Understanding Gradient Boosting as a gradient descent

- For a given sample $\mathrm{x_i}$, the predictions by a gradient-boosting regressor are given by

  $\Rightarrow \hat{\mathrm{y}}_\mathrm{i} = \sum_{\mathrm{m}=1}^{\mathrm{n_{iter}}} \mathrm{h_m}(\mathrm{x_i})$ where $\mathrm{h_m}$ is an instance of the base estimator
- It is often called a weak learner as it does not need to be extremely accurate
- It is mostly a decision tree and hence gradient boosting is generally performed using gradient boosting decision tree\
- The summation of all the base estimators gives the total gradient descent and each base estimator finds the gradient in its way in a function space unlike other algorithms where the most optimal parameter values are defined
- Gradient descent in parameter space: linear regression
  - The lease square loss is given by

    $\Rightarrow \mathcal{L} = \sum_i (\mathrm{y_i} - \hat{\mathrm{y}}_\mathrm{i})^2$

    $\Rightarrow \mathcal{L} = \sum_i (\mathrm{y_i} - \mathrm{x_i^T}\theta)^2$ where $\hat{\mathrm{y}}_\mathrm{i} = \mathrm{x_i^T}\theta$
  - The parameters are updated using the following

    $\Rightarrow \theta^{(\mathrm{m}+1)} = \theta^{(\mathrm{m})} - \alpha \frac{\partial \mathcal{L}}{\partial \theta^{(\mathrm{m})}}$
  - The optimal θ is the summation of all the terms where each term is the product of the learning rate and the negative gradient of the loss function

    $\Rightarrow \theta^{(\mathrm{m}+2)} = \theta^{(\mathrm{m}+1)} - \alpha \frac{\partial \mathcal{L}}{\partial \theta^{(\mathrm{m}+1)}} = (\theta^{(\mathrm{m})} - \alpha \frac{\partial \mathcal{L}}{\partial \theta^{(\mathrm{m})}}) - \alpha \frac{\partial \mathcal{L}}{\partial \theta^{(\mathrm{m}+1)}}$ using the power of induction
  - In gradient boosting, the gradient is performed with respect to the predictions $\hat{\mathrm{y}}_\mathrm{i}$
  - Since gradient descent is used to minimise the loss function with respect to any of the independent variables, the predictions $\hat{\mathrm{y}}_\mathrm{i}$ which is also an independent variable, can be used to minimise the loss function

- Minimising the loss with respect to the predictions
  - Predictions are randomly generated
  - The independent variables are updated using the following

    $\Rightarrow \hat{y}_i^{(m+1)} = \hat{y}_i^{(m)} - \alpha \frac{\partial \mathcal{L}}{\partial \hat{y}_i^{(m)}}$ where $\mathcal{L} = \sum_i (y_i - \hat{y}_i)^2$

    $\Rightarrow \frac{\partial \mathcal{L}}{\partial \hat{y}_i} = \frac{\partial}{\partial \hat{y}_i}(y_i - \hat{y}_i)^2$

    $\Rightarrow -2(y_i - \hat{y}_i)$
- Gradient Boosting: use the base estimator to predict the gradients
  - Prediction takes place on unseen samples
    - Instead of updating the $\hat{y}$ with the actual gradient, use a base estimator to predict the gradients
    - The set of base estimators can output a prediction for any sample
    - This can be represented by

      $\Rightarrow \hat{y}_i = \sum_{m=1}^{n_{iter}} h_m(x_i)$ where h_m is an instance of the base estimator
    - Instead of using gradient descent to estimate a parameter (a vector in a finite-dimensional space), use gradient descent to estimate a function (a vector in an infinite dimensional space)
- Wrapping up
  - Instead of taking the derivative of the loss with respect to the parameter of a parametrised model, take the derivative of the loss with respect to the predictions
    - There is no notion of a parametrised model anymore, but can still minimise our loss on the training samples
  - At each iteration of the gradient boosting procedure, train a base estimator to predict the gradient descent step
    - Saving these base estimators in memory is what enables to output predictions for any future sample