GD — Gradient Descent $\xrightarrow{?}$ SGD — Stochastic Gradient Desent

$$\frac{\partial}{\partial \underline{w}} L(D;\underline{w}) = \sum_{\boxed{i \in D}} \frac{\partial}{\partial \underline{w}} l(\underline{x}_i, y_i; \underline{w})$$

GD is problematic if $|D|$ is big

while $\left|\frac{\partial}{\partial w} L\right| > 0.0001$:

$$\boxed{\underline{w}_{t+1} = \underline{w}_t - \alpha_t \left[\frac{\partial}{\partial \underline{w}} L(D;\underline{w})\right]^T}$$

while
{ for B in D:

$$\underline{w}_{t+1} = \underline{w}_t - \alpha_t \sum_{i \in B} \frac{\partial}{\partial \underline{w}} l(\underline{x}_i, y_i; \underline{w})$$

} → EPOCH

[One iteration over entire Dataset]

Randomly chosen BATCH of data from the dataset D

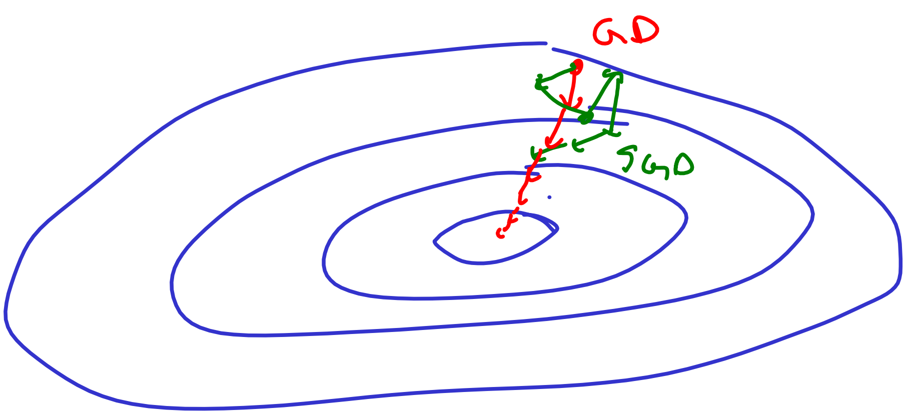
$|B| = 1$ - SGD

$|B| \approx 64, 32, 128, 256, 512$

Batch SGD

Advantages of SGD

1. Lower memory requirement
2. Making progress with weights without having to wait full epoch



Optimization Perspective on ML

- Data $\quad D = \{(\underline{x}_i, y_i) \dots\}$
- Select Model $\quad \hat{y}_i = f(\underline{x}_i; \underline{w})$
- Loss $\quad l(y_i, \hat{y}_i)$

→ - Training by GD

$$\underline{w}^* = \arg\min_{\underline{w}} \; L(D;\underline{w}) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, \hat{y}_i)$$

→ - New data $\underline{x}^* \notin D$

$\boxed{\text{Test Data}}$

$$\hat{y}^* = f(\underline{x}^*; \underline{w}^*)$$

correct?

weight was optimized on the Training data

Training Data ⟺ Test Data ← Expected situation data

(a) R^2 value?

Mathematical

All training and test data samples must be IDENTICALLY distributed.

$$(\underline{x}_i, y_i) \sim \mathbb{P}(X, Y) \quad \text{Training}$$

$$\updownarrow$$

$$(x^*, y^*) \sim \mathbb{P}(X, Y) \quad \text{Test / Expected data where your system is supposed to work}$$

In ML, all data must be (i.i.d. assumption) INDEPENDENT and $\underbrace{\text{IDENTICALLY DISTRIBUTED}}$

$(\underline{x}_i, y_i) \in D$, $(\underline{x}_j, y_j) \in D$

$$(\underline{x}_i, y_i) \perp (\underline{x}_j, y_j)$$

Probabilistic independence $Z_1 \perp Z_2$

① $\mathbb{P}(Z_1, Z_2) = \mathbb{P}(Z_1)\mathbb{P}(Z_2)$ ⟺ ② $\mathbb{P}(Z_1 | Z_2) = \mathbb{P}(Z_1)$

Why do we need Independence assumption?

Optimization $\xleftrightarrow[\text{assumption}]{\text{Independence}}$ Probabilistic perspective

$$\arg\max_{\underline{w}} P(D \mid \underline{w})$$

Maximum likelihood estimation

$$P(D \mid \underline{w})$$

$$= P\big((\underline{x}_1, y_1) \ldots\ldots (\underline{x}_n, y_n) \mid \underline{w}\big)$$

$$= \prod_{i=1}^{n} P\big((\underline{x}_i, y_i) \mid \underline{w}\big)$$

$$\arg\max_{\underline{w}} \prod_{i=1}^{n} P\big((\underline{x}_i, y_i) \mid \underline{w}\big)$$

$$P(Z_1, Z_2) = P(Z_1)P(Z_2)$$

$$\arg\min_{\underline{w}} \frac{1}{n} \sum_{i=1}^{n} l(y_i, \hat{y}_i; \underline{w})$$

$$P\big((\underline{x}_i, y_i) \mid \underline{w}\big) = \frac{1}{Z} \exp\big(-l(y_i, \hat{y}_i; w)\big)$$

Z: normalizing factor

$$= \arg\max_{\underline{w}} -\frac{1}{n} \sum_{i=1}^{n} l(y_i, \hat{y}_i; w)$$

exponential is a monotonically increasing function

$$= \arg\max_{\underline{w}} \exp\Big(-\frac{1}{n} \sum_{i=1}^{n} l(y_i, \hat{y}_i; \underline{w})\Big)$$

$$= \arg\max_{\underline{w}} \prod_{i=1}^{n} \exp\Big(-\frac{1}{n} l(y_i, \hat{y}_i; w)\Big)$$

→ i.i.d assumption

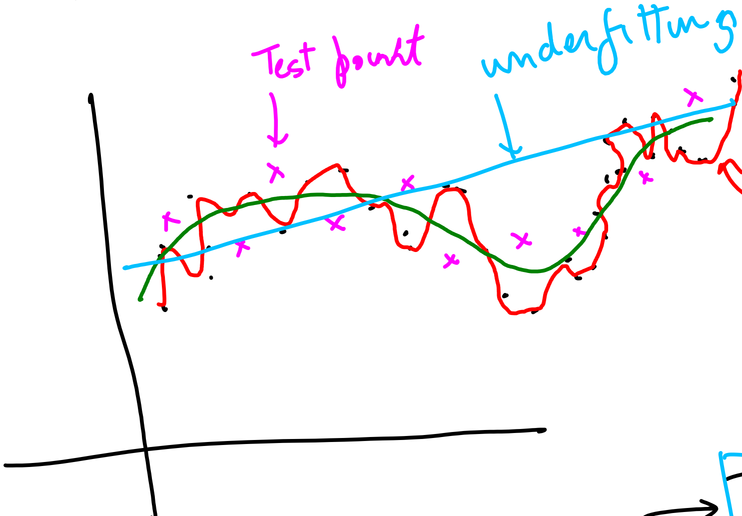
Beyond Linear Models:

→ Does minimizing the training loss minimize the test loss?

a) Always

b) Sometimes → When it does / When it does not? (Overfitting)

→ How can we ensure that the gap is small?



complex model

overfitting

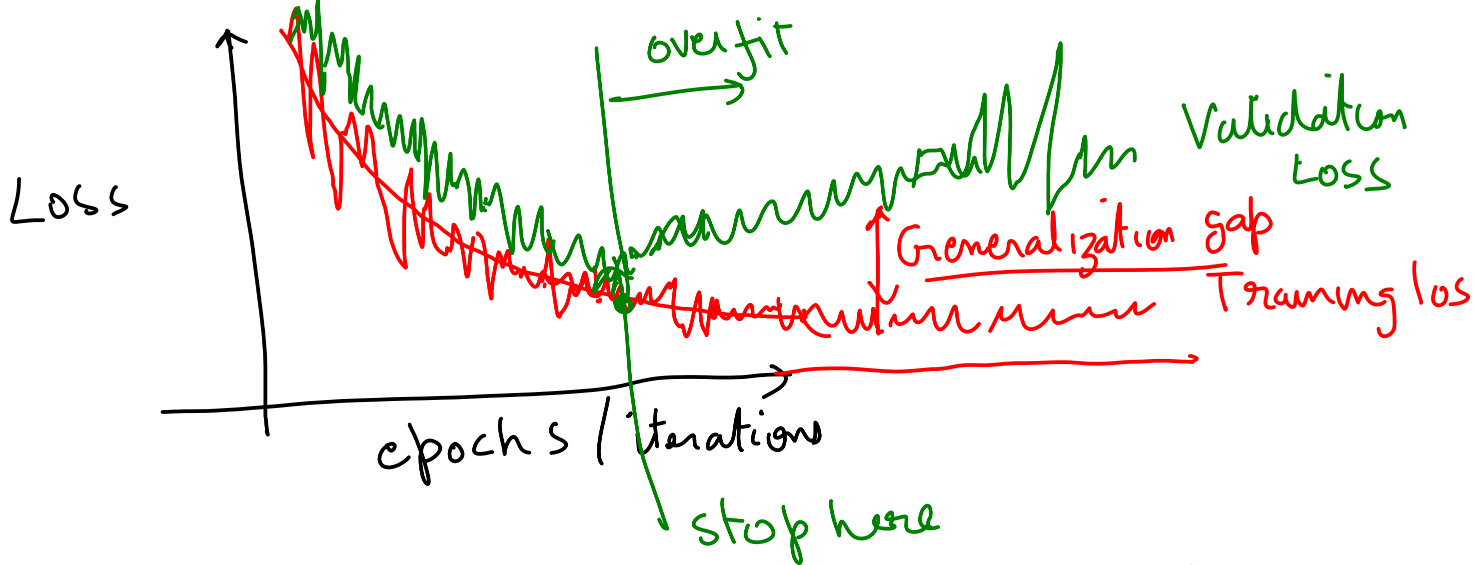
Occam's Razor

simplest explanation

Practically speaking → Data
- Training 60% — minimize loss on
- Validation data 20% — check for overfitting or underfitting
- Test Data 20% — Evaluation only

Overfitting means Test loss >> Training loss

detected using Validation loss >> Training loss



Early stopping technique

$$X_1, \dots, X_n \sim \mathbb{P}_X(X)$$

Expectation

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x\, f(X=x)\, dx = \sum_{x} x\, P(X=x)$$

Sample mean

$$\frac{1}{n}\sum_{i=1}^{n} X_i$$