# LR: Probabilistic Foundations and Weighted Least Squares

## Locally weighted Linear Regression

### 🔷 1. Standard Linear Regression

We fit parameters θ by minimizing:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$$

- Every data point contributes **equally** to the loss.

---

### 🔷 2. Locally Weighted Linear Regression

Instead of giving **equal importance** to all points, LWLR assigns **weights** $w^{(i)}$ based on how close each training point $x^{(i)}$ is to the query point $\bar{x}$ (the point where we want a prediction).

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} w^{(i)} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$$

- Here, $w^{(i)}$ is large if $x^{(i)}$ is close to $\bar{x}$.
- $w^{(i)}$ is small if $x^{(i)}$ is far away.
- This makes the regression **local**.

---

### 🔷 3. Weight Function

Your notes show:

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - \bar{x})^T (x^{(i)} - \bar{x})}{2\tau^2}\right)$$

- Gaussian (exponential) kernel.
- $\tau$ = bandwidth parameter:
  - **Small $\tau$:** only nearby points matter (very local).
  - **Large $\tau$:** distant points also matter (approaches global regression).

---

## 🔷 4. Prediction

Once $\theta^*$ is found (solved for each query $\bar{x}$), the prediction is:

$$\hat{y} = \theta^{*T} \bar{x}$$

---

## 🔷 5. Intuition

- Imagine you want to predict at a point $\bar{x}$.
- Instead of fitting one global line for all data, LWLR fits a **new weighted linear regression** around that point, where nearby data influences more.
- This gives a flexible, non-linear decision boundary even though we're using linear regression locally.

---

✅ **Summary:**

- LWLR is a **non-parametric** algorithm (parameters change for each query).
- It uses a **weighted least squares** approach where weights depend on distance to the query point.
- Good for datasets where the relationship between x and y is **not globally linear** but **locally linear**.

---

# Put simply

## Standard Linear Regression

- You take **all points**.
- You try to find **one best straight line** that goes through them (minimizing overall error).
- Every point is treated as **equally important**.

---

## Locally Weighted Linear Regression (LWLR)

- Suppose you want to predict at a specific point $x_0$.
- LWLR says: **"Pay more attention to the training points near $x_0$, and less attention to points far away."**
- So it builds a **mini line** around that neighborhood.
- Every time you want a prediction at a new place, LWLR makes a new local line.

---

## Analogy 🎯

- Imagine drawing a curve with a ruler:
  - Ordinary Linear Regression = put one **long ruler** across the whole data → one straight line.
  - LWLR = move a **small ruler** point by point → at each spot, fit a little line using nearby points.
  - Stitching those local lines together looks like a smooth curve.

---

## Why it's useful?

- Works well when data is **nonlinear** (curvy).
- But expensive → because you must refit a line for **every prediction**.

---

👉 So, **in one line**:

**Linear Regression = one global line. LWLR = many local lines fitted on the fly, giving flexibility.**

---

In **Locally Weighted Linear Regression (LWLR)**, there's **no traditional "training" step** where you compute and store fixed model parameters (like slope &

intercept).

---

### How it works:

1. **In standard linear regression**
   - You train once → find one global slope & intercept.
   - Then you use them for any new prediction.
2. **In LWLR**
   - You don't pre-compute a global model.
   - For **each new prediction point** $x_0$:
     - Compute weights for all training points (near ones get high weight, far ones get low).
     - Solve a **small weighted regression** just for that location.
   - So the model is built **on the fly** for every prediction.

---

### ✅ Intuition:

It's like asking:

- *"If I only cared about the neighborhood around this point, what line would best fit that?"*
- Then use that line to predict at $x_0$.

---

### 🚨 Downside:

Since you fit a mini-model for each new query, it's **computationally expensive** (slow for large datasets).

---

# Probabilistic Approach of Linear Regression

## 1. Standard Linear Regression Setup

We assume the data is generated like this:

$$y_i = \theta^T x_i + \xi_i$$

- y_i: observed output
- $\theta^T x_i$: deterministic linear part (our prediction)
- $\xi_i$: noise (random error, accounts for randomness/unseen factors)

Now the **assumptions about noise $\xi_i$:**

1. **Noise is Gaussian:**

   $$\xi_i \sim \mathcal{N}(0, \sigma^2)$$

   → Mean = 0, Variance = $\sigma^2$.

   → Means predictions are centered around the true values, but with some spread.

2. **Independent noise:**

   Each data point's noise is independent of the others.

   So, joint probability of all errors = product of their individual probabilities.

---

# 2. Probabilistic Assumption

We assume the noise is **Gaussian distributed**:

$$\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$$

That means:

$$y^{(i)}|x^{(i)} \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$$

👉 Interpretation: Given input x^{(i)}, the output y^{(i)} is normally distributed around \theta^T x^{(i)}.

---

# 3. Likelihood Function

Given the assumptions, the probability of observing one y(i)y^{(i)}y(i) is:

$$P(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

For **all data points (i = 1 to n)**, since errors are independent:

$$L(\theta) = \prod_{i=1}^{n} P(y^{(i)}|x^{(i)};\theta)$$

This is the **likelihood function**: how likely our parameters θ explain the data.

---

## 4. Log-Likelihood

We take the log for simplicity:

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)\right)$$

$$\ell(\theta) = n \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2$$

---

## 5. Maximizing Likelihood ⇔ Minimizing MSE

- First term ($\log \frac{1}{\sqrt{2\pi\sigma^2}}$) is constant w.r.t θ.
- Second term involves the **sum of squared errors (SSE)**.

So:

$$\theta^* = \arg\max_{\theta} \ell(\theta) = \arg\min_{\theta} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2$$

✅ **Key Insight:**

The **probabilistic interpretation** shows that minimizing **MSE in linear regression** is equivalent to **Maximum Likelihood Estimation (MLE)** under the assumption that errors are Gaussian, zero-mean, and independent.

---

## 🎯 Key Takeaway:

- Linear regression with squared error is not just algebra—it's **MLE under Gaussian noise assumption**.
- That's why in your notes it says:

$$\theta^* = \arg\min_{\theta} J(\theta)$$