Linear Regression
Overfitting, Ridge, Lasso

Sujay Sanghavi

Do we have enough training samples ?

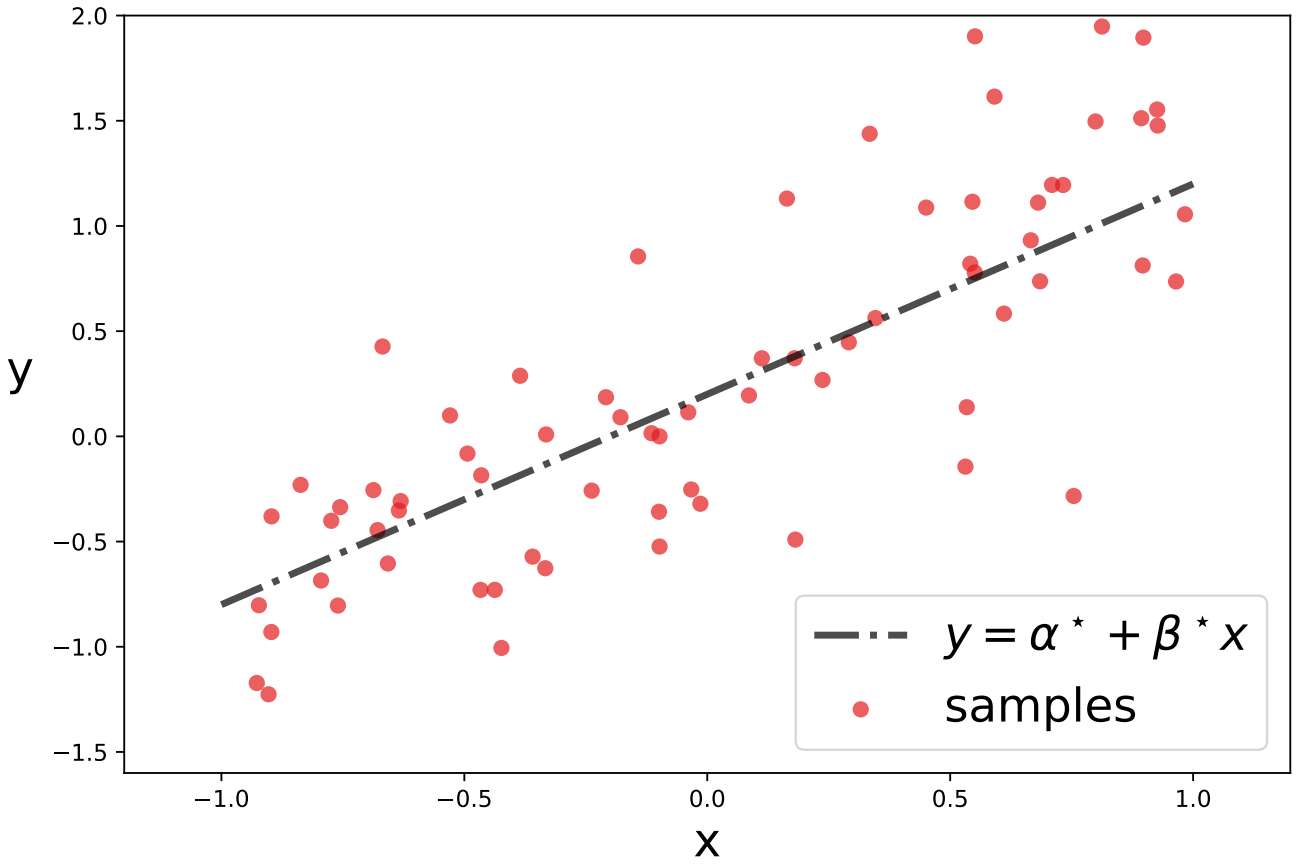
Q: I learnt a model, and it seems to have decent R^2. Does this mean it is great ?

A: Not necessarily ! It also depends on the number of training samples used to learn it.

e.g.: learning a $\widehat{\beta}$ to predict scalar y from scalar x and just two training samples $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})$

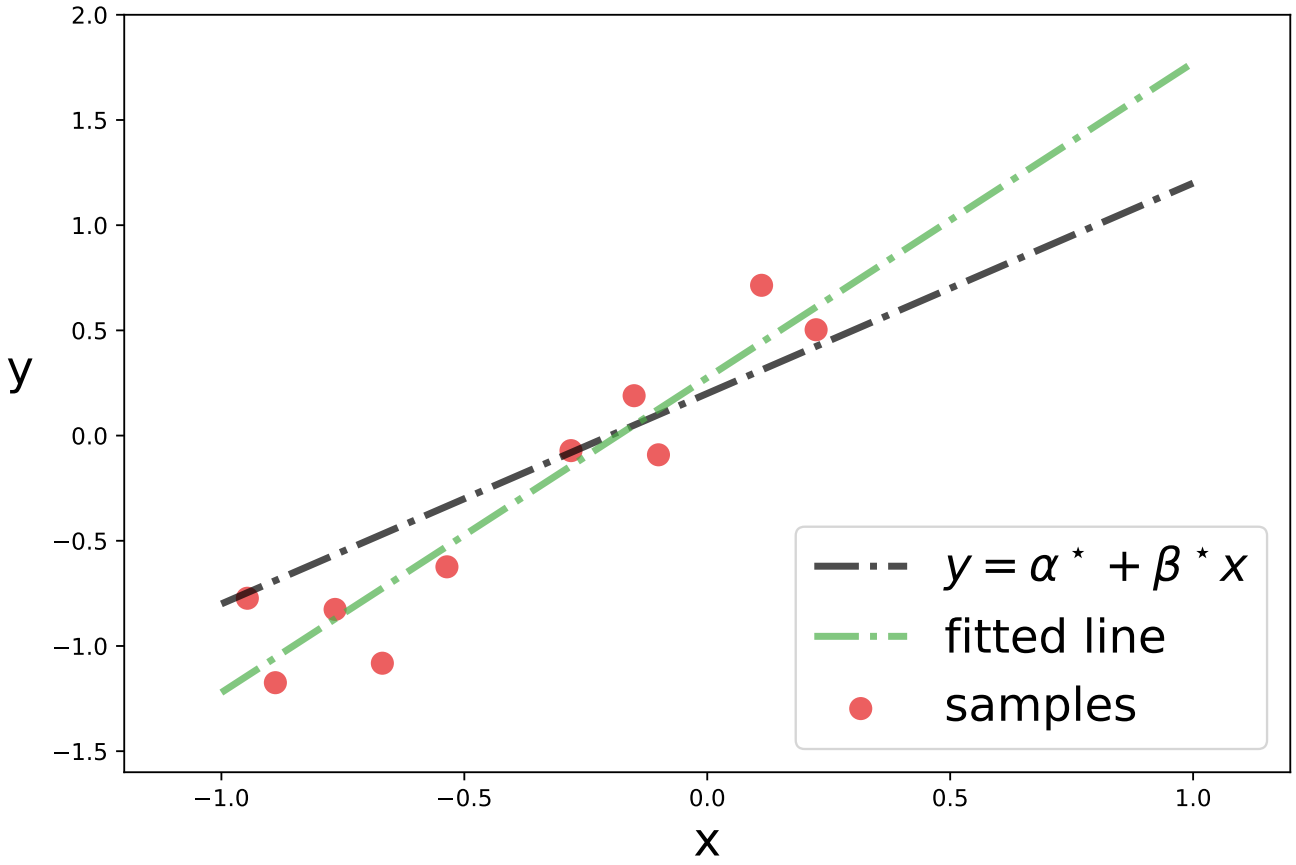
Lets look at this more closely ...

Do we have enough training samples ?



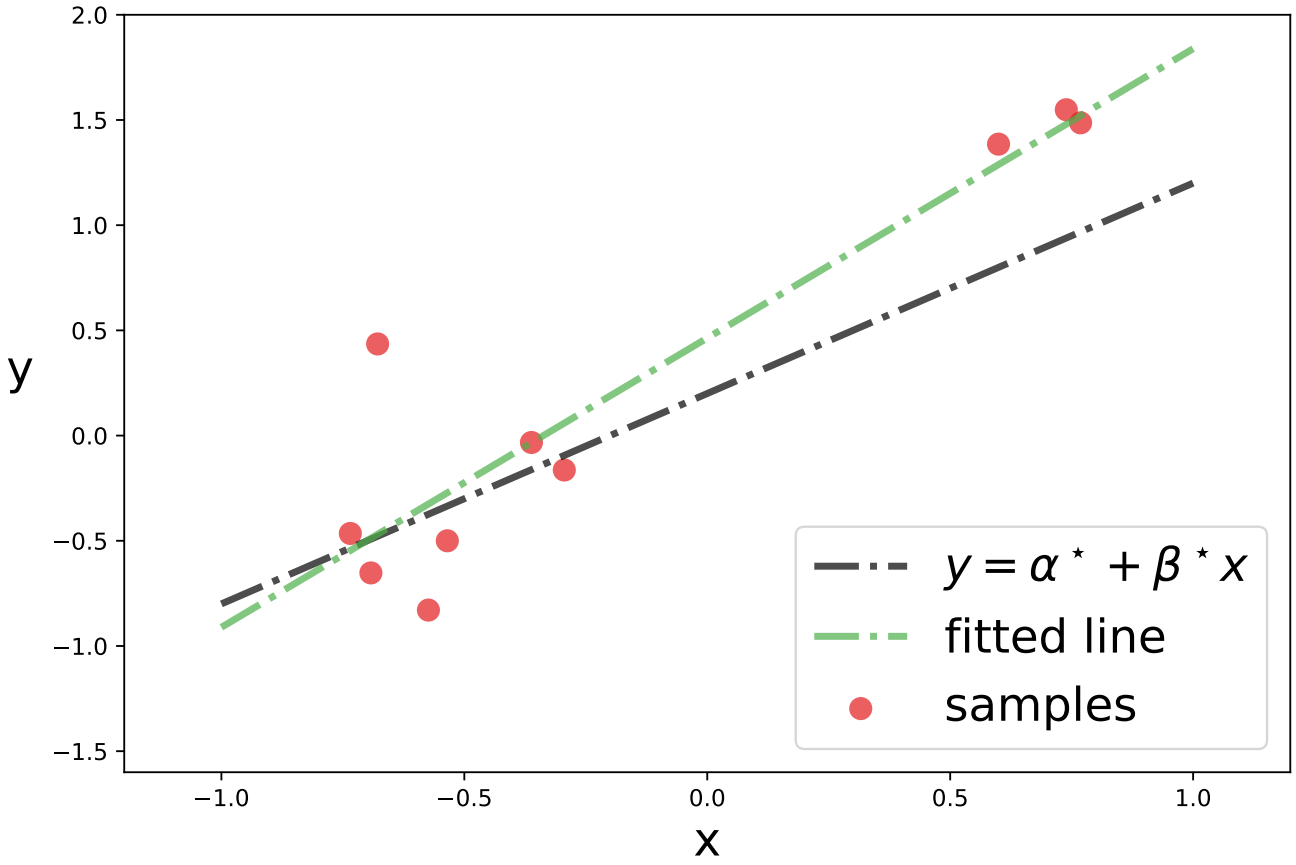
Truth is: $y = \alpha^\star + \beta^\star x + \epsilon$
where intercept $\alpha^\star = 0.2$, slope $\beta^\star = 1.0$ and noise $\epsilon \sim \mathcal{N}(0, 0.25)$

Do we have enough training samples ?



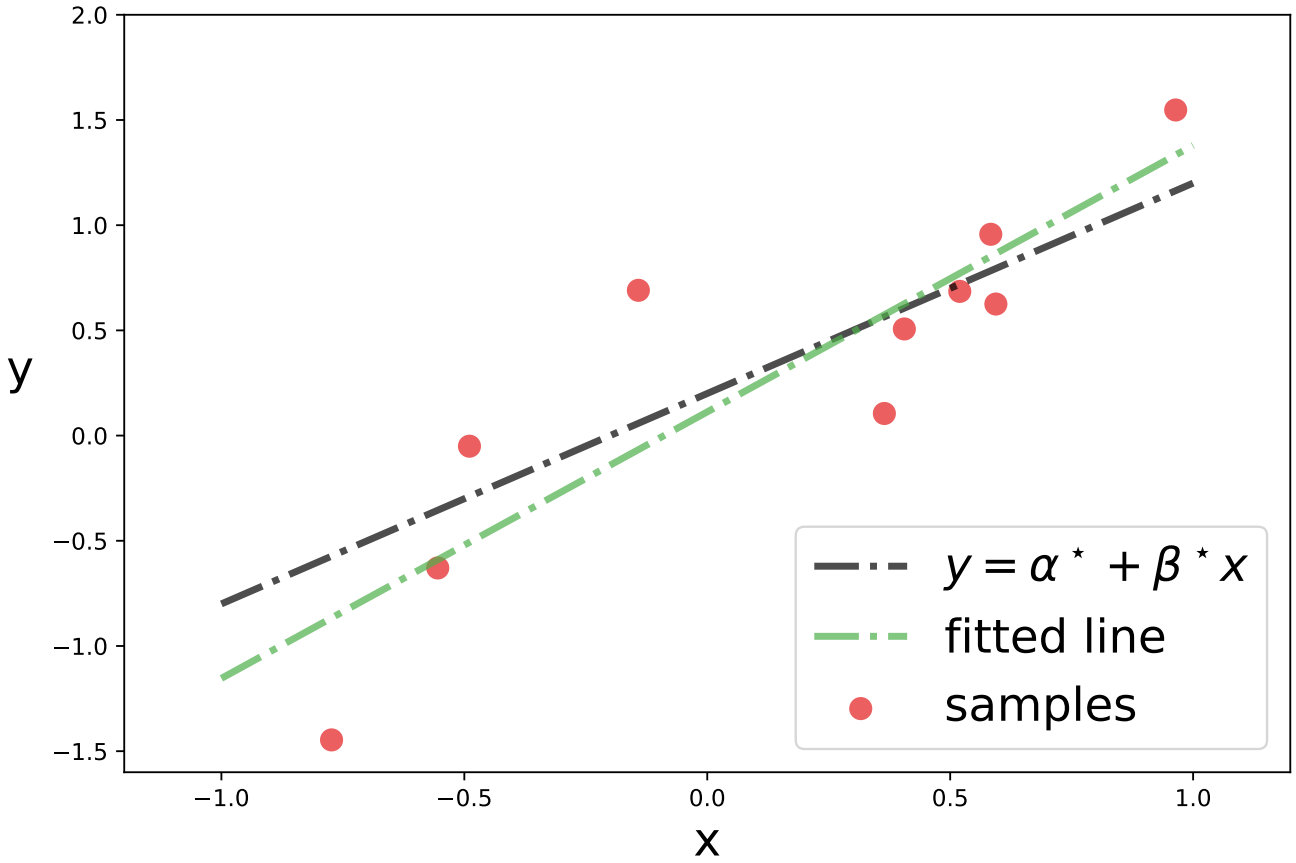
Fitted line when we have 10 samples

Do we have enough training samples ?



Fitted line with a different 10 samples

Do we have enough training samples ?

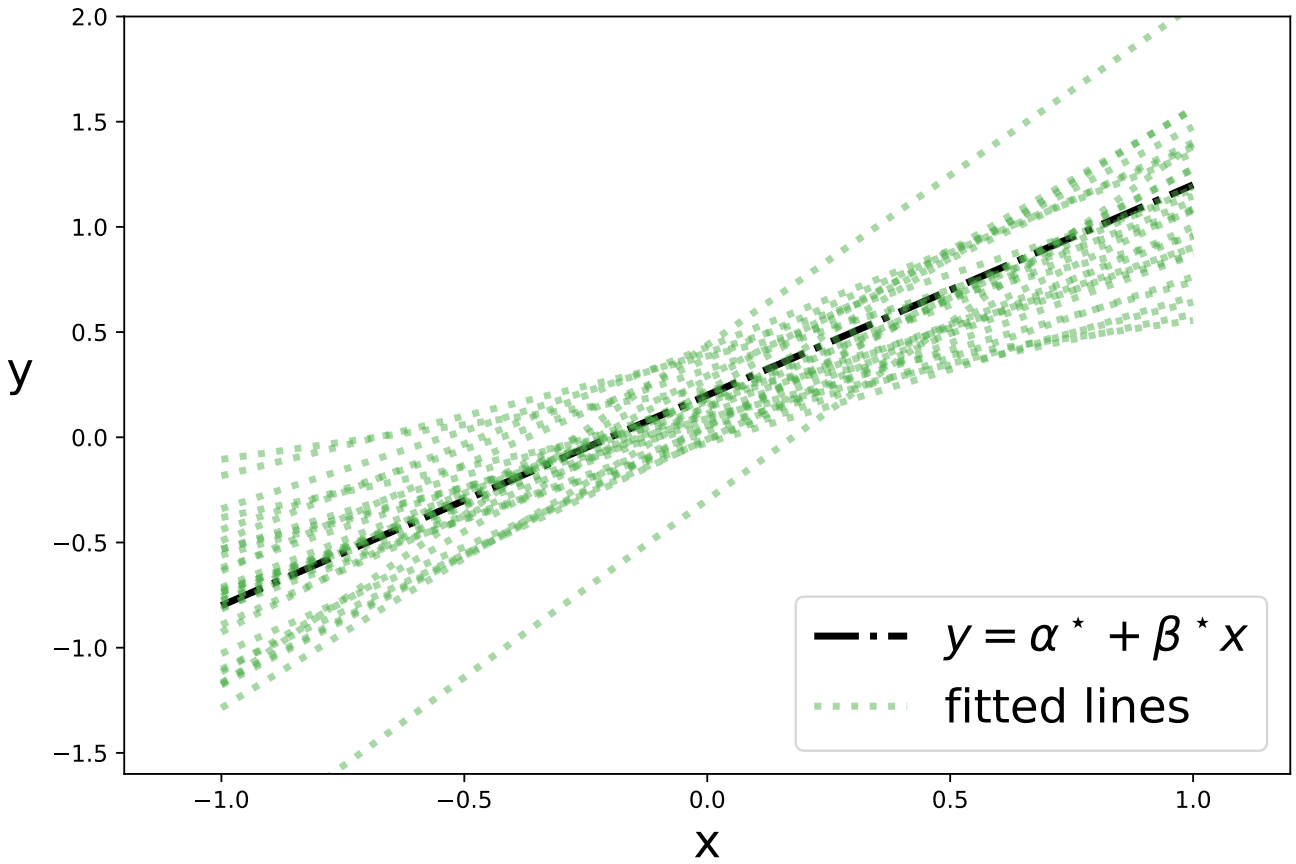


Fitted line yet another set of 10 samples

Do we have enough training samples ?

Every time we get a different $\widehat{\beta}$

Doing this (i.e. fit to 10 new samples every time) many times:

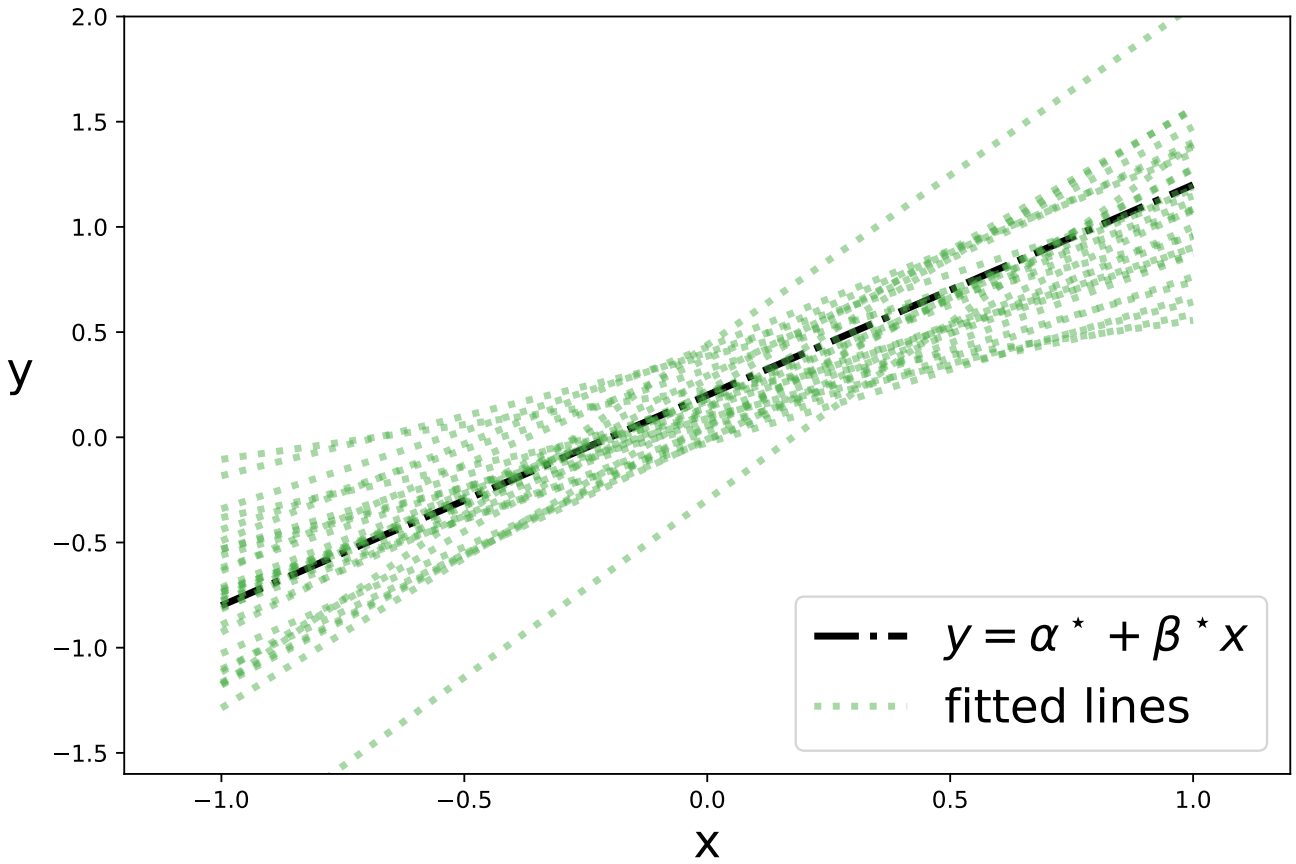


Each of these lines is fit to a **different** set of 10 training samples

Do we have enough training samples ?

Every time we get a different $\widehat{\beta}$

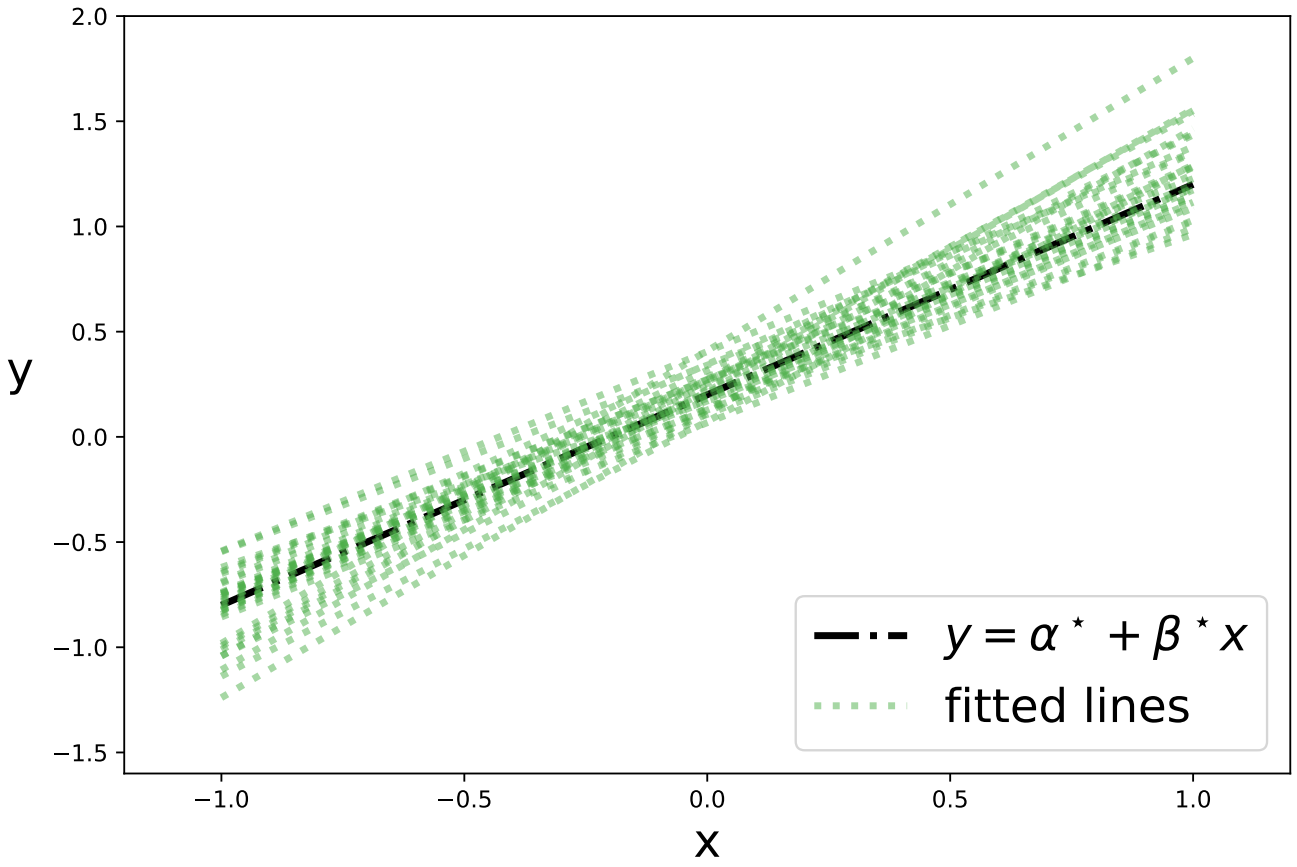
Doing this (i.e. fit to 10 new samples every time) many times:



Each of these lines is fit to a **different** set of 10 training samples

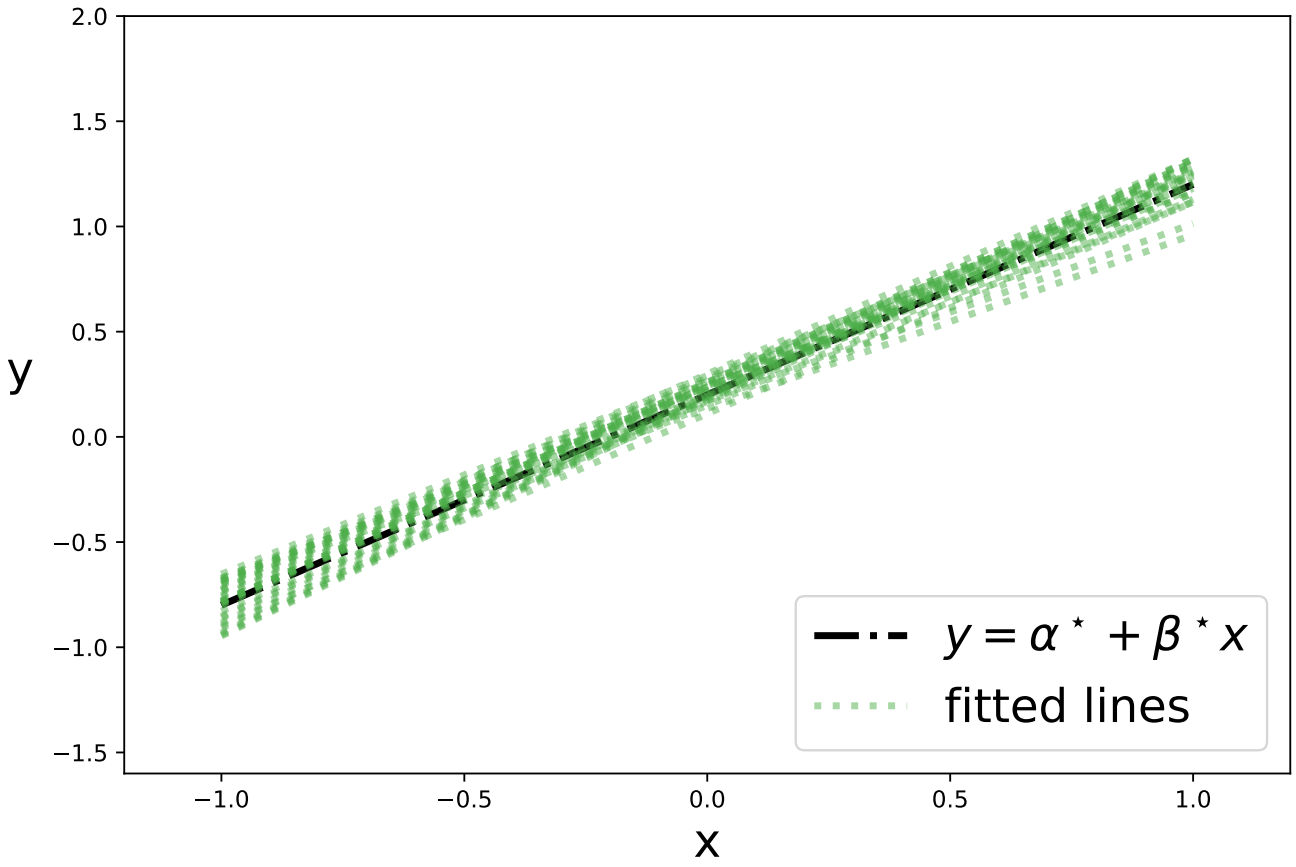
Do we have enough training samples ?

What if we had more training samples every time we fit ?
Here is fit with 30 samples:

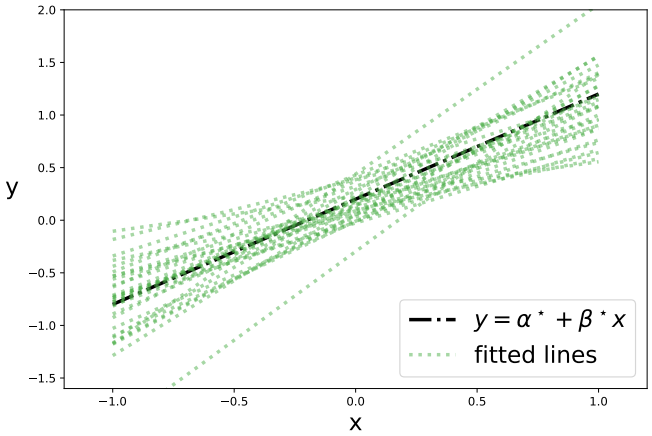


Do we have enough training samples ?

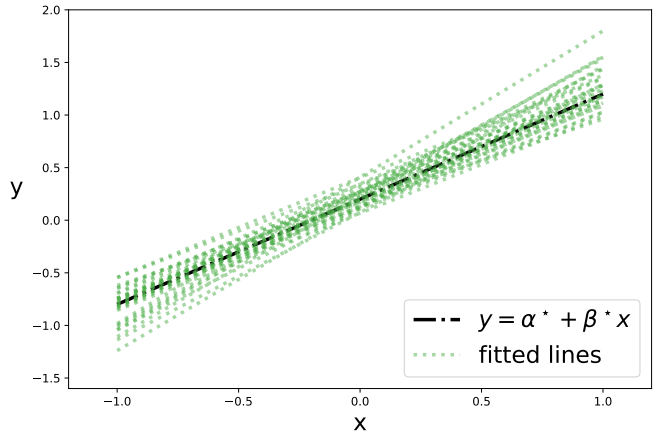
What if we had more training samples every time we fit ?
Here is fit with 100 samples:



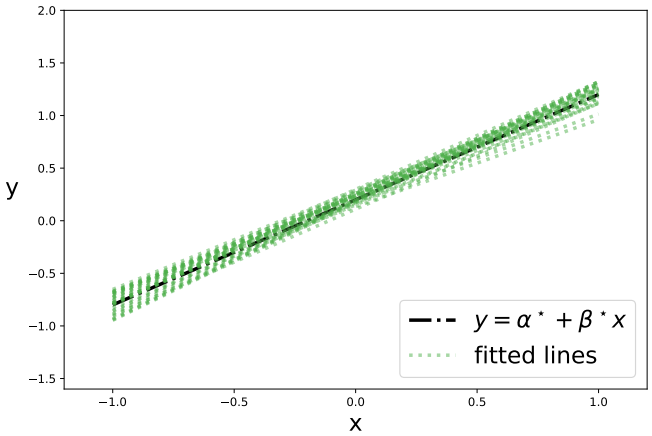
Do we have enough training samples ?



10 samples per line



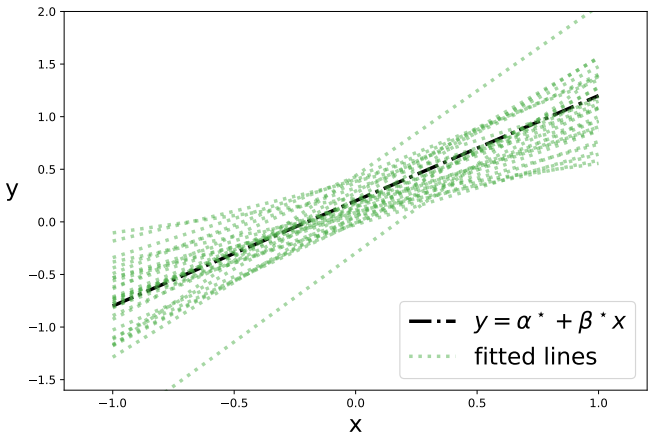
30 samples per line



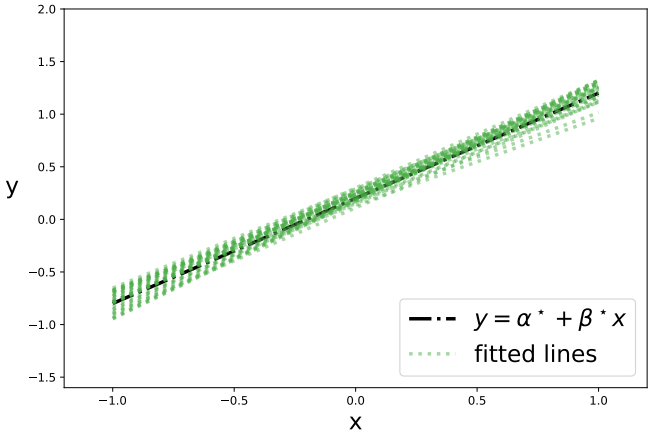
100 samples per line

We get better lines, i.e. better predictors, when we use more training samples to find the predictor.

Consistency



10 samples per line



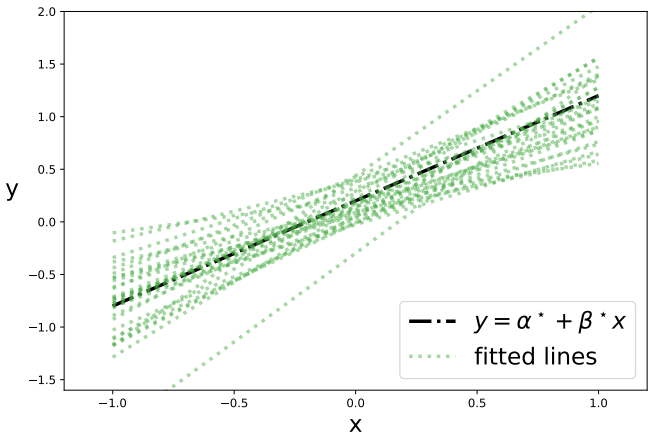
100 samples per line

Suppose samples come from:

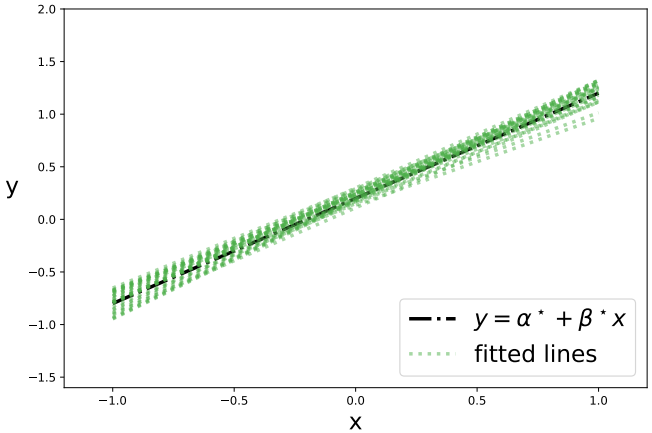
$$y^{(i)} = \beta^\star x^{(i)} + \epsilon^{(i)}$$

Where the $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ and **iid**

Consistency



10 samples per line



100 samples per line

Suppose samples come from:

$$y^{(i)} = \beta^\star x^{(i)} + \epsilon^{(i)}$$

Where the $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ and **iid**

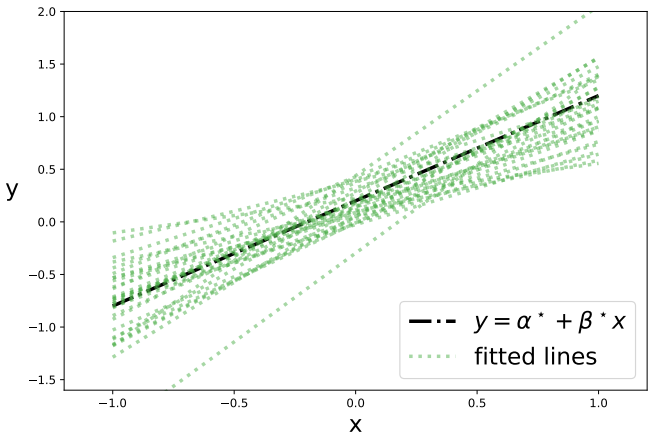
Consider estimate after n samples

$$\widehat{\beta}_n = \arg\min_\beta \sum_{i=1}^{n} (y^{(i)} - \beta^\top x^{(i)})^2$$

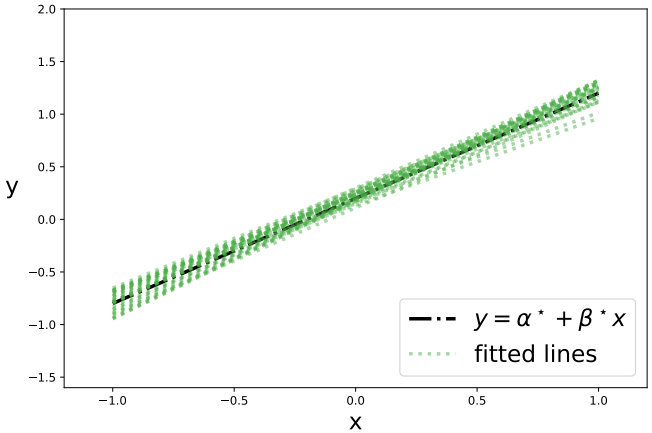
Consistency: as n increases, our estimate becomes correct ...

$$\widehat{\beta}_n \to \beta^\star \qquad \text{as } n \to \infty$$

Consistency



10 samples per line



100 samples per line

Suppose samples come from:

$$y^{(i)} = \beta^\star x^{(i)} + \epsilon^{(i)}$$

Where the $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ and **iid**

Consider estimate after n samples

$$\widehat{\beta}_n = \arg\min_{\beta} \sum_{i=1}^{n} (y^{(i)} - \beta^\top x^{(i)})^2$$

Consistency: as n increases, our estimate becomes correct ...

$$\widehat{\beta}_n \to \beta^\star \qquad \text{as } n \to \infty$$

(not covered in this class): **confidence intervals** for the estimates $\widehat{\beta}$ ...

Interpreting $\widehat{\beta}$

E.g. suppose target is TravelSpend, features are Age, Income, StudentDebt
And we learn a linear model:

$$\widehat{\text{TravelSpend}} = 300 + 20(\text{Age}) + 0.02(\text{Income}) - 0.1(\text{StudentDebt})$$

Interpretations:

- ~~As a person becomes older, their TravelSpend increases~~
- ~~If a person earns more, their TravelSpend increases~~

Interpreting $\widehat{\beta}$

E.g. suppose target is TravelSpend, features are Age, Income, StudentDebt
And we learn a linear model:

$$\widehat{\text{TravelSpend}} = 300 + 20(\text{Age}) + 0.02(\text{Income}) - 0.1(\text{StudentDebt})$$

Interpretations:

- ~~As a person becomes older, their TravelSpend increases~~
- ~~If a person earns more, their TravelSpend increases~~
- If Age & Income held constant, TravelSpend $\downarrow$ as StudentDebt $\uparrow$
- If StudentDebt & Income held constant, TravelSpend $\uparrow$ as Age $\uparrow$

$sign(\widehat{\beta}_i)$ **: how** y **depends on** x_i **when the other** x_j**'s are held fixed**

Interpreting $\widehat{\beta}$

E.g. suppose target is TravelSpend, features are Age, Income, StudentDebt
And we learn a linear model:

$$\widehat{\text{TravelSpend}} = 300 + 20(\text{Age}) + 0.02(\text{Income}) - 0.1(\text{StudentDebt})$$

Interpreting $\widehat{\beta}$

E.g. suppose target is TravelSpend, features are Age, Income, StudentDebt
And we learn a linear model:

$$\widehat{\text{TravelSpend}} = 300 + 20(\text{Age}) + 0.02(\text{Income}) - 0.1(\text{StudentDebt})$$

Interpretations:

- ~~TravelSpend depends more on Age than Income because $\widehat{\beta}_{Age} = 20$ but $\widehat{\beta}_{Income} = 0.02$~~

Interpreting $\widehat{\beta}$

E.g. suppose target is TravelSpend, features are Age, Income, StudentDebt
And we learn a linear model:

$$\widehat{\text{TravelSpend}} = 300 + 20(\text{Age}) + 0.02(\text{Income}) - 0.1(\text{StudentDebt})$$

Interpretations:

- ~~TravelSpend depends more on Age than Income because $\widehat{\beta}_{Age} = 20$ but $\widehat{\beta}_{Income} = 0.02$~~

Relative importance of features much harder to infer. At the minimum, every variable should be *normalized*

Normalization

Normalization is the process of shifting and scaling the features so that they all end up being on the same scale.

Suppose we have n training samples $(x^{(i)}, y^{(i)}), i = 1, \ldots, n$ where each feature vector $x^{(i)}$ is a d-dimensional vector

The first step is to find the mean and standard deviation for each feature

$$\mu_j = \frac{\sum_i x_j^{(i)}}{n} \quad \textit{and} \quad \textit{stdDev}_j = \sqrt{\frac{\sum_i (x_j^{(i)} - \mu_j)^2}{n}} \quad \textit{for} \quad j = 1 \ldots d$$

Then, transform each feature in each sample as follows

$$\tilde{x}_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\textit{stdDev}_j}$$

Normalization

We now have the (transformed) samples $(\widetilde{x}^{(i)}, y^{(i)}), i = 1, \ldots, n$

Then find the β for this transformed problem by solving the least squares problem

$$\widehat{\beta} = \arg\min_{\beta} \sum_{i} \left(y^{(i)} - \beta^{\top} \widetilde{x}^{(i)} \right)^2$$

At **inference time** (i.e. when applying it to the test data) we need to make sure we shift and scale the test samples too, before applying the above $\widehat{\beta}$.

Note: This shifting and scaling of the test data is done with the **same** $\mu_j, stdDev_j$ we computed from the n samples of the training data. We do NOT compute the new mean and standard deviaiton of the test data.

Overfitting

"When an algorithm fits too closely to it's training data, resulting in a model that cannot make accurate predictions / conclusions on new data."

Setup: assume there is a "ground truth: unknown β^*.

Each sample in the training set is

$$y = x^\top \beta^* + w$$

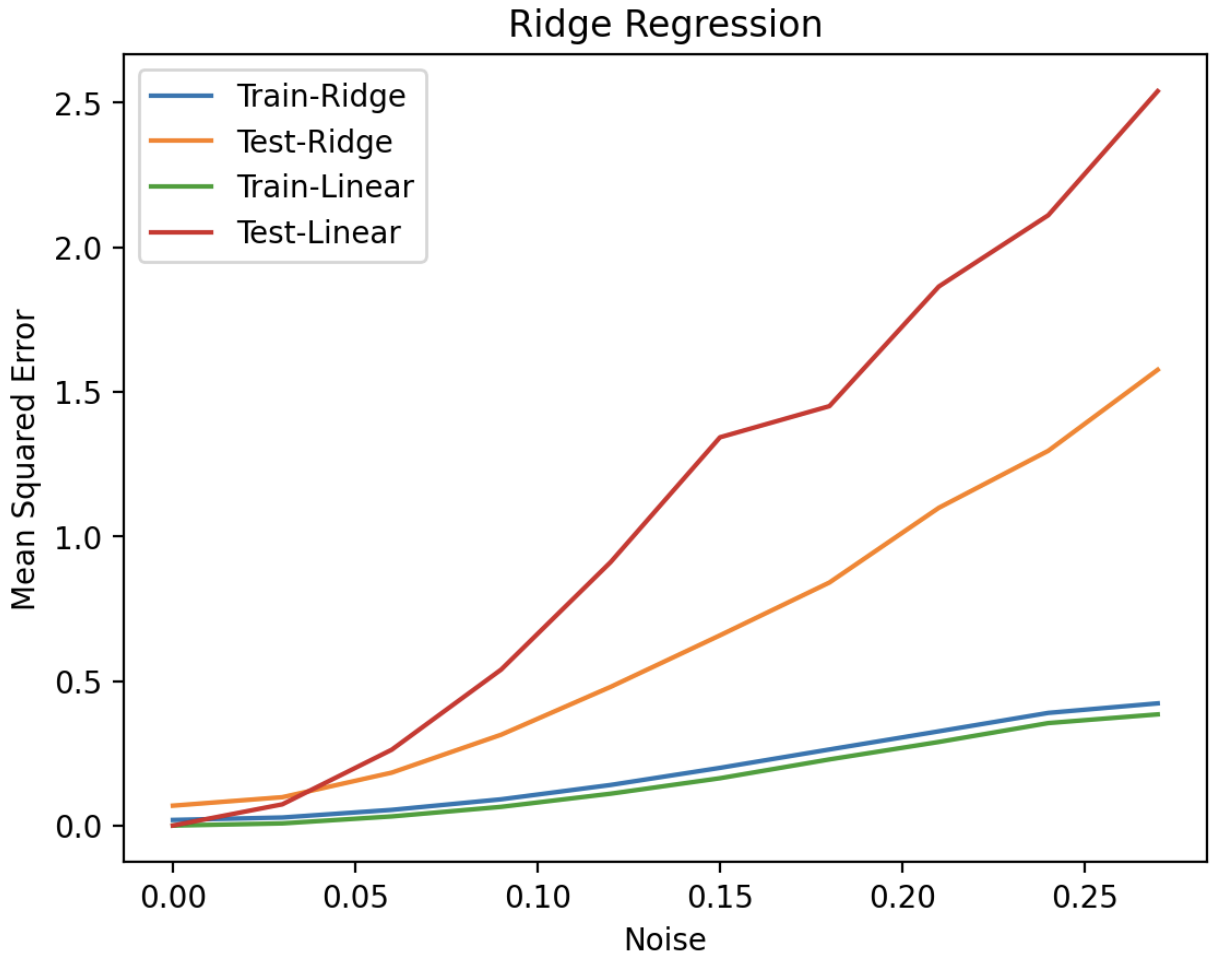
where w is Gaussian noise with mean 0 and **some** variance.

Q: How does estimation error $\|\widehat{\beta} - \beta^*\|$ depend on variance in noise w ?

Here $\widehat{\beta}$ is the estimate we get by doing least-squares linear regression

(obviously, the more the noise the more the error, but can we make this a little better ..?)

Overfitting



dimension 100, number of samples 200

Divergence between training error and test error because there are too many degrees of freedom
(since number of samples is comparable to dimension)

Overfitting

In linear regression, this happens because the β in the minimization is unconstrained – can take large/strange values even for small gains in squared error.

Idea: Penalize β, so that it takes high values **only if** it leads to big gains in squared loss.

$$\arg \min_{\beta} \sum_{i} \left(y^{(i)} - \beta^{\top} x^{(i)} \right)^2 + \mathbf{r}(\beta)$$

$r(\cdot)$ is called the **regularizer** – it is a penalty function

Ridge Regression

The ridge regression estimator is

$$\widehat{\beta}_{ridge} = \arg \min_{\beta} \sum_{i} \left(y^{(i)} - \beta^{\top} x^{(i)} \right)^2 + \alpha \|\beta\|_2^2$$

Here, $\|\beta\|_2^2$ is the square of the euclidean norm, i.e.

$$\|\beta\|_2^2 = \sum_{j=1}^{d} \beta_j^2$$

and α is the penalty parameter. Note $\alpha \geq 0$ always.

Ridge Regression

The ridge regression estimator is

$$\widehat{\beta}_{ridge} = \arg\min_{\beta} \sum_{i} \left(y^{(i)} - \beta^{\top} x^{(i)}\right)^2 + \alpha \|\beta\|_2^2$$

Here, $\|\beta\|_2^2$ is the square of the euclidean norm, i.e.

$$\|\beta\|_2^2 = \sum_{j=1}^{d} \beta_j^2$$

and α is the penalty parameter. Note $\alpha \geq 0$ always.

$\alpha \uparrow$ means worse fit on training data, but less likely to overfit
$\alpha \downarrow$ means better fit on training data, but more likely to overfit
$\alpha = 0$ is back to simple linear regression
Important: Features need to be normalized before this is done.

Ridge Regression

Solving: the ridge estimator

$$\widehat{\beta}_{ridge} = \arg\min_{\beta} \sum_{i} \left(y^{(i)} - \beta^{\top} x^{(i)}\right)^2 + \alpha \|\beta\|_2^2$$

$$= \arg\min_{\beta} \|y - X\beta\|_2^2 + \alpha \|\beta\|_2^2$$

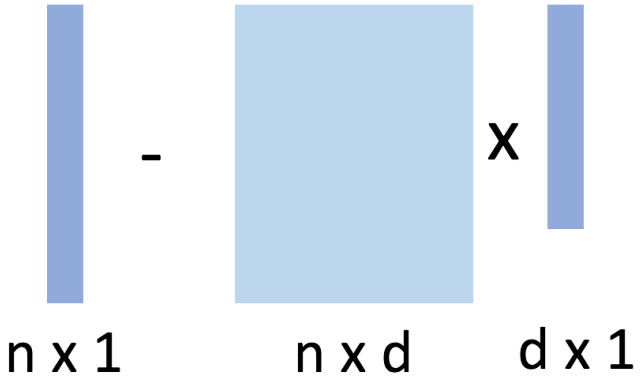
can be written in closed form:

$$\widehat{\beta}_{ridge} = (X^{\top}X + \alpha I)^{-1} X^{\top} y$$

This is true for *both cases* of $n \geq d$ and $n < d$

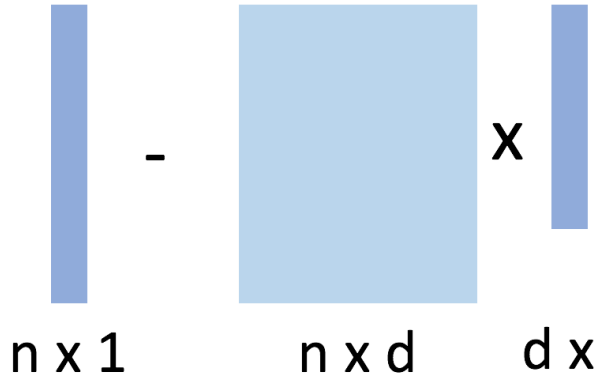
Ridge regression is a good idea when number of samples is too low as compared to the dimension or the level of noise)

Curse of Dimensionality



What happens when we have useless features

Curse of Dimensionality

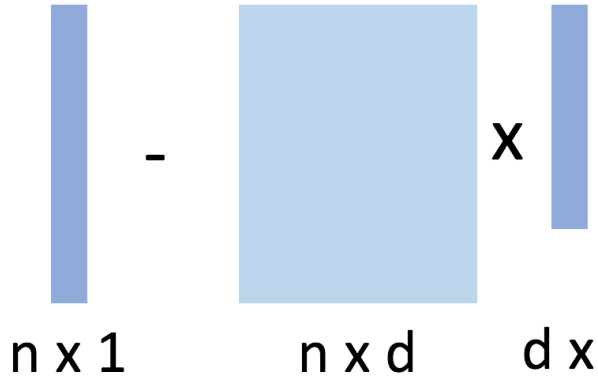


What happens when we have useless features

Illustrative Example: Suppose there are 100 features but only 10 are actually relevant. That is, suppose

$$y = \beta_1^\star x_1 + \beta_2^\star x_2 + \ldots + \beta_{10}^\star x_{10} + w$$

Curse of Dimensionality



What happens when we have useless features

Illustrative Example: Suppose there are 100 features but only 10 are actually relevant. That is, suppose

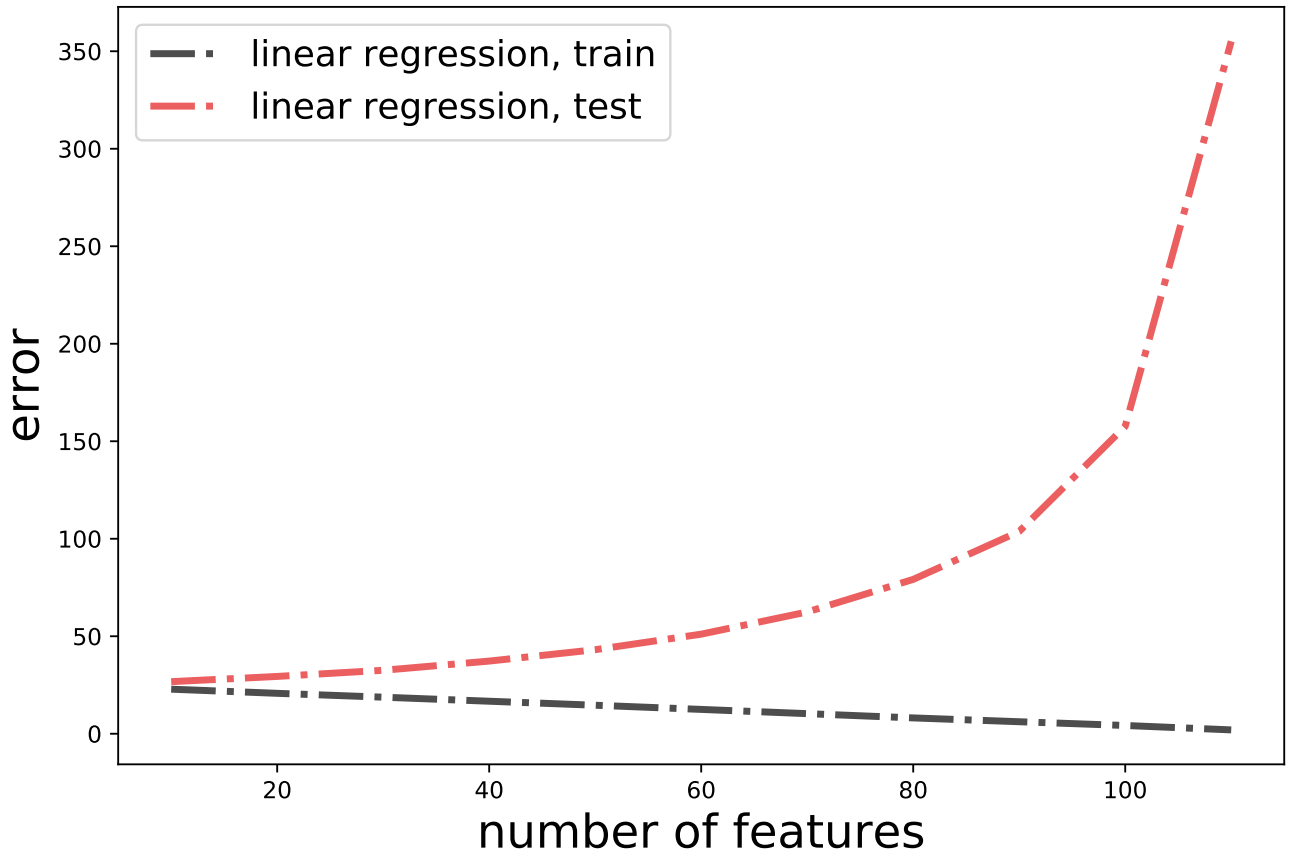
$$y = \beta_1^\star x_1 + \beta_2^\star x_2 + \ldots + \beta_{10}^\star x_{10} + w$$

But we do not know this, so we do least squares with d features, where $d \geq 10$

$$\arg \min_{\beta_1 \ldots \beta_d} \sum_i \left(y^{(i)} - \beta_1 x_1^{(i)} - \ldots - \beta_d x_d^{(i)} \right)^2$$

(i.e. there are *spurious* features)

Curse of Dimensionality

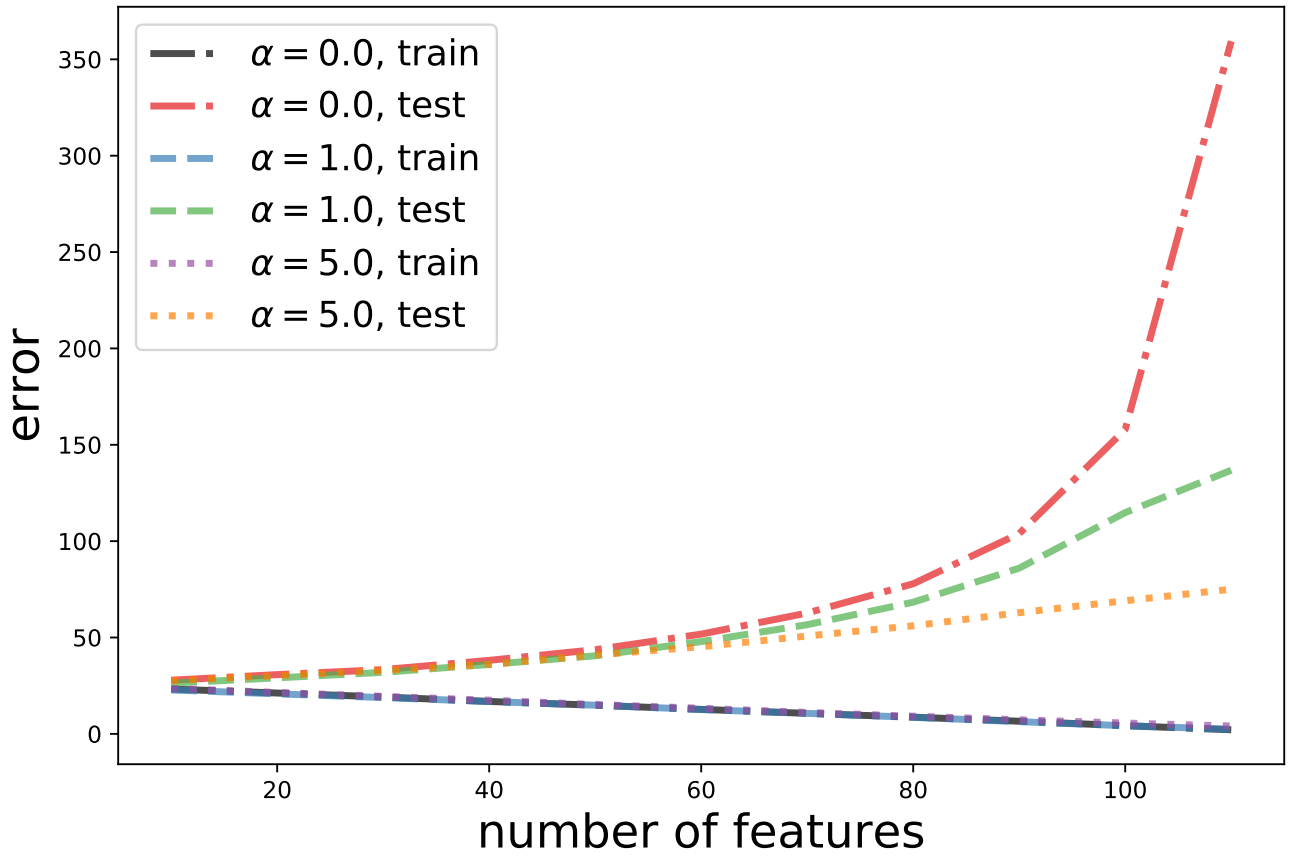


120 samples

only 10 features actually relevant (but we dont know this a-priori)

As # spurious features ↑, train error ↓ but test error ↑
This is overfitting. And it gets worse as number of spurious features increases.

Lets try ridge regression



120 samples

only 10 features actually relevant (but we dont know this a-priori)

Higher the α, less the test error, i.e. less overfitting.

Somewhat of a fix, but can we do better ?

LASSO

The LASSO estimator is

$$\widehat{\beta}_{LASSO} \;=\; \arg \min_{\beta} \sum_{i} \left(y^{(i)} - \beta^{\top} x^{(i)} \right)^2 \;+\; \alpha \, \|\beta\|_1$$

Here, $\|\beta\|_1$ is the ℓ_1 norm, i.e.

$$\|\beta\|_1 \;=\; \sum_{j=1}^{d} |\beta_j|$$

and α is the penalty parameter. Note $\alpha \geq 0$ always.

LASSO

The LASSO estimator is

$$\widehat{\beta}_{LASSO} = \arg\min_{\beta} \sum_{i} \left(y^{(i)} - \beta^{\top} x^{(i)}\right)^2 + \alpha \|\beta\|_1$$

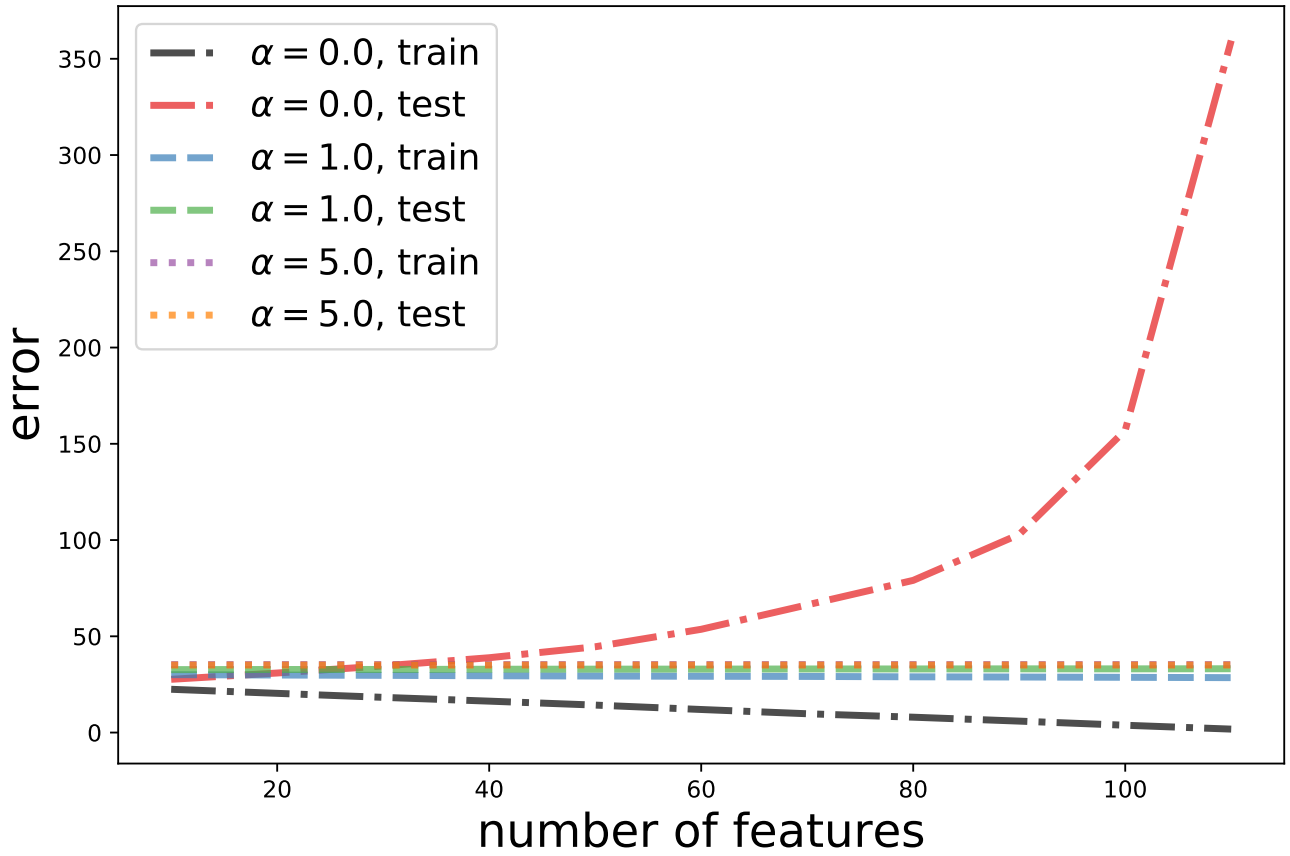
Here, $\|\beta\|_1$ is the ℓ_1 norm, i.e.

$$\|\beta\|_1 = \sum_{j=1}^{d} |\beta_j|$$

and α is the penalty parameter. Note $\alpha \geq 0$ always.

$\alpha \uparrow$ encourages **sparser** βs (more 0 coordinates). And yes, worse fit on training data, but less likely to overfit
$\alpha \downarrow$ encourages denser βs. And yes, means better fit on training data, but more likely to overfit
$\alpha = 0$ is back to simple linear regression
Important: Features need to be normalized before this is done.

LASSO



120 samples

only 10 features actually relevant (but we dont know this a-priori)

Higher the α, less the test error, i.e. less overfitting.
LASSO is better than ridge in this case.

Ridge and Lasso

Overfitting in linear regression controlled via regularization:

$$\min_{\beta} \sum_{i} \left(y^{(i)} - \beta^{\top} x^{(i)} \right)^2 + \mathbf{r}(\beta)$$

Ridge:

$$r(\beta) = \|\beta\|_2^2$$

+ Closed-form answer:

$$\widehat{\beta}_{ridge} = (X^{\top}X + \alpha I)^{-1} X^{\top} y$$

+ Useful to reduce overfitting
- Not tailored for sparsity / variable selection.

Lasso:

$$r(\beta) = \|\beta\|_1$$

+ very effective when β expected to be sparse (i.e. useful for curse of dimensionality settings)

- no closed form, have to use gradient descent or similar.