1. Fundamentals of Prediction

- In prediction problems, we **observe covariates** X and the **labels Y**
- We want to find a **prediction function** that maps the inputs to predict outputs
- To **measure** how good our prediction function **performs** we need **loss function** which penalize incorrect prediction
- Risk: expected loss
- Every prediction problem can be formulated as minimizing expected loss which is the Risk.
- So our goal is to find such predictor f that minimizes the risk

2. Risk Minimization

To perform risk minimization

- To find risk we need to choose two functions:
    - 1. The loss function, which can be a brier loss, MSE, 0-1 loss, etc
    - 2. Need a prediction function f(x)
- How to find the predictions function is, we need to choose a function class
    - Such as a simple linear regression, multiple linear regression, logistic, or any complex function class.
    - Among the chosen class we want to find a prediction function f that minimizes the risk
- From this graph, we wish to find the f_best, but we have to know the true population distribution which is very unlikely to happen in realistic
- So we can perform the Empirical Risk Minimization which is to find the predictor based on our samples or data collected.
- How can we find a better prediction function?
    - 1. If we fix the function class, we can find a better prediction function by increase our **sample size**, or use a **better optimizer**
    - 2. Or better to choose a wider function class, i.e. a more complex function class such as choosing from simple linear regression to multiple linear regression which simple linear regression is a subset of the multiple linear regression.

3. Dataset: `load_breast_cancer`

- We implement some risk minimization algorithm by hand
- Using a breast cancer dataset from `scikit-learn`
- This dataset has binary labels
- 30 features
- And the sample size is 569
- We use 4 ways to find the predictor
    - 1. first we assume the distribution, We assumed each feature follows a normal distribution within each class, but that is a strong assumption

    - 2. So then we find the best predictor, which we can do analytically. We calculate the OLS solution.
    - But we can't always do analytically and it can be hard. So instead we can explicitly minimize the function using gradient descent.

4. Gradient descent - Full-batch:
    - We have our loss function
    - We rewrite it
    - Then take the gradient
    - We want to update the weights until the gradient is 0 ideally, or close enough to 0 in practice
    - Here is how we update the weight in the negative gradient direction,
        - Because for a small enough learning rate we are guaranteed to lower our loss in the negative gradient direction
    - The learning rate is a hyperparameter which we want it to be large for faster convergence, but we don't want it to be too big to not be able to converge.

    - Here are the gradient norm and MSE vs epoch trainings
    - I tried a few learning rate and find out for this particular dataset we want to keep the learning rate under .00015,
    - Learning rate greater than that will not be able to converge, as the bottom two graphs shows

5. Stochastic Gradient Descent
    - Gradient descent updates the weights after each epoch, while SGD updates weights after each data point,
        - For example, we run 100 epochs, gradient descent update weights 100 times
        - While SGD update weights sample size * 100 times which can be very noisy
    - There are some tricks or variations we can do based on SGD, such as, mini-batch, shuffling, step decay, momentum, etc.
    - In my code I implement the standard SGD, shuffling, step decay and mini-batch.

    - And here is the gradient norm and MSE vs epoch graph.
    - We can see that shuffle and mini-batch about same noisy,
    - Step decay is little weird that it oscillate a bit at early epochs then become very stable
    - And the large learning rate of the standard SGD, the gradient norm is largely oscillating.
    - But all of them are converged
    - And have very close weights as the analytical solution
    - As well as shown in the loss

6. Generalization

- In addition to find a best prediction function among the training data
- Our ultimate goal is to find a prediction function that also do well on unseen data
- I.e. We want to minimize the generalization gaps as much as we can.