ECE 6143: Intro to ML
Lecture 4

- Recap: GD
- SGD
- Model regularization.

Recap: Gradient descent

$\{x_i, y_i\}$ $i = 1, \ldots, n$

Find linear f s.t. $y_i \approx f(x_i)$.

Alg 1: Matrix inversion

$y = Xw$ where $w = \underbrace{(X^T X)^{-1} X^T y}$

slow.

Alg 2: Gradient descent

Iterate for $k = 0, 1, \ldots, T$

$w \leftarrow w - \alpha \nabla L(w)$

$L = \frac{1}{2} \|y - Xw\|_2^2$

$w_{k+1} = w_k + \alpha_k X^T (y - X w_k)$.

Pros: 1) Efficient running time
$O(ndT)$ $T = \log_{1/\rho} \frac{\|w^*\|_2}{\epsilon}$

$\rho = \frac{1 - \frac{1}{L}}{1 + \frac{1}{L}}$.

2) Simple to implement.

Cons: 1) Possible that it gets in local minimum.

2) Need to choose α, T.

↳ can be set via "line-search".

3) Requires multiple passes over the data. [not memory efficient].

4) $O(nd)$ can be intractable for very large n, d.

Stochastic Gradient Descent (SGD).

"Backpropagation"

Q. How to speed up gradient descent?

GD:

$$w_{k+1} = w_k + \alpha_k \underbrace{X^T(y - Xw_k)}$$

$$= w_k + \alpha_k \sum_{i=1}^{n} (y_i - \langle w_k, x_i \rangle) x_i$$

$$= w_k + n\alpha_k \underbrace{\frac{1}{n} \sum_{i=1}^{n} (y_i - \langle w_k, x_i \rangle) x_i}$$

↑ Average of $(y - \langle w_k, x \rangle)x$

Idea: Instead of computing average over all data points, pick a subset S uniformly at random.

$$\text{(SGD)} \quad w_{k+1} = w_k + \alpha_k \sum_{i \in S} (y_i - \langle w_k, x_i \rangle) x_i$$

"Stochastic gradient descent"

Size of S → whatever you like! (even a single point).

SGD (single data point)

Iterate:

① Choose $i \in \{1, 2, \dots, n\}$ uniformly at random, select (x_i, y_i)

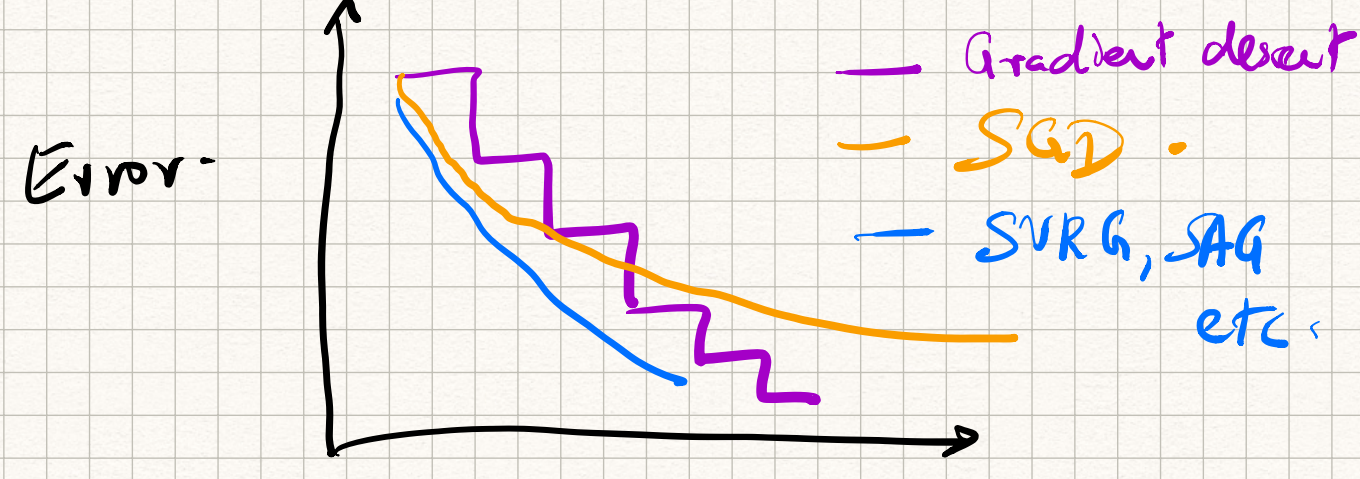
② $w_{k+1} = w_k + \alpha_k x_i (y_i - \langle w_k, x_i \rangle)$

while not max_epochs.

Running time: $O(d \cdot \#\text{epochs})$.

d: per iteration d.

If learning rate $\alpha_k \propto \frac{1}{k}$ then $\#\text{epochs} \propto \frac{1}{\varepsilon}$.



Wall-clock time.

Model selection

Are linear models the right thing to do?

① Data x, label y is a nonlinear function of x.

Solution: Preprocess x to

$[x, x^2, x^3, \ldots, x^d, \ldots, x^{-1}, \exp(-x), \ldots]$

New Problem: Overfitting!

② Challenge in very large datasets:

Not all features are relevant.

Not overfit?

e.g. For glucose prediction, we only need a subset of {weight, age, SpO_2, WBC count, sugar level, ...}.

③ Insufficient data.

LR: $w = (X^T X)^{-1} X^T y$.

$X^T X$: $d \times d$ matrix.

Invertible when $\text{rank}(X^T X) = d$.

$$\begin{bmatrix} 1 & 1 & & 1 \\ x_1 & x_2 & \cdots & x_n \\ 1 & 1 & & 1 \end{bmatrix} \begin{bmatrix} — & x_1 & — \\ — & x_2 & — \\ & \vdots & \\ — & x_n & — \end{bmatrix}$$

$$X^T X = \sum_{i=1}^{n} x_i x_i^T$$

Invertible only if $n \geq d$.

Solution to overfitting: model selection

Setting: Training dataset

$(x_1, y_1), \ldots (x_n, y_n)$.

$y = f(x) + \varepsilon$ ["True" model].

Really, we care about

Test MSE $= E_D (y - f(x))^2$

Distribution of data. model we learn

→ Simulated by hold-out set of training data points.

to avoid artifacts in constructing hold-out set, repeat simulation k times. "k-fold cross validation".

$$\text{Test MSE} = E\left(y - f(x)\right)^2$$

$$= E\left(t(x) + \varepsilon - f(x)\right)^2$$

$$= E(\varepsilon^4) + E\left(\left(t(x) - f(x)\right)^2\right) + 2E(\varepsilon)E\left(t(x) - f(x)\right)$$

typically zero mean

$$= E(\varepsilon^2) + E\left(\left(t(x) - f(x)\right)^2\right),$$

$$E(Z^2) = \left(E(Z)\right)^2 + \text{Var}(Z)$$

$$= E[\varepsilon^2] + [E(\hat{f}(x) - f(x))]^2 + \mathrm{Var}(\hat{f}(x) - f(x))$$

Test MSE $= E[\varepsilon^2] + \text{Bias}^2 + \text{Variance}.$

↳ 1) Noise

2) Bias^2, decrease with model complexity.

3) Variance, increases with model complexity.