INF8953CE - Fall 2020

Machine Learning

- Sarath Chandar

2. Statistical Decision Theory.

2. Statistical Decision Theory

Regression:-

$X \in \mathbb{R}^p$ — real valued random input vector.

$y \in \mathbb{R}$ — real valued random output variable.

$Pr(X, Y)$ - joint distribution of X, Y.

Goal: find a function $f(x)$ for predicting Y given values of X.

$L(Y, f(x))$ = loss function for penalizing errors in prediction.

Common loss function: Squared error loss.

$$L(Y, f(x)) = (Y - f(x))^2$$

Expected prediction error:-

$$EPE(f) = E\left[(Y - f(x))^2\right]$$

$$= \int\int (Y - f(x))^2 P(x, y)\, dx\, dy$$

$$= \int_x \boxed{\int_y (y - f(x))^2 p(y|x)\, dy}\, p(x)\, dx$$

$$= \int_x E_{y|x}\left[(y-f)^2 \,|\, x=x\right] p(x)\, dx$$

$$= E_x\, E_{y|x}\left[(y-f)^2 \,|\, x=x\right]$$

It suffices to minimize EPE pointwise:

$$f^* = \underset{f}{\operatorname{argmin}}\; E_{y|x}\left[(y-f)^2 \,|\, x=x\right]$$

$$\frac{\partial}{\partial f}\left\{ E_{y|x}\left[(y-f)^2 \,|\, x=x\right] \right\} = 0$$

$$\frac{\partial}{\partial f}\left\{ E_{y|x}\left(y^2 | x=x\right) + f^2 - 2\, E_{y|x}\left(y | x=x\right) f \right\} = 0$$

$$2f - 2\, E_{y|x}\left(y | x=x\right) = 0$$

$$\boxed{f^* = E_{y|x}\left(y | x=x\right)}$$

↑ regression function.

The best prediction of Y at any point $X = x$ is the Conditional mean, when best is measured by average squared error.

What is Nearest Neighbor (N.N) doing?

$$\hat{f}(x) = \text{Ave}\left(y_i \mid x_i \in N_k(x)\right)$$

Two approximations to the regression function:

1. Expectation is approximated by averaging over sample data.
2. Conditioning at a point is relaxed to Conditioning on some region "close" to the target point.

Note 1: For large training sample size N, the points in the neighborhood are likely to be close to x.

Note 2: As 'k' gets larger, the average will get more stable.

Under mild regularity Conditions on $Pr(x, y)$, one can show that

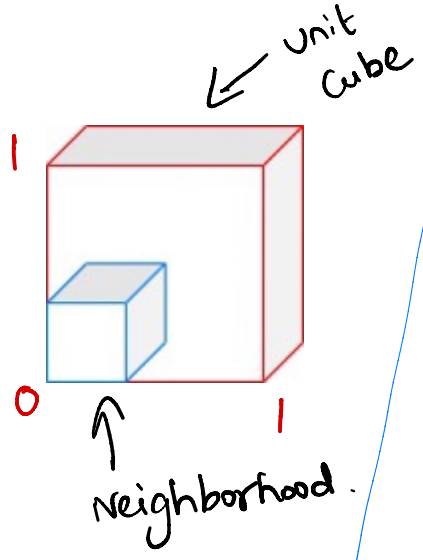
as $N, k \longrightarrow \infty$, such that $k/N \longrightarrow 0$,

$$\hat{f}(x) \longrightarrow E(y \mid x = x)$$

Looks like we have an universal function approximator !?

No! As the number of features increases, in high dimensions, we need very large samples.

Local methods in high dimensions :-



Consider inputs uniformly distributed in a p-dimensional hypercube.

Consider a hypercubical neighborhood around a target point to Capture a fraction 'r' of the observations.

This corresponds to a fraction 'r' of the unit volume.

Expected edge length $e_p(r) = r^{1/p}$

$$e_{10}(0.01) = 0.63$$

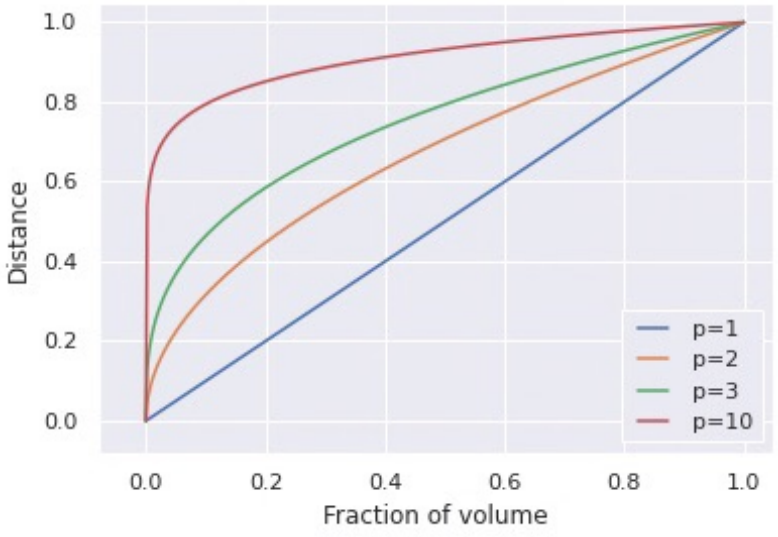
$$e_{10}(0.1) = 0.80$$

Entire range for each input is only 1.0.

To capture 1% or 10% of data, to form a local average, we must cover 63% or 83% of the range of each input variable.

⇒ Such neighborhood are no longer "local".

⇒ if you reduce 'r', with fewer observations, variance will be high.



This is known as "Curse of dimensionality" (Bellman, 1961)

Warning :- Our intuitions often break down in higher dimensions.

What is linear regression doing ?

$$f(x) \approx x^T \omega$$

- This is a model-based approach. We are specifying a model for the regression function.

$$\omega = [E(xx^T)]^{-1} E(xy) \quad \text{---} \circledast$$

Note : We have not conditioned on x. rather we have used our knowledge of the functional relationship to pool over values of x.

Approximation :- Replace the expectation in $\circledast$ by averages over the training data.

Assumptions :-

① Linear regression assumes $f(x)$ is well approximated by a globally linear function.

② k-nearest neighbor assumes $f(x)$ is well approximated by a locally constant function.

These assumptions made by any algorithm is known as the inductive bias of the algorithm.

What happens if we replace L_2 loss with L_1 loss?

$$L_1 : E|Y - f(x)|$$

Solution : $\hat{f}(x) = \text{median}(Y|X=x)$

which is a different measure of location, and its estimates are more robust than those for the Conditional mean.

Historically, L_2 is more popular than L_1 because of the discontinuities in the derivatives of L_1.

Classification :-

Output : Categorical variable G.

Same paradigm works here.

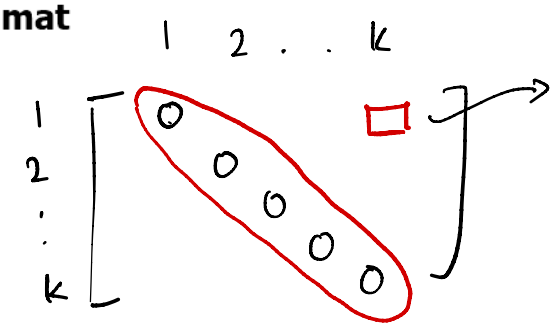
We need a different loss fn. for penalizing prediction errors.

$\mathcal{G}$ - set of possible classes.

$\hat{G}$ - estimate.

loss fn : $K \times K$ matrix where $K = \text{card}(\mathcal{G})$

confusion mat



Price paid for classifying an observation belonging to class 1 to class k.

$L(k, \ell)$ - price for classifying an observation belonging to class k to class ℓ.

Common loss : 0/1 loss function.

0 for correct classification.

1 for misclassification.

$$EPE = E\left[L(G, \hat{G}(x)) \right]$$

the output here is discrete, hence we use summazation. In the top the o/p is continous, hence we used integral

Expectation is w.r.t. $Pr(G, x)$

$$= E_X \sum_{k=1}^{K} L[G_k, \hat{G}(x)] \, Pr(G_k | x)$$

again it suffices to minimize EPE point wise:

$$\hat{G}(x) = \underset{g \in \mathcal{G}}{\operatorname{argmin}} \sum_{k=1}^{K} L(G_k, g) \, Pr(G_k | X = x)$$

For 0/1 loss:

$$\hat{G}(x) = \underset{g \in \mathcal{G}}{\operatorname{argmin}} \left[1 - Pr(g | X = x) \right]$$

(or)

$$\hat{G}(x) = \underset{g \in \mathcal{G}}{\operatorname{argmax}} \left[Pr(g | X = x) \right]$$

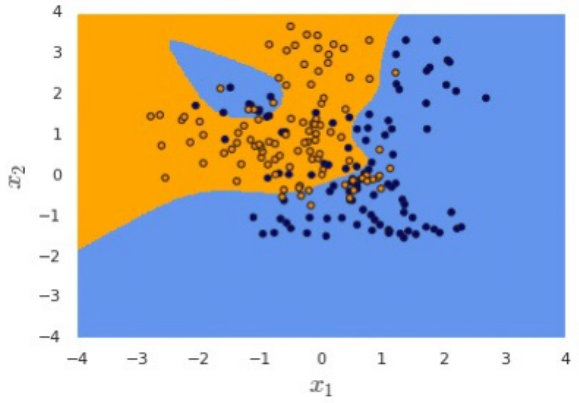
⇒ This solution is known as the Bayes classifier.

⇒ we classify to the most probable class, using the conditional distribution $Pr(G|x)$.

⇒ The error rate of the Bayes classifier is called the Bayes error rate.

We cannot always calculate the bayes error, since we dont the distribution

Note: Bayes error rate for a particular classification problem is the lowest possible error rate any classifier can achieve.



The optimal Bayes decision boundary for the toy task.

K-Nearest neighbor directly approximates this solution.

$\Rightarrow$ Majority voting in the neighborhood.

① Conditional prob. at a point is relaxed to Conditional prob. within a neighborhood of a point.

② Probabilities are estimated by training same proportions.

You should know!

1. Expected prediction error
2. Regression function
3. Curse of dimensionality
4. Inductive bias of an algorithm.
5. Model based algorithms.
6. o/1 loss.
7. Bayes classifier
8. Bayes rate.