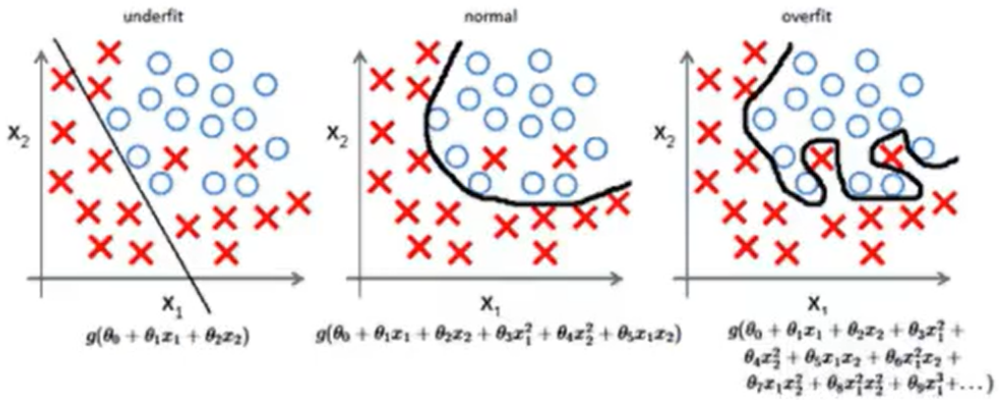
7️⃣

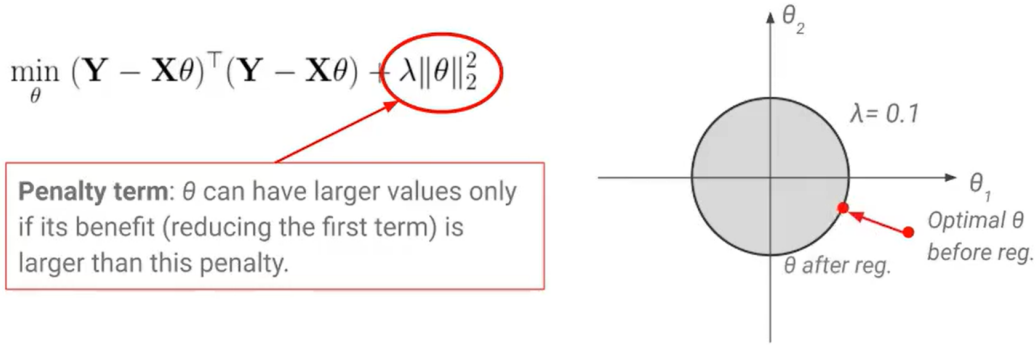
Lecture 7

Regularization

- Overfitting
  - 모델은 데이터의 트랜드뿐만 아니라 training data의 noise까지 fitting한다
  - high dimesional data를 다룰 때 많이 일어남
  - overfitting이 일어나면 파라미터 수 증가

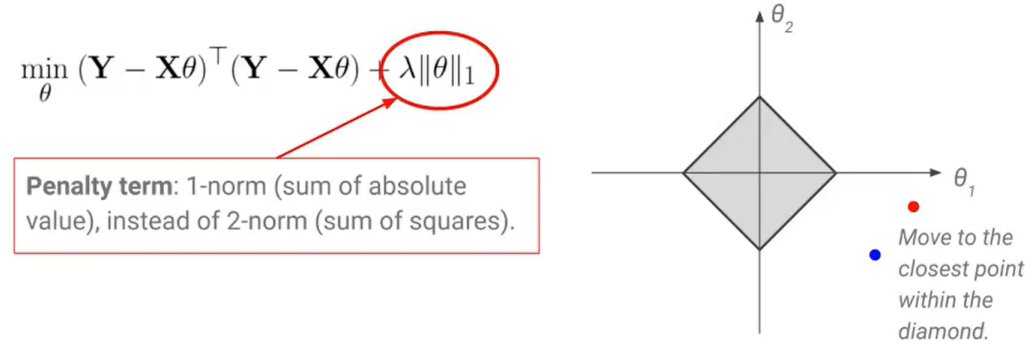


- Regularization
  - 파라미터 크기를 작게 해서 overfitting 방지
  - 모델에 최대한 자유도를 주고 파라미터가 의미 있을 때만 클 수 있게끔 “규제”함
  - Ridge regression (L2 regularization)
    - 값이 균등하게 줄어드는 성질이 있음



- With larger λ (regularization coefficient), θ (parameters) are more strongly regularized to have smaller values.
- With λ = 0, it is equivalent to (unregularized) linear regression.

- Rasso regression (L1 regularization)
  - 세타 값들 중에 상당수들이 0이 되어 sparse해짐 → 식이 단순해져 noise에 robust

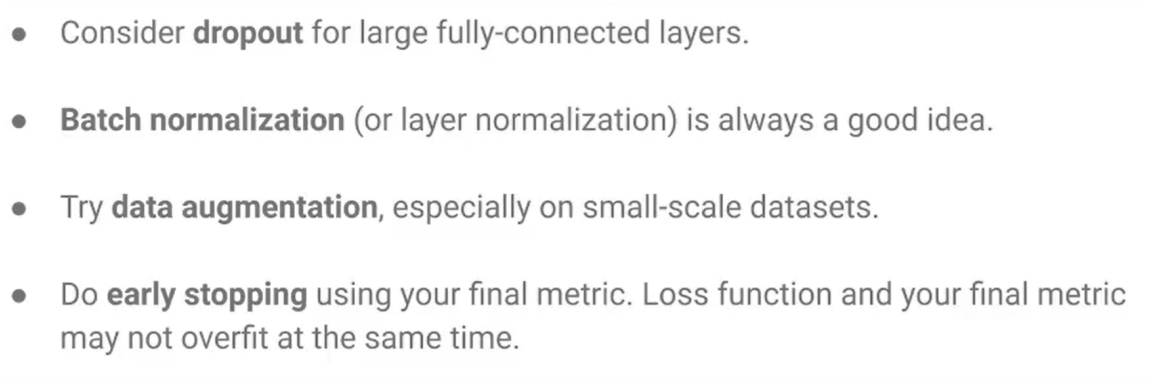


- Lasso encourages sparser representation of θ.
  - For most points, the closest point within the diamond is a vertex.
  - In the example above, red point regularized to have $\theta_2 = 0$.

- Neural network에서의 regularization 방법
  - Weight decay
    - Ridge, Lasso regularization을 더해줘 weight 값이 불필요하게 커지는 것을 방지
  - Early stopping
    - validation loss가 커지기 시작하면 (overfitting되기 시작하면) stop
  - Dropout

        - forward pass할 때 매번 랜덤하게 몇몇 노드들을 제거해 train 진행
    - Stochastic depth
        - dropout과 비슷한데 depth에 초점을 맞춤
        - layer을 몇 개씩 건너뛰어 학습
    - Cutout
        - data augmentation과도 비슷한데, 이미지의 일정 크기 이하를 무작위로 잘라냄

- Regularization in practice

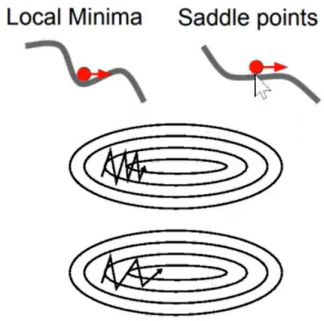


Optimization behind SGD

- SGD의 문제점
    - 한쪽 gradient가 급격하게 변하면 이상적으로 update 안됨
    - saddle point를 만나면 그 점에서 멈춤
    - 미니배치가 크지 못할 때 gradient estimation 자체가 정확하지 않을 수 있음

- 해결방안
    - SGD + Momentum
        - saddle point 해결

- **Momentum** in Physics: the product of the **mass** and **velocity** of an object.
- Here, it is more like **inertia**: an object continues in its existing state or uniform motion in a straight line.
- Build up "velocity" as a running mean of gradients.



SGD

$$x_{t+1} = x_t - \alpha \bigtriangledown f(x_t)$$

SGD + Momentum

$$v_{t+1} = \rho v_t - \alpha \nabla f(x_t)$$
$$x_{t+1} = x_t + v_{t+1}$$

ρ controls the degree of momentum.

◦ AdaGrad

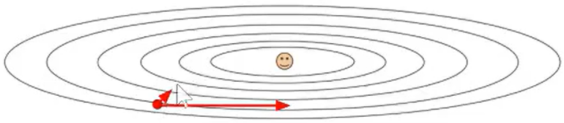
- gradient를 adaptive하게 update시킴
- gradient가 작았던 곳은 키워주고 컸던 곳은 줄여줌

  → element마다 다르게 scaling (element-wise scaling)

```
grad_squared = 0.0
while True:
  dx = compute_gradient(x)
  grad_squared += dx * dx
  divider = np.sqrt(grad_squared) + 1e-7
  x -= learning_rate * dx / divider
```

d-dimensional vector

learning rate applied per dimension



- 항상 양수라 뒤로 갈수록 너무 작게 update된다는 단점도 있음

  → RMSProp

```
grad_squared = 0.0
while True:
  dx = compute_gradient(x)
  grad_squared = dr * grad_squared + (1 - dr) * dx * dx
  divider = np.sqrt(grad_squared) + 1e-7
  x -= learning_rate * dx / divider
```

decay rate

◦ Adam

- RMSProp과 SGD+Momentum을 같이 진행

```
first_moment = 0.0
second_moment = 0.0
while True:
  dx = compute_gradient(x)
  first_moment = beta1 * first_moment + (1 - beta1) * dx
  second_moment = beta2 * second_moment + (1 - beta2) * dx * dx
  divider = np.sqrt(second_moment) + 1e-7
  x -= learning_rate * first_moment / divider
```

momentum

AdaGrad
RMSProp

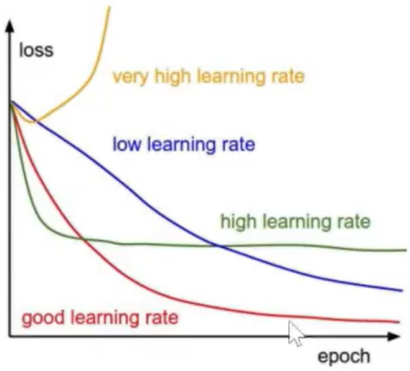
- First vs Second-order Optimization
  - First-order optimization
    - linear optimization
  - Second-order optimization
    - 이차 함수로 optimization 진행
    - convergence가 빠르다는 장점도 있지만, 헤시안 사이즈가 O(n^2)이고 역행렬은 O(n^3)이라 계산상 복잡하다는 단점이 있음

      → 오히려 linear이 더 빠르다는 의견도 있음

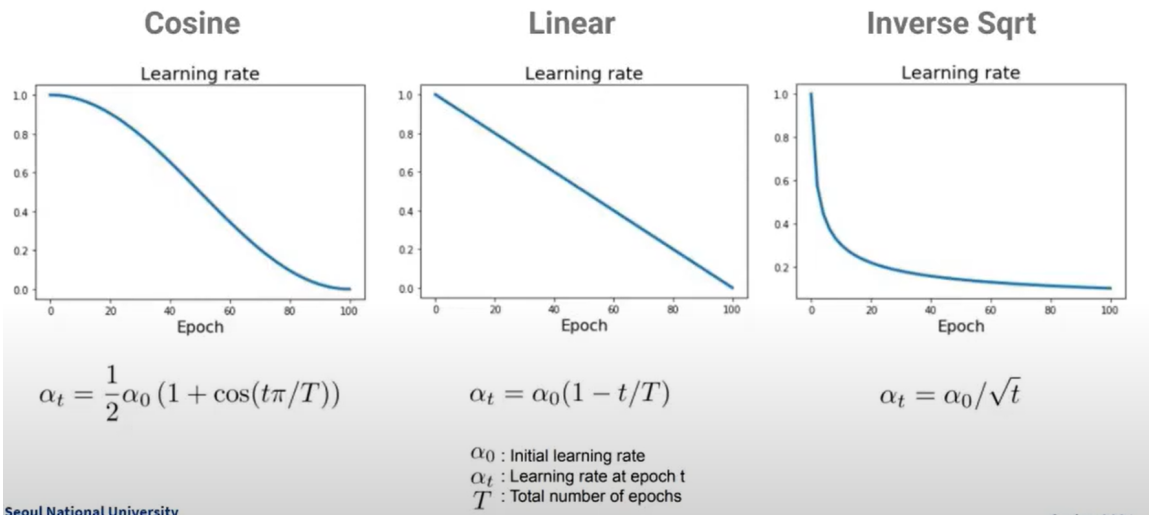
Learning rate scheduling

- learning rate

  - All optimizers we learned (SGD, SGD + Momentum, AdaGrad, RMSProp, Adam) require to specify the learning rate as a hyperparameter.
  - In most cases, we start with a large **initial** learning rate, then **decay** over time.



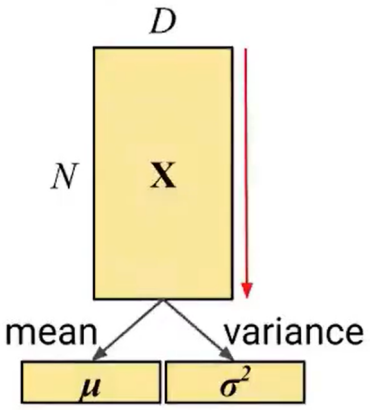
- Decay 방법



Batch normalization

- Batch normalization

  - Estimate them from samples in the same mini-batch.



$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j} \qquad \sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

Q. Dimensionality of μ, σ^2, x^?

A. *D, D, N×D*

- Batch normalization 단점
  - train과 test data의 distribution이 바뀌는 경우엔 정확한 예측 X

  → 해결 방안 : Batch Renormalization

- 최근 normalization 방법들

- Layer Normalization (https://arxiv.org/abs/1607.06450)
- Instance Normalization (https://arxiv.org/abs/1607.08022)
- Group Normalization (https://arxiv.org/abs/1803.08494)