Regression

PHYS591000 2022.03.02

Outline

- Linear regression: Simply fit a line!
- Linear regression with multivariable and polynomials
- Problem of overfitting and regularization
- Non-linear? Ask the neighbors!
  – the k-Nearest Neighbors (kNN) Algorithm

Warming up

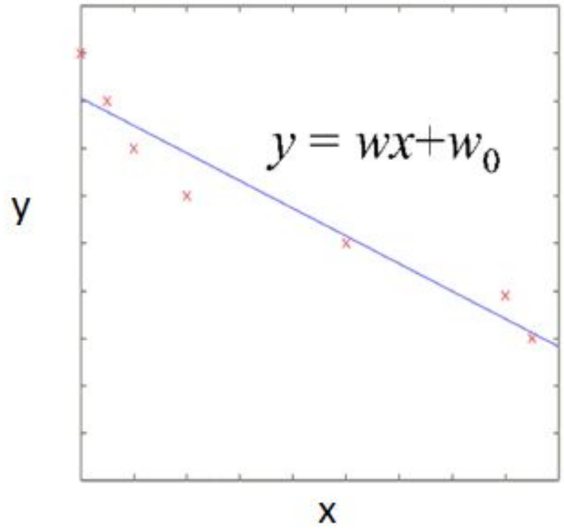
- Access control of the building will be granted soon.

- As usual, take 3 mins to introduce yourself to your teammate for this week!
  – “So you’re really staying in this class!”
  – “How’s your experience so far? What do you think we can do to make this class more enjoyable?”

Regression

- Start with the simplest case: Fit a line!
  → Linear regression

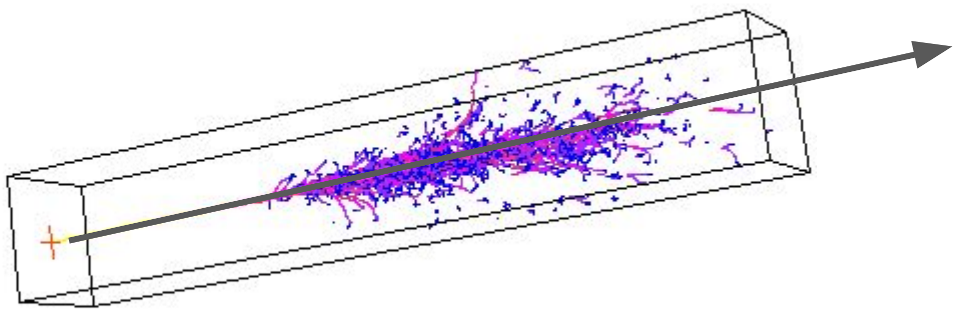
- Physics example: Predict energy of a particle using information of a calorimeter



Linear Regression

- Calorimeters are used to measure the energy of a particle

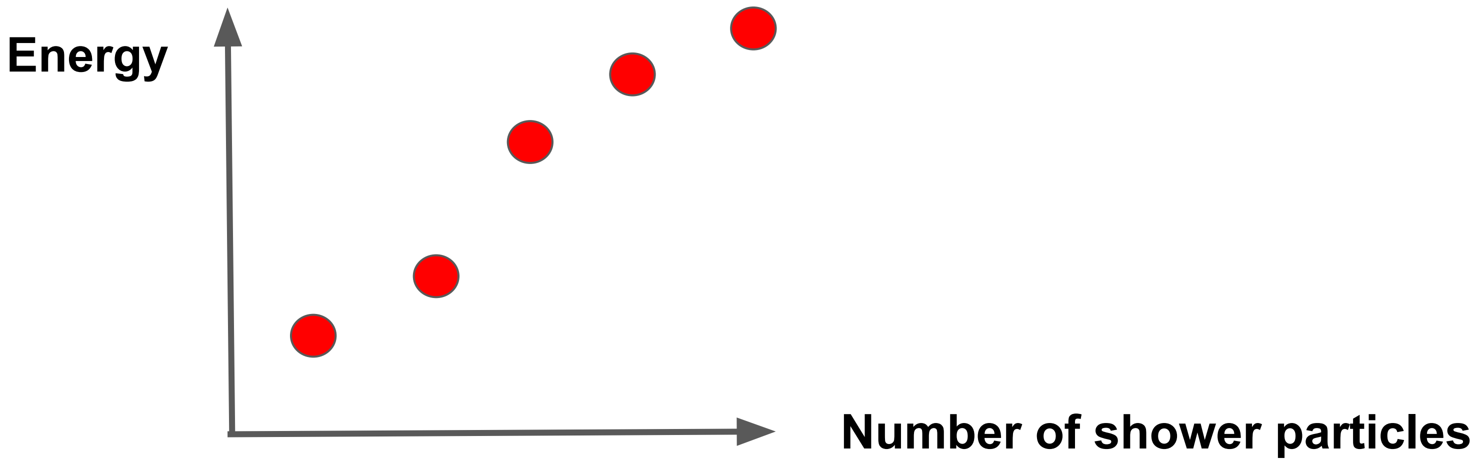
- The incoming particle interacts with materials in the calorimeter and produce a bunch of other particles. → "Shower"



[EM shower simulator by S. Menke]

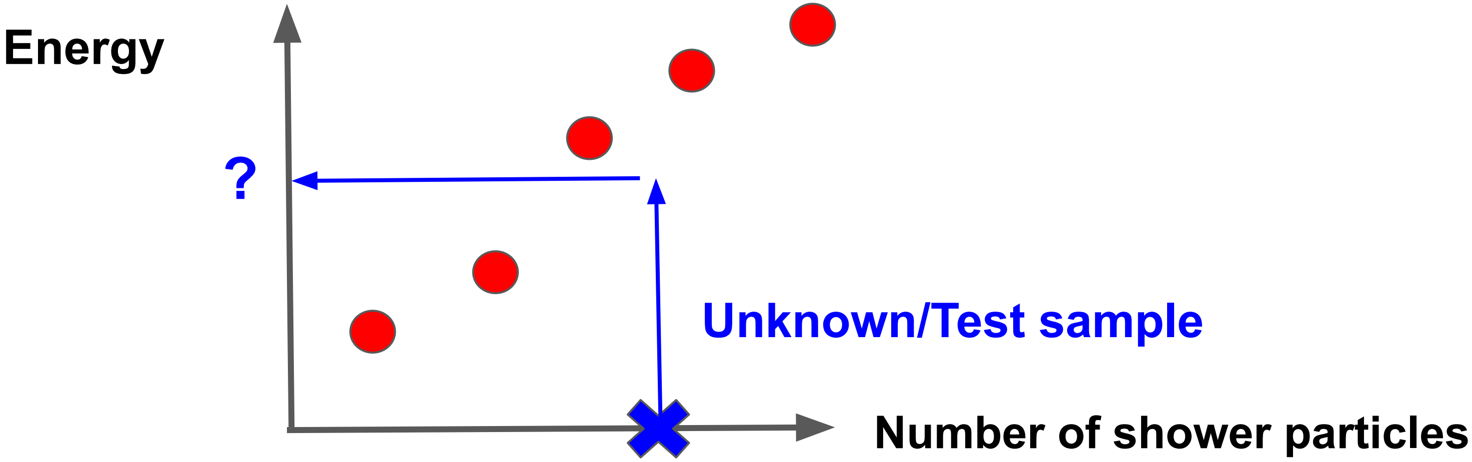
Linear Regression

- Expect the energy of the particle depends on the properties of the shower it creates, e.g. number of particles in the shower



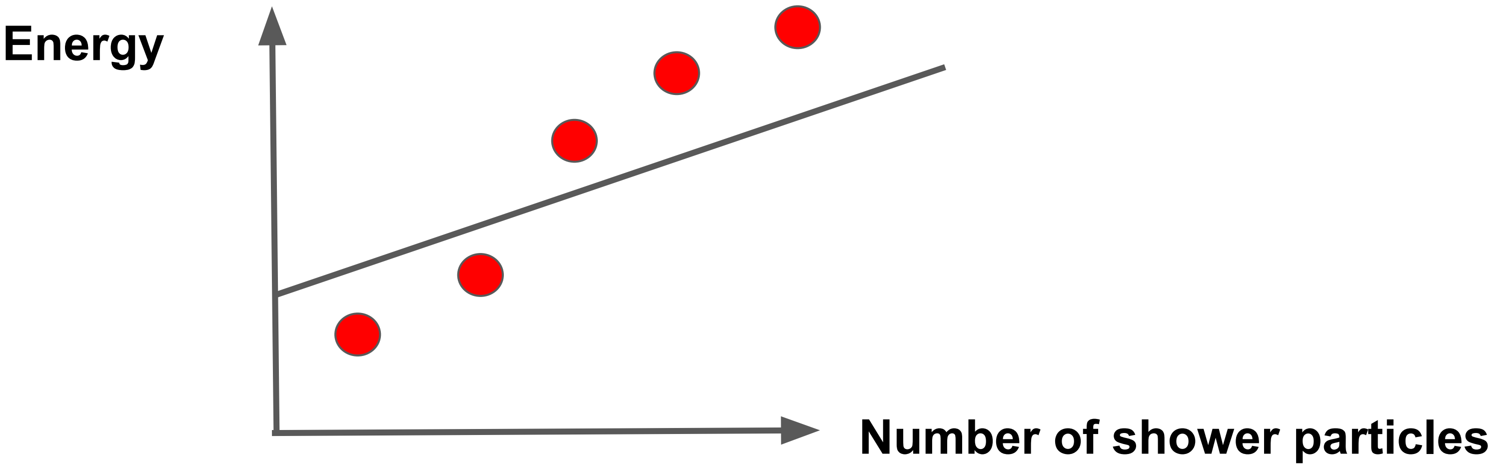
Linear Regression

- Goal: Predict the energy of a particle given the number of shower particles it creates.



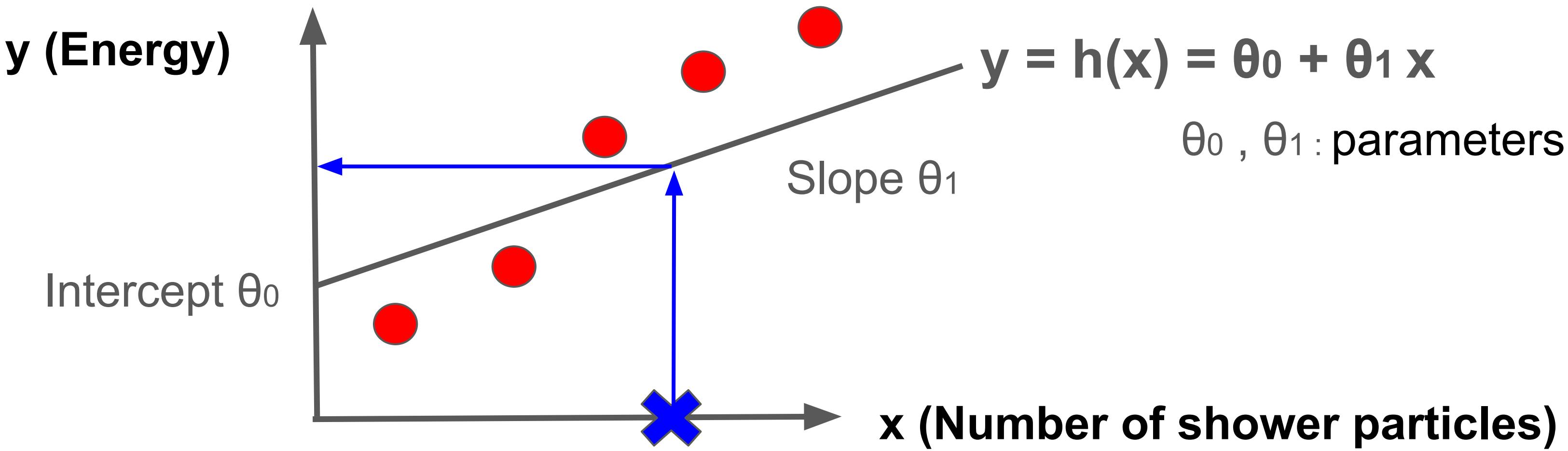
Linear Regression

- Fit a line!



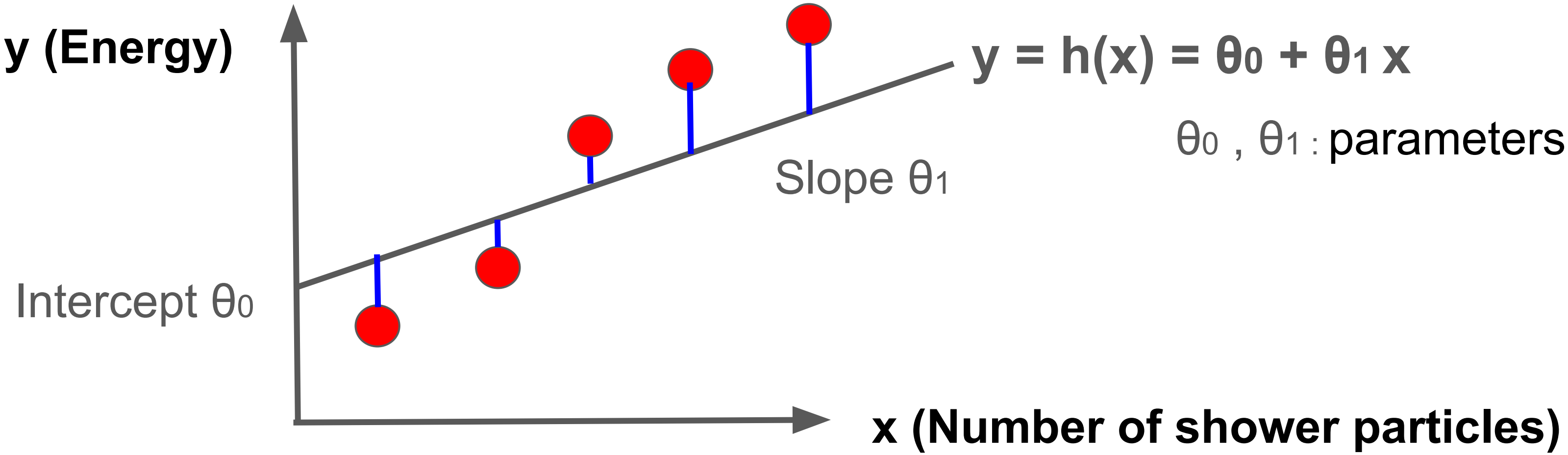
Linear Regression

- Make a prediction of y (**target**) given the value of x (**feature**) using a hypothesis $y = h(x) = \theta_0 + \theta_1 x$



Linear Regression

- The line $y = h(x) = \theta_0 + \theta_1 x$ is the line that gives the *closest* predictions to all the points from the training sample

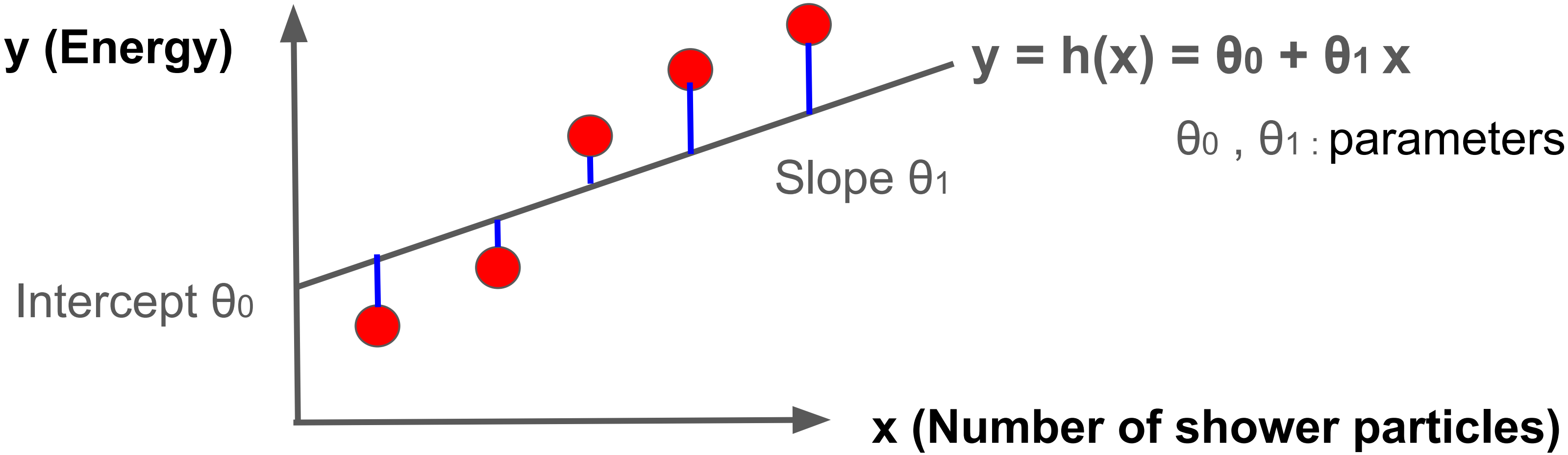


$y = h(x) = \theta_0 + \theta_1 x$

θ_0 , θ_1 : parameters

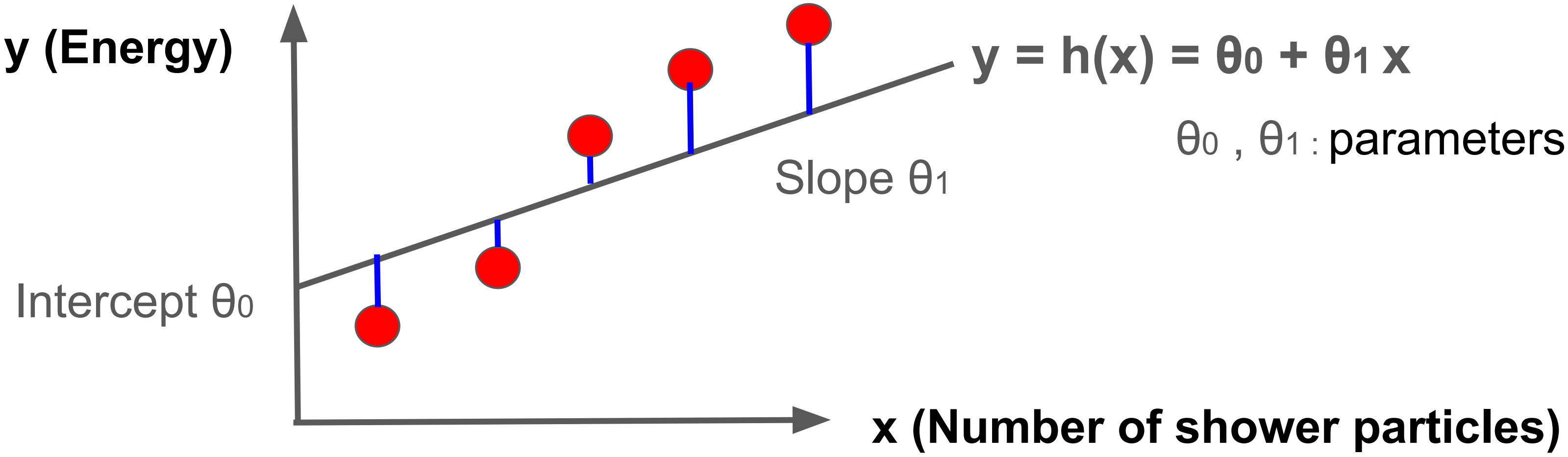
Linear Regression

- Find θ_0 and θ_1 that minimizes the sum of the squares $\Sigma_i(h(x_i)-y_i)^2$ from each (x_i, y_i) of the training sample → method of least squares



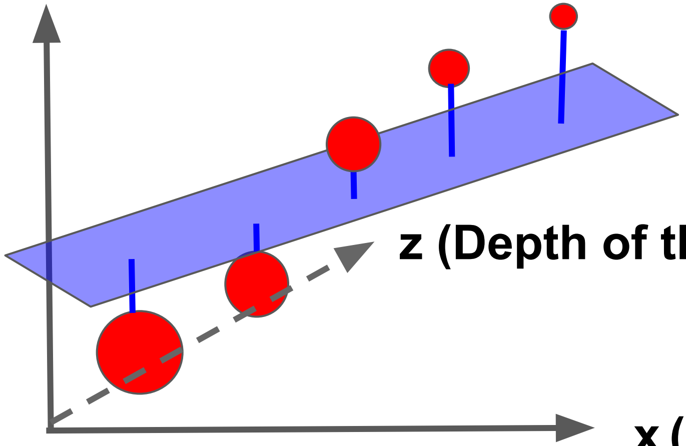
Linear Regression

- $\Sigma_i(h(x_i)-y_i)^2$ is called the objective function (or **cost** function, **loss** function)



Linear Regression

- Can extend to multivariable (more features) regression
  $\rightarrow$ Minimize $\Sigma_i(h(x_i,z_i)-y_i)^2$



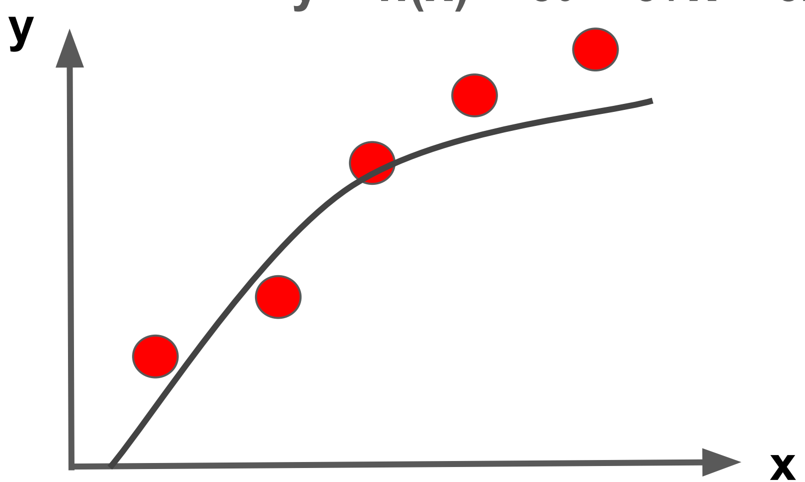
$$y = h(x, z) = \theta_0 + \theta_1 x + \theta_2 z$$

θ_0 , θ_1 , θ_2: parameters

Linear Regression

- What if we want to fit with a polynomial function?

$$y = h(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

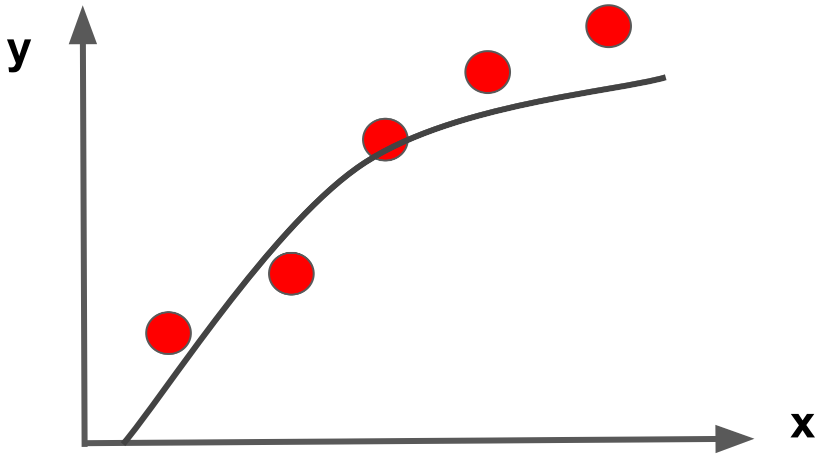


Linear Regression

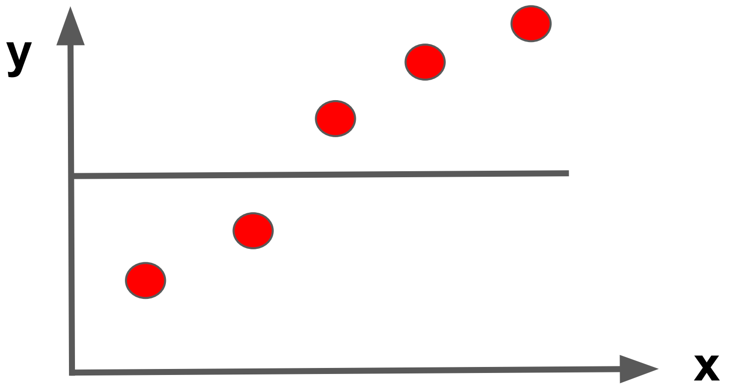
- What if we want to fit with a polynomial function?
  → Same as fitting with multiple features!

$$y = h(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

$$y = h(x, z) = \theta_0 + \theta_1 x + \theta_2 z$$

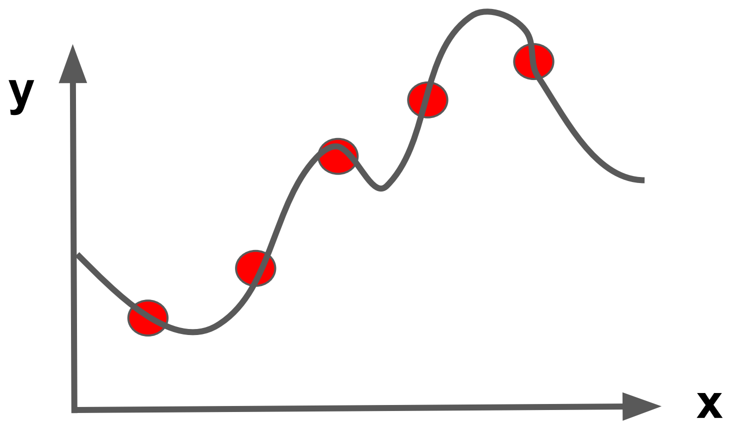


Linear Regression



$$y = h(x) = \theta_0$$

Underfitting



$$y = h(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

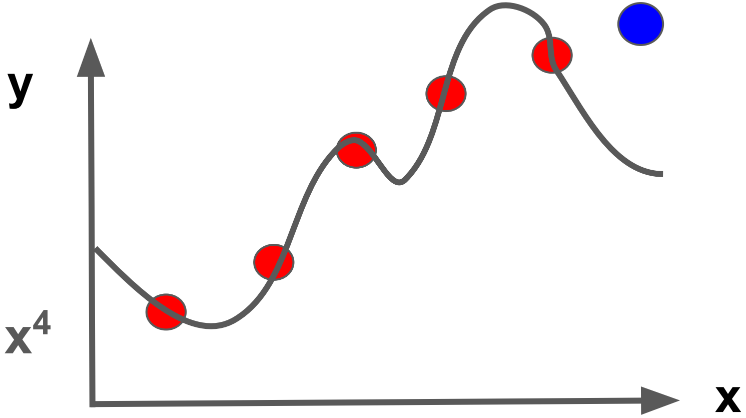
Overfitting

Linear Regression

Training

Testing

$$y = h(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$



Overfitting: When there are too many parameters/features, the learned hypothesis may fit the training set ‘too well’ (sum of least square → 0), but fail to generalize to new samples (fail to make good predictions for test samples).

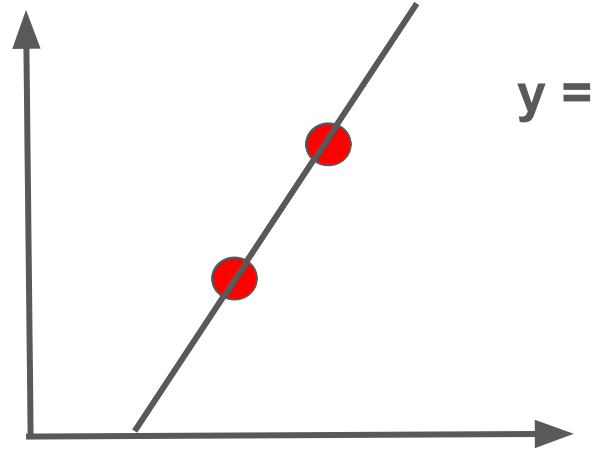
Overfitting

- Options to avoid/fix overfitting:
  - Manually reduce the number of features/parameters
  - **Regularization**: Add a ‘penalty’ for the sizes of parameters

Regularization

- Example: overfitting from two points → Sum of squares

$$\Sigma_i(h(x_i)-y_i)^2 = 0$$

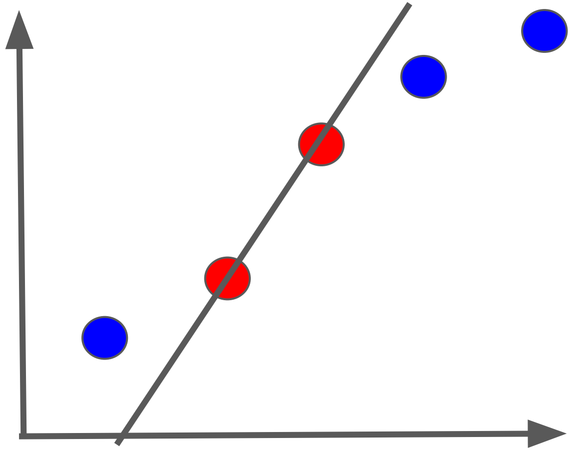


$$\mathbf{y = h(x) = \theta_0 + \theta_1\, x}$$

Regularization

- The line fits training (red) samples perfectly but fails to generalize to test samples (blue)

y (Energy)



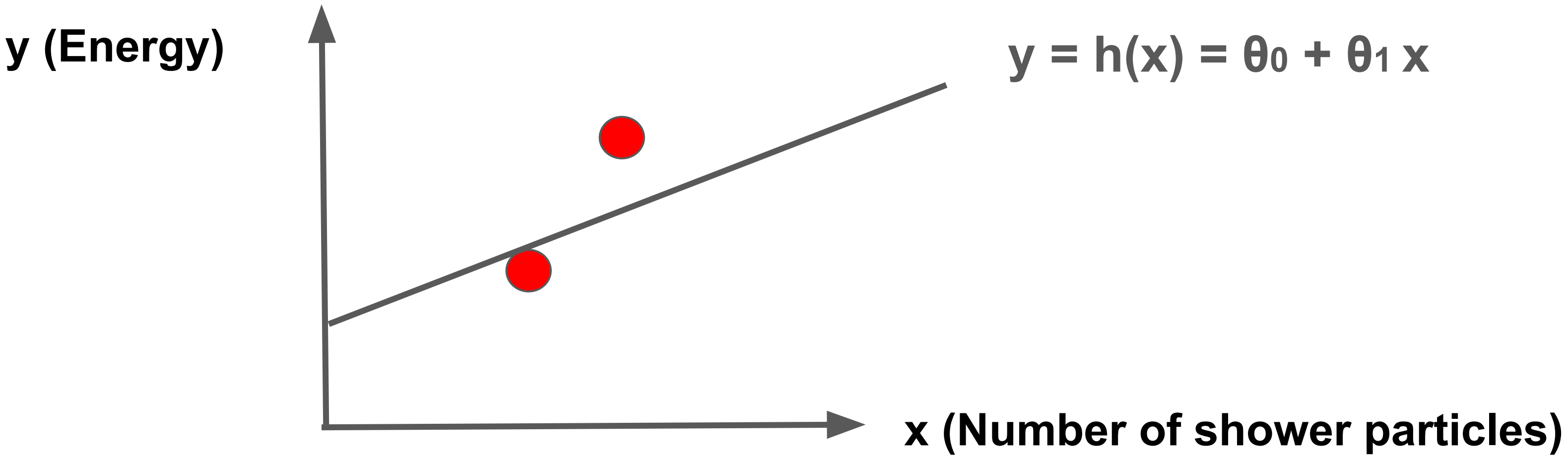
$$y = h(x) = \theta_0 + \theta_1 x$$

x (Number of shower particles)

Regularization

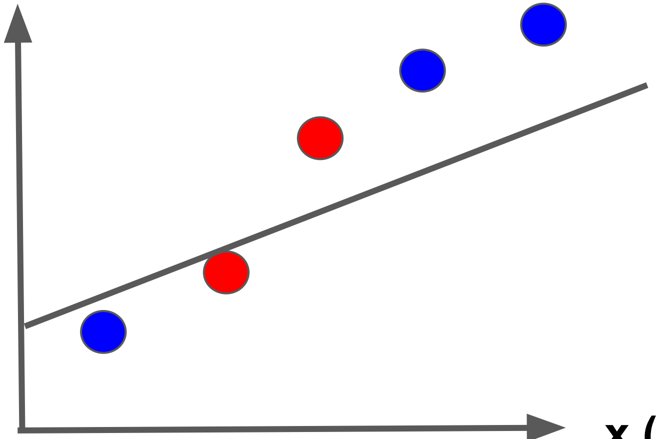
- Regularization: Minimize $\Sigma_i(h(x_i)-y_i)^2$ + $\boldsymbol{\lambda\theta_1^2}$ **λ:**hyperparameter

Slope θ_1 (Parameter)



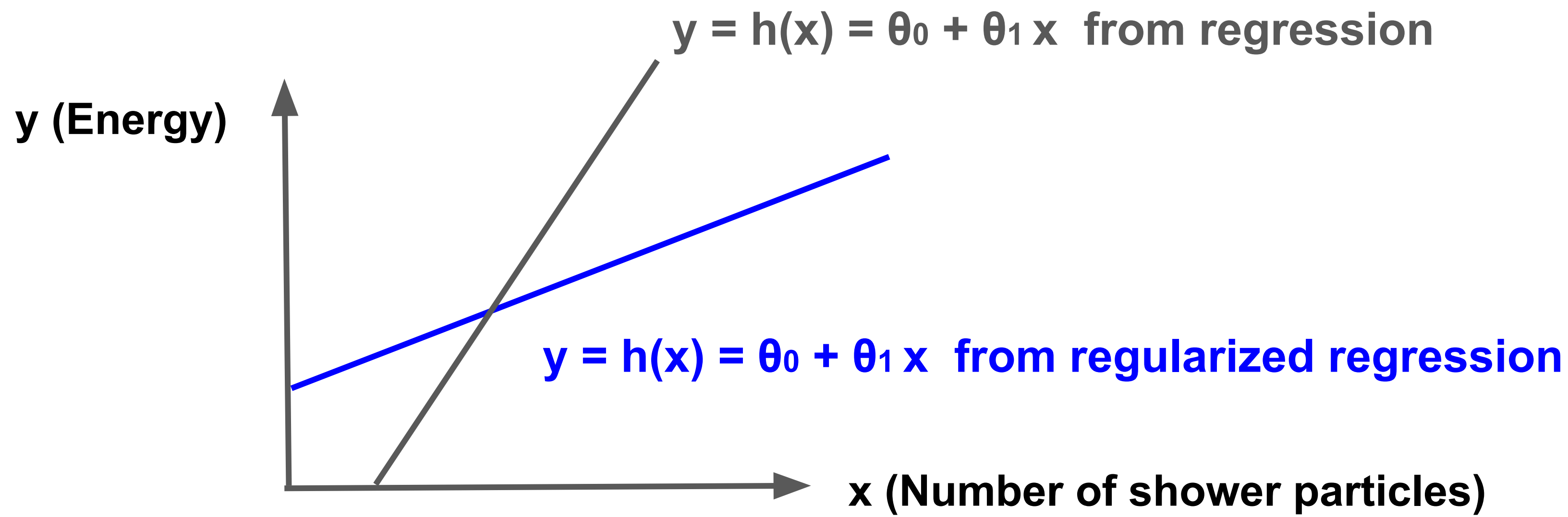
Regularization

- Regularization: Minimize $\Sigma_i(h(x_i)-y_i)^2$ + $\boldsymbol{\lambda\theta_1^2}$ **λ:**hyperparameter



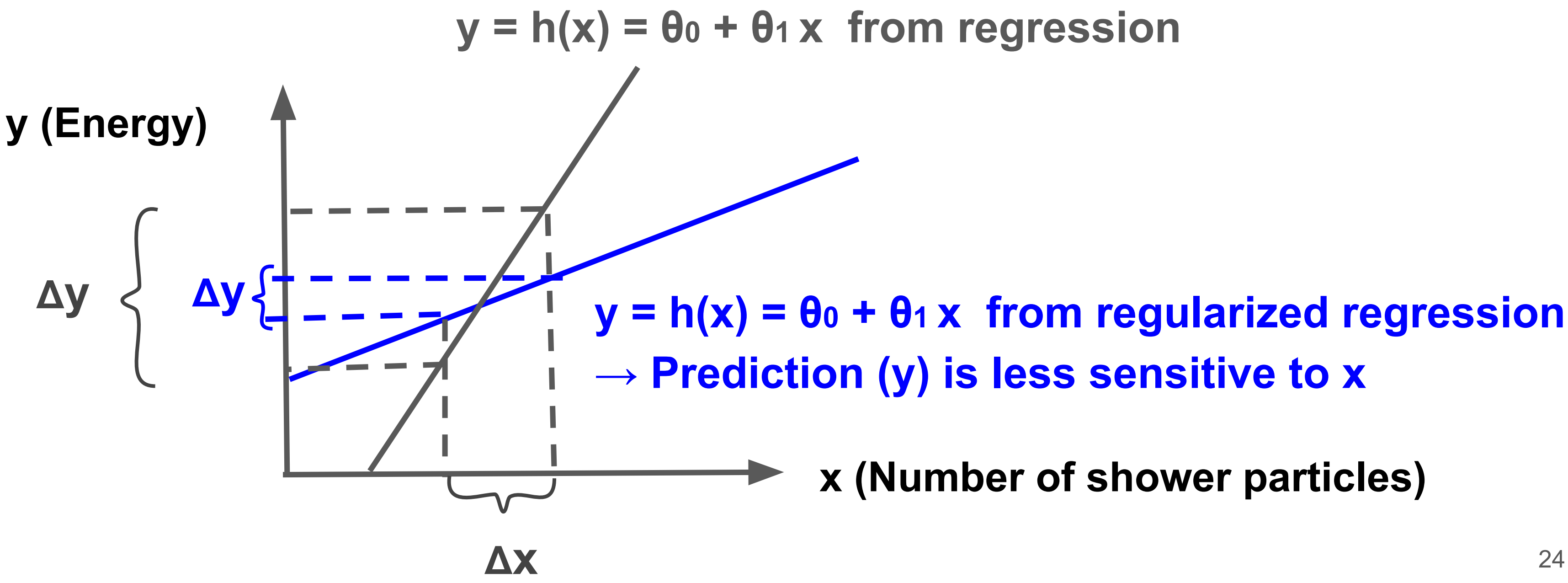
Regularization

- Regularization: Minimize $\Sigma_i(h(x_i)-y_i)^2 + \boldsymbol{\lambda\theta_1^2}$ **λ:**hyperparameter



Regularization

- Regularization: Minimize $\Sigma_i(h(x_i)-y_i)^2$ + $\boldsymbol{\lambda\theta_1^2}$ **λ:**hyperparameter

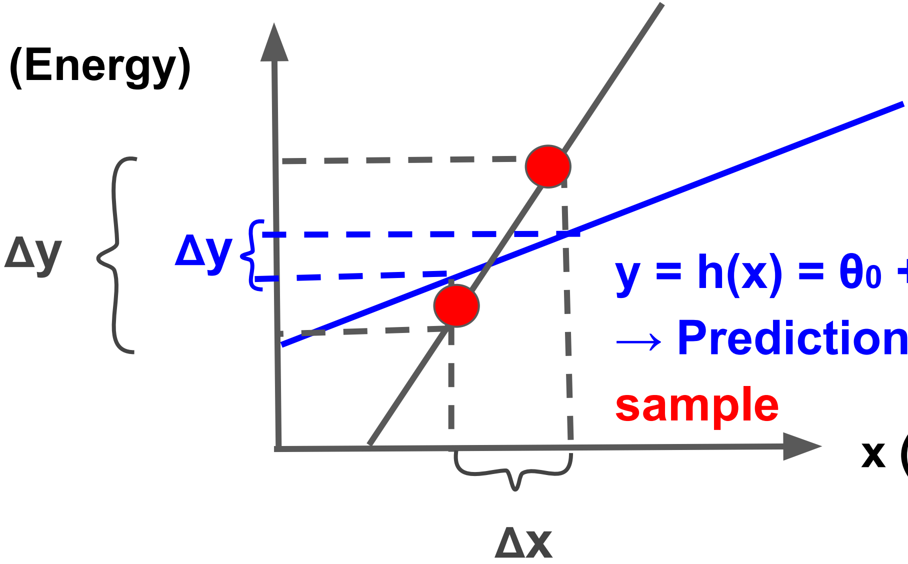


Regularization

- Regularization: Minimize $\Sigma_i(h(x_i)-y_i)^2 + \boldsymbol{\lambda\theta_1^2}$ **λ:**hyperparameter

$y = h(x) = \theta_0 + \theta_1 x$ from regression

y (Energy)



$y = h(x) = \theta_0 + \theta_1 x$ from regularized regression

→ Prediction (y) is less sensitive to the training sample

x (Number of shower particles)

Regularization

- Regularization: Minimize $\Sigma_i(h(x_i)-y_i)^2$ (with θ_0)+ $\boldsymbol{\lambda}\Sigma_i\boldsymbol{\theta_i}^{\mathbf{2}}$
  (from i=1, i.e. without θ_0; we do not penalize the overall constant θ_0)
- **λ:hyperparameter** chosen with an independent **validation** sample
  → Then apply on another independent test sample to evaluate the performance
- Ridge regression: $\boldsymbol{\lambda}\Sigma_i\boldsymbol{\theta_i}^{\mathbf{2}}$ (from i=1)
  Lasso regression: $\boldsymbol{\lambda}\Sigma_i|\boldsymbol{\theta_i}|$ (from i=1)

Linear Regression: Datasets

- For both in-class exercise and lab this week we’ll use the data from a neutrino experiment called OPERA.

- Let’s turn to this week’s in-class exercise to play with what we’ve learned.

Non-linear models

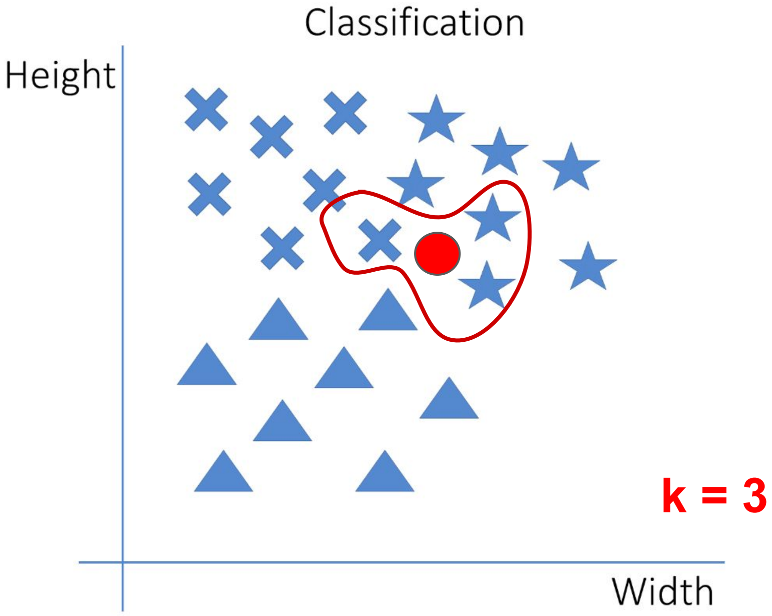
- So far we’ve been working with linear models:
  – Classify two categories with a line (LDA)
  – Making continuous predictions (Regression) by fitting a line (or a curve)

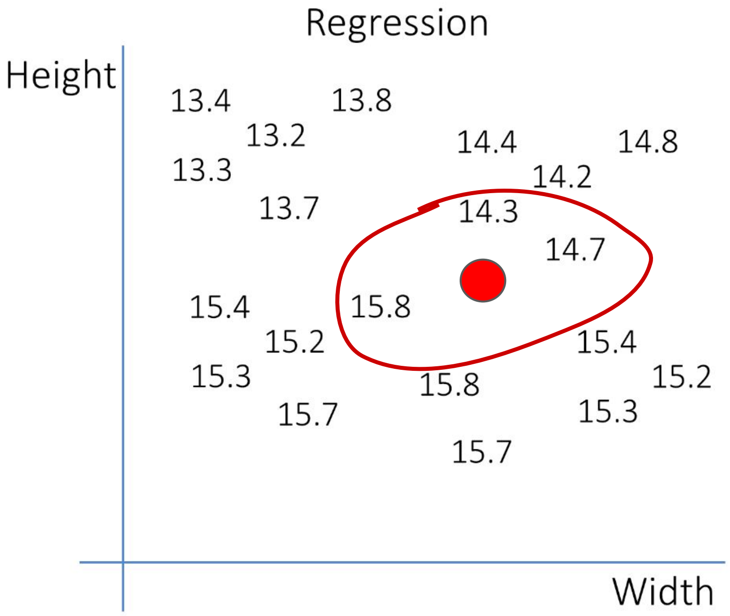
- What if the distributions are not so linear? (As you’ve already seen for Nmax and izmax in the OPERA data)

The k-Nearest Neighbors (kNN) Algorithm

- A simple method is to ‘make it the same as its neighbors’:
  – Pick a positive integer k (=1,2,3,...)
  – For a test (unknown) point, find the k-nearest neighbors of (known/training) data
  – Classification: take votes from the neighbors and classify the test data as the same category which gets the most votes
  – Regression: take average target value of the neighbors

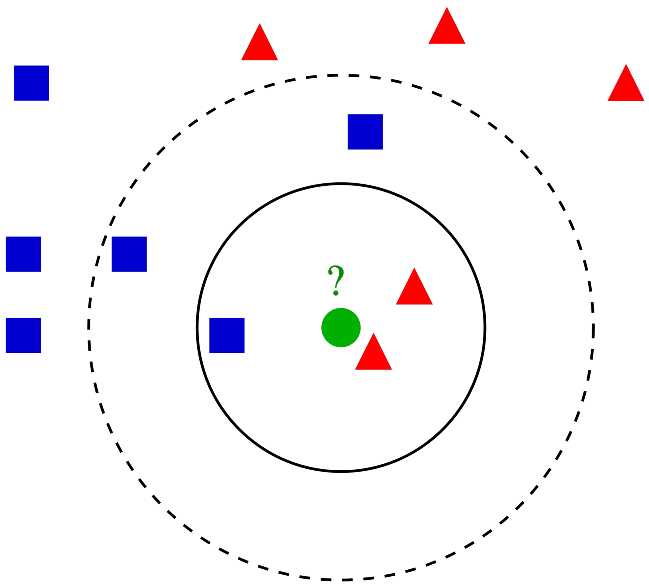
The k-Nearest Neighbors (kNN) Algorithm





The k-Nearest Neighbors (kNN) Algorithm

- kNN is intuitive and fast (no need to 'train' the model)
- The results depends on the choice of k



The k-Nearest Neighbors (kNN) Algorithm

- k too small: The prediction may be affected by noise/outliers
- k too large: Lose sensitivity to categories of small numbers
- Often need to try out a few different k, and validate the performance with test samples.

Lab for this week

- For the Lab this week, you’ll keep studying regression from the task of predicting neutrino energy with OPERA data, and compare the performance of linear regressions to kNN!

Summary-I

- Linear regression: Fit the training sample with a line by minimizing the sum of squares $\Sigma_i(h(x_i)-y_i)^2$

- Linear regression can be generalized to cases with multivariable (more than one feature) or with polynomials.

Summary-II

- When there are too many features one may overfit the training sample
  - Consequence: Cannot generalize to test samples

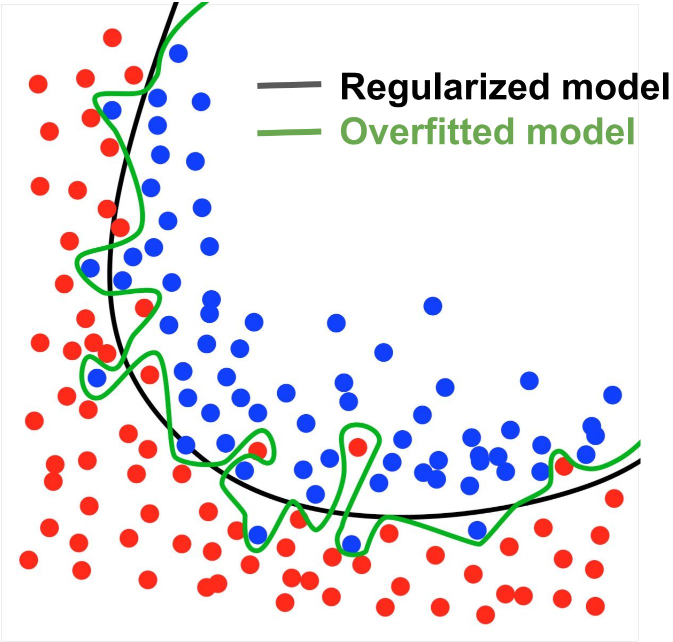
- Regularization: Add a penalty for the size of the parameters to make the prediction less sensitive to the training sample

- E.g. Ridge regression: minimize $\Sigma_i(h(x_i)-y_i)^2 + \lambda\Sigma_i\theta_i^2$ (from i=1)
  - The hyperparameter λ can be tuned with validation samples

Addendum

- Overfitting may happen for classification too!
  → Sometimes called “overtraining”

- Similarly one can ‘regularize’ the classification algorithm. We’ll touch upon the relevant ideas later.



Source: Wikipedia