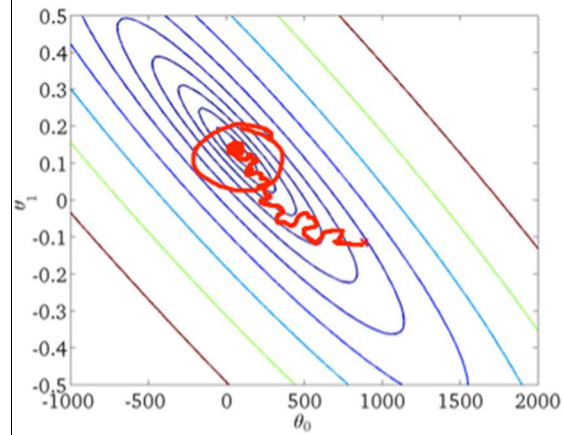
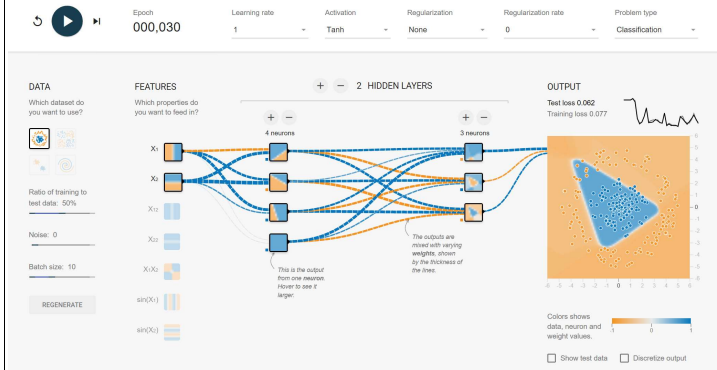
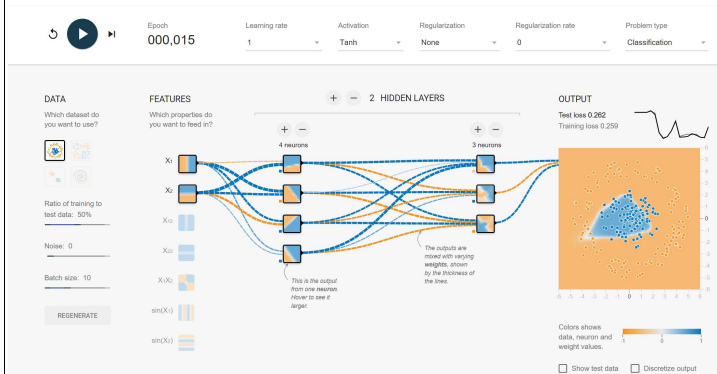


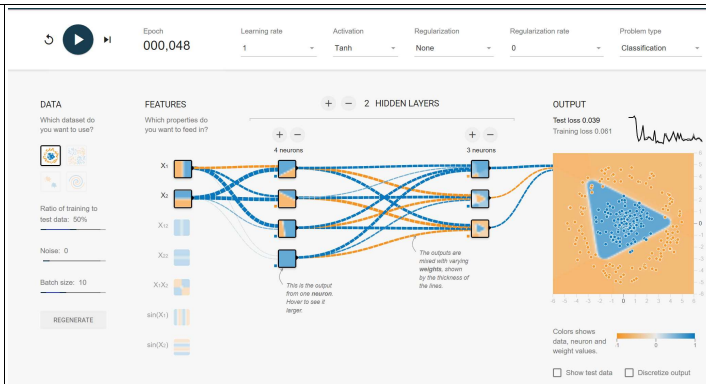
7 주차 조별보고서 (Default)	
작성일: 2019 년 10 월 18 일	작성자: 위성조
조 모임 일시: 2019 년 10 월 18 일 9 교시	모임장소: 학교 앞 카페
참석자: 위성조, 이충현, 최진성, 이재은, 김영연	조원: 위성조, 이충현, 최진성, 이재은, 김영연
구 분	내 용
학습 범위와 내용	<p>3.4 오류 역전파 알고리즘</p> <p>3.5 미니배치 스토캐스틱 경사 하강법</p> <p>3.6 다층 퍼셉트론에 의한 인식</p> <p>3.7 다층 퍼셉트론의 특징</p>
논의 내용	<p>Q. 미니배치 스토캐스틱 경사하강법에서는 미니 배치를 무작위로 뽑는다고 하였다. 무작위로 뽑을 때 선택되지 않은 훈련 집합도 존재할 것인데 이 때의 문제는 무엇인가?</p> <p>미니배치는 무작위로 뽑기 때문에 선택되지 않은 훈련 집합이 존재하므로, 데이터 전체의 경향을 반영하기 힘들다. 그래서 업데이트를 꼭 좋은 방향으로만 하지 않아 현재 학습을 진행하는 데이터 한 개에 대해서는 cost function의 값이 줄어들더라도, 이로 인해 다른 데이터에 대해서는 cost가 증가할 수 있다. 따라서 아래 그림과 같이 갈팡질팡 헤매는 경우가 생길 수 있다.</p>



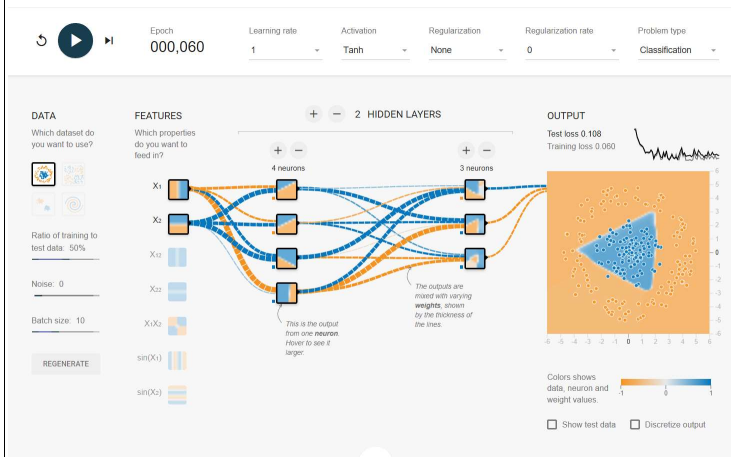
배치와 스토캐스틱 방식에서는 모든 샘플을 사용하지만 너무 느리고, 데이터 한 개를 사용하면 빠르지만 너무 헤맨다. 이러한 문제는 수십 개부터 많게는 수백 개의 데이터를 한 그룹으로 하여 처리함으로써, 적당한 속도로 전체 데이터를 최대한 반영함으로써 해결한다

Q. playgroun.tensorflow.org를 이용한 공동학습

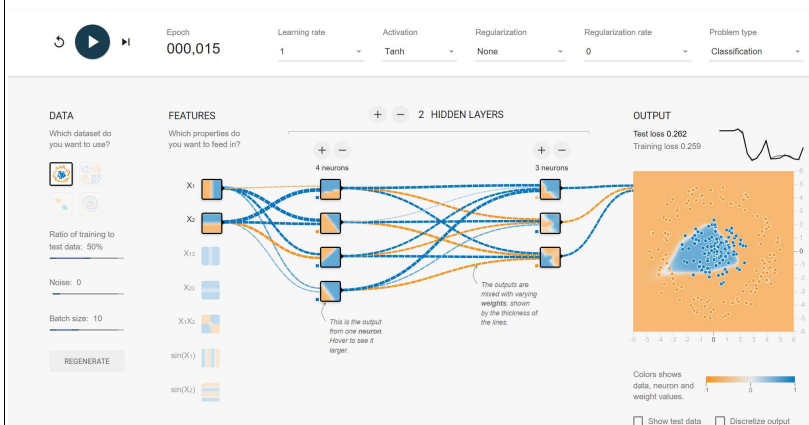
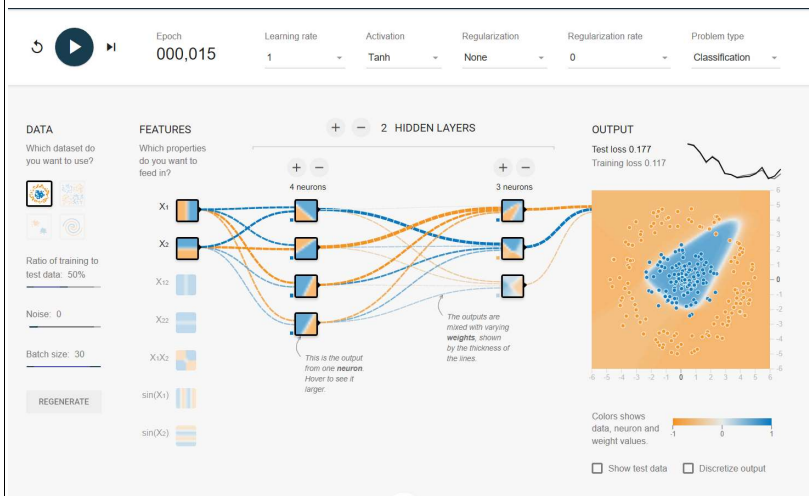


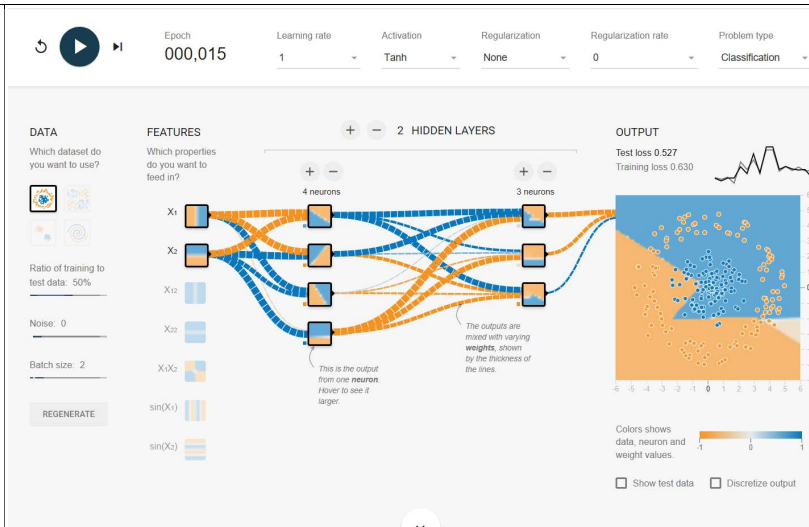


Epochs는 학습을 반복하는 횟수를 말한다. 사람이 일정 숫자의 문제를 푼다고 가정하였을 때 그 문제들을 몇 번이나 반복해서 풀어볼 것인지 정하는 것과 같다. 우리가 같은 문제들을 여러 번 반복하여 풀수록 점차 학습되는 것처럼 신경망 또한 epochs의 수를 늘릴수록 오차가 적어지는 것을 위의 사진을 통하여 알 수 있다.



그러나 위의 사진과 같이 epochs를 크게 하면 문제가 발생할 수 있다 이러한 현상을 우리는 오버피팅이라 하고 위의 사진에서 보면 오버피팅이 발생했음을 알 수 있다.





다음은 batch_size를 늘리고 줄였을 때의 사진이다. Batch_size란 몇 개의 샘플로 가중치를 갱신할 것인지에 관해 지정하는 것이다. 위의 설명과 마찬가지로 사람이 문제를 푸는 것을 이용하여 비유한다면 몇 문제를 풀고 해답을 보는지를 정하는 것과 같다. 즉, batch_size가 작을수록 가중치 갱신이 자주 일어나지만 시간이 오래 걸릴 수 있다는 것을 알아야 한다. 가중치 갱신의 수를 느낄 수 있는 것은 위 사진들에 있는 line들의 굵기를 보면 알 수 있는데, batch_size가 작아질수록 line들의 굵기가 굵어짐을 알 수 있다.

위의 사진에서 보면 batch_size가 작아질수록 오차가 커지는 것을 볼 수 있다. Batch size가 작으면 local minimum에 도달할 가능성이 줄어들고, catastrophic interface 때문에 오차가 늘어날 수 있다. (catastrophic interface란 새로운 정보를 학습할 때 기존에 학습했던 정보를 잊어버리는 것이다.)

질문 내용

Q1. playground.tensorflow.org 사이트를 이용한 공동 학습에서 배치 사이즈를 줄일수록, test 오차가 커지는 현상을 발견할 수 있었는데, 이에 대한 원인을 인터넷 조사를 통해 나름의 결론(바로 위 밑줄과 이탤릭체로 된 부분)을 내렸으나, 이것이 맞는 답인지, 혹은 이 외의 이유가 있는 것인지 확인할 수 없어, 피드백을 받고 싶습니다.

	<p>Q2. 최솟값을 구하기 위해서 배치 경사 하강법을 이용할 경우 최솟값을 정확히 찾아내지만 스토캐스틱 경사 하강법을 이용할 경우에는 대부분 최솟값의 범위를 찾아낼 뿐, 최솟값을 특정하는 데엔 배치 경사 하강법에 비해서 효율이 떨어진다고 알고 있다. 미니 배치 스토캐스틱 경사 하강법은 여러 샘플(미니 배치 스토캐스틱에 최적화된 샘플 숫자인 수 십에서 수 백으로 가정한다고 하였을 때)에서도 최솟값을 찾는 경우, 최솟값을 정확히 찾아낼 수 있는가?</p>
기타	

<개인 리포트>

201402033 위성조

구분	내용
학습 범위	3.4 오류 역전파 알고리즘 3.5 미니배치 스토캐스틱 경사 하강법 3.6 다층 퍼셉트론에 의한 인식 3.7 다층 퍼셉트론의 특징
학습 내용	목적함수 - 평균제곱오차(MSE)로 정의 $J(\theta) = J(\{U^1, U^2\})$ 의 최저점을 찾아주는 경사 하강법 $U^1 = U^1 - \rho \frac{\partial J}{\partial U^1},$ $U^2 = U^2 - \rho \frac{\partial J}{\partial U^2}$ 오차역전파 알고리즘 계산 결과 발생한 오차와, 각 노드의 가중치 값을 미분한 값을 이용하여, 가중치 값을 변화시킨다. 이를 반복하여, 오차가 작아지는 최선의 그레디언트를 찾는 방법이다. 미니배치 방식 한번에 t개의 샘플을 처리함(t는 미니배치 크기) <ul style="list-style-type: none">- T=1이면 스토캐스틱 경사 하강법 (너무 느리다)- T=n(한번에 모두 처리하면) 배치 경사 하강법 (노이즈가 많다) 그레디언트의 잡음을 줄여주는 효과 때문에 수렴이 빨라짐.

	<p>GPU를 이용한 병렬 처리에도 유리</p> <p>오류 역전파 알고리즘이 연산량이 많은 것은 사실이나, 전방 계산에 비해 1.5~2배정도의 계산을 요구하므로 빠른 속도로 계산이 가능</p> <p>학습의 경우, 수렴할 때까지 오류 역전파를 반복해야 하므로 점근적 시간 복잡도는 $\theta((dp + pc)nq)$</p> <p>데이터 희소성, 잡음, 미숙한 신경망 구조 등의 이유로 순수 최적화 알고리즘만으로는 높은 성능을 달성하는 것이 불가능하다.</p> <p>따라서 휴리스틱(경험적 방법)이 중요한데, 휴리스틱 개발에서의 주요 쟁점사항으로는,</p> <p>아키텍처 - 은닉층과 은닉 노드의 개수를 정하는 문제 (은닉층과 은닉 노드를 늘리면 모델의 용량은 커지나, 과적합 위험 발생)</p> <p>초깃값 - 보통 난수를 생성하여 설정하는데, 값의 범위와 분포가 중요하다</p> <p>학습률 - 처음부터 끝까지 같은 학습률을 사용하는 방식과, 처음에는 큰 값으로 시작하고 점점 줄이는 방식이 있다</p> <p>활성함수 - 은닉층의 경우, 로지스틱 시그모이드나 tanh함수를 사용하면, 은닉층이 많아지면서 그레디언트 소멸과 같은 문제가 발생하여, ReLu 함수를 주로 사용한다.</p>
질문 내용	

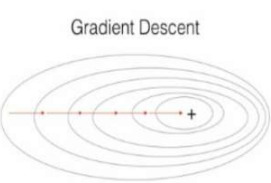
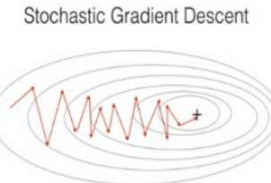
구분	내용
학습 범위	기계학습 3장 3.4절 오류 역전파 알고리즘 3.5절 미니배치 스토캐스틱 경사 하강법 3.6절 다층 퍼셉트론이 인식하는 단계 3.7절 다층 퍼셉트론의 특성
학습 내용	<p><3.4절 오류 역전파 알고리즘></p> <ul style="list-style-type: none"> 역전파 알고리즘은 input과 output을 알고 있는 상태에서(이를 supervised learning이라 함) 신경망을 학습시키는 방법이다. 초기 가중치, weight값은 랜덤으로 주어진다. 각각 노드, node는 하나의 퍼셉트론으로 생각한다. 즉, 노드를 지나칠 때마다 활성화함수를 적용한다. 활성화함수를 적용하기 이전을 net, 이후를 out이라고 하겠다. 다음 레이어의 계산에는 out값을 사용한다. 마지막 out이 output이 된다. 활성함수는 시그모이드 sigmoid 함수로 한다. (미분하기 용이해서 대표적으로 쓰이는 함수이지만, 다른 활성화함수가 아닌 다른 활성화함수를 사용해도 문제없다. <div data-bbox="851 973 1702 1356"> <p>The diagram illustrates a neural network structure with three layers: Input Layer, Hidden Layer, and Output Layer. The Input Layer consists of two nodes, 'input 1' and 'input 2'. The Hidden Layer consists of two nodes, 'h1' and 'h2', each represented as a circle divided vertically into 'net' and 'out' sections. The Output Layer consists of two nodes, 'o1' and 'o2', also represented as circles divided into 'net' and 'out' sections. Connections are shown between the layers: from 'input 1' to 'h1' and 'h2' (labeled with weight W_1), and from 'h1' and 'h2' to 'o1' and 'o2' (labeled with weight W_5). The arrows indicate the flow of information from the input layer through the hidden layer to the output layer.</p> </div>

- 알고리즘에 대해서는
 - 1) 기존에 설정되어 있는 가중치를 사용해 net, out을 계산한다. (forward pass)
 - 2) 전체 오차를 각 가중치로 편미분한 값을 기존의 가중치에서 뺀다(오차를 줄인다. 편미분 값을 더하면 최대값을 찾는 과정).
 - 3) 모든 가중치에 대해 2를 실행한다(output에 가까운 쪽에서부터 먼 쪽으로).
 - 4) 1~3을 학습 회수만큼 반복한다.

<3.5절 미니배치 스토캐스틱 경사 하강법>

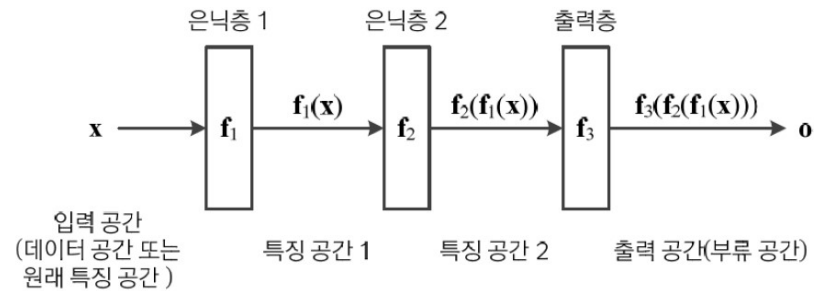
- 만약 우리가 모든 데이터 세트를 가지고 모델을 학습시킨다면 예측의 정확성은 높일 수 있으나, 매우 오랜 시간이 걸릴 것이다. 스토캐스틱 경사하강법은 랜덤으로 단 한 개의 데이터를 추출하여(배치 크기가 1) 기울기를 얻어낸다. 이러한 과정을 반복해서 학습하여 최적점을 찾아내는 것이 스토캐스틱 경사하강법이다. 한 번 학습할 때 마다 **모든 데이터를 계산하여 최적의 한 스텝을 나아가는 경사하강법**과 달리, **확률적 경사하강법은 랜덤하게 추출한 하나의 데이터만 계산하여 빠르게 다음 스텝으로 넘어간다.** 그 결과 더 빠르게 최적점을 찾을 수 있게 되었지만 그 정확도는 낮아진다.

경사하강법과 확률적 경사하강법의 비교

	경사하강법	확률적 경사하강법
1회의 학습에 사용되는 데이터	모든 데이터 사용	랜덤으로 추출된 1개의 데이터 사용(중복 선택 가능)
반복에 따른 정확도	학습이 반복 될 수록 최적해에 근접	학습이 반복 될 수록 최적해에 근접
노이즈	거의 없음	비교적 노이즈가 심함
해를 찾는 과정의 이미지 비교	 <p>Gradient Descent</p>	 <p>Stochastic Gradient Descent</p>

	<ul style="list-style-type: none"> ● 스토캐스틱 경사 하강법은 단일 반복에서 기울기를 구할 때 사용되는 데이터가 1개여서 노이즈가 너무 심하다는 한계가 있다. 이를 보완하는 것이 바로 미니 배치 확률적 경사하강법(Mini-batch SGD)입니다. 미니 배치 스토캐스틱 경사하강법은 전체 배치 반복과 SGD 간의 절충안으로써 배치의 크기가 10 ~ 1000 사이로 설정한다. 미니 배치 확률적 경사하강법은 SGD의 노이즈를 줄이면서도 전체 배치보다는 더 빠르게 최적점을 구할 수 있다. <p><3.7절 다층 퍼셉트론의 특성></p> <ul style="list-style-type: none"> ● 휴리스틱 개발에서의 중요 쟁점 <p><u>아키텍처</u> = 은닉층과 은닉 노드의 개수를 정해야 한다. 은닉층과 은닉 노드를 늘리면 신경망의 용량이 커지는 대신, 매개변수가 많아지고 학습 과정에서 과잉적합 할 가능성이 커진다. 따라서 현대 기계학습은 복잡한 모델을 사용하지, 적절한 규제 기법을 적용한다.</p> <p><u>초기값</u> = 가중치를 초기화하는데 0이 아닌 보통 난수를 생성하여 설정한다.</p> <p><u>학습률</u> = 처음~끝까지 같은 학습률 또는 처음에 큰 값으로 시작해서 점점 줄이는 적응적 방식이 있다.</p> <p><u>활성함수</u> = 로지스틱 시그모이드 → RELU함수 (Gradient소멸문제 해결)</p>
질문 내용	

구분	내용
학습 범위	기계학습 3장 다층 퍼셉트론 3.3 다층 퍼셉트론 3.4 신경망 기초
학습 내용	<p>기계학습 3장, 다층 퍼셉트론</p> <p>3.3.3 다층 퍼셉트론의 구조 입력층과 은닉층을 U로 표기하며, U^1은 입력층을, U^2-을 은닉층으로 여긴다. *0번째 은닉층을 입력층으로, 마지막 은닉층을 출력층으로 간주.</p> <p>3.3.4 다층 퍼셉트론의 동작 2층 퍼셉트론 $\rightarrow o = f(x) = f_2(f_1(x))$ 3층 퍼셉트론 $\rightarrow o = f(x) = f_3(f_2(f_1(x)))$ n층 퍼셉트론($n \geq 4 \rightarrow$ 깊은 신경망) $\rightarrow o = f(x) = f_L(\dots f_2(f_1(x)))$</p> <p>다층 퍼셉트론의 동작을 행렬로 표기할 경우 : $\mathbf{o} = \boldsymbol{\tau}(\mathbf{U}^2 \boldsymbol{\tau}_h(\mathbf{U}^1 \mathbf{x}))$</p> <p>은닉층은 특징 추출기(특징 벡터를 더 분류에 유리한 새로운 특징 공간으로 변환함) \rightarrow 특징 학습이라고 일컫음 딥러닝의 경우에는 많은 단계를 거쳐서 특징 학습을 함.</p>



*다층 퍼셉트론에 의한 인식 단계는 주로 예측, 혹은 테스트 단계에서 이루어진다.

3.4 오류 역전파 알고리즘

오류 역전파 : 출력층의 오류를 역방향으로 전파하여 경사(Gradient)를 계산

보통의 Forwarding 방식 : Input -> Hidden Layer -> Output

Output과 기댓값 y 간의 차이가 발생할 경우 역전파를 통해 Output부터 Hidden Layer로 들어가서 오차를 학습시킴 (Network Weight에 학습)

오류 역전파의 행렬 표기 -> GPU를 사용한 고속 행렬 연산에 적합하다.

3.4.1 목적 함수의 정의

훈련 집합 : 입력 벡터들에 대해서 클래스, 또는 클래스가 가져야 하는 값 정보를 가지고 있다고 가정.

이 경우에 회귀 문제냐, 분류 문제냐에 따라서 데이터 값이 달라질 수있음.

분류 문제의 경우 훈련 집합은 원 핫 코드로 표현됨.(ex : $y_i = (0, 0, 0, \dots, 1, \dots, 0)^t$)

기계학습의 목표 : 모든 샘플을 옳게 분류하는 최적의 함수 T 를 찾는 일

→ 목적 함수를 찾는 일

평균 제곱 오차(MSE)

온라인 모드 : $e = \frac{1}{2} \|y - o\|_2^2$

배치 모드 : $e = \frac{1}{2n} \sum_{i=1}^n \|y_i - o_i\|_2^2$

→ $J(\theta) = \frac{1}{2} \|y - o(\theta)\|_2^2$ (2층 퍼셉트론)

$J(\theta) = J(\{U^1, U^2\})$ 의 최저점을 찾는 경사하강법

$$\left. \begin{aligned} \mathbf{U}^1 &= \mathbf{U}^1 - \rho \frac{\partial J}{\partial \mathbf{U}^1} \\ \mathbf{U}^2 &= \mathbf{U}^2 - \rho \frac{\partial J}{\partial \mathbf{U}^2} \end{aligned} \right\}$$

3.5 미니배치 스토캐스틱 경사 하강법

= 배치 형식 경사 하강법 + 스토캐스틱 경사 하강법

현대에서 많이 사용되는 형태. (경사 잡음을 줄여주는 효과 때문에 수렴이 꽤 빠르다.)

한 번에 t 개의 샘플을 처리함(t 는 미니 배치 크기)

- $t = 1$ -> 스토캐스틱 경사
- $t = n$ -> 배치 경사(보통 수 십 ~ 수 백)

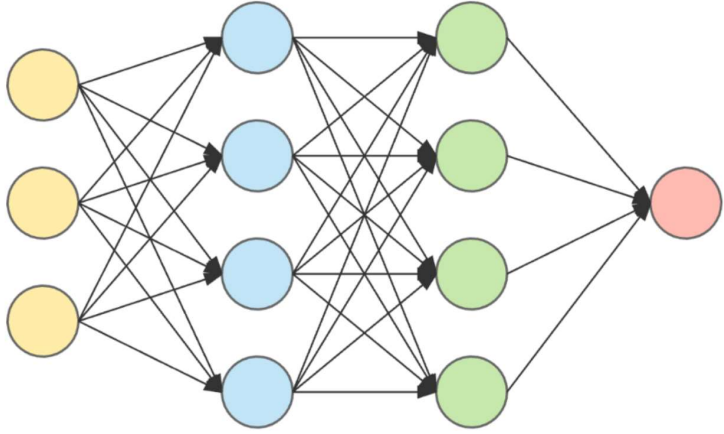
3.7 다층 퍼셉트론의 특성

오류 역전파 알고리즘의 빠른 속도

	<p>전방 계산보다 1.5~2배의 시간이 소요가 된다. -> 여러 데이터에 대하여 빠른 계산이 가능함.</p> <p>-> 수렴할 때 까지 오류 역전파를 반복해야 하는 특성 상 점진적인 시간 복잡도는 전방 계산보다 더 크다.</p> <p>모든 함수를 정확하게 근사할 수 있는 능력</p> <p>실제로는 은닉층 하나를 사용함에도 은닉노드의 수에 따라 효율을 높게 할 수 있다.</p> <p>*은닉노드를 무수히 많게 할 수 없기 때문에 실질적으로 복잡한 구조의 성능을 필요로 하는 곳에선 효율이 떨어진다.</p> <p>성능 향상을 위한 휴리스틱의 중요성</p> <p>휴리스틱 : 불충분한 자원을 토대로 합리적인 판단을 하기 위한 어림 짐작</p> <p>순수한 최적화 알고리즘으로는 높은 성능을 내는 것이 불가능하기 때문에(ex : 데이터 희소성, 잡음, 미숙한 신경망 구조 등) 휴리스틱을 개발하고 공유하여 성능 향상을 이루어 냄.</p> <p>아키텍처, 초깃값, 학습률, 활성화함수가 대표적인 주요 쟁점.</p> <p>다층 퍼셉트론은 실용 시스템 제작에 크게 기여했으나 잡음이 과도하게 섞이거나 데이터 희소성이 심하거나 경우의 수가 매우 많은 경우에 대한 한계가 명확하였다.</p>
질문 내용	<p>최솟값을 구하기 위해서 배치 경사 하강법을 이용할 경우 최솟값을 정확히 찾아내지만 스토캐스틱 경사 하강법을 이용할 경우에는 대부분 최솟값의 범위를 찾아낼 뿐, 최솟값을 특정하는 데엔 배치 경사 하강법에 비해서 효율이 떨어진다고 알고 있다. 미니 배치 스토캐스틱 경사 하강법은 여러 샘플(미니 배치 스토캐스틱에 최적화된 샘플 숫자인 수 십에서 수 백으로 가 정한다고 하였을 때)에서도 최솟값을 찾는 경우, 최솟값을 정확히 찾아낼 수 있는가?</p>

구분	내용
학습 범위	<p>3.3절: 다층 퍼셉트론</p> <p>3.4절: 오류 역전파 알고리즘</p> <p>3.5절: 미니배치 스토캐스틱 경사 하강법</p> <p>3.6절: 다층 퍼셉트론에 의한 인식</p> <p>3.7절: 다층 퍼셉트론의 특성</p>
학습 내용	<ul style="list-style-type: none"> - 특징 공간 변환이란 XOR 해결하기 위하여 퍼셉트론을 여러 개 사용하는 것 이다. 그림 3-9는 사람이 수작업으로 특징 학습을 수행한 것이고, 이를 어떻게 자동으로 학습할 것이냐 -> 3.4절 - 표 3-1 함수 -> 1차 도함수 미분 과정을 이해해야 Back Propagation계산 과정을 이해할 수 있다. - 네트워크를 수렴시킨다라는 말은 다층 퍼셉트론의 가중치 u_{ji}, u_{lj}등의 적절한 값을 찾는 것을 의미한다. - 은닉층은 특징벡터를 분류에 더 유리한 새로운 특징 공간으로 변환: 특징학습 <div data-bbox="465 975 1256 1302"> <p>입력 공간 (데이터 공간 또는 원래 특징 공간)</p> <p>특징 공간 1</p> <p>특징 공간 2</p> <p>출력 공간(분류 공간)</p> <p>그림 3-16 특징 추출기로서의 은닉층</p> </div>

	<ul style="list-style-type: none"> - 문제가 분류(Classification)인지 회귀(Regression)인지에 따라 기대값(부류 벡터 \mathbf{y})가 달라진다. 분류는 class 를 예측하는 것이다. 반면에, 회귀는 보통 연속적인 숫자, 즉 예측 값이 float 형태인 문제들을 해결하는데 사용된다. 주의할 점은 회귀는 확률을 예측하는 것이 아니다. 회귀는 출력은 연속성이 있고, 그 연속성 중에 어디에 점을 찍을지 결정하는 문제이다. - Jacobian 은 어떤 다변수 벡터함수(vector-valued function of multiple variables)에 대한 일차 미분(first derivative)이다. 앞서 나온 gradient 나 Jacobian 모두 함수에 대한 일차 미분(first derivative)을 나타낸다는 점에서는 동일하다. 다만, gradient 는 다변수 스칼라 함수(scalar-valued function of multiple variables)에 대한 일차 미분인 반면 Jacobian 은 다변수 벡터 함수(vector-valued function of multiple variables)에 대한 일차 미분이라는 차이점이 있다. 즉, 두 개 모두 함수에 대한 일차 미분이기 때문에 어떤 함수의 지역적인 변화 특성을 파악할 때, 지역적인 함수의 변화를 선형 근사할 때 또는 함수의 극대(극소)를 찾을 때 활용된다. - 그림 3-18 계산식에서 0.5 를 곱해주는 것은 나중에 미분할 때 편하기 위함. chain rule = 연쇄 법칙 = 합성함수의 미분법. - 오류 역전파 알고리즘: 출력 층의 오류를 역방향(왼쪽)으로 전파하며 gradient 를 계산하는 알고리즘. 이것이 제안됨으로 인해서 다층 퍼셉트론이 가능해졌다. - 미니 배치 스토캐스틱 하강법: 전체 학습 데이터를 배치 사이즈로 등분하여(나눠) 각 배치 셋을 순차적으로 수행함으로써 배치 스토캐스틱 하강법 보다 빠르고 스토캐스틱 하강법보다 낮은 오차율을 보인다. - 오류 역전파는 전방 계산보다 약 1.5 배~2 배 시간 소요. 그러나 오류역전파는 반복해야하므로 점근적 시간 복잡도는 $\Theta((dp + pc)nq)$ - 휴리스틱 개발에서 주요 쟁점: 아키텍처, 초깃값, 학습률, 활성화함수
질문 내용	http://playground.tensorflow.org/ 사이트에서 여러 실습을 해보며, 결과값을 통한 학습을 같이 하고 싶습니다.

구분	내용
학습 범위	<p>3.4 오류 역전파 알고리즘</p> <p>3.5 미니배치 스토캐스틱 경사 강하법</p> <p>3.6 다층 퍼셉트론의 인식</p> <p>3.7 다층 퍼셉트론의 특징</p>
학습 내용	<p>1. 역전파란?</p> <p>신경망에서 경사하강법을 수행하는 기본 알고리즘으로 우선 정방향 단계에서 각 노드의 출력 값을 계산하고 캐시하며, 그런 다음 역방향 단계에서 그래프를 통과하며 각 매개 변수를 기준을 오차의 편미분을 계산</p>  <p>input layer hidden layer 1 hidden layer 2 output layer</p> <p>위의 그림에서 output layer에서 나온 오차를 줄이기 위해서 우선 hidden layer2에서 output layer로 향하는 weight를 업데이트하</p>

	<p>고, 그 후, hidden layer1에서 hidden layer2로 향하는 가중치를 업데이트하고, 마지막으로 input layer에서 hidden layer1으로 향하는 가중치를 업데이트 하는 것이다.</p> <p>2. 미니 배치 경사하강법</p> <ul style="list-style-type: none"> - 각 스텝에서 전체 훈련 세트나 하나의 샘플을 기반으로 경사를 계산하는 것이 아니라 미니배치라 부르는 임의의 작은 샘플 세트에 대한 경사를 계산한다. <p>미니배치를 어느정도 크게 하면 이 알고리즘은 파라미터 공간에서 SGD보다 덜 불규칙하게 움직이므로 결국 미니배치 경사하강법이 SGD보다 최솟값에 더 가까이 도달하게 될 것이다.</p> <p>미니배치는 일반적으로 무작위로 선택한 10개에서 1000개 사이의 예로 구성된다. 미니배치 SGD는 SGD의 노이즈를 줄이면서 전체 배치보다는 더 효율적이다.</p> <p>3. 다층 퍼셉트론에 의한 인식</p> <p>4. 다층 퍼셉트론의 특징</p> <p>1) 오류역전파의 빠른 속도</p> <p>오류 역전파를 구성하는 식들은 전방 계산에 사용하는 식보다 유도 과정과 수식 표현이 훨씬 복잡하다. 그렇기 때문에 계산에 훨씬 많은 시간을 소요할 것이라고 생각할 수 있는데, 연산의 양을 따져보면 비슷하다는 것을 알 수 있다. 덧셈과 곱셈을 구분하지 않고 연산 횟수를 세면, 전방 계산은 $2dp+2pc$, 오류 역전파는 $3dp+5pc$만큼 소요한다. 이 때, d, p, c는 각각 입력 노드의 개수, 은닉 노드의 개수, 출력 노드의 개수이다.</p> <p>2) 모든 함수를 근사할 수 있는 능력</p> <p>3) 성능 향상을 위한 휴리스틱의 중요성</p> <p>대부분의 데이터가 충분하지 않고 잡음이 섞여 있으며 미숙한 신경망구조를 사용하는 등의 여러가지 문제 때문에 기초적인 알고리즘만으로는 만족할 만한성능을 얻기 어렵다. 성능을 높일 수 있는 갖가지 휴리스틱의 중요한 쟁점을 소개하면 다음과 같다.</p> <ul style="list-style-type: none"> - 아키텍처
--	---

	<ul style="list-style-type: none"> - 초깃값 - 학습률 - 활성화함수 <p>4) 실용적인 성능</p> <p>다층 퍼셉트론은 실용적인 시스템을 만드는데 크게 기여를 했지만 한계가 있었다. 이러한 성능의 한계는 딥러닝이 뛰어넘었는데, 딥러닝의 기본원리는 은닉층의 수를 확장하는 것이다.</p>
질문 내용	<p>미니배치 스토캐스틱 경사하강법에서는 미니배치를 무작위로 뽑는다고 하였다. 무작위로 뽑을 때 선택되지 않은 훈련집합도 존재할 것인데 이 때의 문제는 무엇인가?</p>