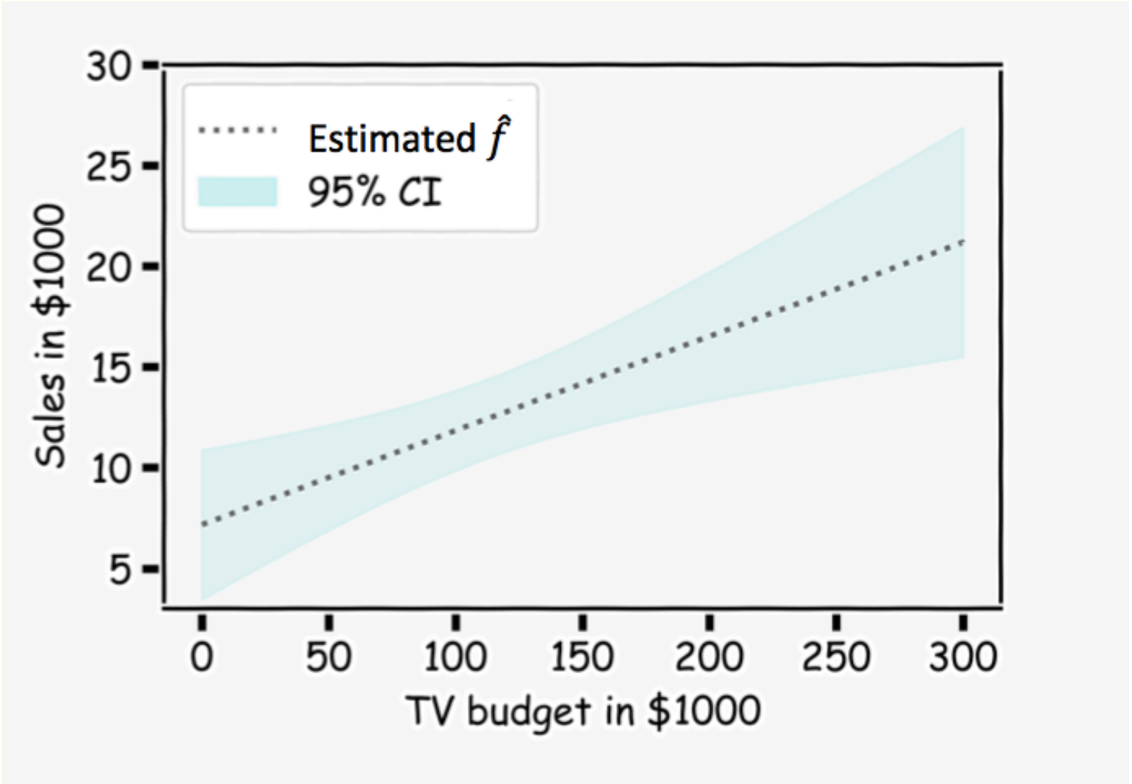
But we must consider the "Null Hypothesis": a given parameter has no effect. We can do this by re-permuting just that column

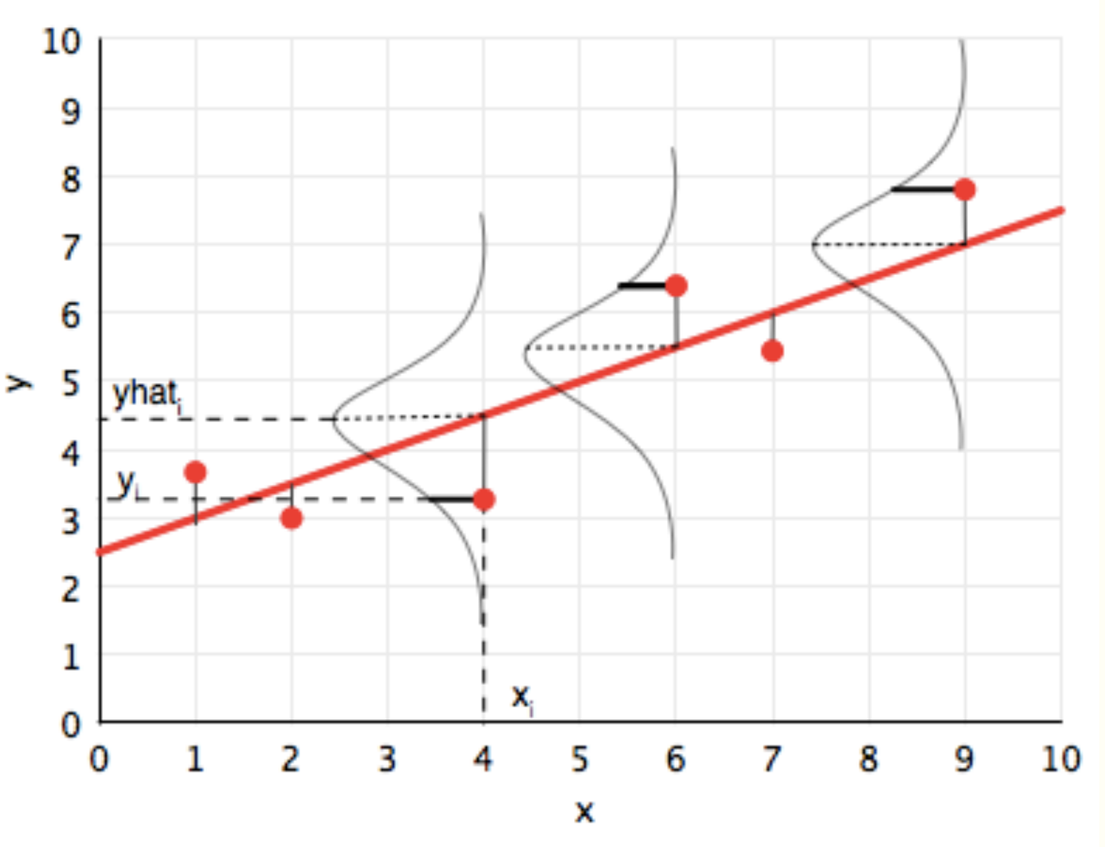
5. Prediction is more uncertain ..

than the mean

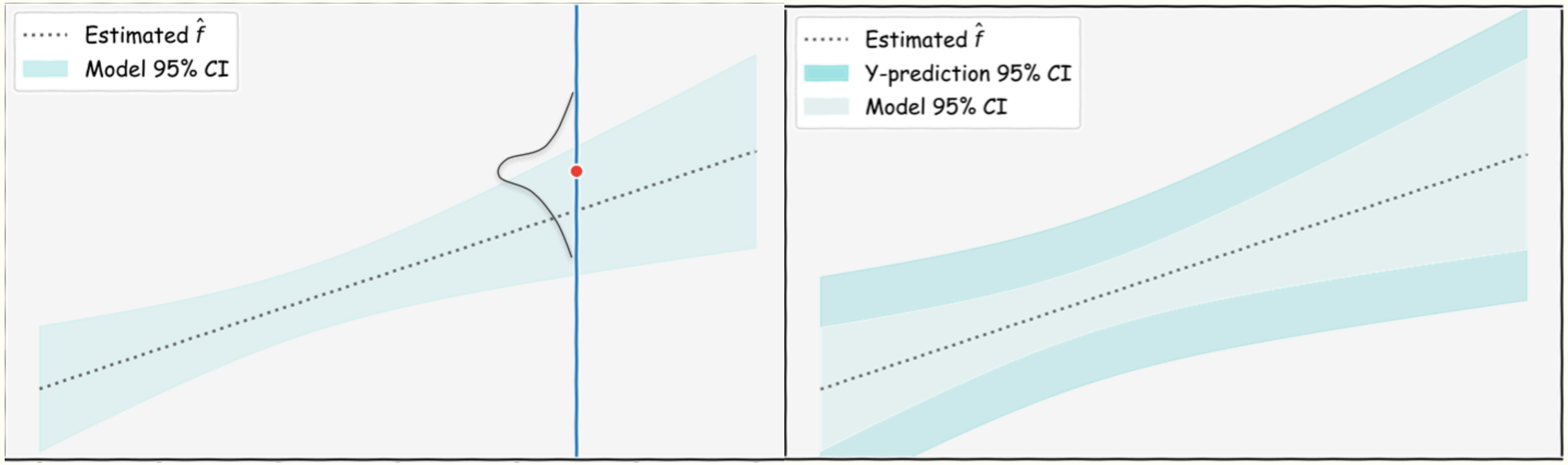
Prediction vs Possible Prediction

- In machine learning we do not care too much about the functional form of our prediction $\hat{y} = \hat{f}(x)$, as long as we predict "well"
- Remember however our origin story for the data: the measured y is assumed to have been a draw from a gaussian distribution at each x: this means that our prediction at an as yet not measured x should also be a draw from such a gaussian
- Still, we use the mean value of the gaussian as the value of the "prediction", but note that we can have many "predicted" data sets, all consistent with the original data we have


Estimated $\hat{f}$
95% CI
Sales in $1000
TV budget in $1000
30
25
20
15
10
5
0
50
100
150
200
250
300


yhat$_i$
y_i
x_i
y
x
0
1
2
3
4
5
6
7
8
9
10

Mean vs Prediction



Concepts to take away

- We see only one sample, and never know the "God Given" model. Thus we always make Hat estimates
- The simplest models are flat and sloped line. To fit these we use linear algebra or gradient descent.
- We use the R^2 to evaluate the models.
- Maximum Likelihood estimation or minimum loss estimation are used to find the best fit model
- Gradient Descent is one way to minimize the loss

Concepts to take away, part 2

- Noise in regression models comes from model-misspecification, measurement noise, and sampling
- sampling can be used to replicate most of these noise sources, and thus we can use multiple samples created or conjured (look up bootstrap) from a population to study the impacts of noise
- these samples can be used to understand the sampling variance of parameters and thus regression lines
- predictions are even more variant since our likelihood indicates a generative process involving sampling from a gaussian

What's next?

- We'll use Polynomial Regression to understand the concepts of model complexity and overfitting
- We'll also see the use of validation sets to figure the value of hyperparameters
- We'll learn how to fit models using `sklearn`.