Linear Regression Introduction

Linear Basis Function Model

1. Least Square Model.

$$y(X, W) = W^T \phi(X)$$

$W = [w_0, w_1, \dots w_{M-1}]^T$ --- parameter $\in \mathbb{R}^M$

$\phi(X) = [1, \phi_1(X) \cdots \phi_{M-1}(X)]^T \in \mathbb{R}^M$

w_0 is the bias term.

$\phi_j(X)$ is known as <u>basis function</u>.

For polynomial basis function. $\phi_j(x) = x^j$

For gaussian kernal. $\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$

For sigmoid function, $\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$ $\sigma(a) = \frac{1}{1+\exp(-a)}$

Fourier basis

We consider the following additive noise ε

$$t = y(x, w) + \varepsilon$$

↑ received data ↑ true value ↑ $\mathcal{N}(0, \beta^{-1})$

β is called <u>precision</u>. which is the reciprocal of variance

So, we can have a series of pdf.

* For input x, the output is t. precision of the noise is β, then.

$$p(t|x, w, \beta) = \mathcal{N}(t|\underbrace{y(x, w)}_{\text{mean}}, \underbrace{\beta^{-1}}_{\text{variance}})$$

if we choose mean-square error. the optimal prediction for t would be $y(x, w)$ with the help of MAP/ML, also,

$$\mathbb{E}[t|x] = \int t \cdot p(t|x)dt = y(x, w)$$

* For many input $\vec{X} = [x_1, x_2 \dots x_N]^T$. corresponding output $T = [t_1 \dots t_N]^T$, we have joint pdf

$$p(T|\vec{X}, w, \beta) = \prod_{i=1}^{N} \mathcal{N}(t_i|y(x_i, w), \beta^{-1})$$

$$\Rightarrow \ln p(T|\vec{X}, w, \beta) = \sum_{i=1}^{N} \ln \frac{\sqrt{\beta}}{\sqrt{2\pi}} e^{-\frac{1}{2}(t_i - y(x_i, w))^2 \beta}$$

$$= \frac{N}{2}\ln\beta - \underbrace{\frac{N}{2}\ln(2\pi)}_{\text{const}} - \frac{1}{2}\beta \sum_{i=1}^{N}(t_i - y(x_i, w))^2$$

Now, use ML to compute w, β separately. $\because y(x_i, w) = w^T\phi(x_i)$ $\therefore \frac{\partial}{\partial w} y = \phi(x_i)$

** For w.

$$\frac{\partial}{\partial w}\ln p(T|\vec{X}, w, \beta) = -\frac{1}{2}\beta \sum_{i=1}^{N} 2(t_i - y(x_i, w)) \cdot (-\underbrace{\frac{\partial}{\partial w} y(x_i, w)}_{\uparrow})$$

$$= \beta \sum_{i=1}^{N} (t_i - w^T\phi(x_i)) \cdot \phi(x_i)^T$$

set the gradient to 0

$$0 = \beta \sum_{i=1}^{N} (t_i - w^T \phi(x_i)) \phi(x_i)^T$$

$$\Rightarrow \sum_{i=1}^{N} t_i \phi(x_i)^T = w^T \sum_{i=1}^{N} \phi(x_i) \phi(x_i)^T$$

$$\Leftrightarrow T^T \phi = w^T \phi^T \phi \Leftrightarrow \phi^T T = \phi^T \phi w \Rightarrow \boxed{w^* = (\phi^T \phi)^{-1} \phi^T T = \phi^\dagger T}$$

Note that

$$\phi = \begin{bmatrix} \phi_0(x_1) & \cdots & \phi_{M-1}(x_1) \\ \vdots & & \vdots \\ \phi_0(x_N) & & \phi_{M-1}(x_N) \end{bmatrix}_{N \times M}$$

** for β

$$\frac{\partial}{\partial \beta} \ln p(T | \underline{x}, w, \beta) = \frac{N}{2} \cdot \frac{1}{\beta} - \frac{1}{2} \sum_{i=1}^{N} (t_i - w^{*T} \phi(x_i))^2 = 0$$

$$\Rightarrow \boxed{\frac{1}{\beta} = \frac{1}{N} \sum_{i=1}^{N} (t_i - w^{*T} \phi(x_i))^2}$$

* The error

$$\boxed{E(w) = \frac{1}{2} \sum_{i=1}^{N} (t_i - w^T \phi(x_i))^2}$$

Definition of Error

$$= \frac{1}{2} \sum_{i=1}^{N} \left(t_i - w_0 - \sum_{j=1}^{M-1} w_j \phi_j(x_i)\right)^2$$

$$\frac{\partial}{\partial w_0} E = -\sum_{i=1}^{N} (t_i - w_0 - \sum_{j=1}^{M-1} w_j \phi_j(x_i)) = 0$$

$$\Rightarrow w_0 = \frac{1}{N} \sum_{i=1}^{N} t_i - \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M-1} w_j \phi_j(x_i)$$

↑ the bias

Geometry Interpretation of least square solution

Our target is to find a parameter W which will minimize the error function:

$$\min_{W} \frac{1}{2} \sum_{i=1}^{N} \left(t_i - W^T \phi(x_i)\right)^2$$

rewritten in matrix form

$$\min_{W} \frac{1}{2} \| T - \Phi(X) W \|_2^2$$

We can have the following interpretation. (two types.)

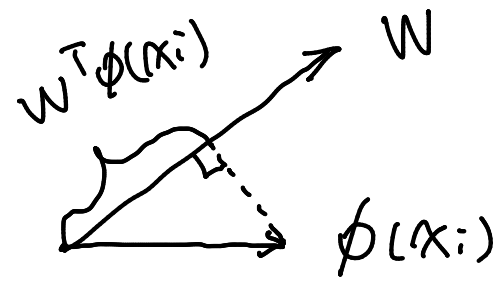
① $T \in \mathbb{R}^N$ target projection distance for N points.

$W \in \mathbb{R}^M$ is a normal vector of M-D plane.

$\phi(X) \in \mathbb{R}^{M \times N}$ is a set of N points.

So, $W^T \phi(X_i)$ is the projection distance of M-D vector $\phi(x_i)$

Our target is to find a normal, or a direction which will minimize the difference of projection distance and target distance.



② $T \in \mathbb{R}^N$ is a point in N-D space

$W \in \mathbb{R}^M$ is a normal vector of a M-D plane

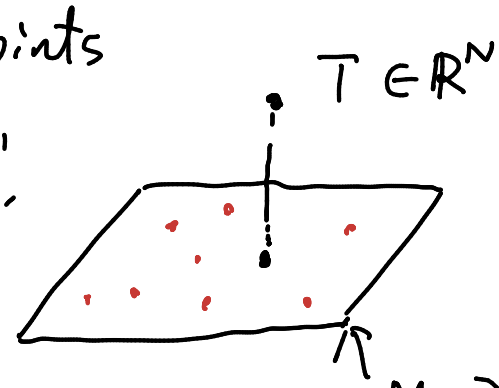
$\phi \in \mathbb{R}^{M \times N}$ are N points on M-D plane

Our target is to find the projection w such that the projection

of T onto the M-D space is orthogonal to the N-D plane.

Suppose the projection points of T on the M-D plane is T'.

Then $\frac{1}{2}\|T-T'\|_2^2$ is the error.



• points in M-D space.

We know that, when TT' is perpendicular to the M-D plane. $\|T-T'\|_2$ is the smallest. In mathmatical words, we should use "orthogonal" instead of "perpendicular".

Sequential Updating Method & Gradient Descent

Sequential Updating Method is an online algorithm, which is very useful when the data is extremely large.

Reasoning

From previous derivation, we know that the analytical solution is

$$W = \Phi^{\dagger} T$$

But the complexity for pseudo inverse is $O(n^3)$, so usually computing $\Phi^{\dagger}$ brutally is inpractical. So, we think of another method. Gradient descent.

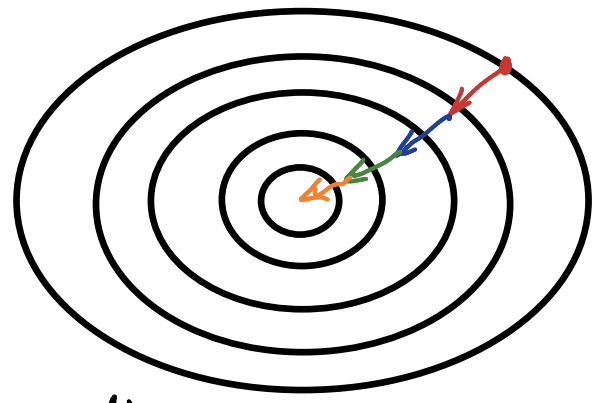
Gradient Descent has performance guarantee for convex problem. But for non-convex problem, no guarantee. but usually have good performance.

For Convex Problem, local maximum <=> global maximum. What Gradient Descent does is to go towards local maximum. So, we will always heading to the destination after iterations.

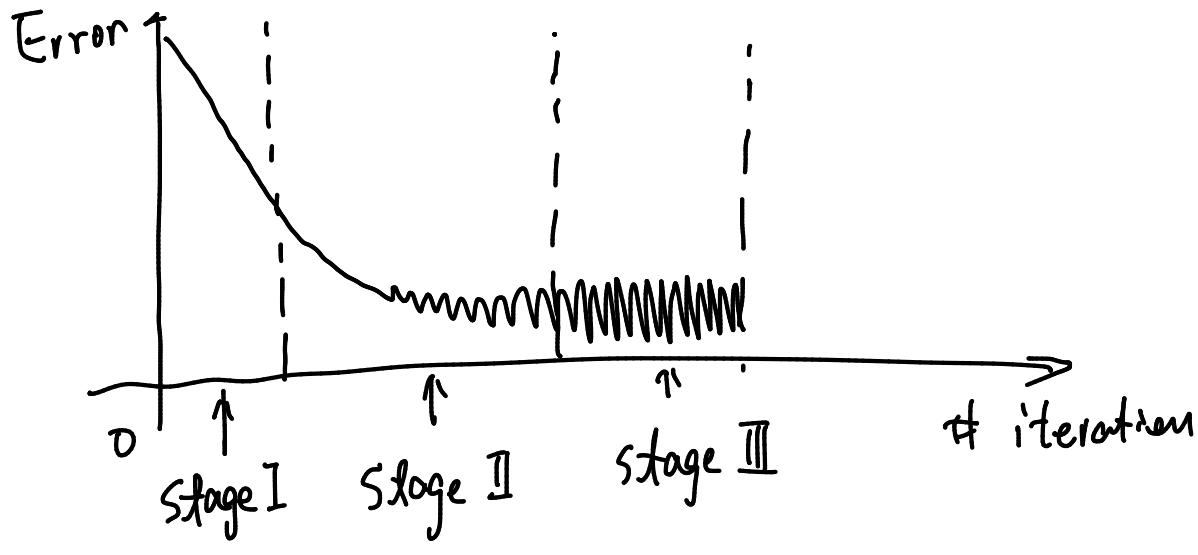
①

$$W = W - \eta \nabla_w E$$

↑ step size.



$-\nabla_w E$ is the gradient updating direction. Each time, we make progress to that direction by step η. Usually, the error would be



Stage I:

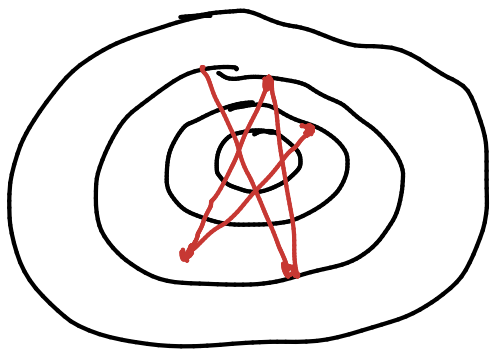
Step size η is not large enough. Each update is making roughly linear improvement.

Stage II.

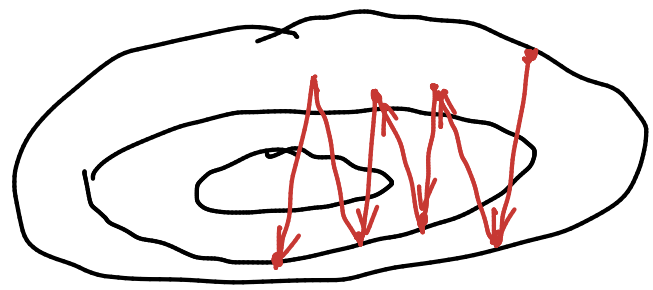
Two explanations

① We are approaching the destination. But η is a little bit larger. We are kind of rambling around the dest.

② We are approaching destination, but the gradient is not quite align with "true" direction.



Case I



case II

We can choose to decrease step size η to cope with case I. but for case II, Momentum Method may be better. Of course. Newton's method is the optimal solution.

Stage III

Each time, we are working cross the true point. So the performance is fluctuating. We should decrease step size η or terminate the iteration.

2. Regularized Least Square.

Sometimes, we know ahead of time that real coefficient W doesn't have weird parts (say big number), or it is sparse (only parts of W's coefficient are non-zero). But simply from previous algorithm, we could not have such constrain.

We want the cost function to consider something of W.

We have the following model.

$$\min_W \underbrace{\frac{1}{2}\|T-\Phi W\|_2^2}_{\text{data-dependent error}} + \underbrace{\lambda}_{\text{regularization coefficient}} \underbrace{f(W)}_{\text{regularization term}}$$

We can have many choices for the regularization term.

Usually, we use p-norm $\|W\|_p$, especially 2-norm, 1-norm.

$$\|W\|_p = \sqrt[p]{\sum_{i=1}^{N} |W_i|^p} \quad . \quad p>0$$

1-norm

$$\|W\|_1 = |W| = \sum_{i=1}^{N} |W_i|$$

2-norm

$$\|W\|_2 = \|W\| = \sqrt{W^T W}$$

∞-norm

$$\|W\|_\infty = \max_i (W_i)$$

Same as before, we start from 2-norm's regularization term.

$$\min_W \frac{1}{2}\sum_{i=1}^{N}\left[t_i - W^T\phi(x_i)\right]^2 + \frac{1}{2}\lambda W^T W$$

rewritten the function to matrix multiplication form.

$$\min_W \frac{1}{2}\left\| T - \Phi W \right\|_2^2 + \frac{1}{2}\lambda W^T W$$

$$\Leftrightarrow \min_W (T-\Phi W)^T (T-\Phi W) + \lambda W^T W$$

$$\Leftrightarrow \min_W \cancel{T^T T} - T^T\Phi W - W^T\Phi^T T + W^T\Phi^T\Phi W + \lambda W^T W$$

$\cancel{T^T T}$: no W

$T^T\Phi W$, $W^T\Phi^T T$: Scalar

$\lambda W^T W \Leftrightarrow \lambda W^T I W$

$W^T\Phi^T\Phi W + \lambda W^T I W$: merge

$$\Leftrightarrow \min_W \underbrace{-2T^T\Phi W + W^T(\Phi^T\Phi + \lambda I)W}_{f(W)}$$

$$\frac{\partial}{\partial W} f(W) = -2(T^T\Phi)^T + 2(\Phi^T\Phi + \lambda I)W = 0$$

$$\Rightarrow (\Phi^T\Phi + \lambda I)W = \Phi^T T \Rightarrow \boxed{W = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T T}$$

inverse!

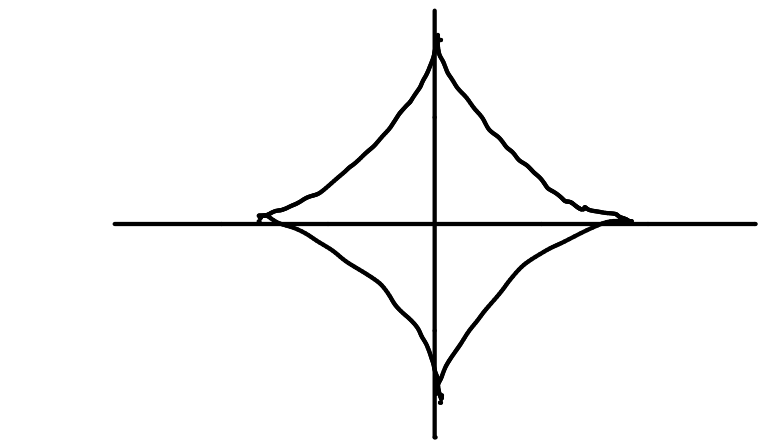
PS: $\Phi^T\Phi + \lambda I$ is PSD, so we can do inverse instead of psendo inverse.

because $\forall x \in \mathbb{R}^M$ $\quad x^T(\Phi^T\Phi + \lambda I)x = x^T\Phi^T\Phi x + \lambda x^T x$

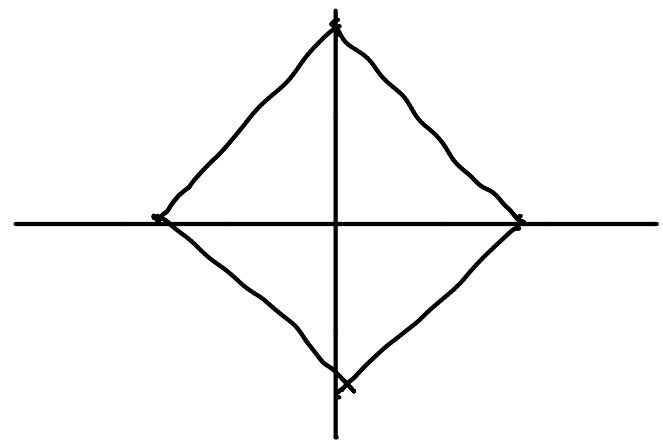
$$= \|\Phi x\|_2^2 + \lambda\|x\|_2^2 \geq 0$$

What's the difference between different norm?

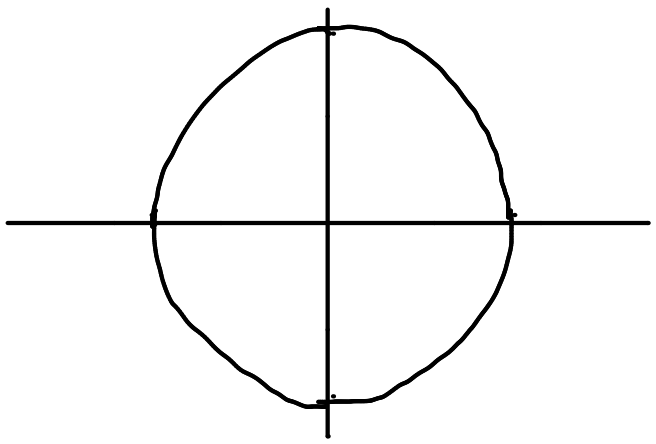
Now, let's see the feasible region for 0.5-norm, 1-norm, 2-norm and 4-norm



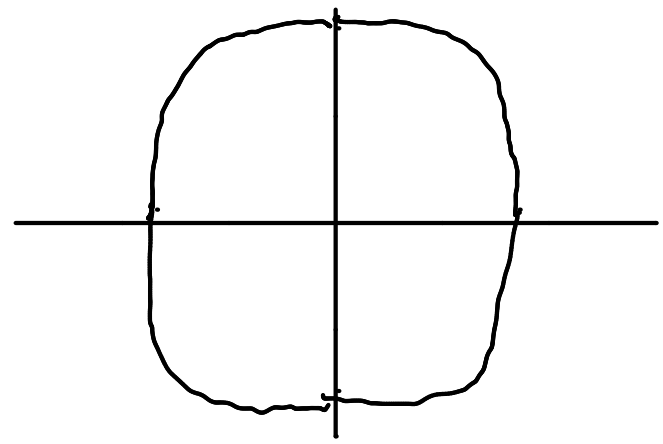
0.5-norm



1-norm



2-norm



4-norm

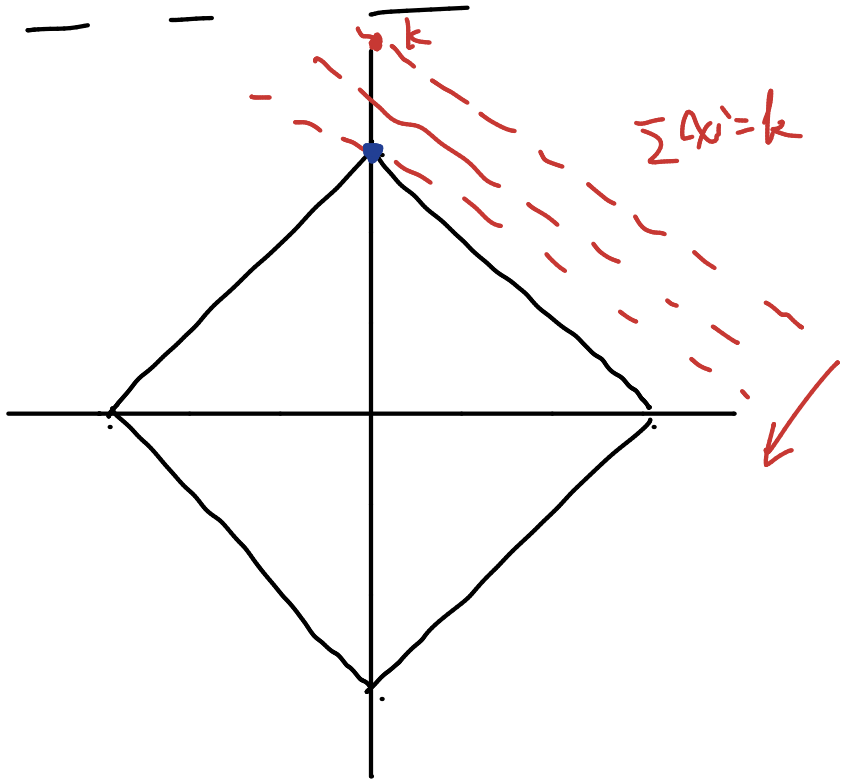
① From the figures, we know that 0.5-norm is non-convex.

② If we use 0.5-norm, 1-norm, we are prone to have sparse output.

Let's see an linear programming problem

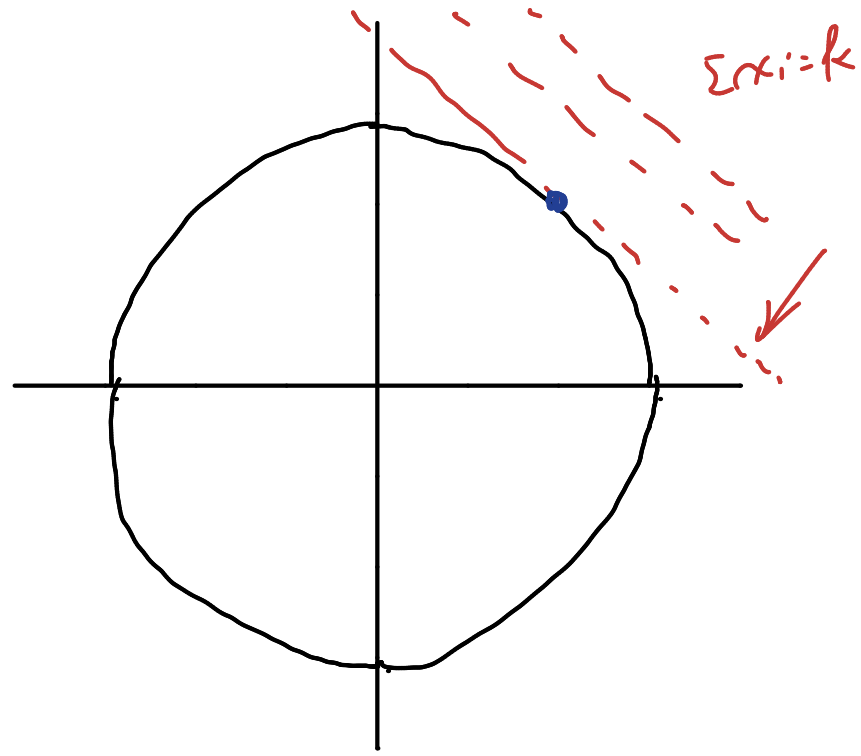
$\min \sum x_i$

s.t. $|x| \le 1$



$\min \sum x_i$

s.t. $\|x\| \le 1$



For 1-norm situation, we will usually have a final point on one axis. So, the final result x's sparse, because many other components are 0

For 2-norm, we won't always have such result.

Typically, for 1-norm case, the method is called lasso.

3. The Bias - Variance Decomposition

If we set our model to be extremely complicated, we may easily fall to over-fitting. If we restrict the complexity, we then limit the flexibility of the model. Regularization term is a good way to balence model complexity and overfitting, but how to set coefficient λ?

Recall that, if our loss function is the "squared" loss function

$$L(x) = x^2$$

the optimal estimator would be the conditional expectation

$$h(X) = \mathbb{E}[t|X]$$

X is the observation, t is the parameter you want to estimate (truth)
$h(X)$ is the optimal predictor

In real application, we can never know $h(x)$ because we don't know the true pdf behind. So, what we can do is to use some other prediction model $y(X)$. Then, we can have our loss

$$\mathbb{E}[L] = \underbrace{\int (y(X) - h(X))^2 p(x)\,dx}_{(I)} + \underbrace{\int (h(x) - t)^2 p(x,t)\,dx\,dt}_{(II)}$$

part(I) is the difference between our estimator and optimal one. This term is always non-negative. If we have sufficiently large dataset, we can make this number small.

part(II) is the error we can never overcome, as you can see, there is nothing we can control. This is the noise.

In reality, what we have is a dataset D. which is sampled from $p(X, t)$.

For any given dataset D, we can run the prediction function and have $y(X; D)$

Different dataset will give different $y(X; D)$.

Now, part(I) will turn to

$$\int (y(X;D) - h(X))^2 p(X) dx$$

$$(y(X;D) - h(X))^2$$

$$= (y(X;D) - \mathbb{E}_D[y(X;D)] + \mathbb{E}_D[y(X;D)] - h(X)]^2$$

$$= (y(X;D) - \mathbb{E}_D[y(X;D)])^2 + [\mathbb{E}_D[y(X;D)] - h(X)]^2$$

$$+ 2(y(X;D) - \mathbb{E}_D[y(X;D)])(\mathbb{E}_D[y(X;D)] - h(X))$$

Now, we take expectation, note that

$$\mathbb{E}_D[2(y(X;D) - \mathbb{E}_D[y(X;D)])(\mathbb{E}_D[y(X;D)] - h(X))]$$

$$= 2(\mathbb{E}_D[y(X;D)] - h(X)) \cdot \underbrace{\mathbb{E}_D[y(X;D) - \mathbb{E}_D[y(X;D)]]}_{=0}$$

$$= 0$$

$$\therefore \mathbb{E}_D[(y(X;D) - h(X))^2]$$

$$= \underbrace{\mathbb{E}_D[(y(X;D) - \mathbb{E}_D[y(X;D)])^2]}_{\text{Variance.}} + \underbrace{[\mathbb{E}_D[y(X;D)] - h(X)]^2}_{(\text{bias})^2}$$

So, we can split the expected loss to

expected loss = $(\text{bias})^2$ + variance + noise.

$$(\text{bias})^2 = \int \left(\mathbb{E}_D[y(x;D)] - h(x)\right)^2 p(x)\,dx$$

$$\text{variance} = \int \mathbb{E}_D\left[\left(y(x;D) - \mathbb{E}_D[y(x;D)]\right)^2\right] p(x)\,dx$$

$$\text{noise} = \int (h(x) - t)^2 p(x,t)\,dx\,dt$$

Remind that, our target is to minimize the overall loss, i.e. minimize the expected loss.

Noise is unsolvable. it's there, and we cannot deal with that.

But there is a trade-off between bias - variance.