딥러닝 모델용 하이퍼 파라미터 최적화 실천 가이드

DL모델의 베이비시터가 싫증나지 않으세요? 그거면 딱 좋겠네요. 이 기사에서는 모든 딥러닝 모델에 최적인 하이퍼 파라미터를 효과적으로 검색할 동기와 전략에 대해 설명합니다. Floyd Hub에서 이것이 어떻게 가능한지 그리고 연구가 어떤 방향으로 진행되고 있는지를 나타낸다. 이 글을 다 읽고 나면 당신의 데이터 사이언스 툴-belt에 강력한 새로운 툴이 추가될 것이다. 그러면 딥러닝 태스크에 가장 적합한 구성을 찾는 프로세스가 가능한 자동화가 될 것이다.

기계 학습 모델과는 달리 딥러닝 모델은 말 그대로 하이퍼 파라미터로 가득 차 있습니다. 뭔가 증거를 원하십니까? Transformer base v1 하이퍼 파라미터의 정의를 봐 주세요.

물론 이러한 모든 변수가 모델의 학습 과정에 똑같이 기여하는 것은 아니지만, 이러한 복잡성을 감안할 때 이러한 고차원 공간에서 이러한 변수에 가장 적합한 구성을 찾는 것은 사소한 일이 아닙니다.

운 좋게도, 우리는 검색 문제에 대처할 수 있는 다른 전략과 도구를 가지고 있습니다. 뛰어 들어갑시다!

HOW?  
  
하이퍼 파라미터의 최적의 구성을 찾아 검증/테스트 세트에서 중요한 지표에 대해 최고의 점수를 얻을 수 있습니다.

WHY?

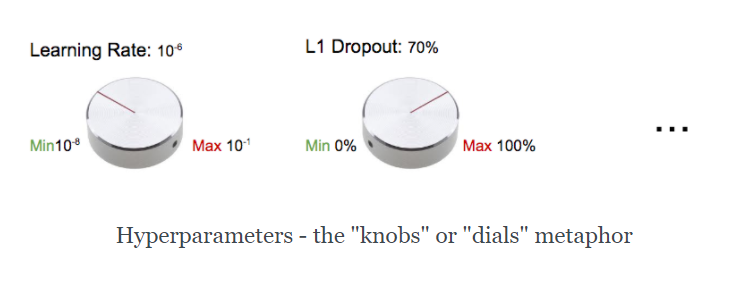
모든 과학자와 연구자는 이용 가능한 자원 💻, 💰 및 ⏳(컴퓨팅, 비용, 시간)을 고려하여 태스크에 가장 적합한 모델을 요구하고 있습니다. 효과적인 하이퍼 파라미터 검색은 이 목표를 향해 나아가는데 도움이 되는 퍼즐의 빠진 조각 부분입니다.

WHEN?

-연구원이나 취미주의자들 사이에서는 개발의 마지막 단계에서 이들 탐색전략 중 하나를 시험하는 경우가 자주 있습니다. 이를 통해 몇 시간의 작업 후에 이미 취득된 가장 좋은 모델에서 개선이 가능합니다.

-하이퍼 파라미터 검색은 반자동/자동 딥러닝 파이프라인의 스테이지 또는 컴포넌트로도 일반적입니다. 이것은 확실히, 기업의 데이터 사이언스 팀 사이에서 많이 일반적입니다.

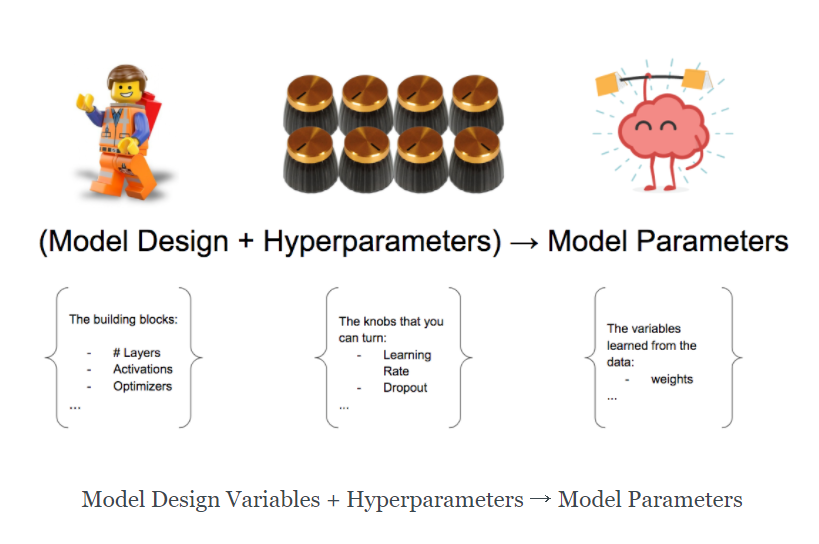
잠깐만요. 하이퍼 파라미터가 뭐예요?  
우선은 가능한 한 간단한 정의부터 시작합시다.  
  
하이퍼 파라미터는 머신 딥러닝 모델을 구축할 때 사용할 수 있는 노브입니다.



하이퍼 파라미터 - '노브' 또는 '다이얼' 비유

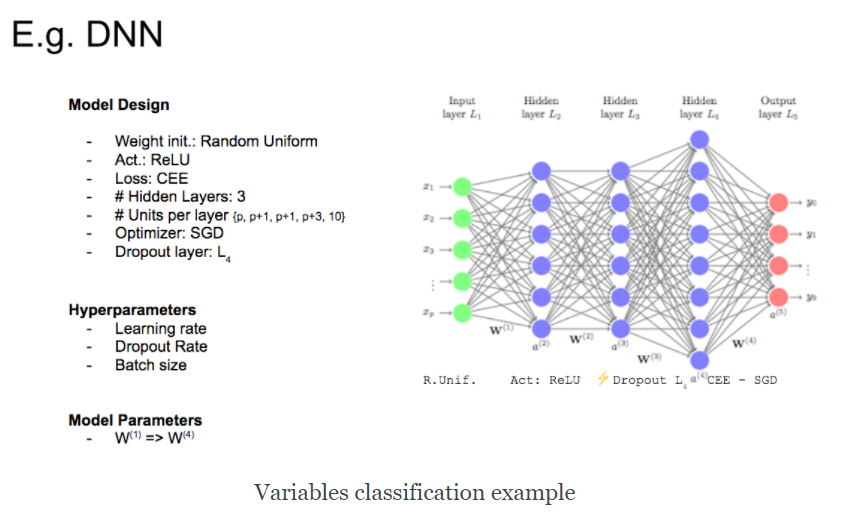
또는 다음과 같이 됩니다.  
  
하이퍼 파라미터는 트레이닝을 시작하기 전에 미리 정해진 값으로, 수동으로 설정된 모든 트레이닝 변수입니다.

러닝 레이트(Learning Rate)와 드롭아웃 레이트(Dropout Rate)는 하이퍼 파라미터로 간주되기도 하는데 모델 설계 변수는 어떨까요? 여기에는 내장(embeddings), 레이어 수(number of layers), 액티베이션 기능(activation function) 등이 포함됩니다. 이들 변수를 하이퍼 파라미터로 생각해야 할까요?

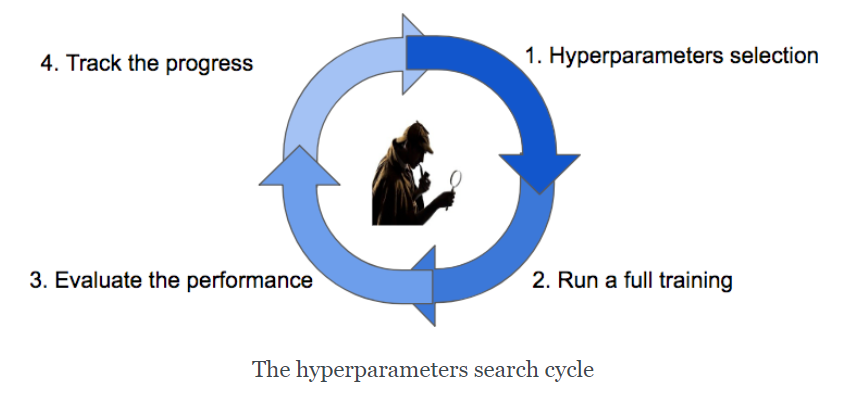


모델 설계 변수 + 하이퍼 파라미터 -> 모델 파라미터

간단하게 하기 위해서, 네. 하이퍼 파라미터 세트의 일부로 모델설계 컴포넌트를 검토할 수도 있습니다.  
  
마지막으로 트레이닝 프로세스에서 취득한 파라미터(데이터에서 학습한 변수)는 어떨까요? 이들 무게는 모델 파라미터로 불립니다. 하이퍼 파라미터 세트에서 제외합니다.  
  
실제의 예를 보겠습니다 딥러닝 모델에서 변수가 다른 분류 예시는 아래 그림을 참조하십시오.



다음 문제: 검색에 비용이 많이 든다.  
하이퍼 매개 변수의 최적 구성을 검색하고자 하는 것으로 나타난 지금, 하이퍼 매개 변수 검색은 💻, 💰 및 ⏳에 의해서 제약되는 반복 프로세스이라는 과제에 직면하고 있습니다.



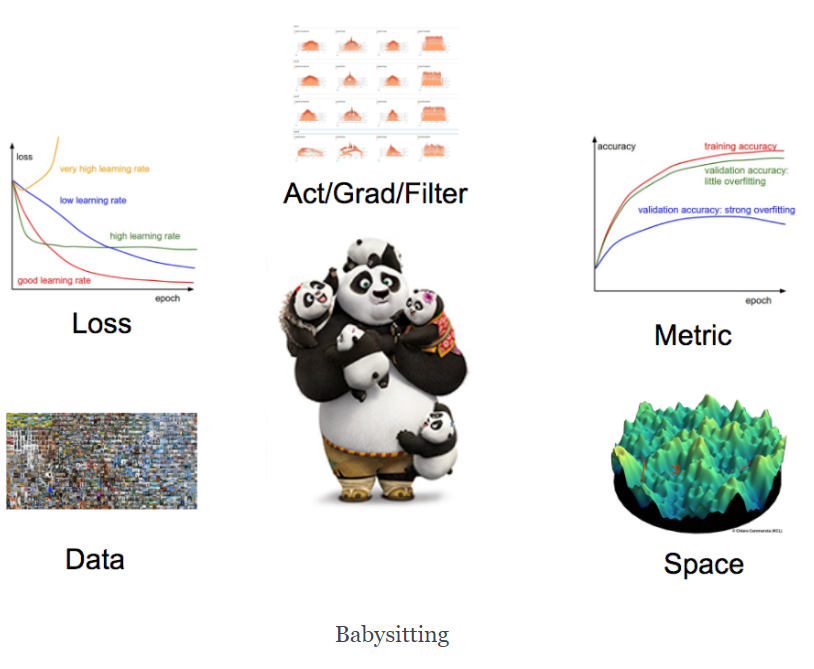
1. 하이퍼 파라미터 선택 2. 충분한 훈련을 하다

3.퍼포먼스를 향상시키다 4. 진척 상황을 추적하다

모든 것은 유망한 구성의 추측(단계1)에서 시작되며 관심 있는 메트릭에 관한 실제 평가를 얻으려면 트레이닝(단계2)이 완료될 때까지 기다려야 합니다 (단계 3). 검색 진행 상황을 추적하고 (단계 4), 검색 전략에 따라 새로운 추측을 선택합니다(단계 1).  
  
종료상태(⏳나💰이 없어지는 등)가 될 때까지 이대로 계속합니다.

전략에 대해 이야기합시다  
최적의 구성을 검색하기 위한 네 가지 주요 전략이 있습니다.  
  
- 베이비시터(시행착오) \_ **Babysitting (aka Trial & Error)**  
- 그리드 검색 \_ **Grid Search**  
- 랜덤 검색 \_ **Random Search**  
- 베이즈 최적화 \_ **Bayesian Optimization**

**Babysitting**   
베이비시터는 학계에서 시행착오 또는 대학원생의 후손으로도 불립니다. 이 접근법은 100% 매뉴얼이며, 연구자, 학생, 취미주의자들에게 가장 널리 채택되고 있습니다.  
  
엔드 투 엔드 워크플로우(The end-to-end workflow )는 매우 단순합니다. 학생들은 학습 프로세스의 모든 단계(데이터 수집에서 feature map 가시화까지)를 수행하는 새로운 실험을 고안하고 시간이 없어질 때까지(일반적으로는 마감이 원인이 되어) 하이퍼 파라미터를 순서대로 반복합니다.



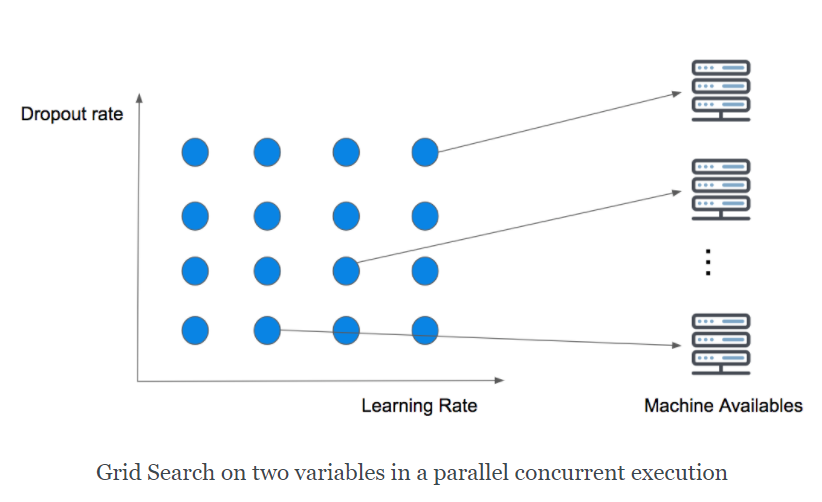
deeplearning.ai 코스에 등록해 보신 분들은 이 방법을 아실 수 있을 겁니다. 바로 Andrew Ng 교수님이 설명한 Panda workflow입니다.  
  
이 접근법은 매우 교육적이지만 데이터 사이언티스트의 시간이 정말 귀중한 팀이나 기업 내에서는 확장될 수 없습니다.   
  
이렇게 해서 다음 질문으로 넘어갑니다.

더 좋은 투자 방법이 있나요?  
물론 그렇습니다! 하이퍼 파라미터 검색의 자동전략을 정의함으로써 고객님의 시간을 최적화할 수 있습니다.

**Grid Search**  
"Justtry everything!" 이라고 하는 명령어에서 파생된 Grid Search는 모든 가능한 구성을 시험하는 단순한 접근법입니다.  
  
워크플로우는 다음과 같습니다.

-그리드를 n차원으로 정의합니다. 여기에서 이들 맵은 하이퍼 파라미터에 대응하고 있습니다. 예: n = (learning\_rate, dropout\_rate, batch\_size)  
-각 차원에 대해 가능한 값의 범위를 정의합니다. 예를 들어 batch\_size = [4, 8, 16, 32, 64, 128, 256]  
-가능한 모든 설정을 검색하고 최선의 설정이 확립될 때까지 결과를 기다립니다. 예를 들어 C1 = (0.10.3, 4) -> acc = 92%, C2 = (0.10.35, 4) -> acc = 92.3% 등입니다.

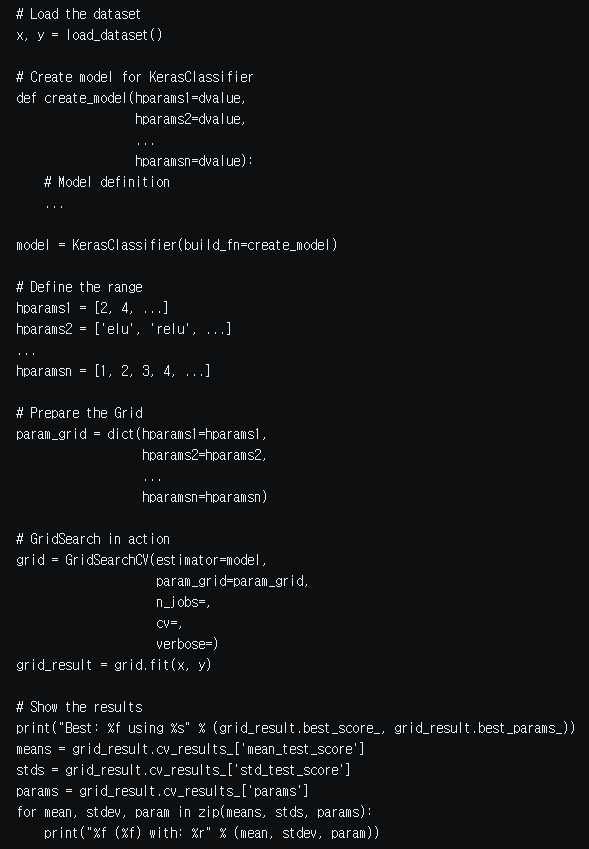
아래의 그림은 드롭아웃율(Dropout)과 러닝률(Learning rate)의 2차원의 단순한 그리드 검색을 나타내고 있습니다.



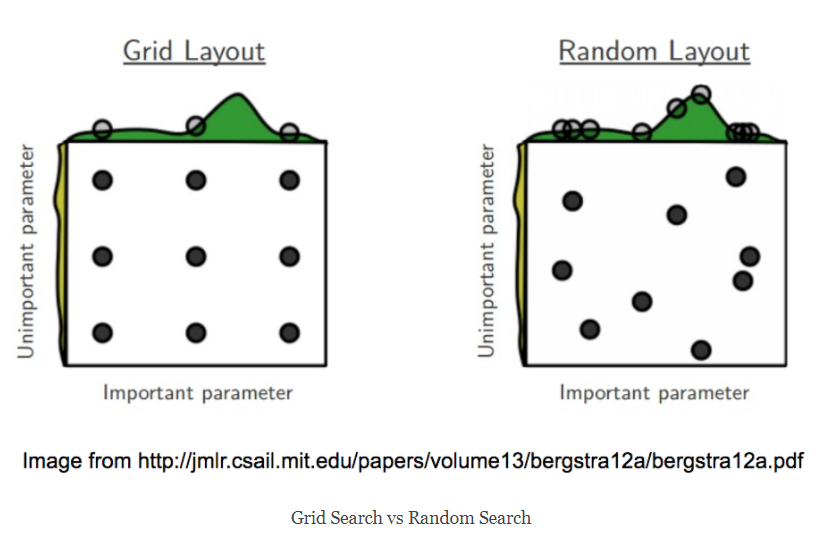
병렬 동시 실행의 두 변수에 대한 그리드 검색

이 전략은 계산이력을 고려하지 않았기 때문에 부끄러울 정도로 병행하고 있습니다 곧 확장할 것입니다. 그러나 이는 즉 이용 가능한 계산 자원이 많을수록 동시에 많은 추측을 할 수 있다는 것입니다.  
  
이 접근법의 진짜 과제는 차원의 저주로 알려져 있습니다 즉, 더 많은 차원을 추가할수록 검색은 시간의 복잡함(통상은 지수 함수적인 요인)으로 폭발해 결국에는 이 전략은 실행 불가능 해집니다.  
  
이 방법은 치수가 4 이하인 경우에 일반적으로 사용됩니다. 그러나 실제로는 최적의 구성을 찾는다는 것이 최종적으로 보증된다 하더라도 권장하지는 않습니다. 그 대신에, 랜덤 검색을 사용하는 것이 좋다. 이에 대해서는 다음에 설명한다.

지금 바로 그리드 검색을 시도해 보세요.  
이 버튼을 클릭해서 Floyd Hub에 워크스페이스를 엽니다. workspace를 사용해 완전히 구성된 클라우드 시스템에서 이하의 코드(Scikit-learn과 Keras를 사용한 그리드 검색)를 실행할 수 있습니다.

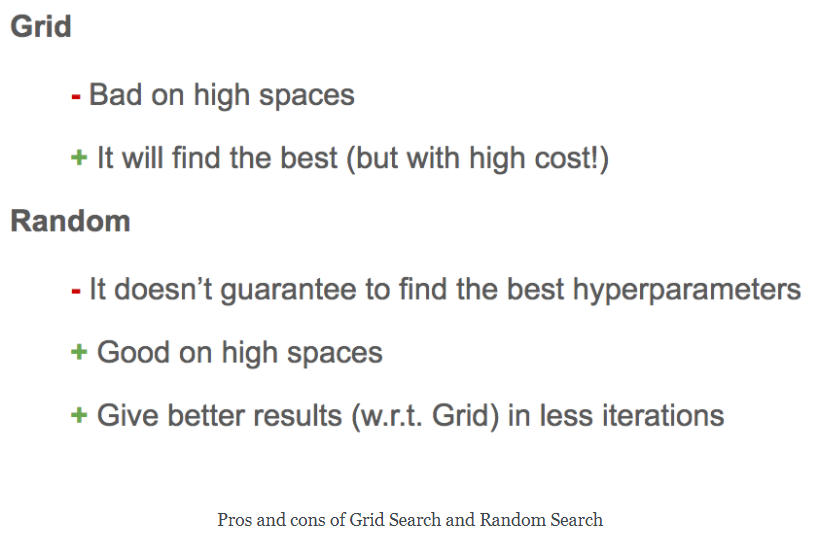


랜덤 검색  
몇 년 전, Bergstra와 Bengio는 놀라운 논문을 발표하여 Grid Search의 비효율성을 증명하였다.  
  
그리드 검색과 랜덤 검색의 유일한 진정한 차이는 전략 사이클의 스텝 1에 있습니다. 랜덤 검색은 구성 공간에서 랜덤하게 포인트를 선택합니다.  
  
이하의 화상(논문 제공)을 사용하여, 연구자로부터 보고된 클레임을 표시합시다.

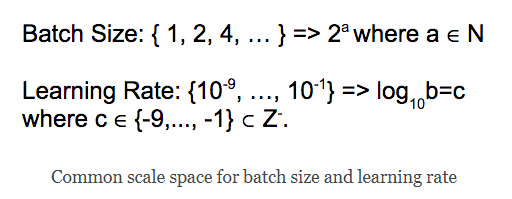


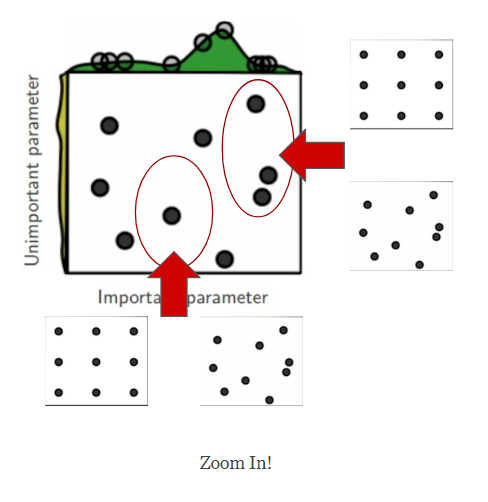
이 이미지에서는 2개의 하이퍼 파라미터 공간에서 최적의 구성을 검색함으로써 2가지 접근방식을 비교합니다. 또, 한쪽 파라미터가 다른 쪽 파라미터보다 중요하다는 것을 전제로 하고 있습니다. 이것은 안전한 전제 조건입니다.왜냐하면, 딥러닝 모델은, 서두에서 말한 것처럼, 실제로는 하이퍼 파라미터로 가득 차 있어, 통상, 연구자 과학자 학생은, 어느 것이 트레이닝에 가장 큰 영향을 주는지를 파악하고 있기 때문입니다.  
  
Grid Layout 에서는 9개의 모델을 훈련을 한다고 해도 변수별로 3개의 값만 사용하고 있다는 것을 쉽게 알 수 있습니다. 한편 랜덤 레이아웃에서는 동일한 변수를 여러 번 선택하는 경우가 거의 없습니다. 두 번째 접근방식에서는 변수별로 9개의 서로 다른 값을 사용해서 9개의 모델을 훈련하게 됩니다.

이미지의 각 레이아웃 상단에 있는 공간탐색에서 알 수 있듯이 우리는 랜덤 탐색(특히 더 중요한 변수)을 통해 하이퍼 파라미터 공간을 보다 넓게 탐색했습니다. 그러면 더 적은 반복으로 최적의 구성을 찾을 수 있습니다.  
  
정리: 검색 공간에 3 ~ 4 차원이 포함되어 있으면 그리드 검색을 사용하지 마십시오. 대신에, 랜덤 검색을 사용합니다. 랜덤 검색은 각 검색 태스크에 매우 우수한 기준선을 제공합니다.



지금 바로 랜덤 검색을 시도해 보세요.  
달리다  
이 버튼을 클릭해서 Floyd Hub에 워크스페이스를 엽니다. 워크스페이스를 사용하여 완전히 구성된 클라우드 시스템에서 이하의 코드(Scikit-learn과 Keras를 사용한 랜덤 검색)를 실행할 수 있습니다.

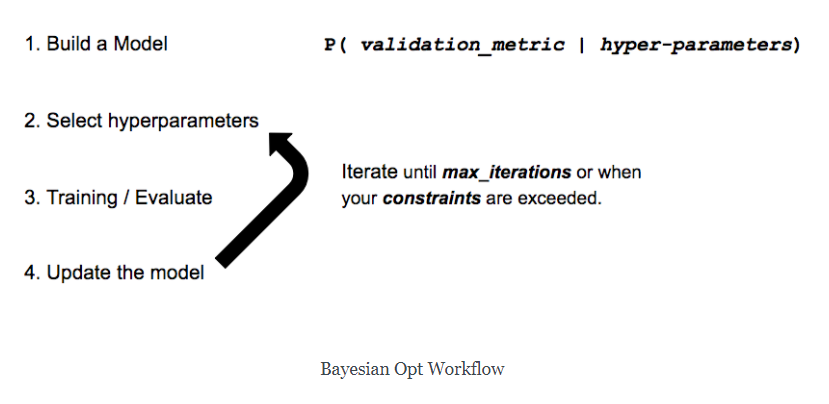
한걸음 물러서 두걸음 앞으로  
여담이지만, 각 차원의 공간을 설정해야 할 경우에는 변수별로 적절한 스케일을 사용하는 것이 매우 중요합니다.  
  
  
배치 크기와 학습률의 공통 스케일 스페이스   
예를 들어 배치 크기 값을 2의 제곱으로 사용하여 로그 스케일로 학습률을 샘플링하는 것이 일반적입니다.



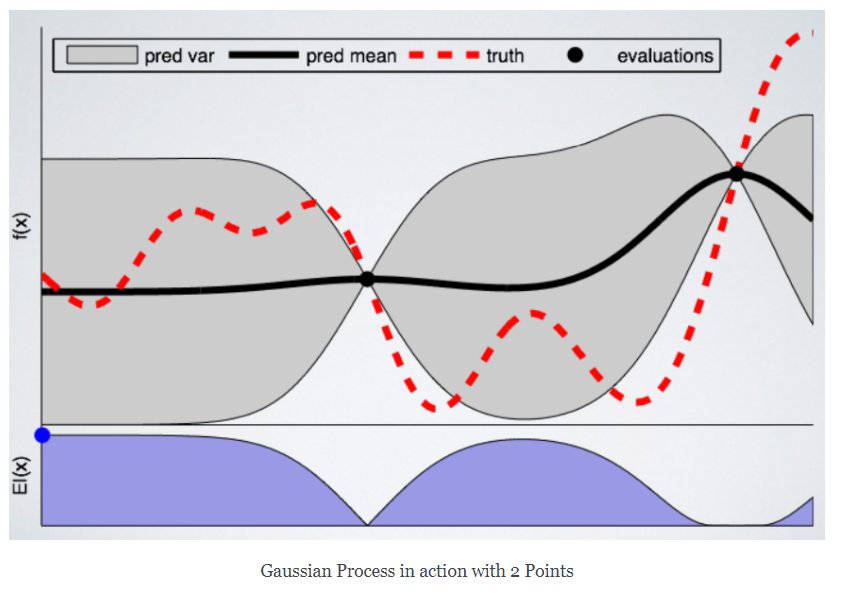
또, 상기의 레이아웃의 하나로부터 시작해, 각 변수의 범위를 보다 조밀하게 샘플링 해 유망한 부분 공간에 줌인 해, 같거나 다른 검색 전략으로 새로운 검색을 시작하는 것도 자주 있다.

또 다른 문제가 있습니다 독자적인 추측입니다  
안타깝게도 Grid와 Random Search 모두 공통적으로 결점이 있습니다.  
  
하나하나 새로운 추측은 앞의 예상과 무관합니다!  
기묘하고 놀라운 일처럼 들릴지 모르지만, Babysitting이 효과적으로 기능하고 있는 것은 필요한 시간임에도 불구하고 과학자가 과거를 자원으로 이용함으로써 탐색과 실험을 효과적으로 진행하는 능력이다.  
  
잠깐만요. 들은 기억이 있는 것 같군요. 하이퍼 파라미터 검색을 기계학습 태스크로 모델화하려면?!  
  
베이즈 최적화를 소개합니다.

베이즈 최적화  
이 검색 전략은 하이퍼 파라미터 구성으로부터 우리가 관심을 가지는 메트릭을 예측하려고 하는 대리 모델을 구축한다.  
  
새로운 반복을 할 때마다 우리는 어떤 새로운 추측이 개선될지에 대해 점점 더 자신감을 가질 것입니다 다른 검색 전략과 마찬가지로 종료 조건도 동일합니다.

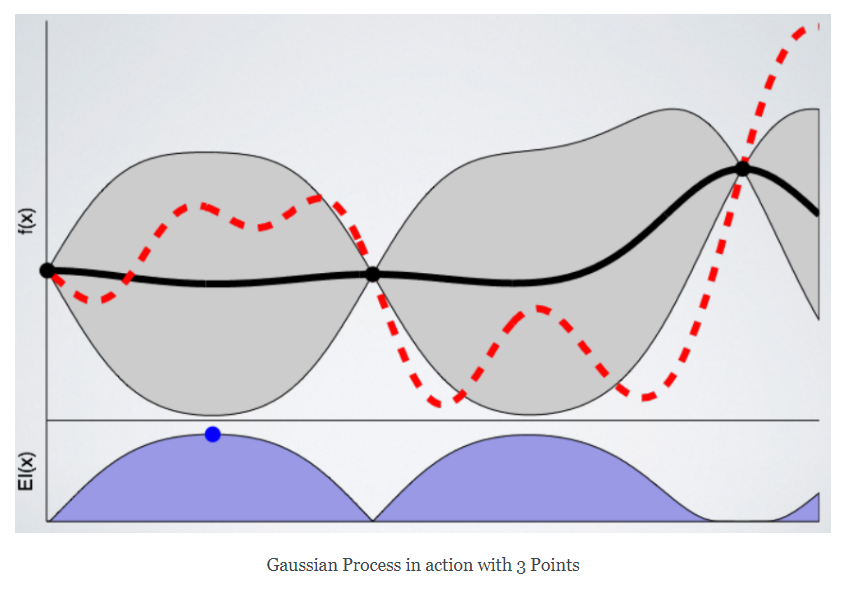


만약 지금 이것이 혼란스러운 것 같으면 걱정하지 마세요. 지금이 또 다른 시각적인 예시를 들 때입니다.  
  
동작중인 가우스 (Gausian Process)  
우리는 가우스 프로세스를 하이퍼 파라미터 구성으로부터 관심 있는 메트릭으로의 매핑을 학습하는 대리로서 정의할 수 있습니다. 예측치로서 뿐만 아니라 불확실성의 범위(평균값과 분산값)도 제시합니다.  
  
이 놀라운 튜토리얼의 예를 자세히 살펴보겠습니다.

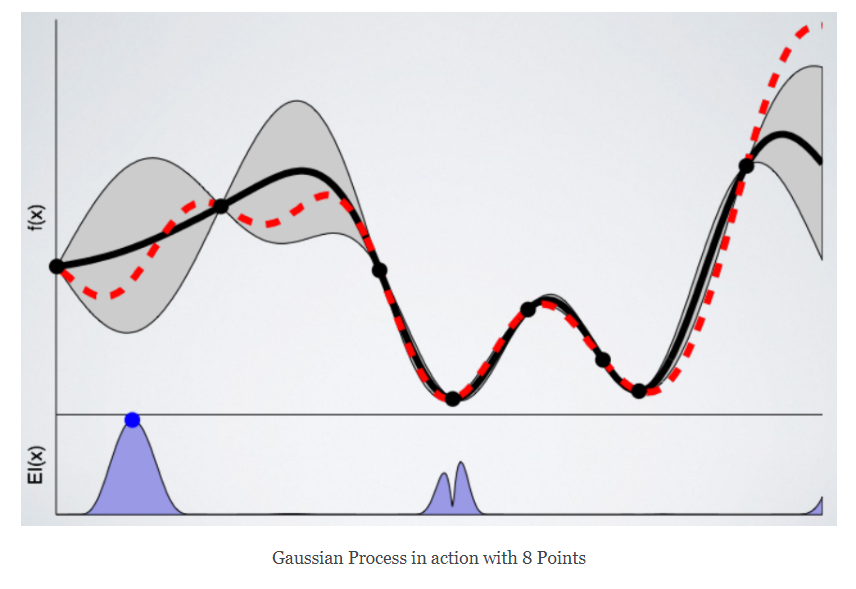


위 그림에서는 단일 변수(수평축상)에 대한 가우스·프로세스 최적화의 첫 번째 스텝을 따르고 있습니다. 상상의 예시에서 이는 학습률 또는 드롭아웃률을 나타낼 수 있습니다.  
  
세로축에서는 단일 하이퍼 파라미터의 함수로 관심 있는 메트릭을 플롯하고 있습니다. 되도록 낮은 값을 찾고 있기 때문에 손실함수라고 생각할 수 있습니다.  
  
검은 점은 지금까지 훈련을 받은 모델을 나타내고 있어요. 빨간 선은 우리가 배우려 하고 있다 기본적인 진실 즉 기능입니다 흑선은 기저진리함수에 대해 우리가 갖고 있는 실제 가설의 평균을 나타내고 회색 영역은 공간에서 관련된 불확실성 또는 분산을 나타내고 있습니다.

아시다시피 점 주위의 불확실성이 줄어듭니다. 이것은, 이러한 점에 관해서 얻을 수 있는 결과에 매우 자신이 있기 때문입니다(여기서 모델을 트레이닝을 했기 때문입니다). 그리고 정보가 적은 지역에서는 불확실성이 커진다.  
  
출발점을 정의했으므로 모델을 훈련할 다음 유망 변수를 선택할 준비가 되었습니다. 그러기 위해서는 다음 구성을 어디에서 샘플링하면 되는지를 나타내는 취득기능을 정의할 필요가 있습니다.  
  
이 예에서는 '기대되는 개선'을 사용하고 있습니다. 이것은 불확실성 영역에서 제안된 구성을 사용할 경우에 가능한 한 낮은 값을 찾는 것을 목적으로 하는 함수입니다. 위의 「개선 전망」차트의 푸른 점은, 다음 트레이닝에서 선택한 포인트를 나타내고 있습니다.



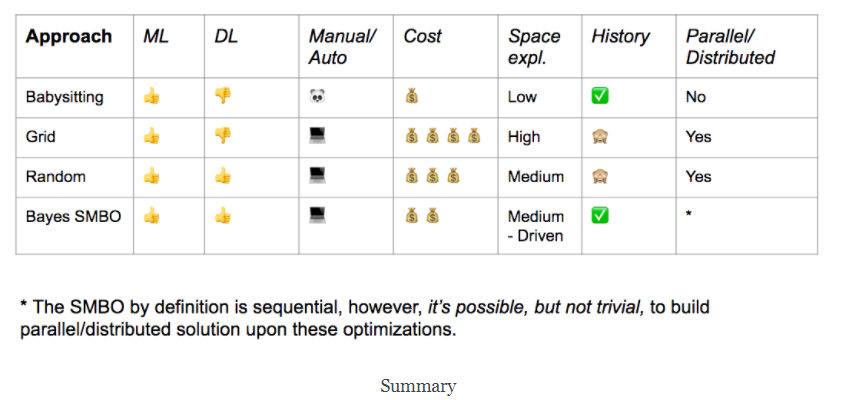
훈련하는 모델이 늘어날수록 대리인은 다음에 기대할 수 있는 샘플 포인트에 대해 자신감을 갖게 됩니다. 그 다음 8개의 트레이닝 받은 모델의 차트를 보여줍니다.



가우스 프로세스는 Sequential Model Based Optimization(SMBO; 시퀀셜 모델베이스 최적화)라 불리는 알고리즘의 클래스에 속합니다. 지금까지 살펴본 바와 같이 이들 알고리즘은 최적의 하이퍼 파라미터 설정 검색을 개시하기 위한 매우 뛰어난 베이스라인을 제공합니다. 그러나 모든 툴과 마찬가지로 아래와 같은 단점이 있습니다.  
  
정의상 프로세스는 시퀀셜입니다.  
수치 파라미터만을 처리할 수 있습니다.  
성적이 나쁘면 훈련을 중지하는 메커니즘은 없습니다.  
이 흥미로운 토픽에 대해서, 우리는 아직 표면적인 설명을 했기 때문에 주의하세요. 좀 더 자세한 읽기와 SMBO를 확장하는 방법에 관심이 있으신 분들은 이 논문을 읽어보시기 바랍니다.  
  
지금 바로 베이즈 최적화를 시도해 보세요.

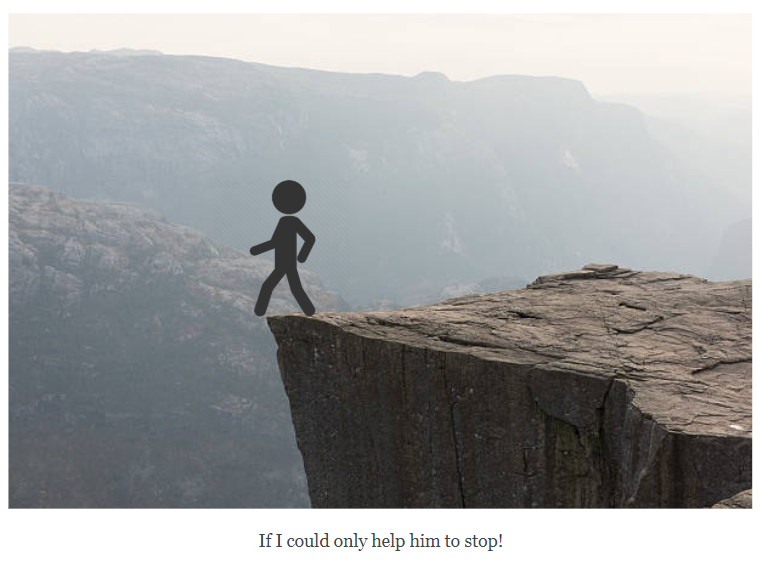
이 버튼을 클릭해서 Floyd Hub에 워크스페이스를 엽니다. 워크스페이스를 사용하여 완전히 구성된 클라우드 시스템에서 이하의 코드(Hyperas를 사용한 Bayesian Optimization(SMBO-TPE))를 실행할 수 있습니다.

검색전략의 비교  
마지막으로, 각 제안의 장점과 단점을 이해하기 위해 지금까지 설명한 내용을 정리해보겠습니다.

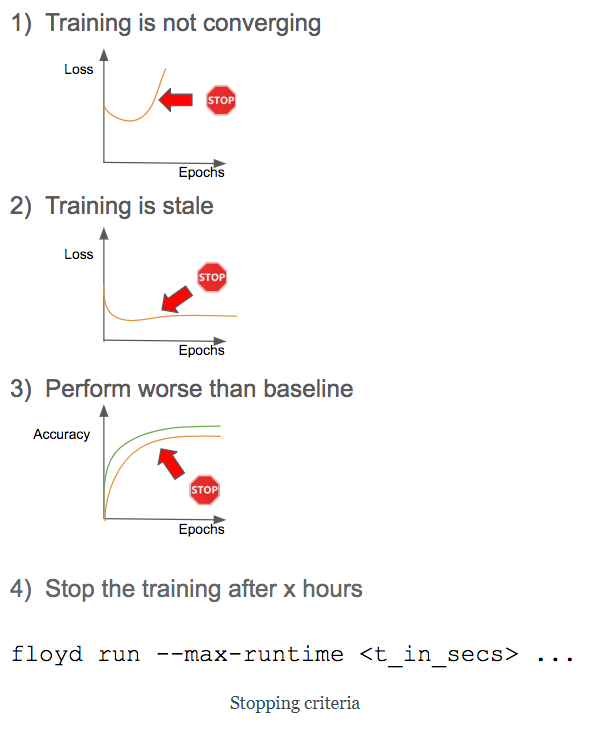


리소스가 당신이나 당신의 팀에 제약이 되지 않는 한, 아마 베이스라인 SMBO가 최선의 후보입니다만, 랜덤 검색을 통한 베이스라인 확립도 검토해야 합니다.  
  
한편 아직 배우고 있거나 개발 단계에 있다면 비록 우주 탐사와 관련해서 실용적이지 못하더라도 베이비시터가 최선의 방법입니다.  
  
중소기업 섹션에서 설명한 바와 같이 훈련 성과가 부진하거나 더 악화되면 이들 전략 모두 자원을 절약하는 메커니즘을 제공하지 않습니다. 계산이 끝날 때까지 기다려야 합니다.  
  
이렇게 해서, 우리의 훌륭한 탐구의 마지막 질문에 도착합니다.

훈련시간을 최적화할 수 있나요?  
알아봅시다.  
  
빨리 멈추는 힘

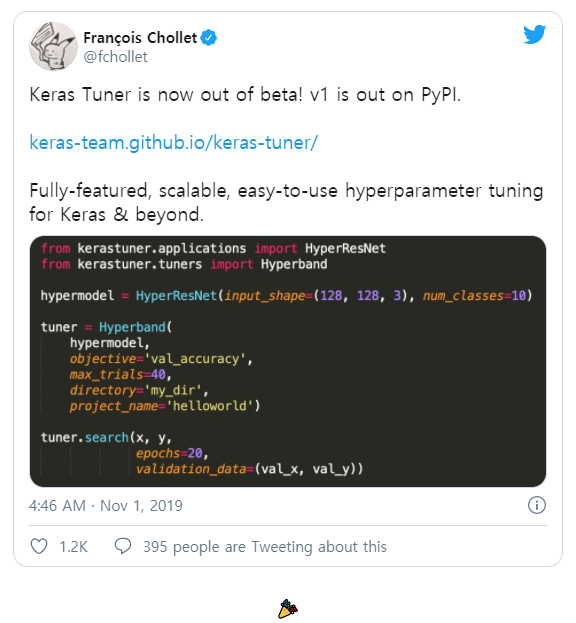


조기 정지는 유명한 정규화 기술일 뿐만 아니라 훈련이 적절한 방향으로 진행되지 않았을 때 자원 낭비를 막기 위한 뛰어난 메커니즘을 제공합니다.  
  
다음으로 가장 많이 사용되고 있는 정지기준의 그림을 나타냅니다.



처음 세가지 기준은 자기 설명이니까 마지막 기준을 주목해봅시다.  
  
연수 시간은 연구실내의 실험의 클래스에 따라서 제한하는 것이 일반적입니다. 이 정책은 실험의 깔때기 역할을 하며 팀 내 자원을 최적화합니다. 이렇게 해서 우리는 가장 유망한 실험에만 더 많은 자원을 할당할 수 있을 것이다.  
  
Floyd-cli(사용자가 Floyd Hub과 통신하기 위해 사용하고 Github 상에서 오픈 소스를 제공하고 있는 소프트웨어)는 우리의 파워유저가 실험을 규제하기 위해 그것을 대량으로 사용하고 있다는 것을 의미하는 플래그를 제공하고 있다.  
  
이러한 기준은 학습 프로세스의 베이비시터 시 수동으로 적용하기도 하고, 가장 일반적인 프레임워크에서 제공하는 후크콜백을 사용하여 실험에서 이러한 규칙을 통합함으로써 더욱 효과적으로 적용할 수 있습니다.

Keras는 우수하다 Early Stopping 기능을 제공하여 더욱 편리한 콜백 스위트를 제공합니다. Keras는 최근 TensorFlow 내에 통합되어 TensorFlow 코드 내에서 콜백을 사용할 수 있습니다.  
Tensor Flow는 트레이닝 훅을 제공합니다. 이들은 아마도 Keras 콜백(또는 tf. keras API)으로서는 직관적이지는 않지만 실행 상태를 보다 상세하게 제어할 수 있습니다. TensorFlow 2.0 (현재 베타판) 에서는 하이퍼 파라미터 최적화를 관리하기 위한 새로운 API가 도입되어 있습니다. 상세한 것에 대하여는, TensorFlow 의 공식 문서를 참조해 주세요.  
Tensor Flow Keras 영역에는 더 많은 것이 있습니다. Keras 팀은 TensorFlow 2.0을 탑재한 tf. keras 전용 하이퍼 파라미터 튜너를 막 출시했습니다.



현시점에서 PyTorch는 아직 훅이나 콜백 컴포넌트를 제공하지 않지만, Torch Sample 리포트나 훌륭한 포럼에서 확인할 수 있습니다.  
fast.ai 라이브러리에서 콜백을 받을 수 있습니다자세한 내용은, fastai의 공식 콜백 문서 페이지를 봐 주세요. 길을 잃으셨거나 도움이 필요하시면 훌륭한 fast.ai 커뮤니티에 연락하시길 권합니다.  
Ignite(PyTorch 의 고급 라이브러리)는 Keras 와 같이 콜백을 제공합니다. 그 도서관은 실제로 개발 중인데, 확실히 흥미로운 선택지인 것 같아요.

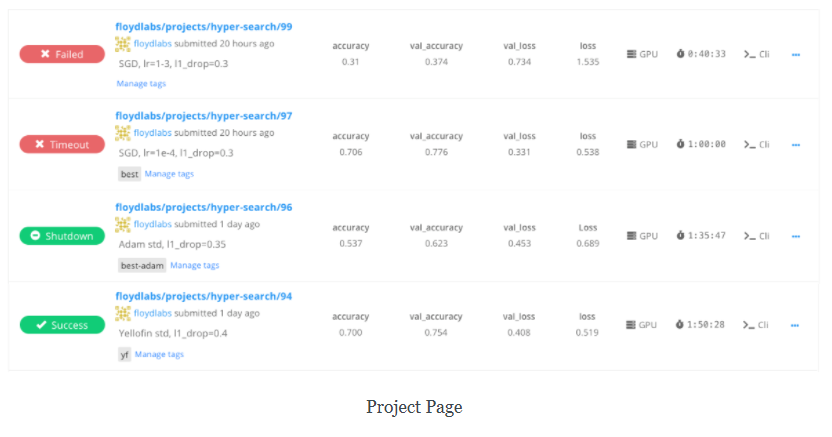
논의를 가장 많이 사용되고 있는 트렌드 프레임워크에 한정하기 위해 저는 여기서 목록을 정지하겠습니다(다른 프레임워크 저자의 감성을 손상시키지 않기를 바랍니다). 만약 그렇다면, 불평을 저에게 전해 주시면 기꺼이 내용을 갱신하겠습니다.!)  
  
이게 끝이 아닙니다.  
기계 학습에는 「AutoML(자동 기계 학습)」라고 불리는 서브 필드가 있어, 모델 선택의 방법, 특징 추출 및 또는 하이퍼 파라미터 최적화를 자동화하는 것을 목적으로 하고 있다.  
  
이 툴은 마지막 질문에 대한 답입니다(약속합니다!).

우리는 모든 과정을 배울 수 있나요?  
AutoML은 Baeysian Optimization에서 했던 것과 같은 또 다른 머신 학습 태스크를 해결하는 머신 학습 태스크로 여겨집니다. 이것이 메타 머신 러닝입니다.

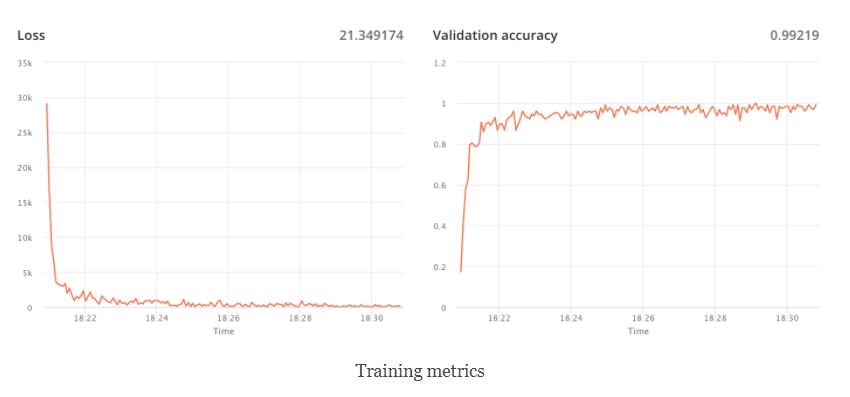
조사: AutoML 및 PBT  
Google의 AutoML은 뉴럴 아키텍처 검색을 위한 브랜드 변경이다. 이 기사의 첫머리에서 모델설계 컴포넌트를 하이퍼 파라미터 변수로 마지하기로 했습니다. 뉴럴 아키텍처 탐색은 AutoML 서브필드에서 주어진 태스크에 가장 적합한 모델을 찾는 것을 목적으로 합니다. 이 문제에 대해 충분한 논의를 하려면 일련의 기사가 필요할 것이다. 다행히도, fast.ai의 레이첼 토머스 박사는 멋진 일을 해줘서 우리는 기꺼이 링크할 수 있습니다.  
  
Deep Mind의 이제 1개의 흥미로운 연구 성과를 드리고 싶어요. 거기에서는 진화전략 알고리즘의 바리에이션을 사용해 Population Based Training이라 불리는 하이퍼 파라미터 검색을 실행했습니다.(PTB는 보도진에서는 별로 다루지 않고 있습니다만, 제가 강력히 추천하는 DeepMind도 이미 하나의 놀라운 연구의 기초가 되기도 합니다. 스스로 체크아웃하세요). DeepMind의 말을 인용:

PBT - 랜덤 검색과 같이 - 많은 뉴럴 네트워크를 랜덤 하이퍼 파라미터와 병행하여 훈련하는 것으로 시작합니다. 그러나 네트워크 훈련을 개별적으로 하는 것이 아니라 다른 사람들로부터의 정보를 사용하여 하이퍼 파라미터를 개량하고 장래성을 나타내는 모델에 계산 자원을 할당합니다. 이는 노동자로 알려진 집단의 각 구성원이 나머지 집단의 정보를 이용할 수 있는 유전적 알고리즘으로부터 영감을 얻고 있습니다. 예를 들어 워커는 보다 뛰어난 퍼포먼스를 발휘하는 워커부터 모델 파라미터를 복사할 수 있습니다. 또한 현재 값을 랜덤으로 변경해 새로운 하이퍼 파라미터를 탐색할 수도 있습니다.  
물론이 분야에는 아주 흥미로운 연구들이 많이 있겠죠. 최근에 뉴스로 유명해진 사람들에 대해 이야기했어요.

Floyd Hub에서의 실험 관리  
Floyd Hub의 가장 큰 특징 중에 하나는 서로 다른 하이퍼 파라미터 세트를 사용할 때 훈련 중에 다른 모델을 비교할 수 있다는 것입니다.  
  
아래 사진은 Floyd Hub 프로젝트의 업무 목록입니다. 이 사용자가 작업의 메시지 필드(예: floyd run --message "SGD, lr=1e-3, l1\_drop= 0.3"…)를 사용해 이들 각 작업에서 사용되는 하이퍼 파라미터를 강조 표시하고 있는 것을 알 수 있습니다.  
  
게다가 각 작업의 트레이닝 메트릭도 표시할 수 있습니다. 이들 태스크 중 어느 작업이 가장 잘 실행되었는지 사용하는 머신의 종류와 총 훈련시간을 쉽게 파악할 수 있습니다.



Floyd Hub 대시보드는 하이퍼 파라미터의 최적화에 관한 훈련을 모두 비교하는 간단한 방법을 제공하고 실시간으로 갱신합니다.  
  
우리의 조언은 당신이 해결해야 할 과제들마다 다른 Floyd Hub 프로젝트를 작성하는 것입니다. 이렇게 하면 작업을 정리하고 팀과 공동 작업하는 것이 쉬워집니다.  
  
훈련 지표  
위와 같이 Floyd Hub 일에서 간단히 트레이닝 지표를 내줄 수 있습니다. Floyd Hub 대시보드에서 당신의 일을 보면 당신이 정의한 각 지표의 실시간 차트를 찾을 수 있다.  
  
이 기능은 Tensorboard를 대체하는 것이 아니라(이 기능도 제공하고 있습니다), 선택한 하이퍼 파라미터의 설정을 바탕으로 트레이닝 동작을 강조하는 것을 목적으로 하고 있습니다.  
  
예를 들면 트레이닝 프로세스의 베이비시터를 하고 있는 경우, 트레이닝 지표는 확실히 정지 기준을 결정하고 적용하는 데 도움이 됩니다



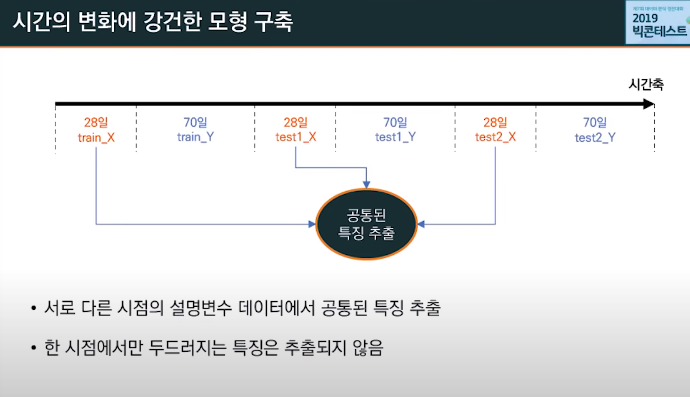
Floyd Hub Hyper Search (조만간 출시 예정)  
현재 Floyd Hub에서 하이퍼 파라미터 검색을 효과적으로 실행하기 위해서 제안된 전략으로 floyd-cli 커맨드라인 툴을 랩하는 방법의 예를 몇 가지 출시할 예정입니다. 그러니까 기대해주세요!  
  
마지막으로 하나 더!  
Floyd Hub 유저 중에는 Floyd Hub 내에서 심플한 하이퍼 파라미터 검색 솔루션(Google Vizier 페이퍼에서 제안된 솔루션과 동일)을 요구하는 사람도 있다. 도움이 될 것 같으면 지원자에게 연락하거나 포럼에 투고해 주십시오.  
  
Floyd Hub를 개선하여 여러분의 트레이닝 요구에 부응하기를 기대합니다.

### 당신은 생활 모델을 하고 있습니까? 👩‍💻 🤖

MLDL 사용자 조사 참여 매월 시원한 AI셔츠 선물💥  
  
MLDL 사용자 연구를 위해 전임 데이터 사이언티스트를 모집하고 있습니다. 조정된 사용자 조사 실험에 45분간 참여합니다. 그 조사는 화상통화로 행해질 예정이다. 팀 동료들에게 과시할 만한 재미있는 티가 많이 있어요. 1년 동안은 매달 다른 상품을 발송합니다!  
  
자세한 사항은 여기를 클릭해 주세요.

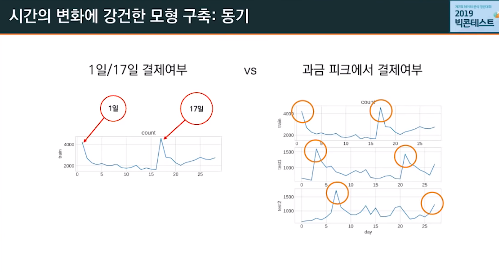
AI 라이터를 위한 Floyd Hub 콜  
알레시오 같은 훌륭한 기사를 써서 인공 종합 지능으로 가는 긴 여정에서 당신의 역할을 다하고 싶으십니까? 획기적인 A의 실용화를 위해, 세계 최고의 블로그를 구축하기 위해 열정적인 라이터를 모집하고 있습니다. I. 테크닉 Floyd Hub은 AI 커뮤니티 내에서 큰 영향력을 가지고 있어 당신의 도움을 받아 AI의 다음 파도를 자극할 수 있다. 지금 바로 신청해 주세요.

-전처리

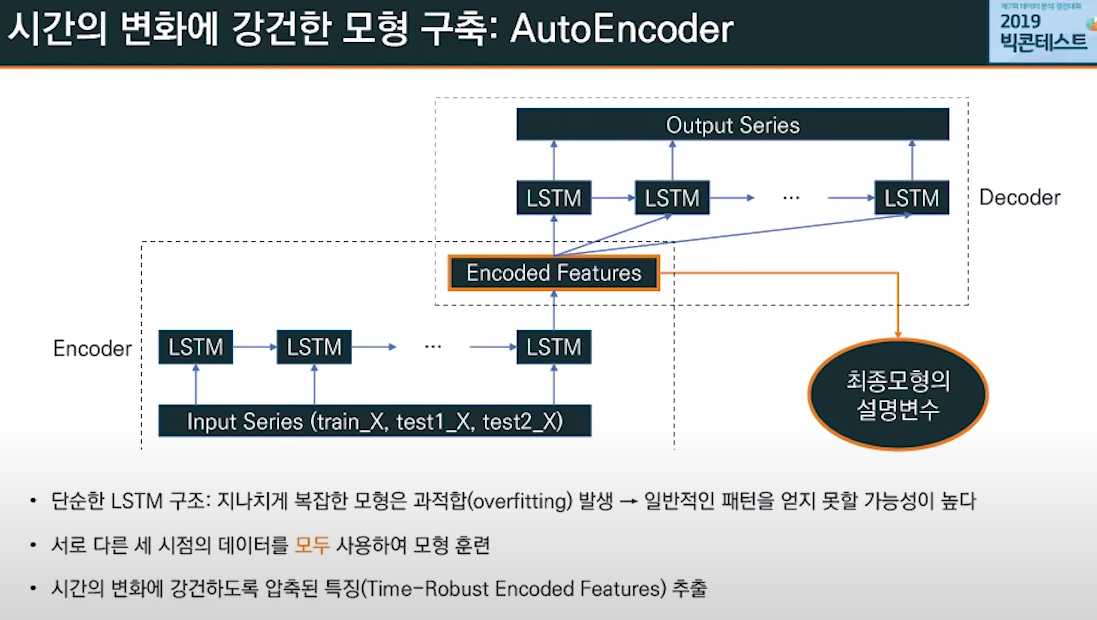


작년 문제 핵심 키워드: 시간의 변화에 강건한 모형 구축

서로 다른 시점에서 수집된 자료에 모두 존재하며, 시점에 의존하지 않는 공통적인 패턴을 찾기 =>



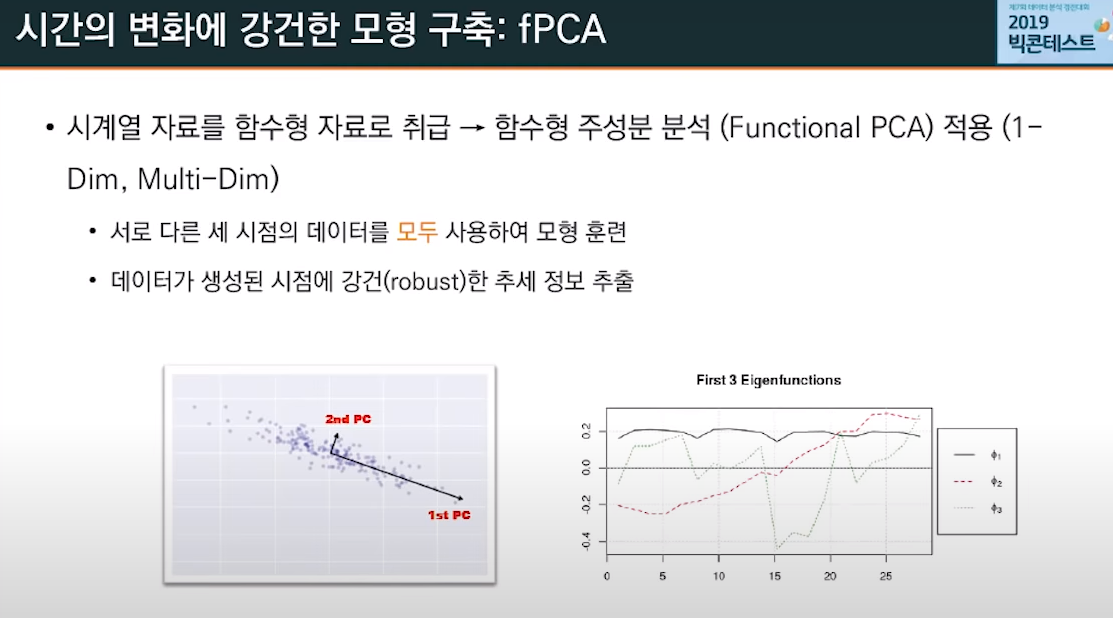
자료 수집 시점에 영향을 덜 받으면서, 강건한 설명변수를 작용적으로 찾아내고자 함



시간의 변화에 강건한 모형을 구축하기 위해 Auto Encoder 라는 딥러닝 구조를 적용

Auto Encoder=일반적으로 데이터의 압축, 복원에 사용-> 공통적이고 일반적인 특징을 잘 뽑아줄 것이다

시간의 변화에 강건한 특징을 추출하기 위해 3가지 시점을 모두 모형 훈련에 사용

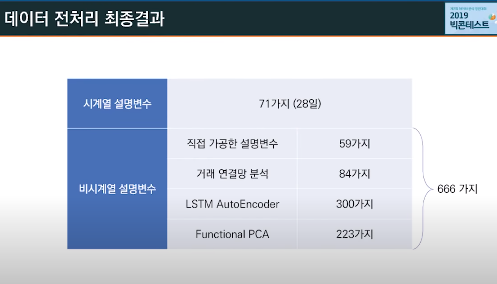


함수형 주성분 분석을 사용해 시간의 변화에 강건한 모형을 구축하고자 함

fPCA는 PCA를 함수형 데이터에도 적용할 수 있게 확장한 것

28일간의 시계열 설명변수 데이터를 하나의 함수로 생각하고 본 방법론 적용

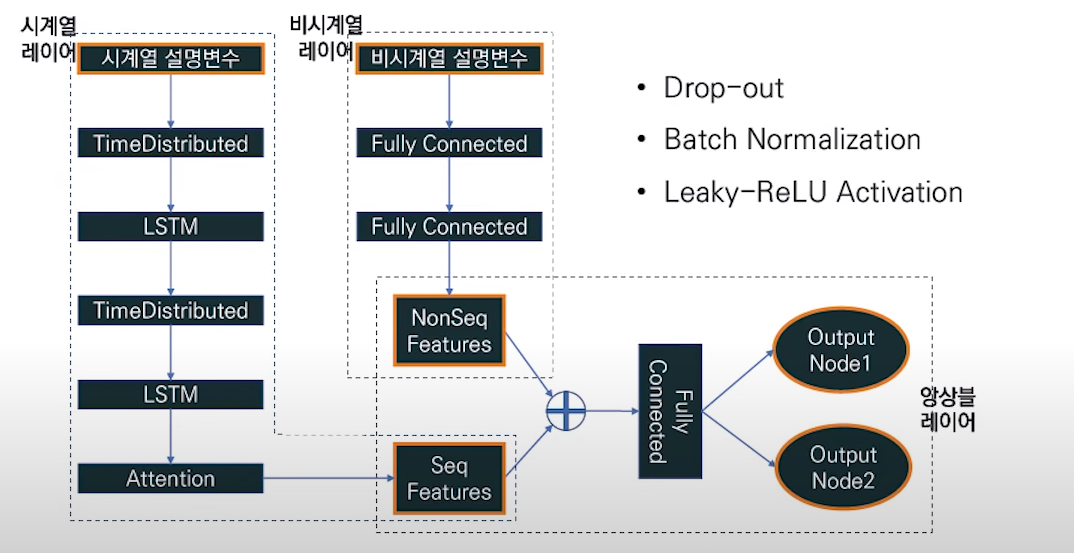
fPCA는 그림과 같이 EigenFunction이 결합된 형태로 원자료를 설명하기 때문에 다양한 함수의 패턴을 유용하게 캐치



데이터 전처리 진행 이후 결과: 전처리로 얻은 설명변수는 모두 731가지로

시계열 설명변수 71가지 + 비시계열 설명변수 666가지

-모델 구축 및 훈련

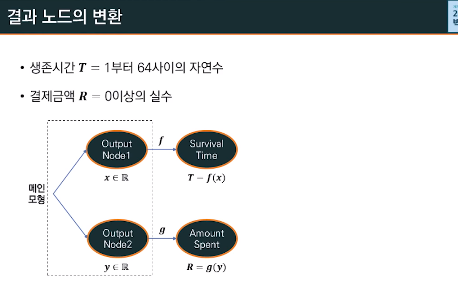


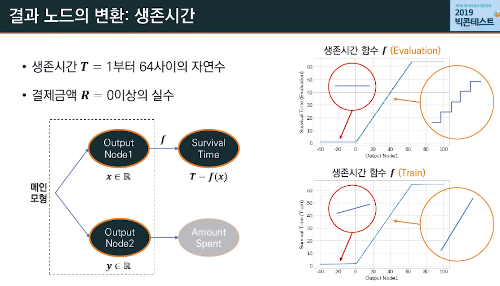
최신의 딥러닝 방법 사용

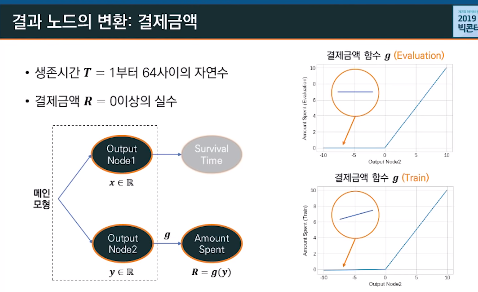
시계열, 비시계열, 앙상블레이어

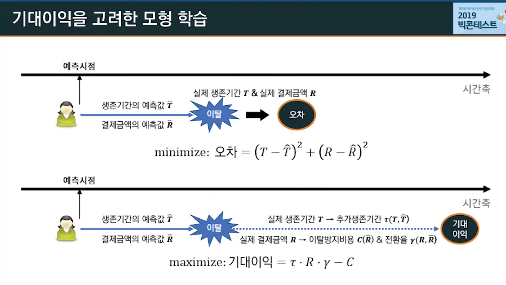
시계열 레이어와 비시계열 레이어는 각각 시계열, 비시계열 데이터를 입력으로 받아 시계열특징(seq Features)과 비시계열 특징(Non seq Features)을 출력하는 구조

앙상블 레이어는 이 2가지 특징을 병합한 데이터를 입력으로 받아 2가지 결과값(생존기간,결제금액)을 내놓음



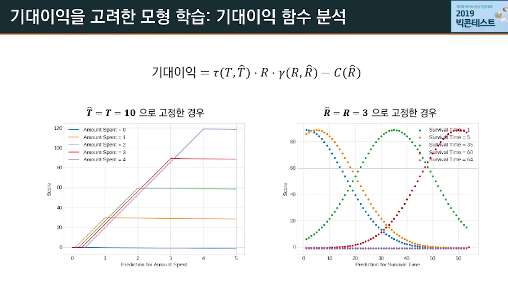


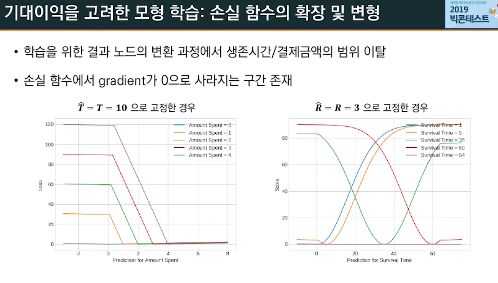


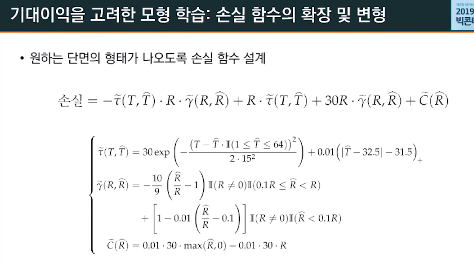


작년 문제 핵심 키워드: 기대이익 고려한 모형

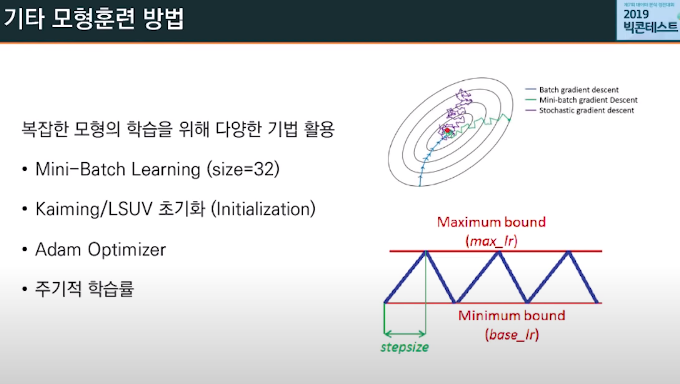
해당 공식 NC소프트에서 제공





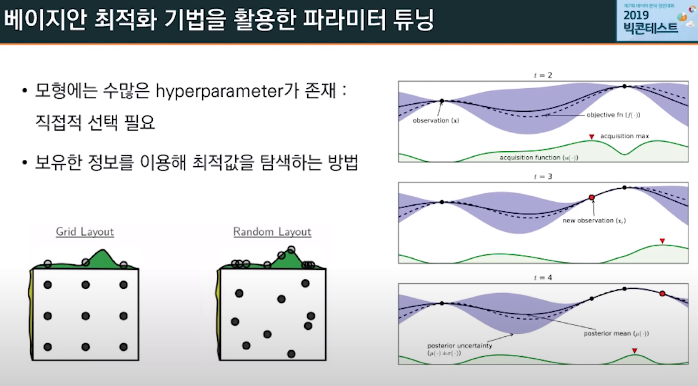


손실함수 직접 구해 모델 학습에 적용

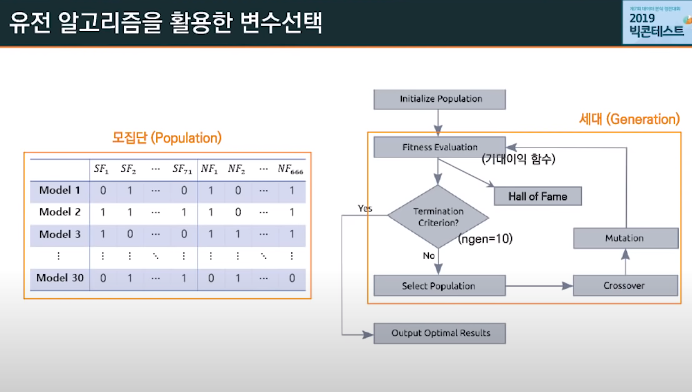


다음은 이외에 적용한 딥러닝 훈련방법들

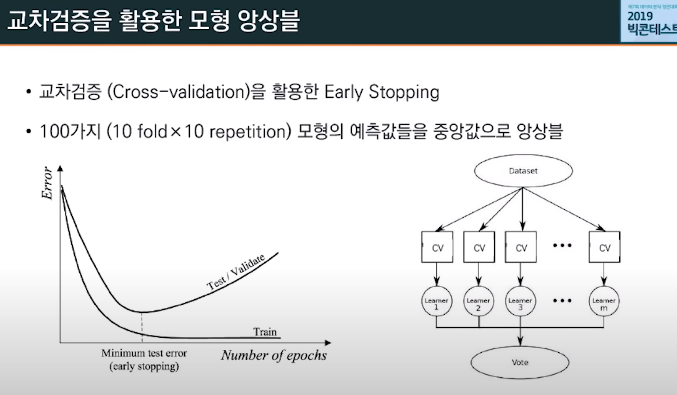
모두 성공적인 딥러닝 모델 학습을 위해서 널리 사용하는 방법론으로 위 4가지 사용함



수많은 하이퍼파라미터를 선택하기 위해서 베이지안 최적화 방법론 사용



데이터의 수많은 변수 중 유의미한 변수들을 선별하기 위해서 일종의 변수선택 방법론으로 유전 알고리즘을 활용



교차검증 방법론을 적용해서 딥러닝의 에폭이라고 부르는 훈련데이터 전체를 한번 훈련한 횟수를 결정해줌. 마지막으로 훈련과정에서 발생했을지 모르는 과적합을 방지해주기 위해 100가지 모형의 예측값들을 중앙값으로 앙상블 해줌