Optimization: How to make the learning go faster

Lecture 6

목차

- Overview
  - 도입
  - Reminder : Gradient descent
  - Full / Online /mini batch gradient 설명

- A bag of tricks for mini-batch gradient descent
  - Learning rate
  - Shifting / scaling the inputs
  - De-correlate the input components (PCA)
  - Common problems that occur in multi-layer networks

- Methods to speed up mini-batch learning
  - Momentum method
  - Nesterov momentum method
  - Separate, adaptive learning rate for each connection
  - rmsprop

도입

- Many gradient descent optimization methods
  - Tensorflow 같은 machine learning librar들이 여러 gradient descent optimization functio을 지원한다
    → 잘 알고 사용해야

class tf.train.AdadeltaOptimizer
class tf.train.AdagradOptimizer
class tf.train.AdagradDAOptimizer
class tf.train.MomentumOptimizer
class tf.train.AdamOptimizer
class tf.train.FtrlOptimizer
class tf.train.RMSPropOptimizer

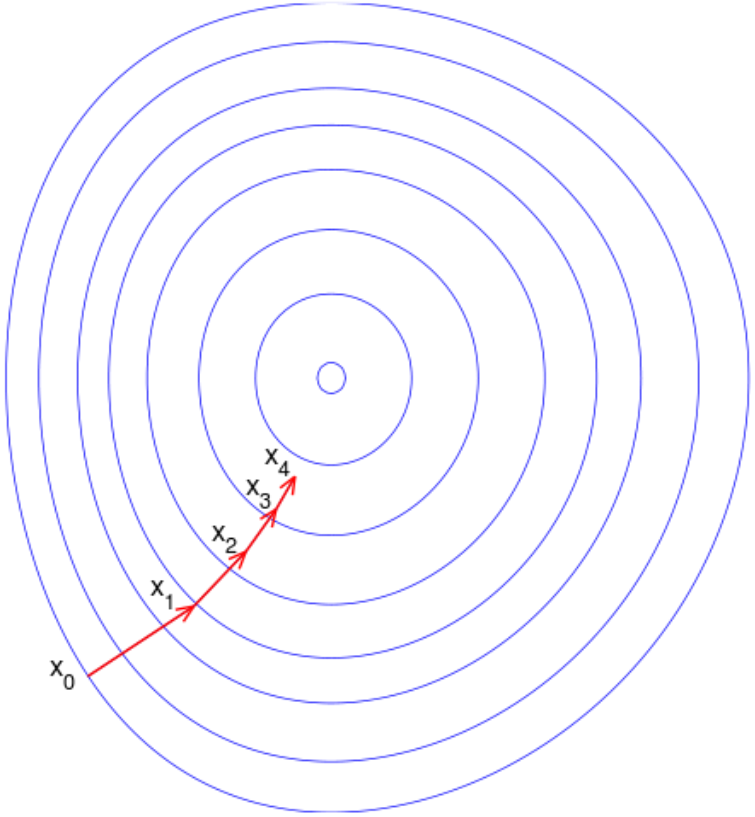
Reminder : Gradient descent

$$\mathbf{x}_{k+1}=\mathbf{x}_k-\lambda_k\nabla f(\mathbf{x}_k),\ \ k\geq 0$$

- 장점
  - 구현이 쉽고 다 차원에서 적용 가능

- 단점
  - 연산 량이 많음
  - Local minima
  - 수렴속도가 늦음 (해에 근접할 수록 기울기가 0에 수렴)

    - step size(lamda)가 크면 발산, 작으면 수렴속도가 늦어 해를 구하지 못함.



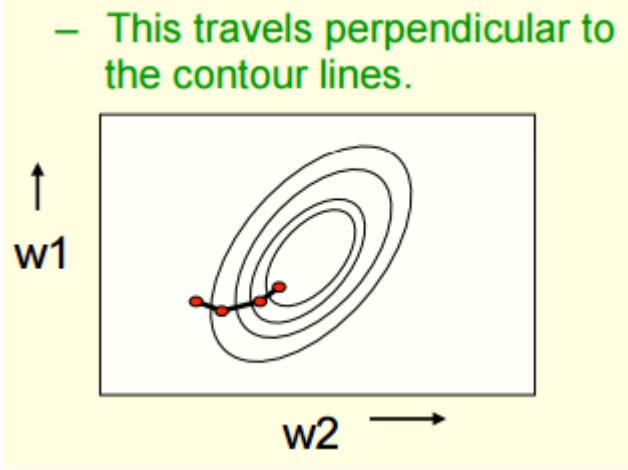
Full / Online /mini batch gradient

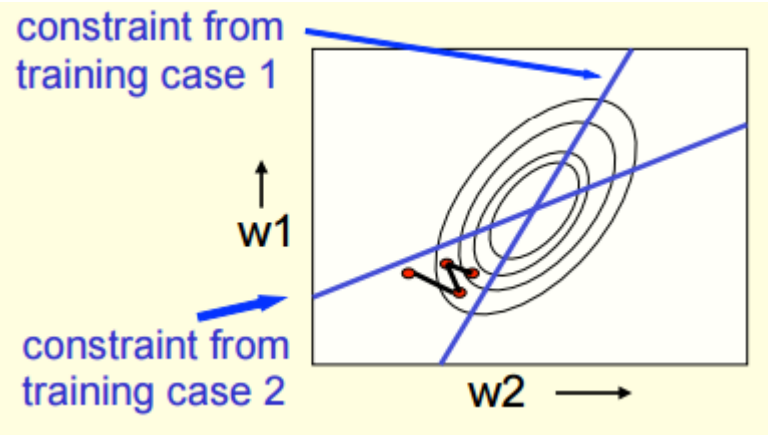
- Full batch gradient
  - 전체 Data를 사용하여 Weight를 갱신 (연산량 多), 작은 Data set에 유용
  - Steepest descent 방향으로 수렴
  - 빠른 대체 방법 존재 (Newton's method, conjugate gradient)

- On-line (stochastic) batch gradient
  - 한건의 Data를 사용하여 Weight를 갱신
  - Zig-zag 방향으로 수렴

  (error function 모양이 찌그러진 형태에서는
  Zig-zag로 이동하다 eclipse 밖으로 나갈 수 있다.)

- Mini-batch gradient
  - 전체 Data의 일부를 사용하여 Weight를 갱신
  - On-line / Full batch의 절충안
  - Very large and highly redundant training set에 유용
  - GPU 병렬처리 가능
  - Most common way to train ANN (combined with backpropagation)





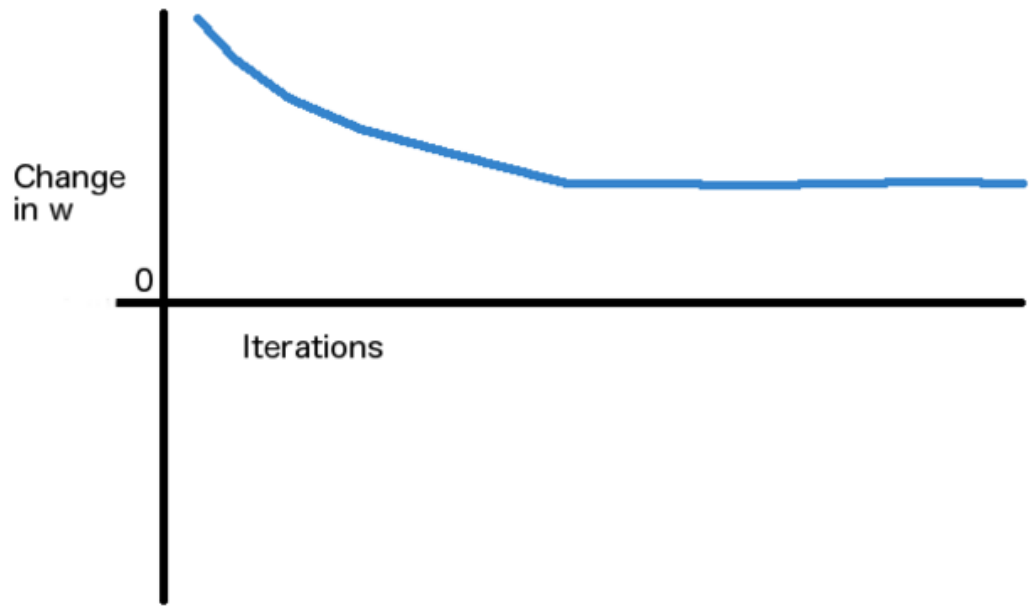
1 / 1 points

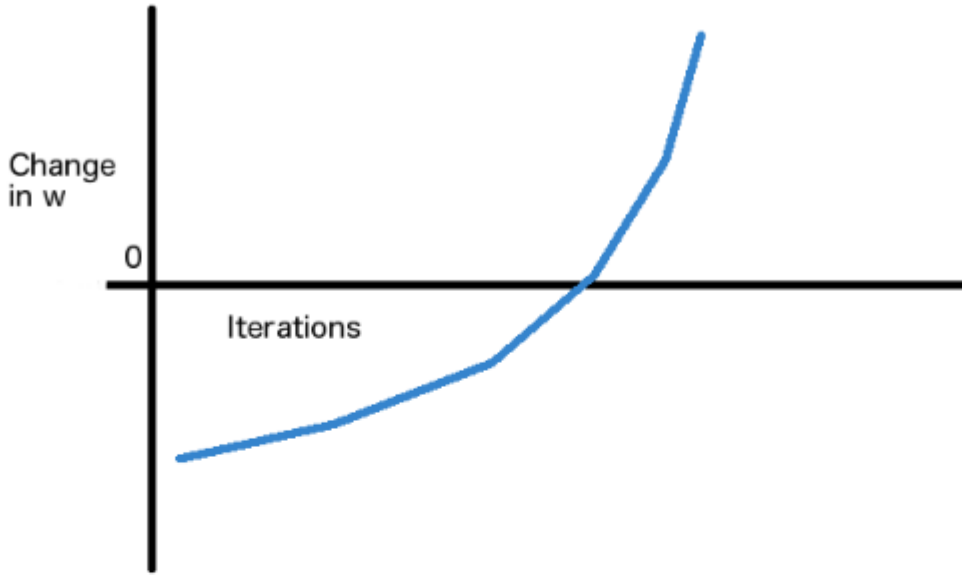
1. Suppose w is the weight on some connection in a neural network. The network is trained using gradient descent until the learning

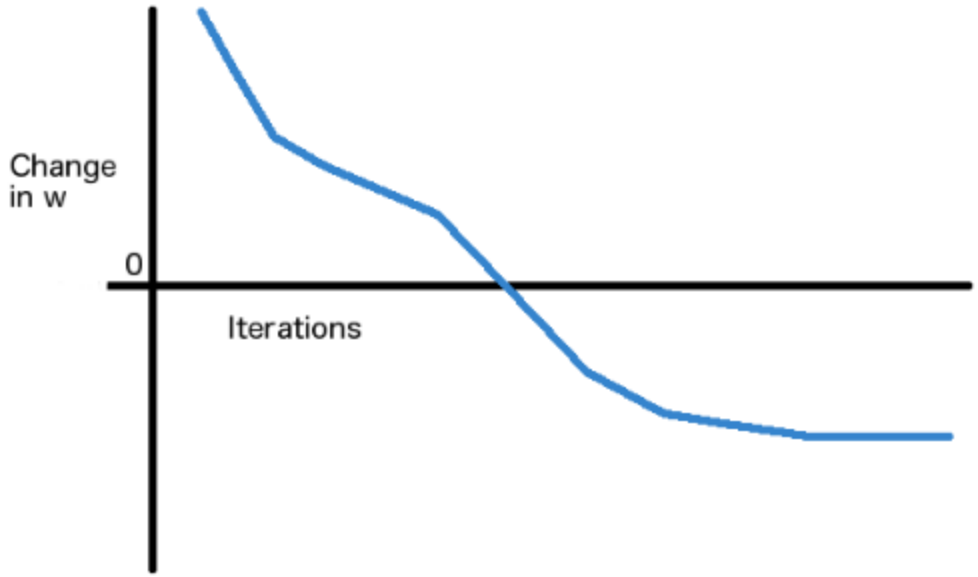
converges. However, the dataset consists of two mini-batches, which differ from each other somewhat. As usual, we alternate between the mini-batches for our gradient calculations, and that has implications for what happens

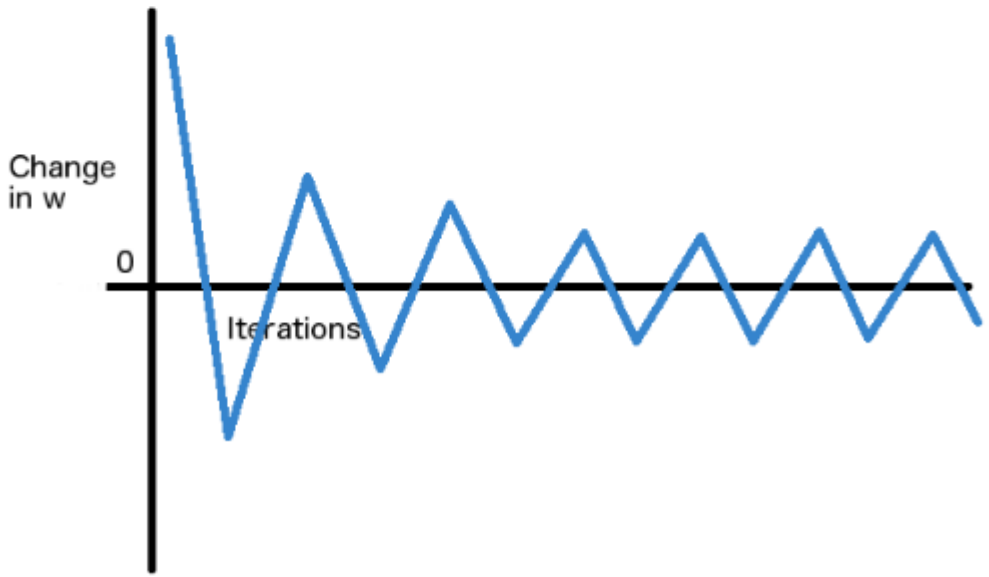
after convergence. We plot the change of w as training progresses. Which of the following scenarios shows that convergence has occurred? **Noticethat we're plotting the change in w, as opposed to w itself.**

Note that in the plots below, each *iteration* refers to a single *step* of steepest descent on a *single minibatch*.




Change
in w
0
Iterations


Change
in w
0
Iterations

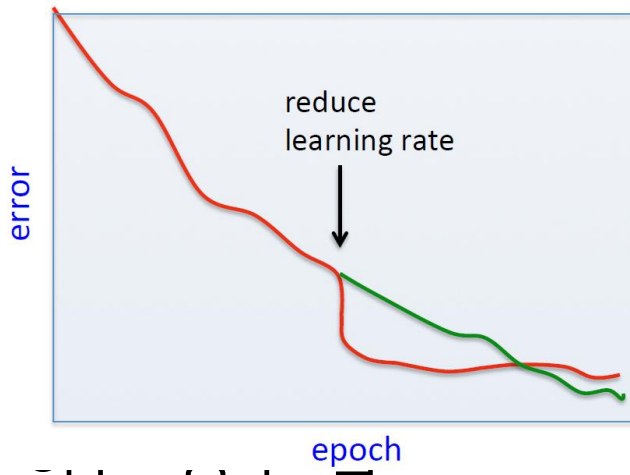

Change
in w
0
Iterations

A bag of tricks for mini-batch gradient descent

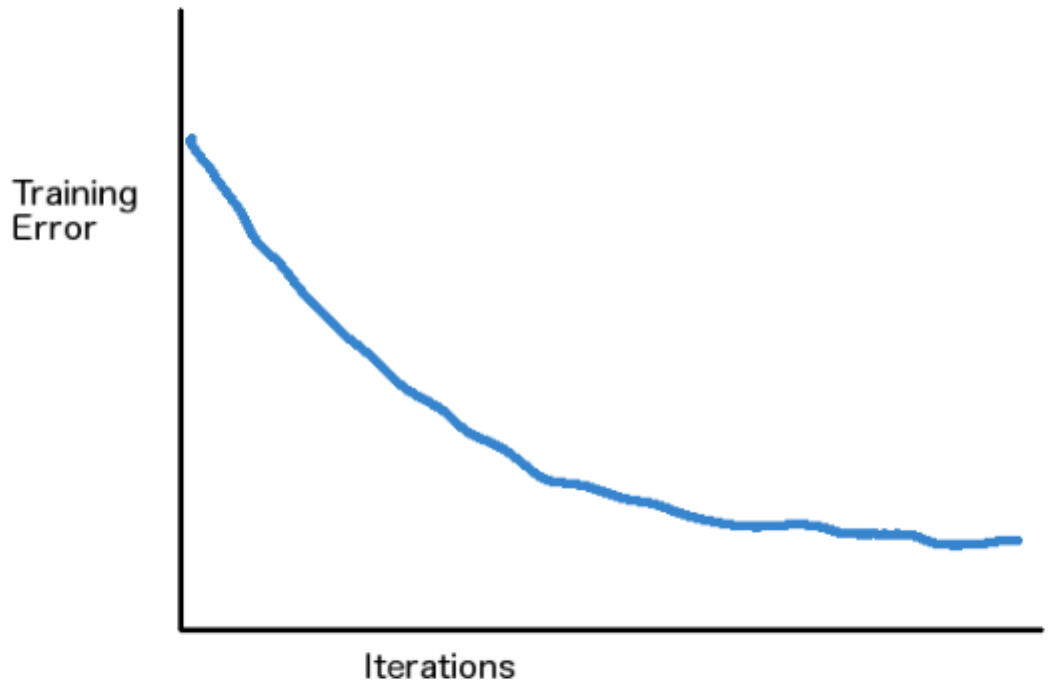
Learning rate

- Turning down learning rate
  - Reduces the random fluctuations
  - Turn down learning rate when error stops decreasing

- Initializing the weights
  - Hidden unit이 같은 weight를 가지지 않도록 한다.
  (다른 feature를 학습하도록)
  - 입력의 개수가 많은 경우 (big fan-in) optimal position을 overshoot (최종 optimal position 주위에서 fluctuation) 생
  - Fan-in의 제곱근 값에 비례해서 weight을 초기화

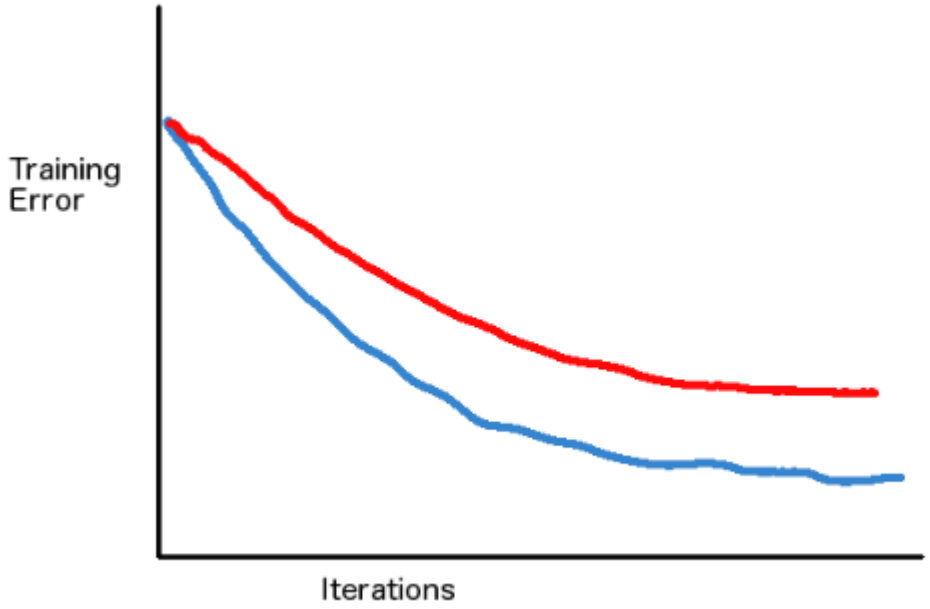


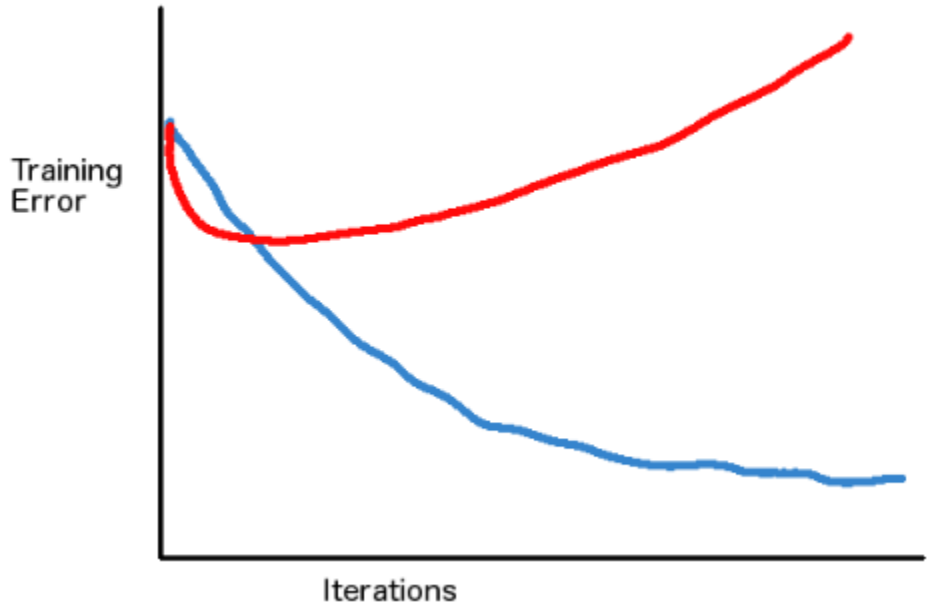
4. Claire is training a neural net using mini-batch gradient descent. She chose a particular learning rate and found that the training error decreased as more iterations of training were performed as shown here in blue

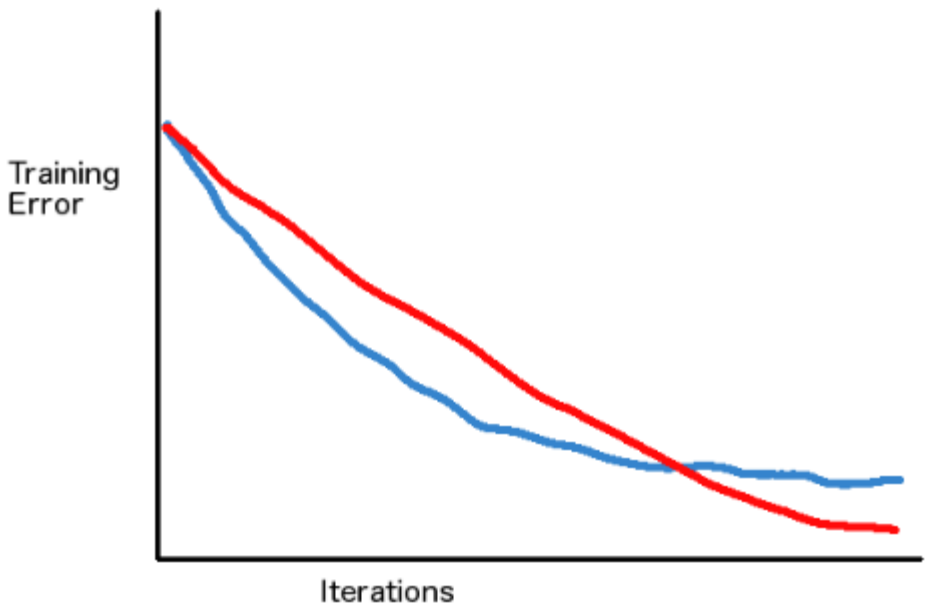


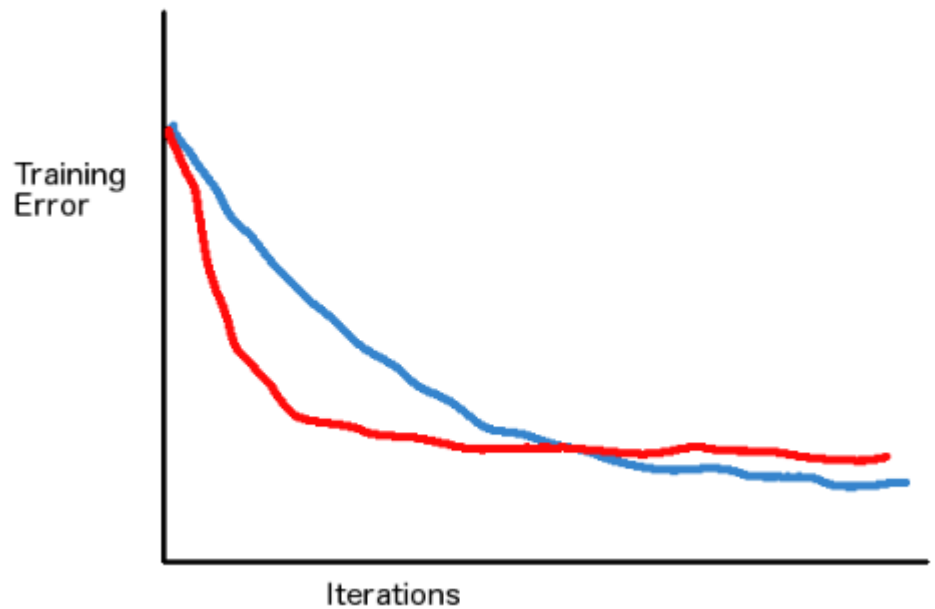
She was not sure if this was the best she could do. So she tried a **smaller** learning rate. Which of the following error curves (shown in red) might she observe now? Select the two most likely plots.

Note that in the plots below, each *iteration* refers to a single *step* of steepest descent on a *single minibatch*.


Training
Error
Iterations


Training
Error
Iterations


Training
Error
Iterations


Training
Error
Iterations

Shifting / scaling the inputs

- Error surface를 타원에서 원형으로 만드는 법
  - Shifting the inputs (mean of the shift is 0)

  Data → unit function → activation function (tanh -1 ~ 1) → output

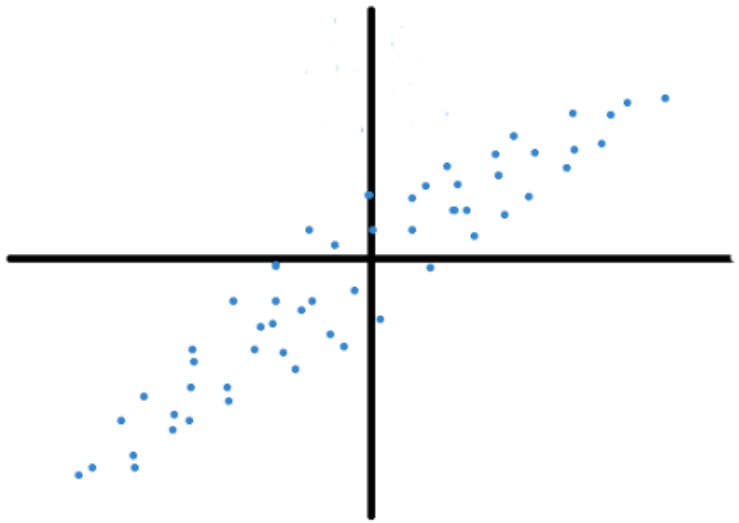
  Ex) 101, 101 → 2, 101, 99 → 0 ➔ 1, 1 → 2, 1 -1 → 0

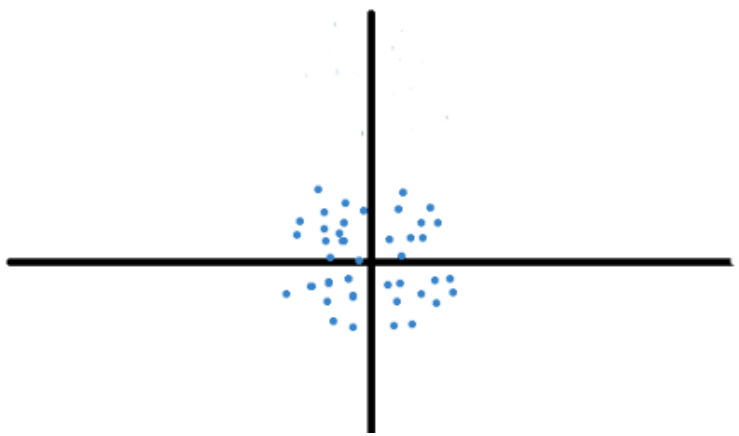
  - Scaling the inputs (단위 분산 scaling)

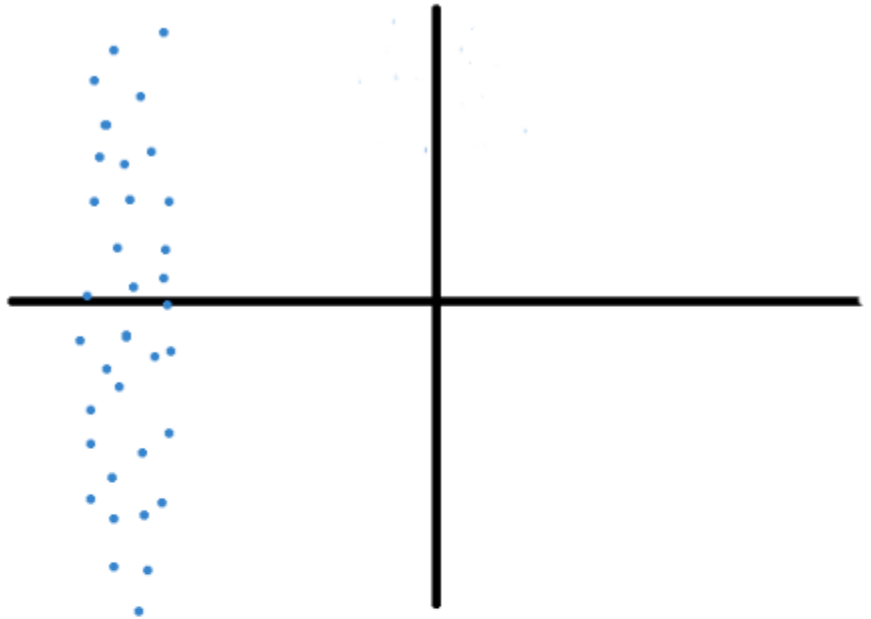
  It usually helps to transform each component of the input vector so that it has unit variance over the whole training set.

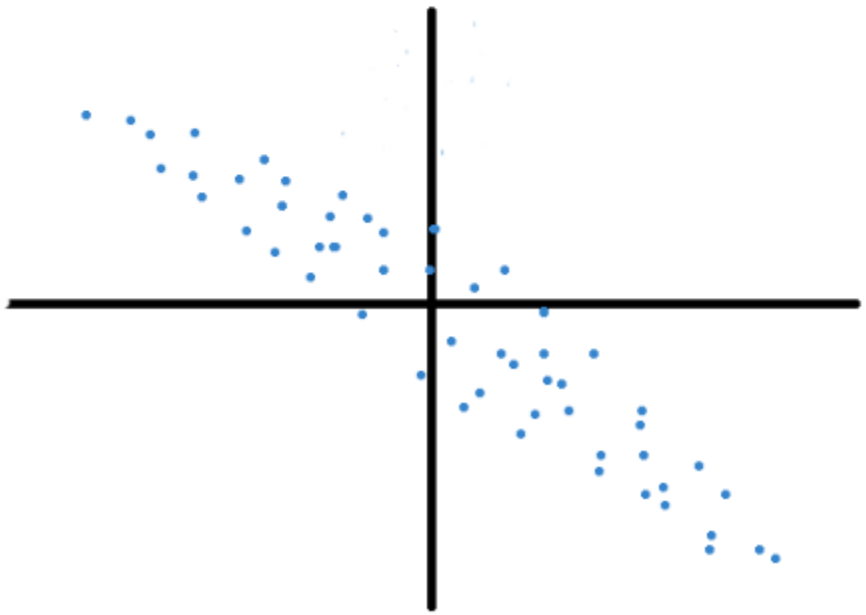
  Ex) 0.1, 10 → 2, 0.1 -10 → 0 ➔ 1, 1 → 2, 1 -1 → 0

3. Four datasets are shown below. Each dataset has two input values (plotted below) and a target value (not shown). Each point in the plots denotes one training case. Assume that we are solving a classification problem. Which of the following datasets would most likely be easiest to train using neural nets ?









De-correlate the input components (PCA)

- PCA (Principal components Analysis – 주성분 분석)
  - Data의 차원을 낮춘다.
    - 보다 적은 Data를 확보해도 됨. (차원의 저주)
  - De-correlate 가능 (eclipse → circle)
  - Data의 차원 수 만큼의 성분이 존재, 이 중 중요한 축 성분을 추출 나머지 의미없는 차원은 사상시킴

Common problems that occur in multi-layer netoworks

- Learning rate가 높으면
  - Gradient vanishing backpropagation시, 하위 hidden layer로 갈 수록 미분값이 곱해지면서 gradient가 vanish하는 현상
    (학습해도 error가 줄지 않아 local minima에 머문다.)
  - exploding - gradient 발산

In classification networks that use a squared error or a cross-entropy error, the best guessing strategy is to make each output unit always produce an output equal to the proportion of time it should be a 1.

- The network finds this strategy quickly and may take a long time to improve on it by making use of the input.
- This is another plateau that looks like a local minimum.

Methods to speed up mini-batch learning

Momentum method - 1

- Gradient에 관성 모멘트 속성을 부여하는 개념
- 이전에 구한 gradient 값에 일정 percent를 반영하여 현재 gradient를 구한다.

$$\mathbf{x}_{k+1}=\mathbf{x}_k-\mathbf{v}(t)$$

$$\mathbf{v}(t)=\alpha\,\mathbf{v}(t-1)-\varepsilon\frac{\partial E}{\partial \mathbf{w}}(t)$$

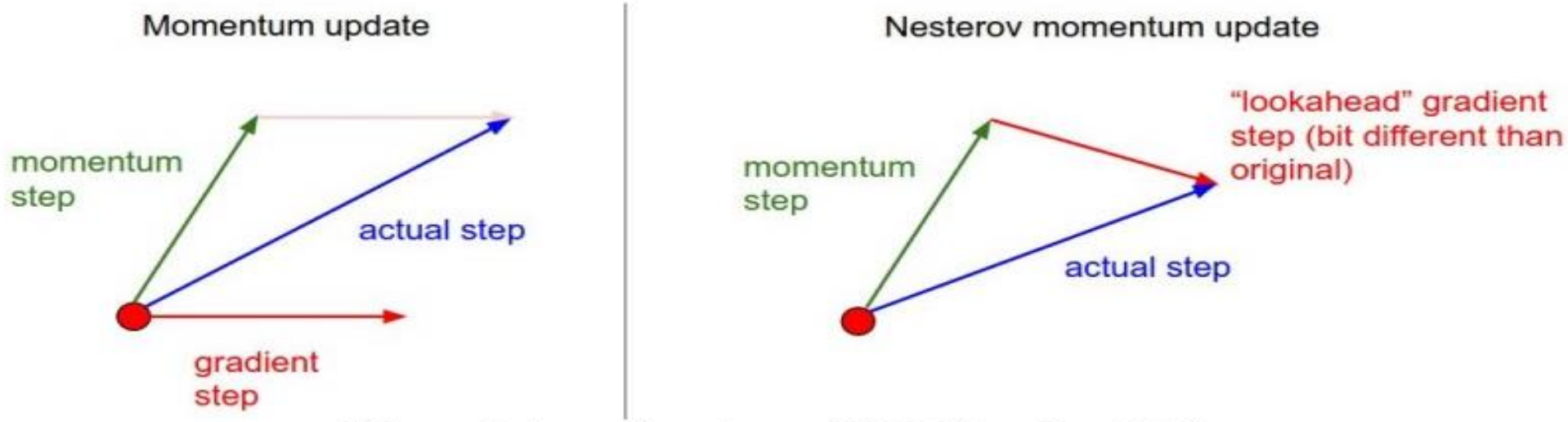
$$\Delta\mathbf{w}(t)=\alpha\,\Delta\mathbf{w}(t-1)-\varepsilon\frac{\partial E}{\partial \mathbf{w}}(t)$$

$$\Delta\mathbf{w}(t)=\mathbf{v}(t)$$

Momentum method - 2

- α 가 1에 수렴할 수록 gradient descent 속도는 증가함
  (α → 1에 수렴 v → 무한대)
- 최초 learning 시 0.5로 하고, weigh가 stuck되고 large gradien 가 사라지면 0.9정도까지 서서히 올린다.

Nesterov momentum method



Difference between Momentum and NAG. Picture from CS231.

Gradient와 momentum step 을 독립적으로 구해 더한다.

momentum step을 이동한 후, gradient를 구해서 이동한다.

Momentum 방식보다 효과적

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$

Separate, adaptive learning rate for each connection

- Local gains
  - Initialize local gains to one
  - Increase if sign not changed
  - Additive increase, multiplicative decrease
    - Oscillation dies out
  - Limit the gains
  - Can be used with momentum (sign Agreement between gradient and velocity)

$$\Delta w_{ij} = -\varepsilon \, g_{ij} \, \frac{\partial E}{\partial w_{ij}}$$

$$if \quad \left(\frac{\partial E}{\partial w_{ij}}(t) \frac{\partial E}{\partial w_{ij}}(t-1) \right) > 0$$

$$then \quad g_{ij}(t) = g_{ij}(t-1) + .05$$

$$else \quad g_{ij}(t) = g_{ij}(t-1) * .95$$

Check last two signs

Minima를 안 지났다.

Minima를 지났다.

Rprop

- Resilient Backpropagation (탄성역전파)
- Local adaptive learning scheme
- Full batch only

$$if \quad \left(\frac{\partial E}{\partial w_{ij}}(t) \frac{\partial E}{\partial w_{ij}}(t-1) \right) > 0$$

- Check last two signs, if they are the same, multiply by 1.2 otherwise by 0.5

RMSProp

- 변수를 updat할 때 각각의 변수마다 step size를 다르게 함.
  - 많이 변하지 않은 변수는 optimum에 가까울 확률이 높다.
- G 부분을 지수 이동평균을 사용
- 감마 0.9 사용

$$G = \gamma G + (1 - \gamma)(\nabla_\theta J(\theta_t))^2$$

$$\theta = \theta - \frac{\eta}{\sqrt{G + \epsilon}} \cdot \nabla_\theta J(\theta_t)$$

Summery

- Small datasets – Use full-batch
– Adaptive learning rates
– Conjugate gradient

- Big redundant dataset – Mini-batch
- Try momentum
- Try rmsprop

- Problematic because Neural Nets are very different(structure, goals etc.)

2. Suppose you are using mini-batch gradient descent for training some neural nets on a large dataset. All neurons are logistic. You have to decide the mini-batch size and learning rate. You try some values and find that the value of the objective function on the training set keeps fluctuating and does not converge. What could be going wrong? Check all that apply.

- [ ] The mini-batch size could be too small.
- [ ] The size of the dataset may be too large.
- [ ] The learning rate may be too big.
- [ ] The weights may have been initialized to large values.

5. In the lectures, we discussed two kinds of gradient descent algorithms: mini-batch and full-batch. For which of the following problems is mini-batch gradient descent likely to be **a lot better** than full-batch gradient descent?

- Sentiment Analysis: Decide whether a given movie review says that the movie is 'good' or 'bad'. The input consists of the word count in the review, for each of 50,000 words. The training set consists of 100 movie reviews written by experts for a newspaper.

- Sentiment Analysis: Decide whether a given movie review says that the movie is 'good' or 'bad'. The input consists of the word count in the review, for each of 50,000 words. The training set consists of 1,000,000 movie reviews found on the internet.

- Object detection: Identify which of 1000 categories an object image belongs to, given 10 million 256 X 256 pixel images.

- Predict if an experiment at the Large Hadron Collider is going to yield positive results. The input consists of 25 experiment parameters (energy level, types of particles, etc). The training set consists of the 200 experiments that have already been completed (some of those yielded positive results; some yielded only negative results).