2021 Spring

Artificial Intelligence & Deep Learning

Prof. Minsuk Koo

Department of Computer Science & Engineering
Incheon National University

5.5 하이퍼 매개변수 최적화

- 학습 모델에는 두 종류의 매개변수가 있음
  - 내부 매개변수
    - 신경망의 경우 에지 가중치로서 이 책은 $\boldsymbol{\Theta}$로 표기 (예, 식 (4.1)의 가중치 행렬 $\mathbf{U}^{l}$)
    - 학습 알고리즘이 최적화함
  - 하이퍼 매개변수
    - 모델의 외부에서 모델의 동작을 조정함
    - 예, 은닉층의 개수, CNN 마스크 크기와 보폭, 학습률, 모멘텀과 관련된 매개변수 등

5.5 하이퍼 매개변수 최적화

하이퍼 매개변수 선택

- 표준 문헌이 제시하는 기본값을 사용하면 됨
  - 보통 여러 후보 값 또는 범위를 제시
- 후보 값 중에서 주어진 데이터에 최적인 값 선택 ← 하이퍼 매개변수 최적화

알고리즘 5-9 하이퍼 매개변수 최적화

입력: 훈련집합 $\mathbb{X}$, 검증집합 $\mathbb{X}_{validate}$, 하이퍼 매개변수집합 $\mathcal{H}$
출력: 최적 하이퍼 매개변숫값 $\hat{\mathrm{H}}$

```
1  q_max = 0
2  repeat
3     하이퍼 매개변수 조합 H̃을 생성한다.
4     H̃으로 설정된 모델을 𝕏로 학습한다.
5     학습된 모델을 𝕏_validate로 성능 q를 측정한다.
6     if (q > q_max) q_max = q, Ĥ = H̃
7  until(멈춤 조건)
```

- 라인 3을 구현하는 방법에 따라 수동 탐색, 격자 탐색, 임의 탐색

Cross-validation strategy

coarse -> fine cross-validation in stages

First stage: only a few epochs to get rough idea of what params work
Second stage: longer running time, finer search
… (repeat as necessary)

Tip for detecting explosions in the solver:
If the cost is ever > 3 * original cost, break out early

5.5 하이퍼 매개변수 최적화 - example

Run coarse search

```
max_count = 100
for count in xrange(max_count):
      reg = 10**uniform(-5, 5)
      lr = 10**uniform(-3, -6)

      trainer = ClassifierTrainer()
      model = init_two_layer_model(32*32*3, 50, 10) # input size, hidden size, number of classes
      trainer = ClassifierTrainer()
      best_model_local, stats = trainer.train(X_train, y_train, X_val, y_val,
                                              model, two_layer_net,
                                              num_epochs=5, reg=reg,
                                              update='momentum', learning_rate_decay=0.9,
                                              sample_batches = True, batch_size = 100,
                                              learning rate=lr, verbose=False)
```

note it's best to optimize in log space!

```
val_acc: 0.412000, lr: 1.405206e-04, reg: 4.793564e-01, (1 / 100)
val_acc: 0.214000, lr: 7.231888e-06, reg: 2.321281e-04, (2 / 100)
val_acc: 0.208000, lr: 2.119571e-06, reg: 8.011857e+01, (3 / 100)
val_acc: 0.196000, lr: 1.551131e-05, reg: 4.374936e-05, (4 / 100)
val_acc: 0.079000, lr: 1.753300e-05, reg: 1.200424e+03, (5 / 100)
val_acc: 0.223000, lr: 4.215128e-05, reg: 4.196174e+01, (6 / 100)
val_acc: 0.441000, lr: 1.750259e-04, reg: 2.110807e-04, (7 / 100)
val_acc: 0.241000, lr: 6.749231e-05, reg: 4.226413e+01, (8 / 100)
val_acc: 0.482000, lr: 4.296863e-04, reg: 6.642555e-01, (9 / 100)
val_acc: 0.079000, lr: 5.401602e-06, reg: 1.599828e+04, (10 / 100)
val_acc: 0.154000, lr: 1.618508e-06, reg: 4.925252e-01, (11 / 100)
```

nice

From cs231n Stanford Univ.

5.5 하이퍼 매개변수 최적화 - example

Run finer search

```
max_count = 100
for count in xrange(max_count):
      reg = 10**uniform(-5, 5)
      lr = 10**uniform(-3, -6)
```

adjust range

```
max_count = 100
for count in xrange(max_count):
      reg = 10**uniform(-4, 0)
      lr = 10**uniform(-3, -4)
```

범위 제한
영향이 있는지도 고려

```
val_acc: 0.527000, lr: 5.340517e-04, reg: 4.097824e-01, (0 / 100)
val_acc: 0.492000, lr: 2.279484e-04, reg: 9.991345e-04, (1 / 100)
val_acc: 0.512000, lr: 8.680827e-04, reg: 1.349727e-02, (2 / 100)
val_acc: 0.461000, lr: 1.028377e-04, reg: 1.220193e-02, (3 / 100)
val_acc: 0.460000, lr: 1.113730e-04, reg: 5.244309e-02, (4 / 100)
val_acc: 0.498000, lr: 9.477776e-04, reg: 2.001293e-03, (5 / 100)
val_acc: 0.469000, lr: 1.484369e-04, reg: 4.328313e-01, (6 / 100)
val_acc: 0.522000, lr: 5.586261e-04, reg: 2.312685e-04, (7 / 100)
val_acc: 0.530000, lr: 5.808183e-04, reg: 8.259964e-02, (8 / 100)
val_acc: 0.489000, lr: 1.979168e-04, reg: 1.010889e-04, (9 / 100)
val_acc: 0.490000, lr: 2.036031e-04, reg: 2.406271e-03, (10 / 100)
val_acc: 0.475000, lr: 2.021162e-04, reg: 2.287807e-01, (11 / 100)
val_acc: 0.460000, lr: 1.135527e-04, reg: 3.905040e-02, (12 / 100)
val_acc: 0.515000, lr: 6.947668e-04, reg: 1.562808e-02, (13 / 100)
val_acc: 0.531000, lr: 9.471549e-04, reg: 1.433895e-03, (14 / 100)
val_acc: 0.509000, lr: 3.140888e-04, reg: 2.857518e-01, (15 / 100)
val_acc: 0.514000, lr: 6.438349e-04, reg: 3.033781e-01, (16 / 100)
val_acc: 0.502000, lr: 3.921784e-04, reg: 2.707126e-04, (17 / 100)
val_acc: 0.509000, lr: 9.752279e-04, reg: 2.850865e-03, (18 / 100)
val_acc: 0.500000, lr: 2.412048e-04, reg: 4.997821e-04, (19 / 100)
val_acc: 0.466000, lr: 1.319314e-04, reg: 1.189915e-02, (20 / 100)
val_acc: 0.516000, lr: 8.039527e-04, reg: 1.528291e-02, (21 / 100)
```

53% - relatively good f
or a 2-layer neural net
with 50 hidden neurons.

But this best
cross-validation result is
worrying. Why?

From cs231n Stanford Univ.

5.5 하이퍼 매개변수 최적화

격자 탐색과 임의 탐색

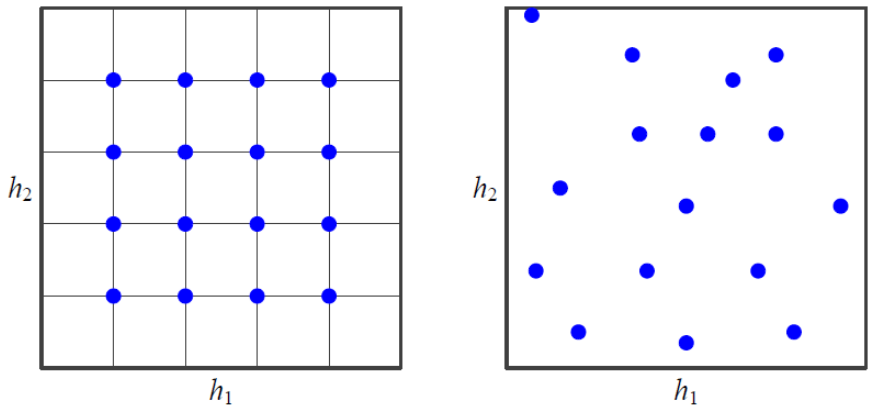


그림 5-29 하이퍼 매개변수 탐색 방법

- 임의 탐색은 난수로 매개변수 조합을 생성함

로그 규모 간격

- 어떤 매개변수는 로그 규모를 사용해야 함
- 예, 학습률 범위가 [0.0001~0.1]일 때
  - 등간격은 0.0001, 0.0002, 0.0003, ..., 0.0998, 0.0999로 총 1000개 값을 조사
  - 로그 규모는 2배씩 증가시킴. 즉 0.0001, 0.0002, 0.0004, 0.0008, ..., 0.0256, 0.0512, ...를 조사함

5.5 하이퍼 매개변수 최적화

- 차원의 저주 문제 발생
  - 매개변수가 m개이고 각각이 q개 구간이라면 q^m개의 점을 조사해야 함
- 임의 탐색이 우월함
  - [Bergstra2012]의 실험 (32개의 매개변수) → 임의 탐색이 유리함
  - [그림 5-30]은 임의 탐색이 유리한 이유를 설명

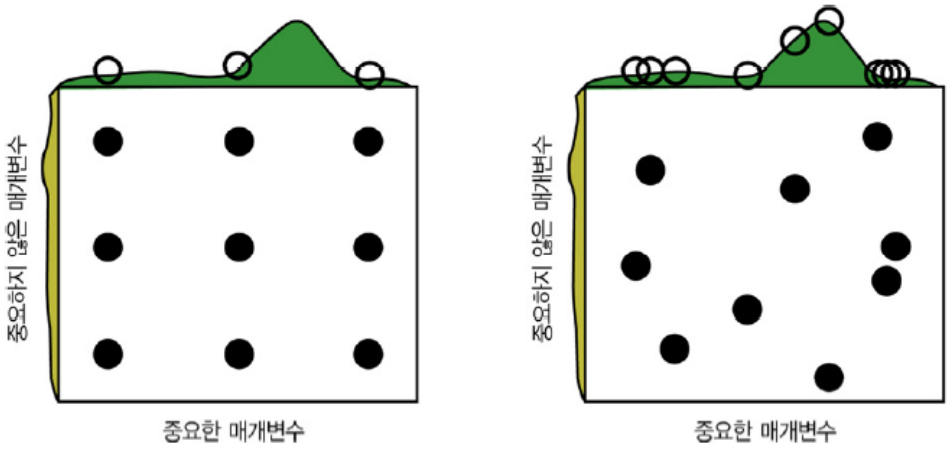
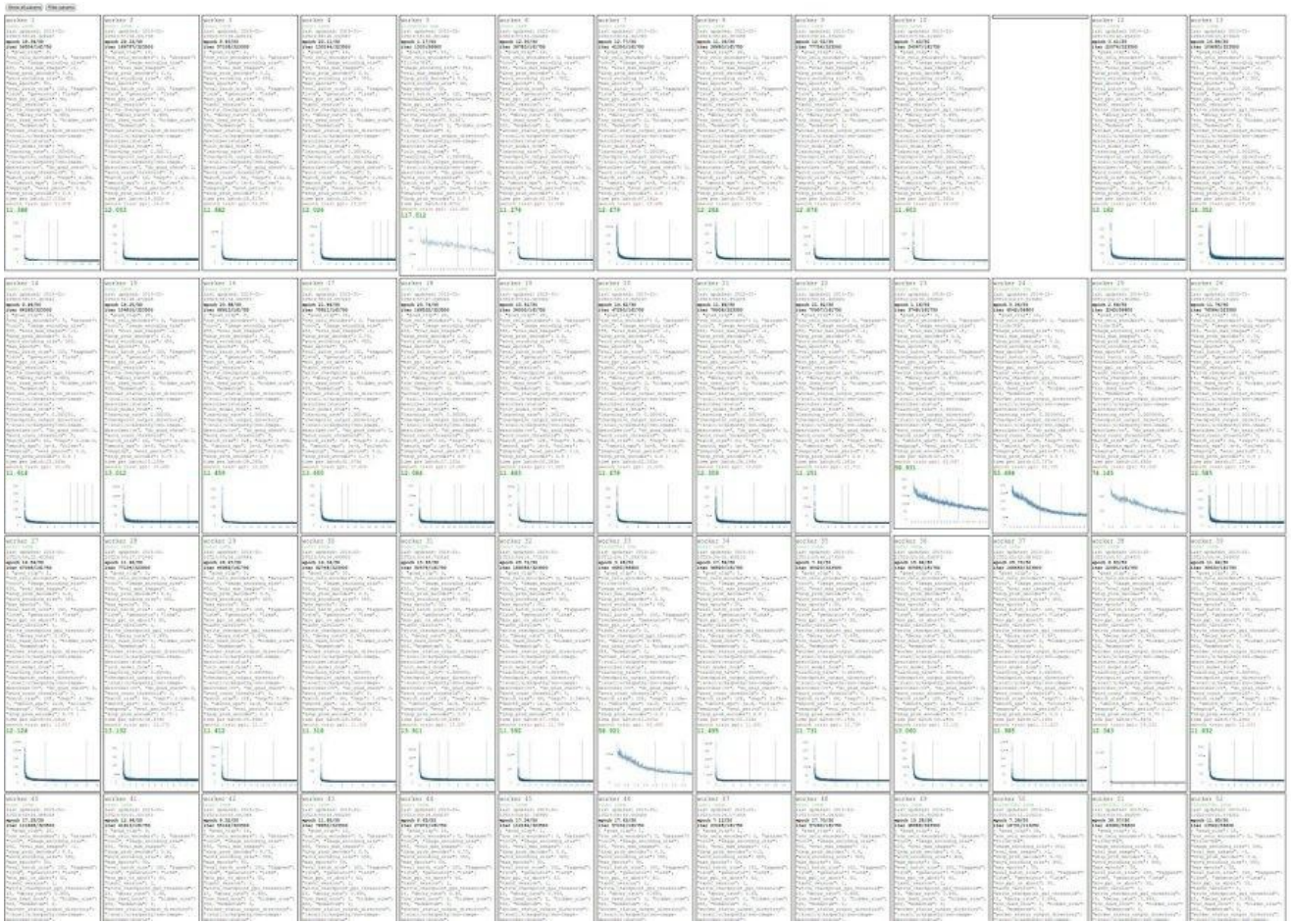


그림 5-30 격자 탐색과 임의 탐색의 성능 비교

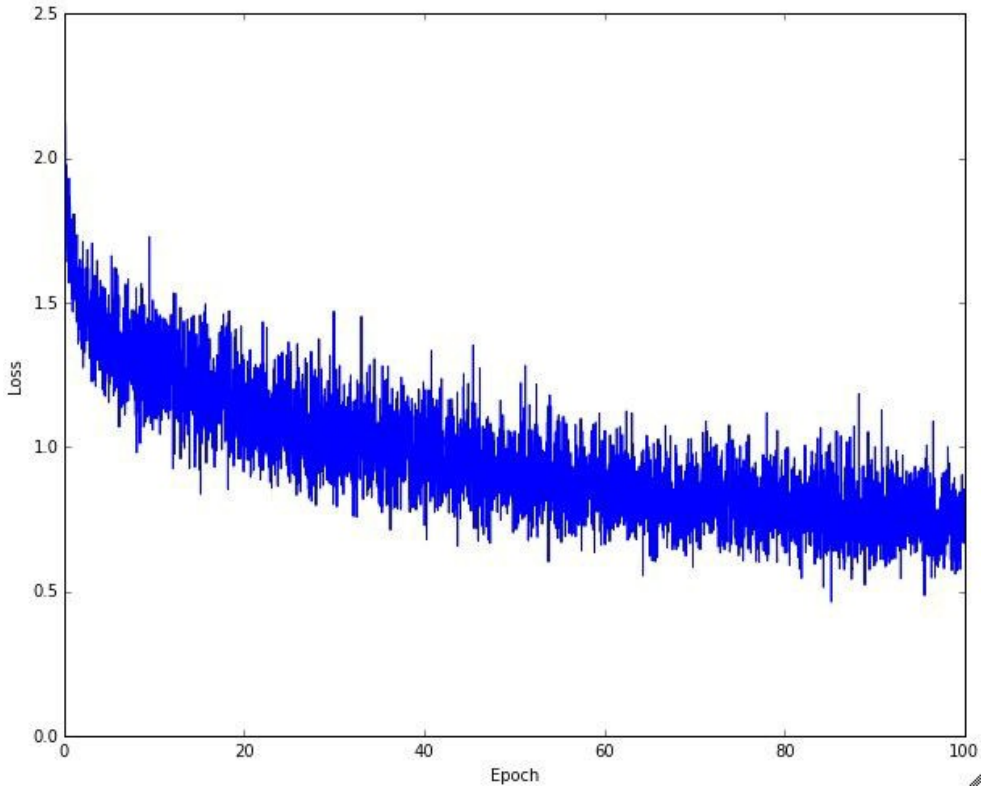
5.5 하이퍼 매개변수 최적화

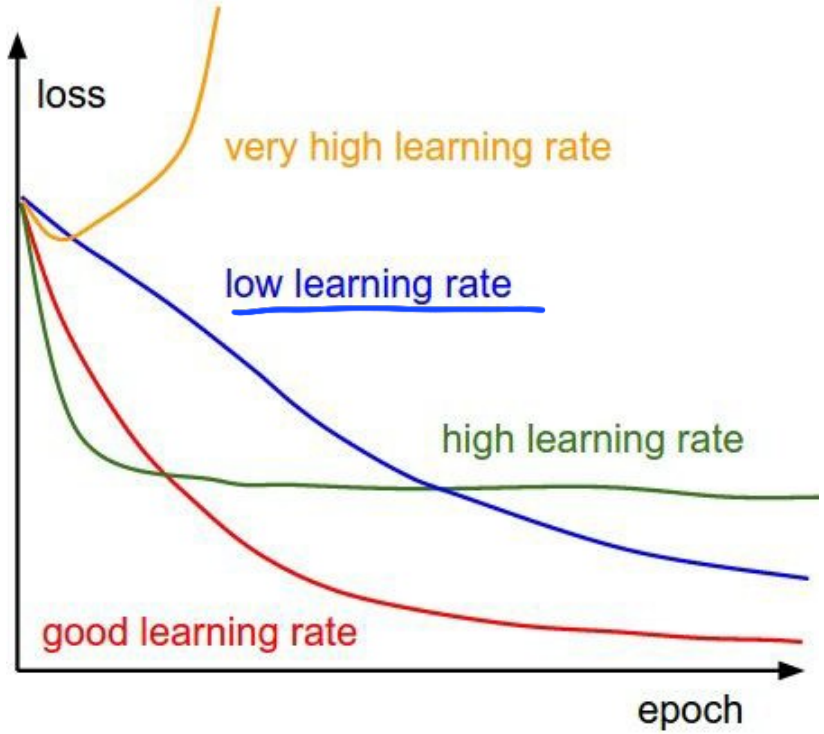


From cs231n Stanford Univ.

5.5 하이퍼 매개변수 최적화

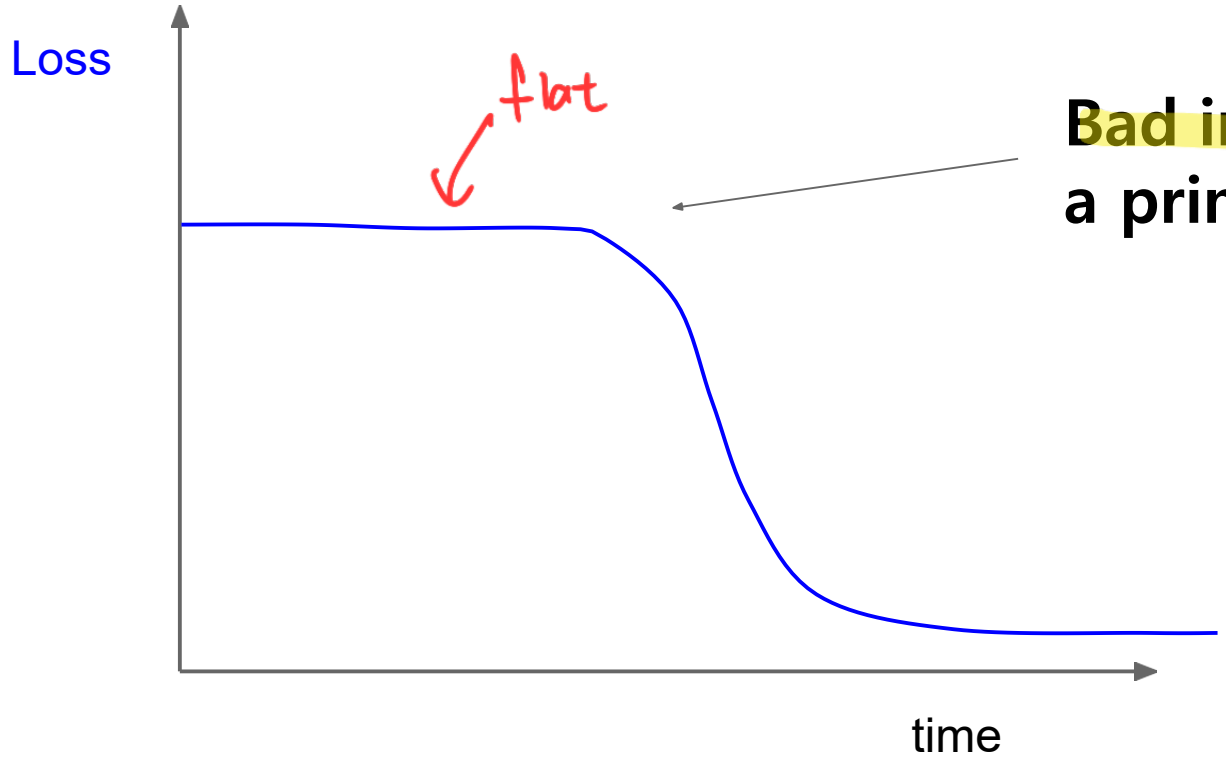
Monitor and visualize the loss curve





From cs231n Stanford Univ.

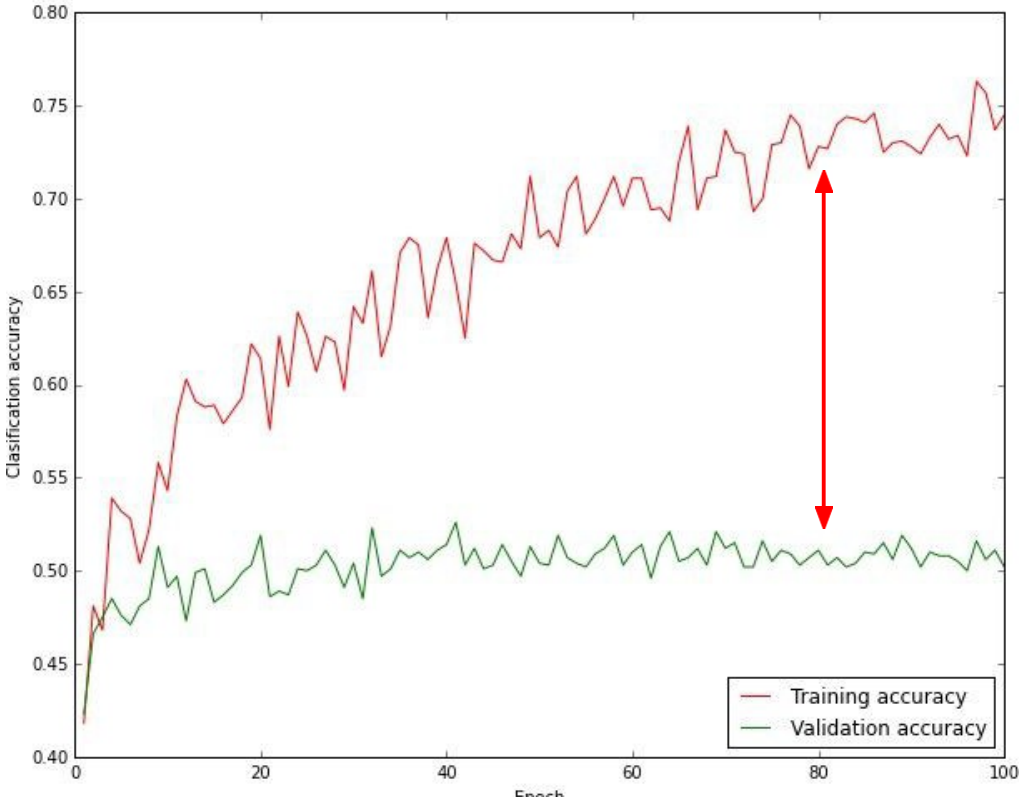
5.5 하이퍼 매개변수 최적화



Bad initialization
a prime suspect

From cs231n Stanford Univ.

5.5 하이퍼 매개변수 최적화



big gap = overfitting

=> increase regularization strength?

no gap

=> increase model capacity?

From cs231n Stanford Univ.

5.6 2차 미분을 이용한 방법

- 5.6.1 뉴턴 방법
- 5.6.2 켤레 그레이디언트 방법
- 5.6.3 유사 뉴턴 방법

- 그레이디언트(1차 미분)를 사용하는 경사 하강법
  - 현재 기계 학습의 주류 알고리즘
  - 두 가지 방향의 개선책 [Bottou2017]
    - 그레이디언트의 잡음을 줄임 (예, 미니배치 사용 등)
    - 2차 미분 정보를 활용 → 5.6절의 주제

5.6 2차 미분을 이용한 최적화

- 경사 하강법을 더 빠르게 할 수 있나
  - [그림 5-31]에서 파란 경로는 경사 하강법이 해를 찾아가는 과정
  - 1차 미분 정보로는 빨간 경로를 알 수 없음
    - 1차 미분은 현재 위치에서 지역적인 기울기 정보만 알려주기 때문

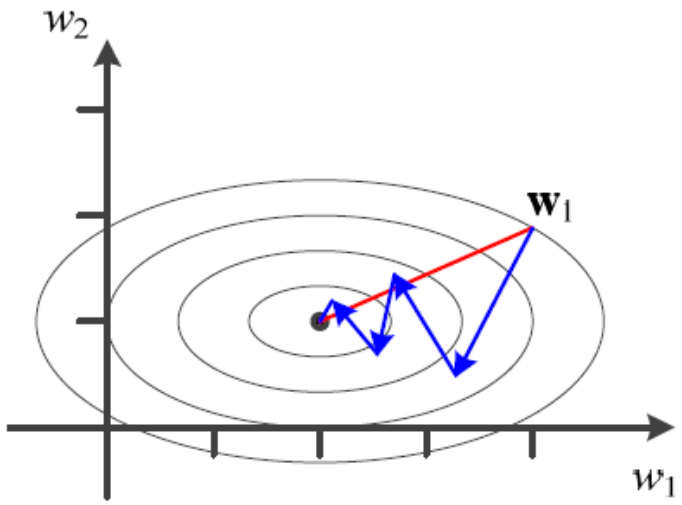


그림 5-31 1차 미분을 사용하는 경사 하강법을 더 빠르게 할 수 있는가

- 뉴턴 방법은 2차 미분 정보를 활용하여 빨간 경로를 알아냄

5.6.1 뉴턴 방법

- 테일러 급수를 적용하면,

$$J(w+\delta) \approx J(w) + J'(w)\delta + \frac{J''(w)}{2}\delta^2 + \cdots$$

  - 식 (5.37)은 변수가 여러 개일 때 (**H**는 헤시언 행렬)

$$J(\mathbf{w}+\boldsymbol{\delta}) \approx J(\mathbf{w}) + \nabla J^{\mathrm{T}}\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^{\mathrm{T}}\mathbf{H}\boldsymbol{\delta}$$

  - δ로 미분하면,

$$\frac{\partial J(w+\delta)}{\partial \delta} \approx J'(w) + \delta J''(w)$$

  - [그림 5-32]처럼 $w+\delta$를 최소점이라 가정하면,

$$\frac{\partial J(w+\delta)}{\partial \delta} \approx J'(w) + \delta J''(w) = 0$$

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix},$$

$$(\mathbf{H}_f)_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

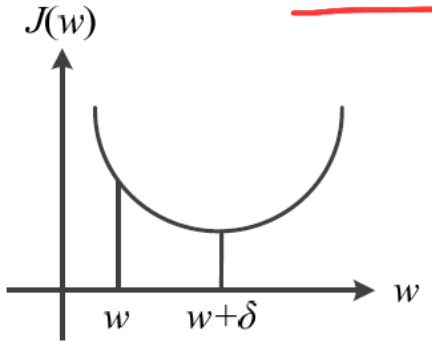


그림 5-32 변수가 하나인 상황의 뉴턴 방법

5.6.1 뉴턴 방법

- 식을 조금 정리하면,

$$\delta = -\frac{J'(w)}{J''(w)} = -\left(J''(w)\right)^{-1} J'(w) \qquad (5.38)$$

- 변수가 여러 개인 경우로 확장하면,

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\nabla J \qquad (5.39)$$

5.6.1 뉴턴 방법

뉴턴 방법의 적용

- [예제 5.3]은 2차 함수에 뉴턴 방법을 적용했으므로, 3차 항 이상을 무시한 식 (5.36)을 사용했음에도 최적 경우 제시
- 기계 학습이 사용하는 목적함수는 2차 함수보다 복잡한 함수이므로 한 번에 최적해에 도달 불가능 → [알고리즘 5-10]과 같이 반복하는 뉴턴 방법을 사용해야 함

알고리즘 5-10 뉴턴 방법

입력 : 훈련집합 $\mathbb{X}$, $\mathbb{Y}$
출력 : 최적해 $\hat{\boldsymbol{\Theta}}$

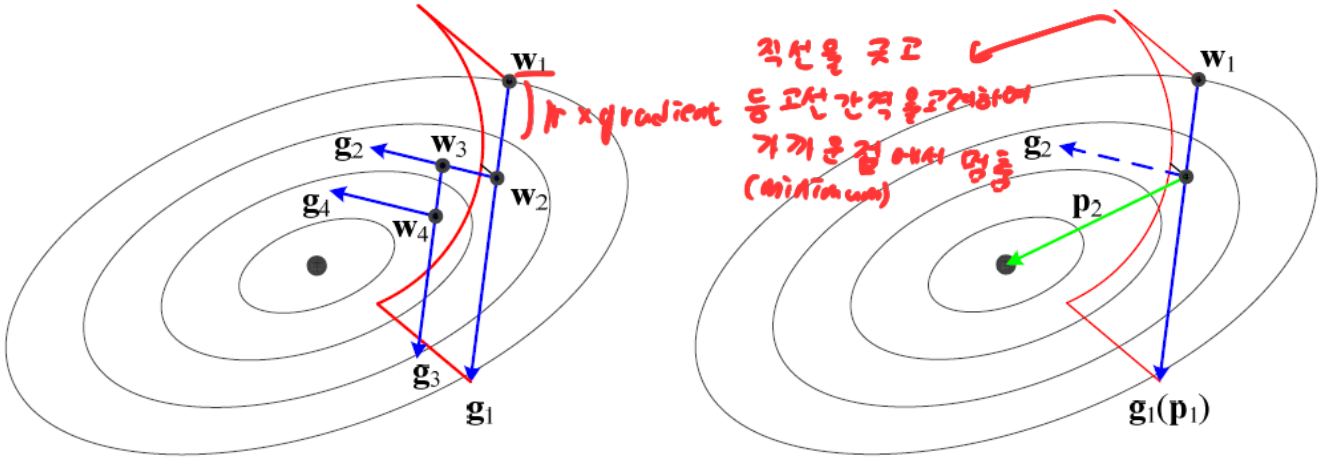
```
1  난수를 생성하여 초기해 Θ를 설정한다.
2  repeat
3     δ = −H^{-1}∇J를 계산한다.   // 식 (5.39)
4     Θ = Θ + δ
5  until (멈춤 조건)
6  Θ̂ = Θ
```

- 라인 3에서 헤시언 $\mathbf{H}$를 구해야 함 → 매개변수의 개수를 m이라 할 때 $O(m^3)$이라는 과다한 계산량 → 켤레 그레이디언트 방법이 대안 제시

5.6.2 켤레 그레이디언트 방법

직선 탐색line search

- [그림 5-33(a)]는 경사 하강법이 학습률을 직선 탐색으로 찾는 상황을 예시



(a) 경사 하강법 (b) 켤레 그레이디언트 방법

그림 5-33 경사 하강법과 켤레 그레이디언트 방법의 비교

- 켤레 그레이디언트 방법
  - [그림 5-33(a)]의 경사 하강법은 근본적으로 이전 경사 하강법과 같음
    - $\mathbf{g}_2$로 직선 탐색할 때 직전에 사용한 $\mathbf{g}_1$ 정보를 전혀 고려하지 않음
  - [그림 5-33(b)]의 켤레 그레이디언트는 직전 정보를 사용하여 해에 빨리 접근

5.6.3 유사 뉴턴 방법

- 유사 뉴턴 방법의 기본 아이디어
  - 헤시언 **H**의 역행렬을 근사하는 행렬 **M**을 사용
  - 처음에 단위 행렬 **I**로 시작하여 그레이디언트 정보를 이용하여 점점 개선
  - LFGS가 많이 사용됨
  - 기계 학습에서는 **M**을 저장하는 메모리를 적게 쓰는 L-BFGS를 주로 사용함
- 기계 학습에서 2차 미분 정보의 활용
  - 현재 널리 활용되지는 않지만 연구 계속되고 있음

"… These methods, such as those built on noise reduction and second-order techniques, offer the ability to attain improved convergence rates, overcome the adverse effects of high nonlinearity and ill-conditioning, and exploit parallelism and distributed architectures in new way. … 잡음 감소와 2차 미분과 같은 방법은 수렴 속도를 향상하고, 심한 비선형과 불량 조건 같은 부정적 효과를 극복하며, 새로운 방식으로 병렬분산 계산 구조를 이용하는 길을 제시한다." [Bottou2017]

7.4 전이학습

- 7.4.1 과업 전이
- 7.4.2 도메인 전이

- 일상 생활에서 전이 학습
  - 피아노를 칠 줄 아는 사람은 못 치는 사람보다 바이올린을 빨리 배움
  - C언어에 익숙한 학생은 파이썬을 금방 배움
  - 두 영역의 공통 지식을 공유하기 때문
- 기계 학습에서 전이 학습
  - 어떤 도메인에서 제작한 프로그램을 데이터가 적어 애를 먹는 새로운 도메인에 적용하여 높은 성능을 얻는 기법
  - 현대 기계 학습에서 널리 활용되고 있음

7.4 전이 학습

과업이 다른 경우와 도메인이 다른 경우로 구분

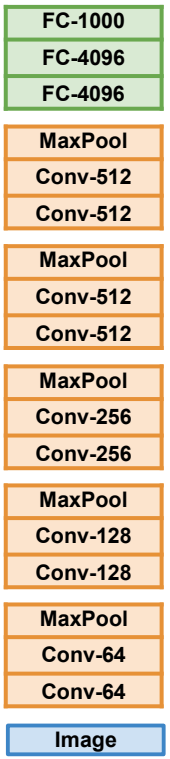
- 과업task이 달라지는 경우: 영상 인식에서 음성 인식으로 전이하는 것처럼 응용분야가 달라질 수 있다. 이는 두 과업 사이의 거리가 아주 먼 상황이며, 아직 이러한 상황에 적용할 수 있는 실용적인 연구 결과는 없다. 반면, 자연영상을 1,000부류로 인식하는 과업을 200종의 나뭇잎을 인식하는 과업으로 전이하는 경우를 생각할 수 있다. 응용분야가 다른 경우보다는 과업 간 거리가 훨씬 짧다. 현재 이런 종류의 전이 학습은 보편적으로 적용할 수 있는 기술이 되었다. 예를 들어, 공개된 VGGNet을 소량의 나뭇잎 영상만 있는 상황에 전이하여 실용적인 나뭇잎 인식 프로그램을 만들어 앱 스토어에 공개한 사례가 여럿 있다.[5]

- 도메인이domain 달라지는 경우: 특징 공간이 다른 경우와 특징 공간은 같은데 데이터의 확률분포가 다른 경우로 구분한다. 전자 사례로는 영불 번역기를 영한 번역기로 전이하거나 한국어 정보 검색기를 베트남어 정보 검색기로 전이하는 상황을 들 수 있다. 두 도메인은 단어 집합이 달라 특징 공간이 다를 수 밖에 없다. 후자 사례는 한국인이 쓴 필기 숫자 데이터베이스로 만든 인식기를 인도에 수출하는 상황에서 발생할 수 있다. 두 도메인은 같은 크기의 숫자 영상을 사용하면 되는데, 필기 습관이 달라 데이터의 확률분포가 다를 것이다. 이때 전이 학습을 사용한다면, 인도인을 대상으로 소량의 데이터만 수집해도 높은 성능을 얻을 수 있다.

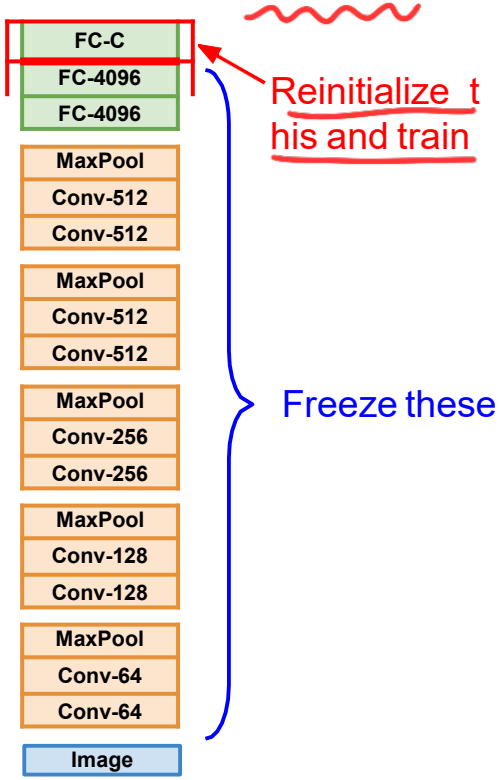
7.4 전이 학습

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014 Raz avian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

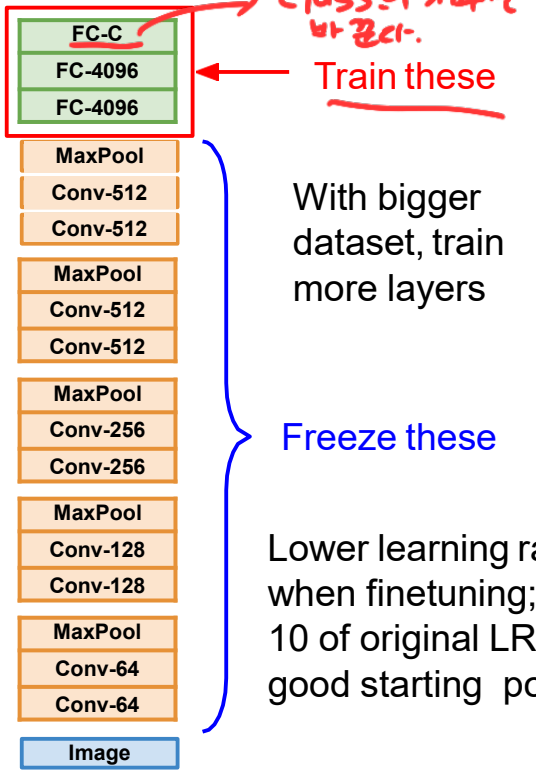
1. Train on Imagenet



2. Small Dataset (C classes)



3. Bigger dataset

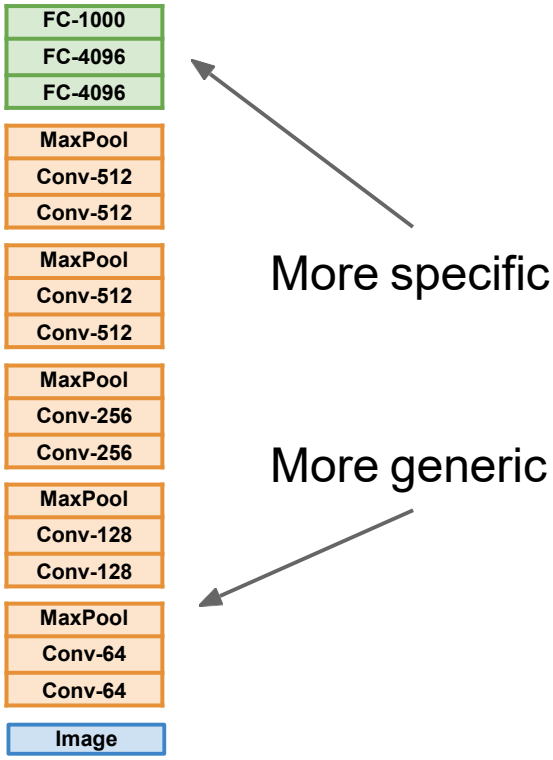


With bigger dataset, train more layers

Lower learning rate when finetuning; 1/10 of original LR is good starting point

From cs231n Stanford Univ.

7.4 전이 학습



	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

From cs231n Stanford Univ.

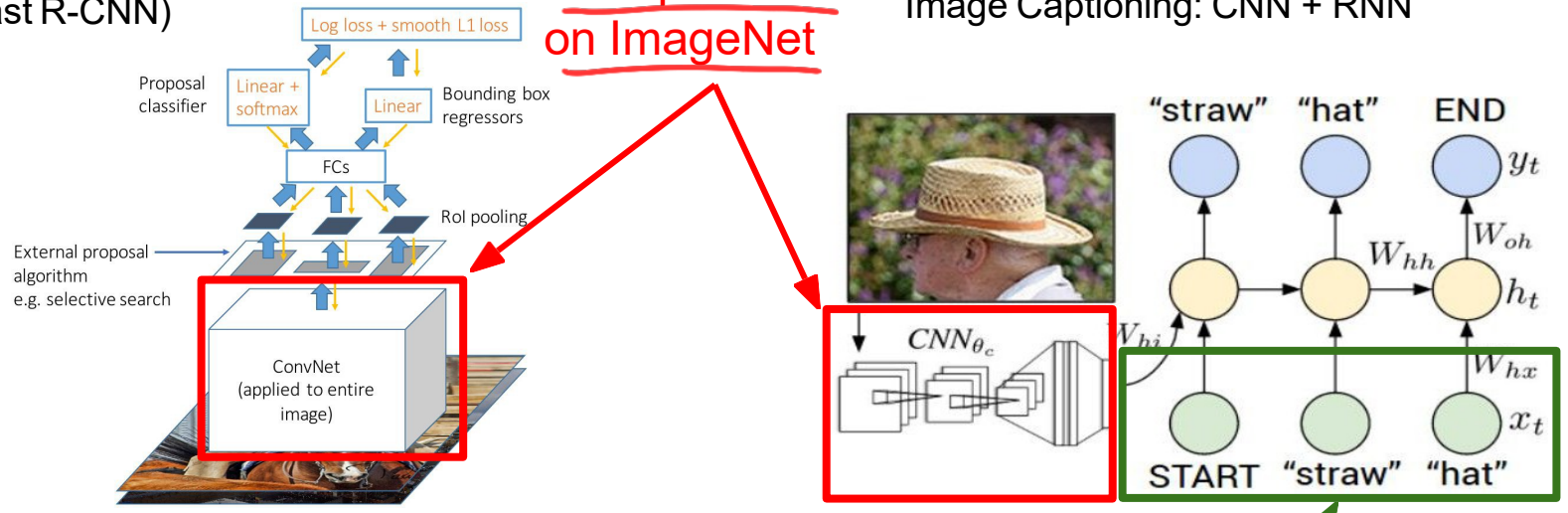
7.4 전이 학습

CNN을 이용하는 전이학습은 몇몇의 예외적인 경우가 아니라
새로운 기준처럼 매우 널리 이용되고있다.

Object Detection
(Fast R-CNN)

CNN pretrained
on ImageNet

Image Captioning: CNN + RNN



Word vectors pretrained
with word2vec

Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

From cs231n Stanford Univ.

5장 연습문제

2 softmax를 적용한 후 출력이 $(0.001, 0.9, 0.001, 0.098)^T$이고 레이블 정보가 $(0,0,0,1)^T$일 때, 세 가지 목적함수, 평균제곱 오차, 교차 엔트로피, 로그우도를 계산하시오.

MSE Cross Entropy Log likelihood

3 [예제 5-1]에서 $\lambda = 0.1$, $\lambda = 0.5$일 때를 계산하고 λ에 따른 효과를 설명하시오. 이때 [그림 5-21]을 활용하시오.

예제 5-1 **리지 회귀** Ridge

훈련집합 $\mathbb{X} = \{\mathbf{x}_1 = \begin{pmatrix}1\\1\end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix}2\\3\end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix}3\\3\end{pmatrix}\}$, $\mathbb{Y} = \{y_1 = 3.0, y_2 = 7.0, y_3 = 8.8\}$이 주어졌다고 가정하자. 특징 벡터가 2차원이므로 d=2이고 샘플이 3개이므로 n=3이다. 훈련집합으로 설계행렬 **X**와 레이블 행렬 **y**를 다음과 같이 쓸 수 있다.

$$\mathbf{X} = \begin{pmatrix}1 & 1\\ 2 & 3\\ 3 & 3\end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix}3.0\\ 7.0\\ 8.8\end{pmatrix}$$

이 값들을 식 (5.29)에 대입하여 다음과 같이 $\hat{\mathbf{w}}$을 구할 수 있다. 이때 $\lambda = 0.25$라 가정하자.

$$\hat{\mathbf{w}} = \left(\begin{pmatrix}1 & 2 & 3\\ 1 & 3 & 3\end{pmatrix}\begin{pmatrix}1 & 1\\ 2 & 3\\ 3 & 3\end{pmatrix} + \begin{pmatrix}0.5 & 0\\ 0 & 0.5\end{pmatrix}\right)^{-1}\begin{pmatrix}1 & 2 & 3\\ 1 & 3 & 3\end{pmatrix}\begin{pmatrix}3.0\\ 7.0\\ 8.8\end{pmatrix} = \begin{pmatrix}1.4916\\ 1.3607\end{pmatrix}$$

따라서 하이퍼 평면은 $y = 1.4916x_1 + 1.3607x_2$이다. 새로운 샘플로 $\mathbf{x} = (5 \quad 4)^T$가 입력되면 식 (5.30)을 이용하여 12.9009를 예측한다.

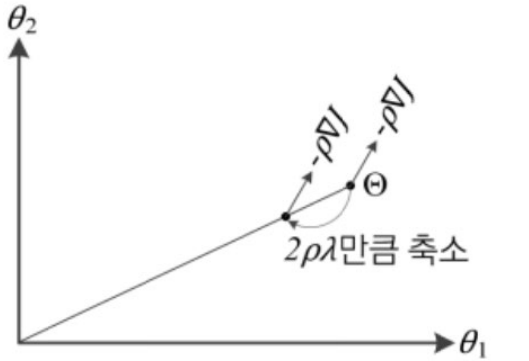


그림 5-21 L2 놈을 사용한 가중치 감쇠 기법의 효과

5장 연습문제

5 혈압, 키, 몸무게가 특징 벡터를 이룬다. 다음과 같이 훈련집합이 주어졌다.

$$\begin{pmatrix}121\\1.72\\69.0\end{pmatrix},\begin{pmatrix}140\\1.62\\63.2\end{pmatrix},\begin{pmatrix}120\\1.70\\59.0\end{pmatrix},\begin{pmatrix}131\\1.80\\82.0\end{pmatrix},\begin{pmatrix}101\\1.78\\73.5\end{pmatrix}$$

$$x_i^{new} = \frac{x_i^{old} - \mu_i}{\sigma_i} \tag{5.9}$$

(1) 퍼셉트론의 가중치 벡터가 $(-0.01, 0.5, -0.23)^T$이고 바이어스가 0이라고 했을 때, 훈련집합을 가지고 규모 문제를 설명하시오.

(2) 식 (5.9)의 전처리를 적용한 후의 훈련집합을 쓰시오.

(3) 전처리가 규모 문제를 완화하는지를 설명하시오.