모두의연구소

AI 최신 기술은 어떻게 찾아 보고 계신가요?

모두의연구소
박은수 Research Director

최신 트렌드 따라가 보는 방법

모두의연구소

- 테리의 딥러닝 토크
  - #0.3. SNS로 딥러닝 소식 팔로우 하는 법 (1/2)
    - https://youtu.be/Z1OdPpq9w0o
  - #0.4. SNS로 딥러닝 소식 팔로우 하는 법 (2/2)
    - https://youtu.be/w1oQQmu8NKo

몇가지 좋은 AI 뉴스레터들을 정리해 놓은 곳

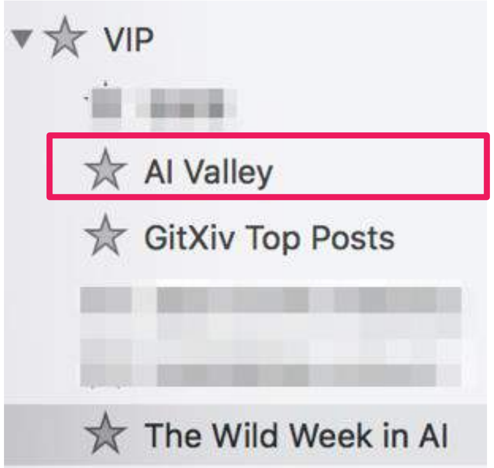
모두의연구소

- Artificial Intelligence Newsletters to Subscribe to
- 5 Must-Read AI Newsletters
- My Curated List of AI and Machine Learning Resources from Around the Web

제가 하는 것들...

모두의연구소

- 메일로 소식 받기 : AI Valley



Top Posts This Week
AI-VALLEY.COM

매주 토요일 새벽 따끈
따끈한 AI소식을 묶어서
보내줍니다

얼마나 따끈따끈한지 살펴보죠

AI That Creates AI

Video - 128 shares

Next-Level Surveillance: China Embraces Facial Recognition

Video - 1224 shares

Will the Future Be Human?

Video - 2559 shares

Building a Deep Neural Net In Google Sheets

Article - 71 shares

Will Artificial Intelligence Replace Doctors?

Video - 30 shares

RedditSota/state-of-the-art-result-for-machine-learning-problems

Repo - 5921 stars

Deep Generative Models

Article - 19 shares

Machine Learning From Scratch - eriklindernoren/ML-From-Scratch

Repo - 7666 stars

See More

AI Valley

제목만 봐도 재밌고
좋은 글들이 많은 것
같습니다

모두의연구소

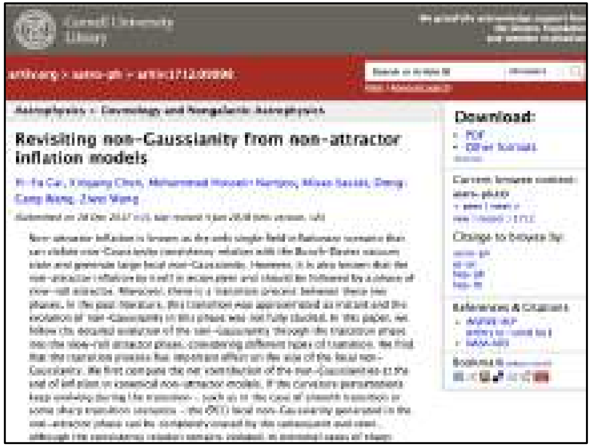
제가 하는 것들...

- 메일로 소식 받기 : GitXiv Top Posts

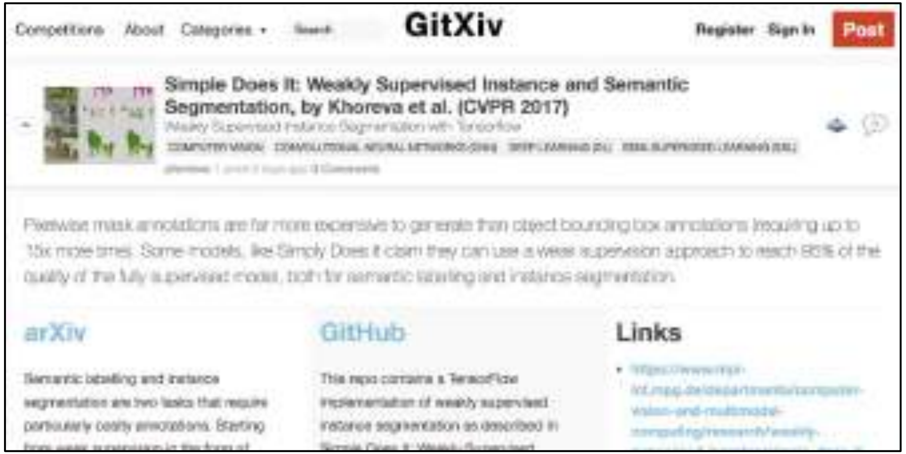
매주 월요일 GitXiv의 소식을 전해 줍니다

제가 하는 것들...

- 메일로 소식 받기 : GitXiv Top Posts



논문을 볼 수 있음



1. 논문을 볼 수 있음
2. 코드를 볼 수 있음
3. 기타 관련 링크를 볼 수 있음

제가 하는 것들...

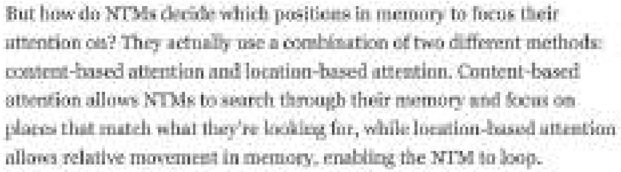
모두의연구소

- 새로운 형태의 논문 : Distill

내용을 이해하기 좋게 상호작용 할 수 있는 요즘 시대에 맞는 형태의 출간 형태로 보시면 됩니다

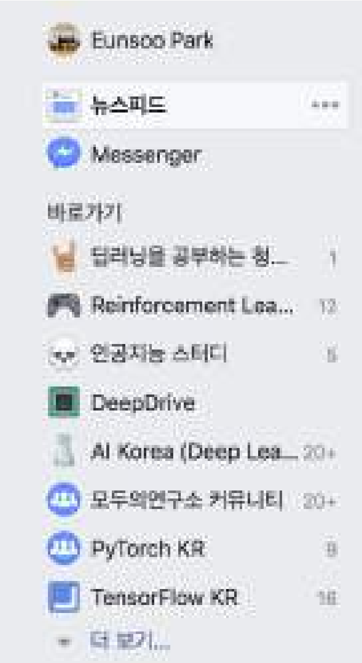
직접 조작가능

Citation


But how do NTMs decide which positions in memory to focus their attention on? They actually use a combination of two different methods: content-based attention and location-based attention. Content-based attention allows NTMs to search through their memory and focus on places that match what they're looking for, while location-based attention allows relative movement in memory, enabling the NTM to loop.

제가 하는 것들...

- SNS로 소식받기 : 페이스북

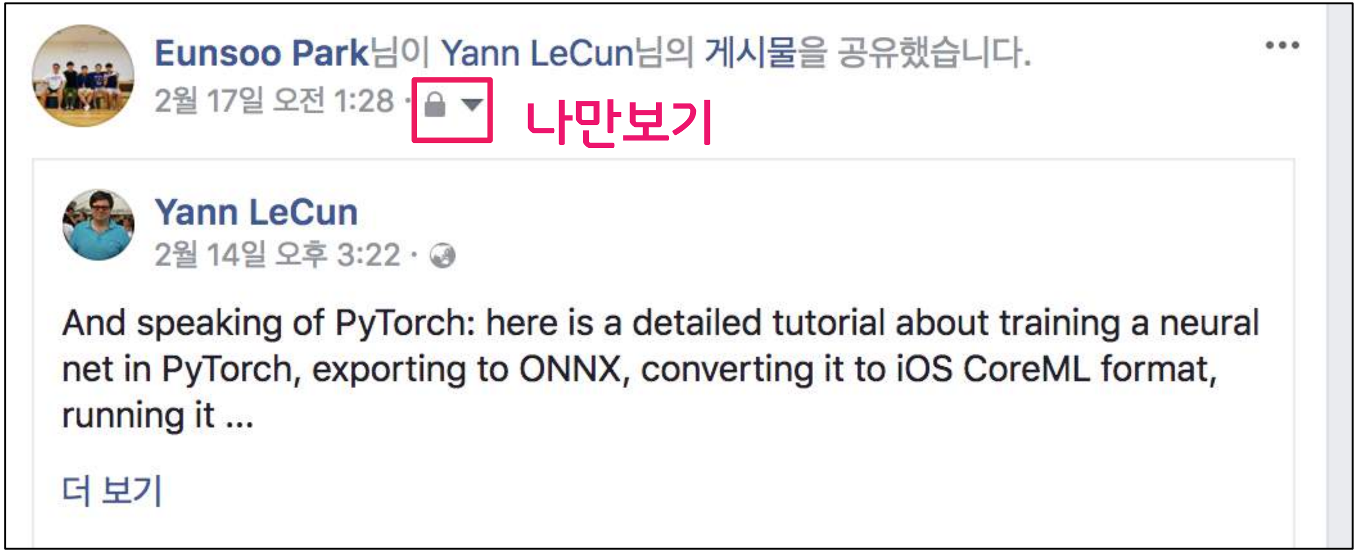


아마 이미 여러분은 저보다 더 많이 가입하셨을것 같습니다

제가 하는 것들...

- **SNS로 소식받기 : 페이스북**

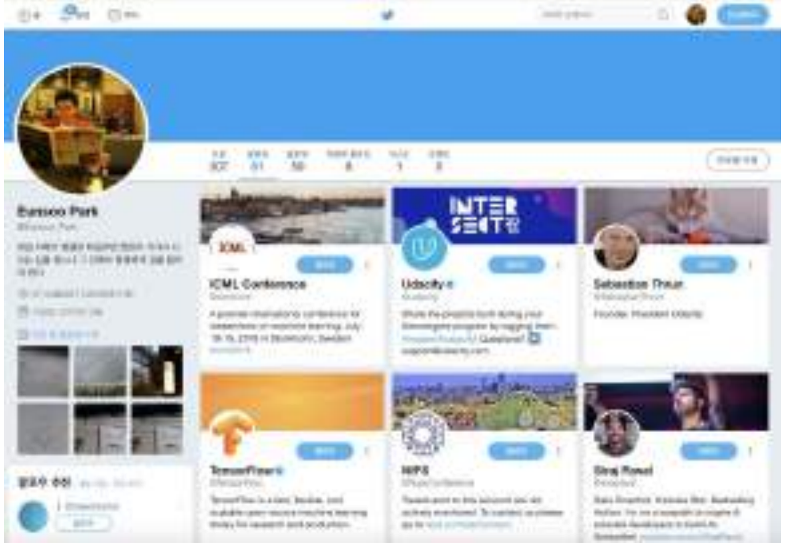
전 소심해서 나만보기로 공유해서 스크랩 해 두고 거의 보지 않는 수백(천?)개의 글이 쌓여있습니다



제가 하는 것들...

모두의연구소

- SNS로 소식받기 : 트위터



제가 하는 것들...

모두의연구소

- SNS로 소식받기 : 트위터

제가 하는 것들...

모두의연구소

- SNS로 소식받기 : 트위터는 메일도 보내줘요

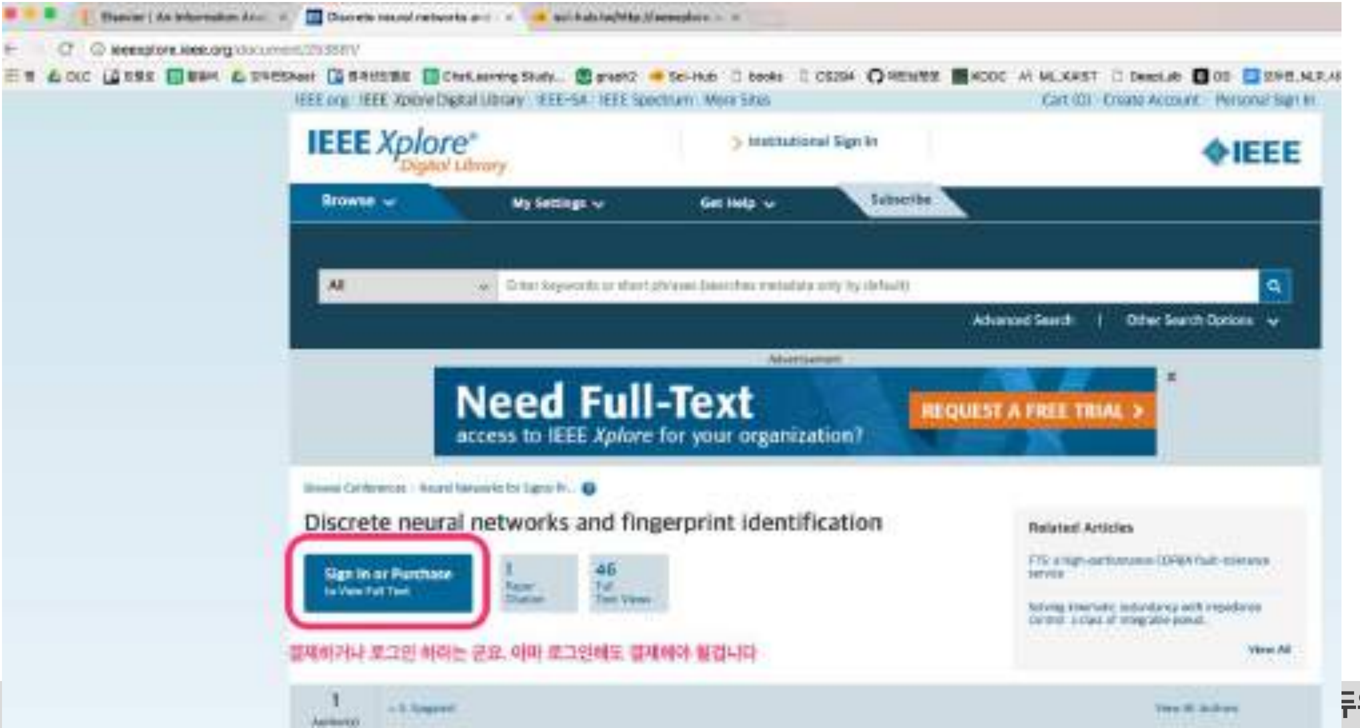
제가 하는 것들...

- Arxiv 논문 필터링 (http://www.arxiv-sanity.com/)

  사용법 : https://youtu.be/S2GY3gh6qC8

제가 하는 것들...

- 좋은 논문이 있는데 결재하라고 해요~



제가 하는 것들...

- 좋은 논문이 있는데 결재하라고 해요~

제가 하는 것들...

- 좋은 논문이 있는데 결재하라고 해요~
  - SCI-Hub : http://sci-hub.tw/

제가 하는 것들...

- 좋은 논문이 있는데 결재하라고 해요~
  - SCI-Hub : http://sci-hub.tw/

성공

제가 하는 것들...

- 추천하는 연구주제 : OpenAI Research 2.0

Requests for Research 2.0

JANUARY 31, 2018

We're releasing a new batch of **seven unsolved problems** which have come up in the course of our research at OpenAI. Like our original Requests for Research (which resulted in several papers), we expect these problems to be a fun and meaningful way for new people to enter the field, as well as for practitioners to hone their skills (it's also a great way to get a job at OpenAI). Many will require inventing new ideas. Please email us with questions or solutions you'd like us to publicize!

(Also, if you don't have deep learning background but want to learn to solve problems like these, please apply for our Fellowship program!)

⭐⭐⭐ **Parameter Averaging in Distributed RL.** Explore the effect of parameter averaging schemes on sample complexity and amount of communication in RL algorithms. While the simplest solution is to average the gradients from every worker on every update, you can save on communication bandwidth by independently updating workers and then infrequently averaging parameters. In RL, this may have another benefit: at any given time we'll have agents with different parameters, which could lead to better exploration behavior. Another possibility is use algorithms like EASGD that bring parameters partly together each update.

⭐⭐⭐ **Transfer Learning Between Different Games via Generative Models.** Proceed as follows:

- Train 11 good policies for 11 Atari games. Generate 10,000 trajectories of 1,000 steps each from the policy for each game.
- Fit a generative model (such as the Transformer) to the trajectories produced by 10 of the games.
- Then fine-tune that model on the 11th game.
- Your goal is to quantify the benefit from pre-training on the 10 games. How large does the model need to be for the pre-training to be useful? How does the size of the effect change when the amount of data from the 11th game is reduced by 10x? By 100x?

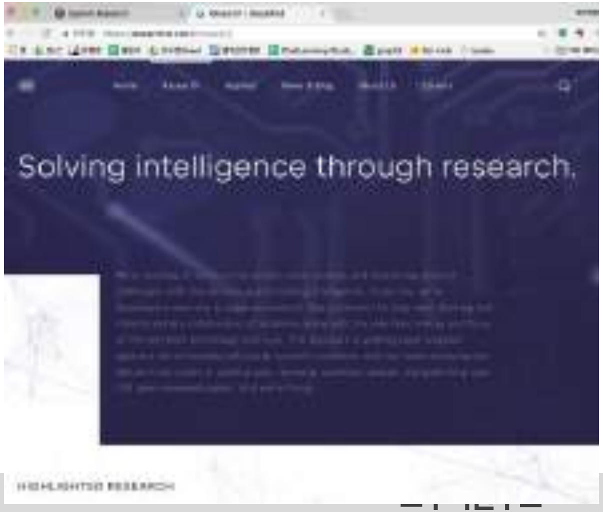
제가 하는 것들...

모두의연구소

- OpenAI research나 Deep Mind 의 research 도 좋습니다

https://openai.com/research/

https://deepmind.com/research/



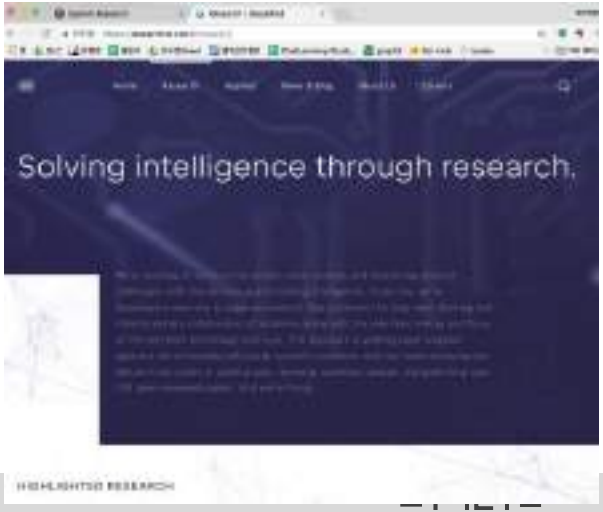
제가 하는 것들...

모두의연구소

- 이들의 연구주제는 대부분 Artificial General Intelligence 입니다 (AGI)

https://openai.com/research/

https://deepmind.com/research/



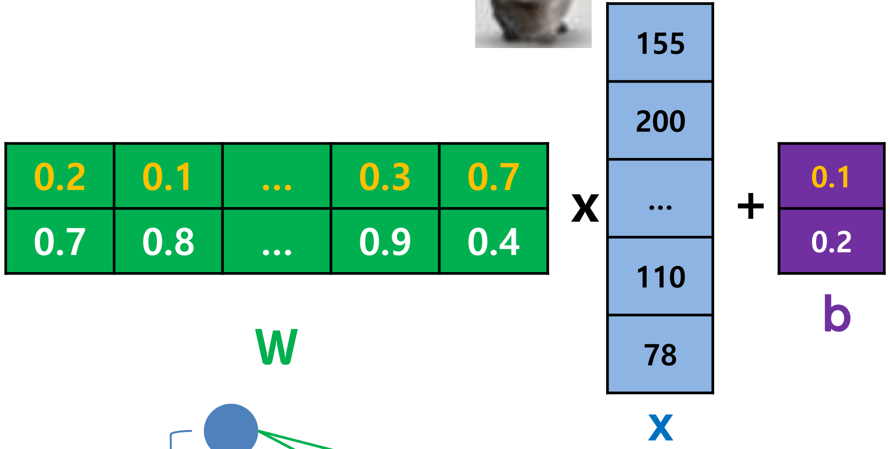
제가 하는 것들... 정리

- 메일로 소식 받기 (예시 2개)
  - AI valley
  - Gitxiv
  - Arxiv-sanity (arxiv 논문 필터링)
- 페이스북 커뮤니티 활동하기
- 트위터로 유명인, 유명회사, 유명 컨퍼런스 팔로윙 하기
- 기타
  - 새로운 형태의 인터렉티브 논문 : Distill
  - 논문결재 돈 요구 : SCI-hub (http://sci-hub.tw/)
  - OpenAI, DeepMind 등의 연구 따라가 보기
  - Research 2.0 주제 확인해 보기

지난시간 Review

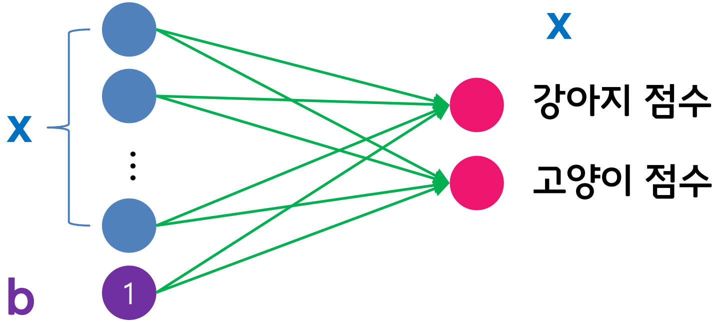
모두의연구소
박은수 Research Director

지난시간 돌아보기 ...



- 분류기의 구성
  - Score function
  - Loss function
  - Optimization

고양이가 입력이면 고양이
점수가 높아야 함



지난시간 돌아보기 ...

- 분류기의 구성
  – Score function
  – Loss function
  – Optimization

$$\underbrace{\begin{bmatrix} 0.2 & 0.1 & \dots & 0.3 & 0.7 \\ 0.7 & 0.8 & \dots & 0.9 & 0.4 \end{bmatrix}}_{\mathbf{W}} \times \underbrace{\begin{bmatrix} 155 \\ 200 \\ \dots \\ 110 \\ 78 \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}}_{\mathbf{b}}$$

Cross-entropy loss (Softmax)

강아지 점수	2.3	exp →	9.97	normalize →	0.97	강아지 확률
고양이 점수	-1.2		0.3		0.03	고양이 확률

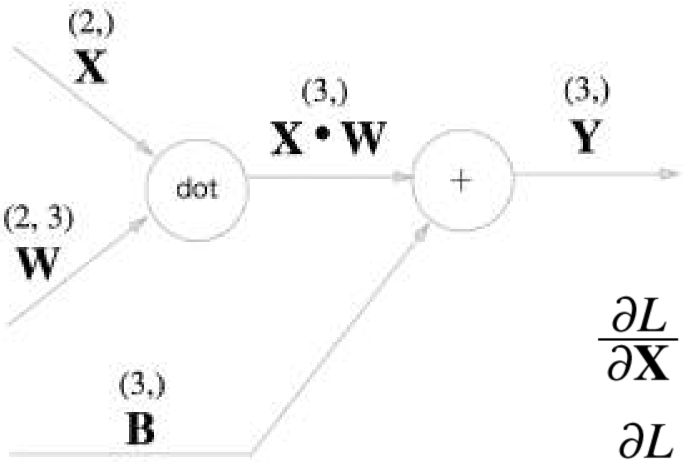
$-log(0.03) = 3.5$

현재의 분류기는 3.5만큼 안 좋음. 이 loss 값을 줄이는게 목표

지난시간 돌아보기 ...

- 분류기의 구성
  - Score function
  - Loss function
  - Optimization

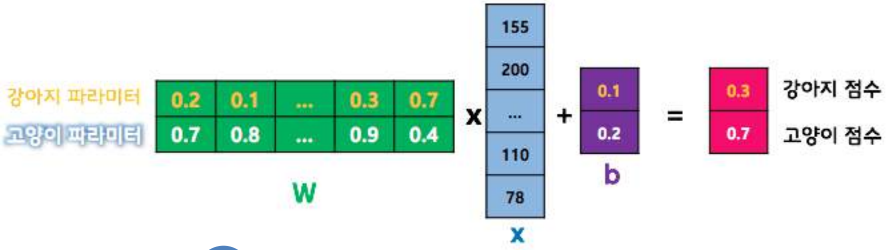
$$w = w - \eta \frac{\partial L}{\partial w}$$

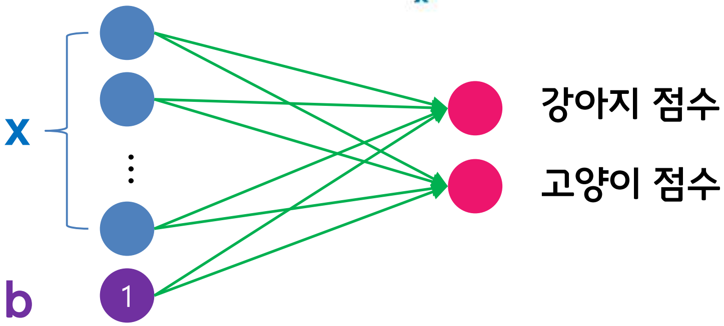


$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

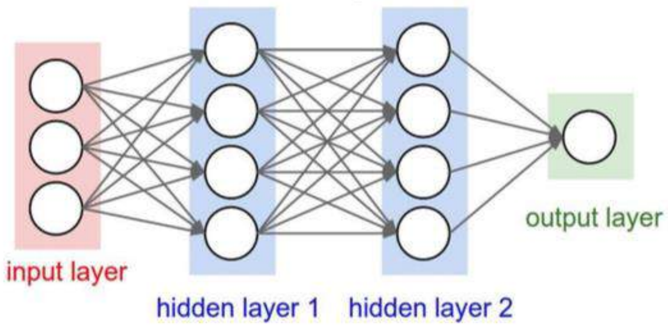
$$\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$

지난 시간 돌아보기 ...





Before



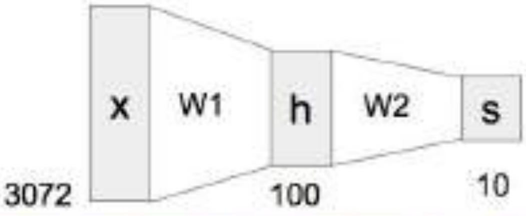
레이어를 쌓아서 더 깊게

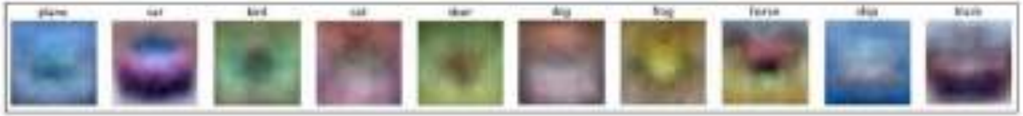
Neural Networks

Now

지난 시간 돌아보기 ...

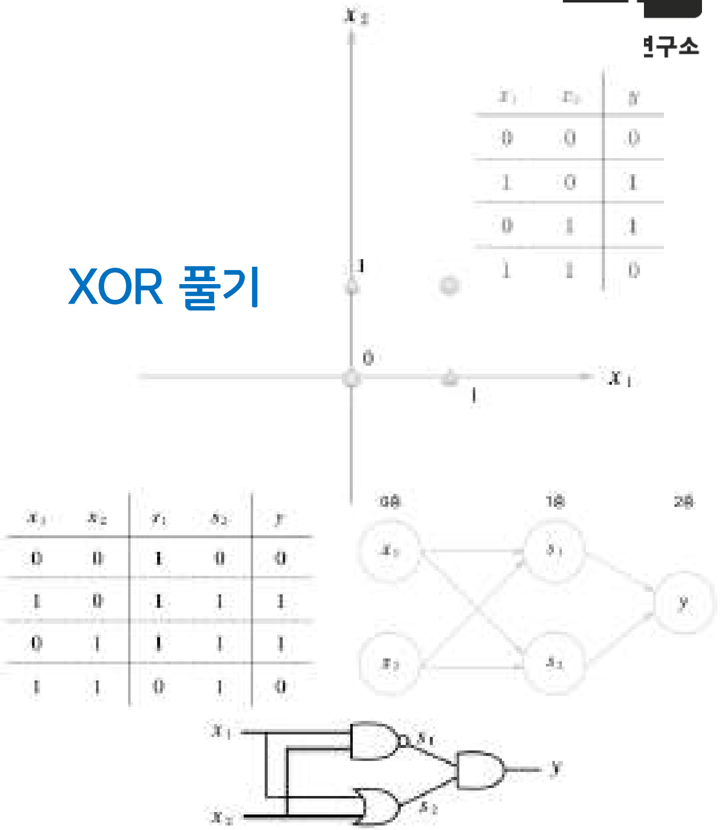
- 레이어를 쌓는 다는 것은 ...





Cifar-10 파라미터 시각화

XOR 풀기

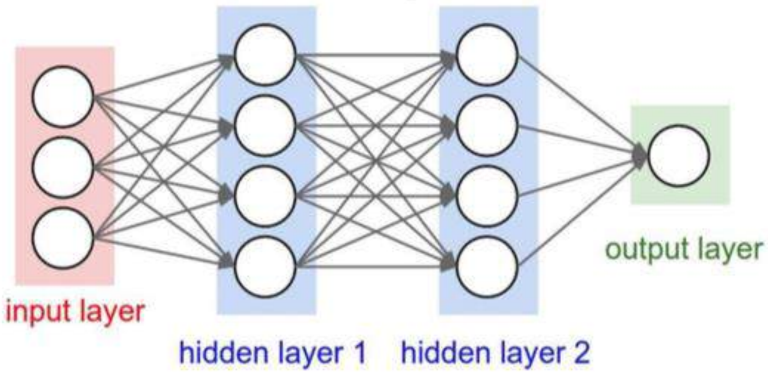


x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

지난 시간 돌아보기 ...

모두의연구소



$$output\ layer = W_2(W_1\mathrm{x}) = W_2W_1\mathrm{x} = \mathrm{Wx}$$

레이어를 하나 쌓는 것 과 같음 → **비선형 Activation function 이 필요**

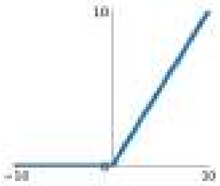
지난 시간 돌아보기 ...

모두의연구소

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$
or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

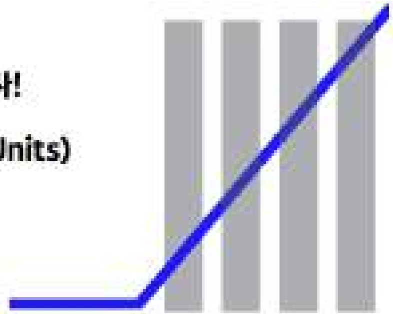


ReLU
(Rectified Linear Unit)

max function 으로 구현

사그라드는 sigmoid대신
죽지않는 activation func을 쓰자!

→ **ReLU** (Rectified Linear Units)



이녀석은 양의 구간에서 전부 미분 값(1)이 있다!

끝 줄 학생까지 이야기가 전달이 잘 되고 위치를 고친다!

지난시간 돌아보기 ...

모두의연구소

- Momentum

Gradient 이동누적 스러운 방법

$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \eta\frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

업데이트 1) $v_1 \leftarrow \alpha * 0 - K_0 : -K_0$
업데이트 2) $v_2 \leftarrow \alpha v_1 - K_1 : -\alpha K_0 - K_1$
업데이트 3) $v_3 \leftarrow \alpha v_2 - K_2 : -\alpha^2 K_0 - \alpha K_1 - K_2$
업데이트 4) $v_4 \leftarrow \alpha v_3 - K_3 : -\alpha^3 K_0 - \alpha^2 K_1 - \alpha K_2 - K_3$

- 1) $\mathbf{W} \leftarrow \mathbf{W} + v_1$
- 2) $\mathbf{W} \leftarrow \mathbf{W} + v_2$
- 3) $\mathbf{W} \leftarrow \mathbf{W} + v_3$
- 4) $\mathbf{W} \leftarrow \mathbf{W} + v_4$

- Adagrad

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{1}{\sqrt{\mathbf{h}}}\frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1) $\frac{1}{\sqrt{\boxed{K_0^2}}}K_o$

업데이트 2) $\frac{1}{\sqrt{\boxed{K_1^2+K_0^2}}}K_1$

업데이트 3) $\frac{1}{\sqrt{\boxed{K_3^2+K_1^2+K_0^2}}}K_3$

업데이트 4) $\frac{1}{\sqrt{\boxed{K_4^2+K_3^2+K_1^2+K_0^2}}}K_4$

Gradient Normalization 스러운 방법

- RMSprop

$$\mathbf{h} \longleftarrow \alpha\mathbf{h} + (1-\alpha)\left(\frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}\right)$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{1}{\sqrt{\mathbf{h}}}\frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1) $\mathbf{h}_1 = (1-a)\mathbf{K}_1^2$
업데이트 2) $\mathbf{h}_2 = a(1-a)\mathbf{K}_1^2 + (1-a)\mathbf{K}_2^2$
업데이트 3) $\mathbf{h}_3 = a^2(1-a)\mathbf{K}_1^2 + a(1-a)\mathbf{K}_2^2 + (1-a)\mathbf{K}_3^2$
업데이트 4) $\mathbf{h}_4 = a^3(1-a)\mathbf{K}_1^2 + a^2(1-a)\mathbf{K}_2^2 + a(1-a)\mathbf{K}_3^2 + (1-a)\mathbf{K}_4^2$

지난시간 돌아보기 ...

모두의연구소

- Momentum

$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \eta\frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

업데이트 1) $v_1 \leftarrow \alpha * 0 - K_0 : -K_0$
업데이트 2) $v_2 \leftarrow \alpha v_1 - K_1 : -\alpha K_0 - K_1$
업데이트 3) $v_3 \leftarrow \alpha v_2 - K_2 : -\alpha^2 K_0 - \alpha K_1 - K_2$
업데이트 4) $v_4 \leftarrow \alpha v_3 - K_3 : -\alpha^3 K_0 - \alpha^2 K_1 - \alpha K_2 - K_3$

- 1) $\mathbf{W} \leftarrow \mathbf{W} + v_1$
- 2) $\mathbf{W} \leftarrow \mathbf{W} + v_2$
- 3) $\mathbf{W} \leftarrow \mathbf{W} + v_3$
- 4) $\mathbf{W} \leftarrow \mathbf{W} + v_4$

- Adagrad

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{1}{\sqrt{\mathbf{h}}}\frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1) $\frac{1}{\sqrt{\boxed{K_0^2}}}K_0$
업데이트 2) $\frac{1}{\sqrt{\boxed{K_1^2+K_0^2}}}K_1$
업데이트 3) $\frac{1}{\sqrt{\boxed{K_3^2+K_1^2+K_0^2}}}K_3$
업데이트 4) $\frac{1}{\sqrt{\boxed{K_4^2+K_3^2+K_1^2+K_0^2}}}K_4$

두 방법의 같이쓰자
Adam

- RMSprop

$$\mathbf{h} \longleftarrow \alpha\mathbf{h} + (1-\alpha)(\frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}})$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{1}{\sqrt{\mathbf{h}}}\frac{\partial L}{\partial \mathbf{W}}$$

업데이트 1) $\mathbf{h}_1 = (1-a)\mathbf{K}_1^2$
업데이트 2) $\mathbf{h}_2 = a(1-a)\mathbf{K}_1^2 + (1-a)\mathbf{K}_2^2$
업데이트 3) $\mathbf{h}_3 = a^2(1-a)\mathbf{K}_1^2 + a(1-a)\mathbf{K}_2^2 + (1-a)\mathbf{K}_3^2$
업데이트 4) $\mathbf{h}_4 = a^3(1-a)\mathbf{K}_1^2 + a^2(1-a)\mathbf{K}_2^2 + a(1-a)\mathbf{K}_3^2 + (1-a)\mathbf{K}_4^2$

지난시간 돌아보기 ...

모두의연구소

- Adam
  - RMSProp + 모멘텀

Adam update

[Kingma and Ba, 2014]

(incomplete, but close)

```
# Adam
m = beta1*m + (1-beta1)*dx # update first moment
v = beta2*v + (1-beta2)*(dx**2) # update second moment
x += - learning_rate * m / (np.sqrt(v) + 1e-7)
```

momentum

RMSProp-like

m 은 분자에 곱, v는 분모에 곱

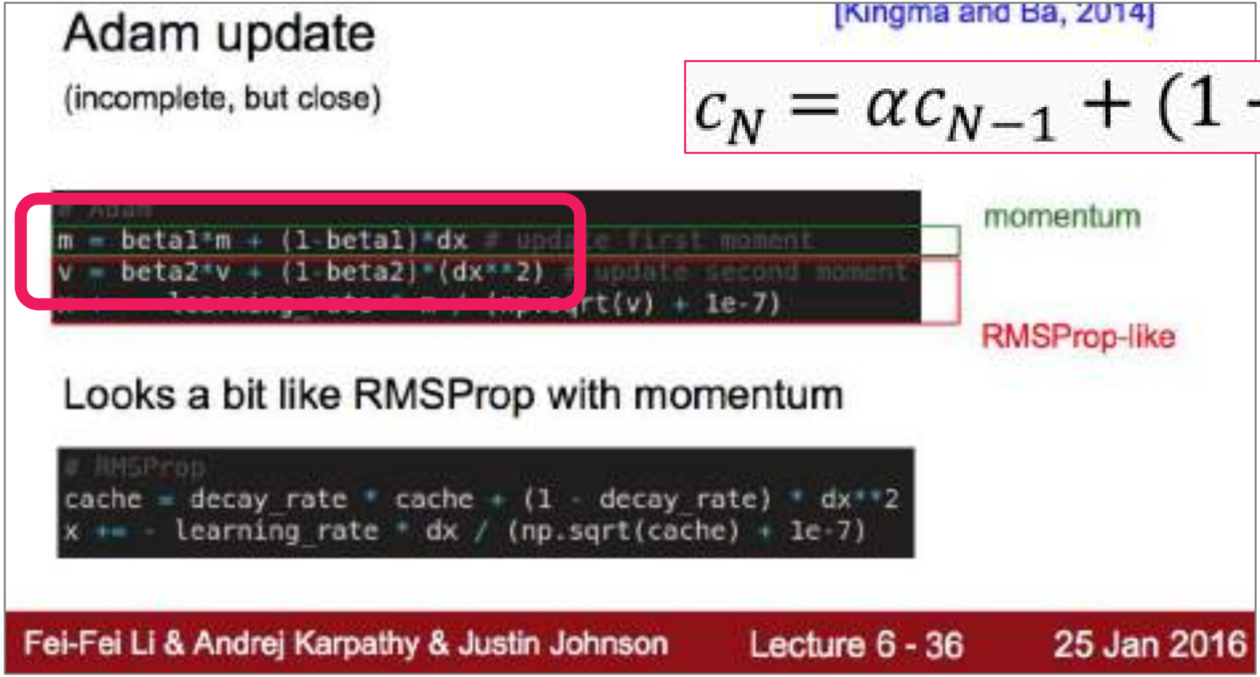
Looks a bit like RMSProp with momentum

```
# RMSProp
cache = decay_rate * cache + (1 - decay_rate) * dx**2
x += - learning_rate * dx / (np.sqrt(cache) + 1e-7)
```

Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 6 - 36 25 Jan 2016

지난시간 돌아보기 ...

- Adam
  - RMSProp + 모멘텀



$$c_N = \alpha c_{N-1} + (1-\alpha)x_N$$

많이 등장하는 패턴이군요

누적 패턴 : 이전의 값은 감소 시키고..

평균을 구해보자

N개의 sample에 대한
평균 c_N 을 구해보자.

$$c_N = \frac{1}{N}(x_1 + x_2 + x_3 + \cdots + x_N)$$
$$= \frac{1}{N}\sum_{i=1}^{N} x_i$$

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$
$$= \frac{1}{N}\left(\sum_{i=1}^{N-1} x_i + x_N\right)$$
$$= \frac{N-1}{N}\underbrace{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i}_{c_{N-1}} + \frac{1}{N}x_N$$
$$= \alpha c_{N-1} + (1-\alpha)x_N \quad 0 < \alpha < 1$$

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N-1} x_i + x_N\right)$$

$$c_{N-1}$$

$$= \frac{N-1}{N}\boxed{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i} + \frac{1}{N}x_N$$

$$= \alpha \boxed{c_{N-1}} + (1-\alpha)x_N \quad 0 < \alpha < 1$$

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\underbrace{\sum_{i=1}^{N-1} x_i}_{} + x_N\right)$$

$$= \boxed{\frac{N-1}{N}}\boxed{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i} + \frac{1}{N}x_N \qquad c_{N-1}$$

$$= \boxed{\alpha}\boxed{c_{N-1}} + (1-\alpha)x_N \qquad 0 < \alpha < 1$$

$$\alpha = \frac{N-1}{N}$$

$$1 - \alpha = 1 - \frac{N-1}{N}$$

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N-1} x_i + x_N\right)$$

c_{N-1}

$$= \boxed{\frac{N-1}{N}}\boxed{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i} + \boxed{\frac{1}{N}}x_N$$

$$= \boxed{\alpha}\boxed{c_{N-1}} + \boxed{(1-\alpha)}x_N \quad 0 < \alpha < 1$$

$$\alpha = \frac{N-1}{N}$$

$0 < a < 1$

$$1 - \alpha = 1 - \frac{N-1}{N}$$

$$1 - \alpha = \frac{N}{N} - \frac{N-1}{N} = \frac{1}{\underline{N}}$$

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N-1} x_i + x_N\right)$$

$$= \frac{N-1}{N}\overbrace{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i}^{c_{N-1}} + \frac{1}{N}x_N$$

$$= \alpha c_{N-1} + (1-\alpha)x_N \quad 0 < \alpha < 1$$

$$\alpha = \frac{N-1}{N} = 1 - \frac{1}{N}$$

α 가 크다는 것은?

N이 ?

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N-1} x_i + x_N\right)$$

$$= \frac{N-1}{N}\underbrace{\frac{1}{N-1}\sum_{i=1}^{N-1} x_i}_{c_{N-1}} + \frac{1}{N}x_N$$

$$= \alpha c_{N-1} + (1-\alpha)x_N \quad 0 < \alpha < 1$$

$$\alpha = \frac{N-1}{N} = 1 - \frac{1}{N}$$

α 가 크다는 것은?

N이 ? 이전 결과의 반영 비율 크

평균을 구하는 또 다른 방법

$$c_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N-1} x_i + x_N\right)$$

c_{N-1}

$$= \frac{N-1}{N}\frac{1}{N-1}\sum_{i=1}^{N-1} x_i + \frac{1}{N}x_N$$

$$= \alpha c_{N-1} + (1-\alpha)x_N \quad 0 < \alpha < 1$$

$$\alpha = \frac{N-1}{N} = 1 - \frac{1}{N}$$

α 가 크다는 것은?

N이 ? 커야 겠군요

1. 더 많은 이동평균을 고려한 관점이라고 해석할 수 있을 것 같아요 평균의 관점에서
2. 이전의 평균값에 더 많은 가중치를 준 합이라고 볼 수 있네요 (수식 그대로 해석)

지난시간 돌아보기 ...

모두의연구소

Batch Normalization

[Ioffe and Szegedy, 2015]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Note: at test time BatchNorm layer functions differently:

The mean/std are not computed based on the batch. Instead, a single fixed empirical mean of activations during training is used.

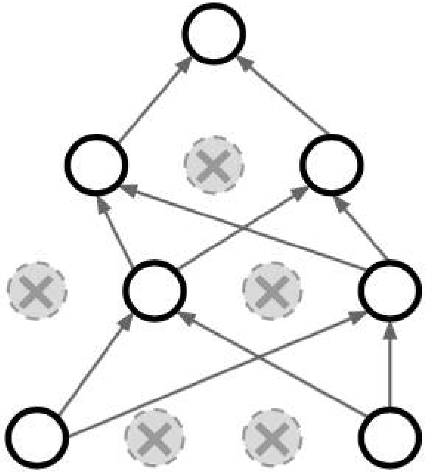
(e.g. can be estimated during training with running averages)

지난시간 돌아보기 ...

모두의연구소

Regularization: Dropout

How can this possibly be a good idea?



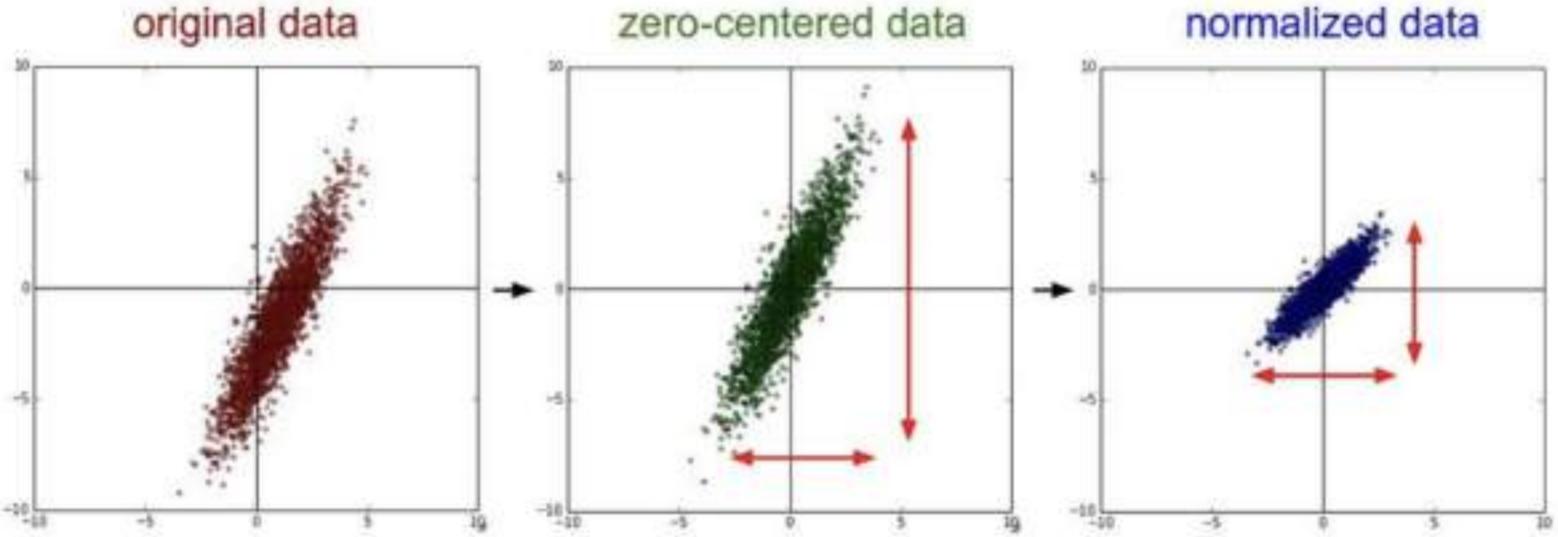
Another interpretation:

Dropout is training a large **ensemble** of models (that share parameters).

Each binary mask is one model

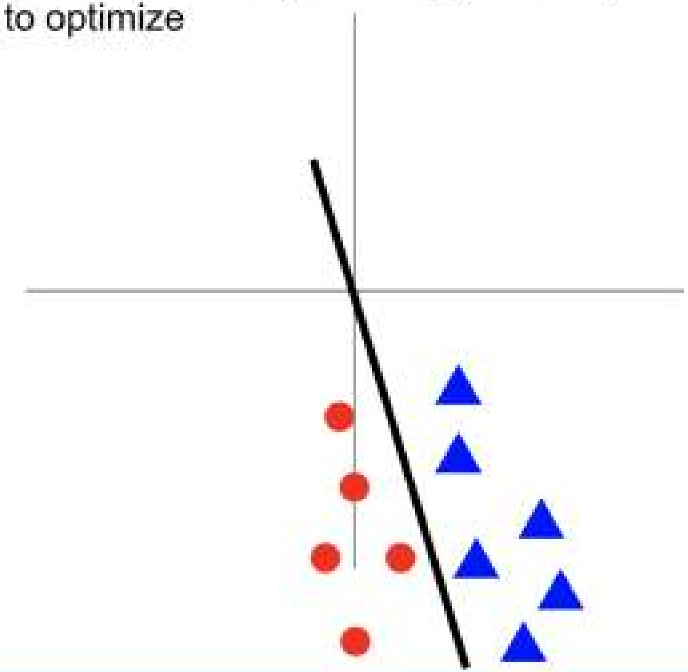
An FC layer with 4096 units has $2^{4096} \sim 10^{1233}$ possible masks!
Only ~ 10^{82} atoms in the universe...

Last time: Data Preprocessing

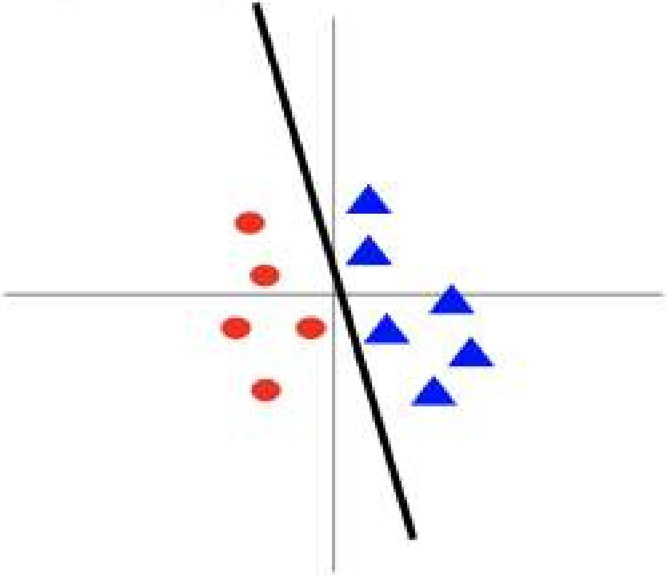


Last time: Data Preprocessing

Before normalization: classification loss very sensitive to changes in weight matrix; hard to optimize



After normalization: less sensitive to small changes in weights; easier to optimize



지난시간 돌아보기 ...

Summary

TLDRs

We looked in detail at:

- Activation Functions (use ReLU)
- Data Preprocessing (images: subtract mean)
- Weight Initialization (use Xavier init)
- Batch Normalization (use)
- Babysitting the Learning process
- Hyperparameter Optimization
  (random sample hyperparams, in log space when appropriate)

Training Neural Networks – 2

모두의연구소
박은수 Research Director

오늘의 계획

모두의연구소

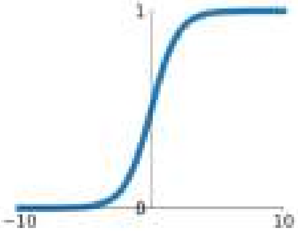
- 좀 더 자세히 (from : cs231n)
  - Activation Functions
  - Fancier Optimization
  - Regularization : Data augmentation

- Transfer Learning

Activation Functions

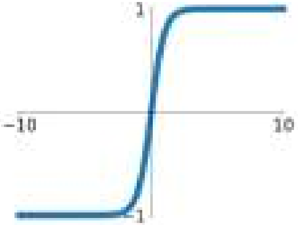
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



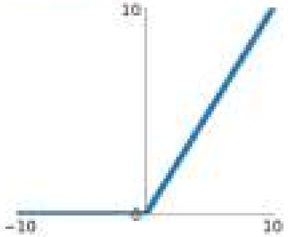
tanh

$\tanh(x)$



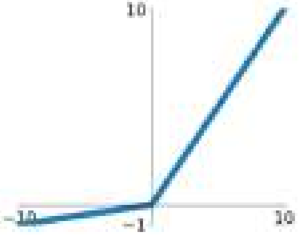
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

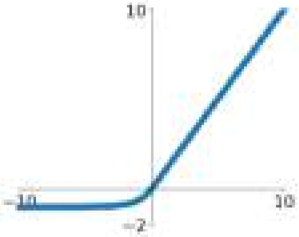


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

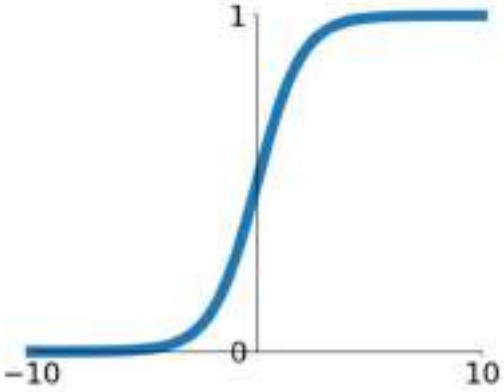
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation Functions

$$\sigma(x) = 1/(1 + e^{-x})$$



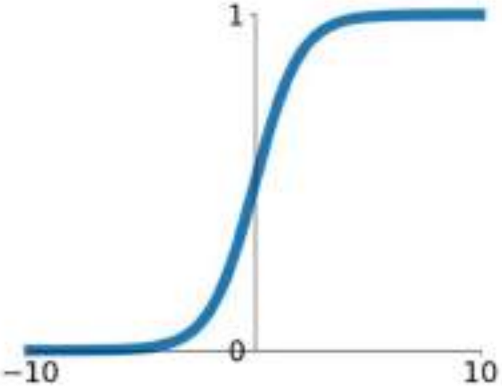
Sigmoid

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

neruon의 firing rate를 잘 모방 하고 있다.. 라고 생각해서많이 사용 했었다.

Activation Functions

$$\sigma(x) = 1/(1 + e^{-x})$$

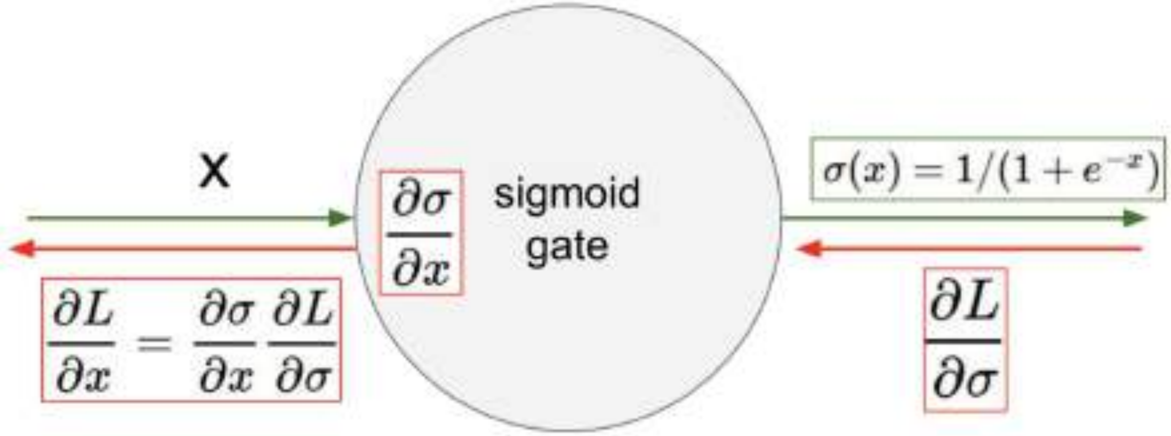


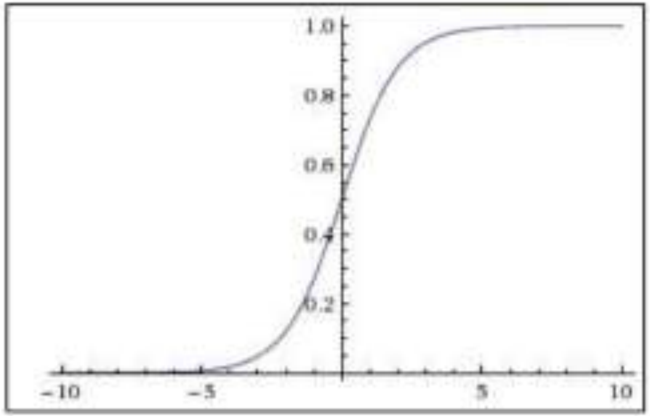
Sigmoid

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

3 problems:

1. Saturated neurons "kill" the gradients





Backward 에서 문제가 많다!

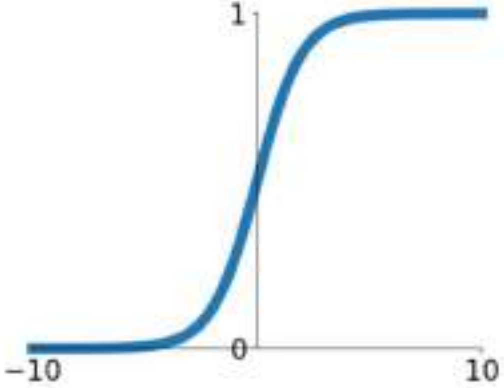
What happens when x = -10? = 0

What happens when x = 0? = y(1-y) = 1/4

What happens when x = 10? = 0

Activation Functions

$$\sigma(x) = 1/(1 + e^{-x})$$



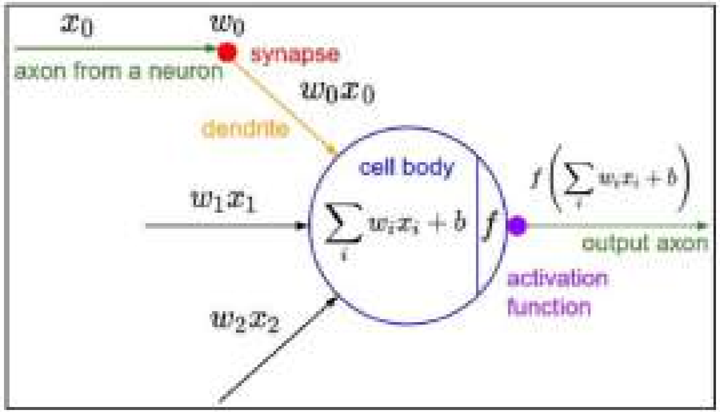
Sigmoid

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

3 problems:

1. Saturated neurons "kill" the gradients
2. Sigmoid outputs are not zero-centered

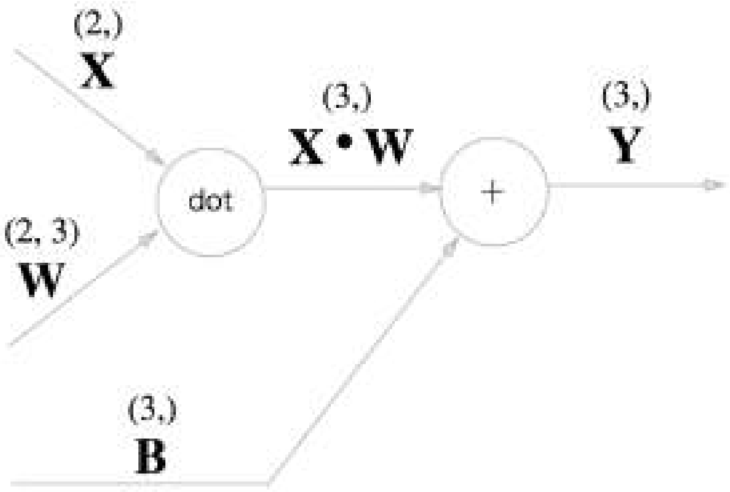
Consider what happens when the input to a neuron (x) is always positive:



$$f\left(\sum_i w_i x_i + b\right)$$

What can we say about the gradients on **w**?

역전파



$$\frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{W}^{\mathrm{T}}$$

$$\boxed{\frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}}$$

복습 : Computational Graph

모두의연구소

- **내적연산**

$$\mathrm{x}^{T}\mathbf{w} = \mathbf{y}$$

$$[x_1 \;\; x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

Gradient **w**

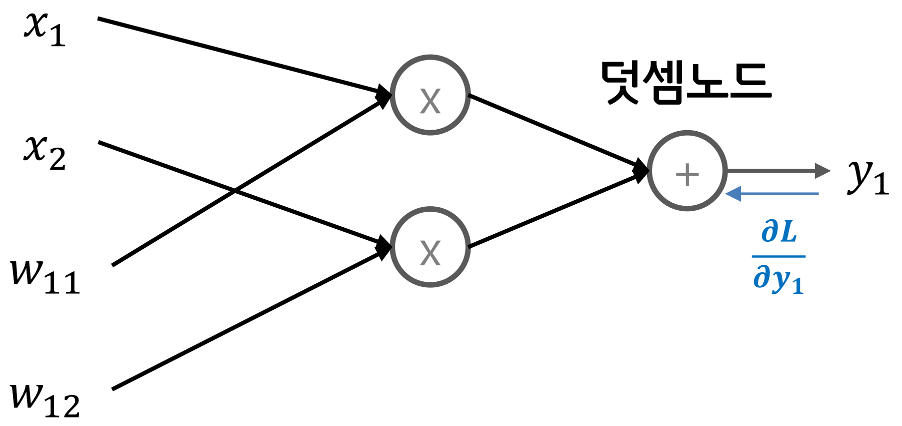
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



같은 벡터단위로 살펴봅시다

복습 : Computational Graph

- Affine 계층

$$\mathbf{x}^T\mathbf{w} = \mathbf{y}$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$[x_1 \ x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

Gradient **w**

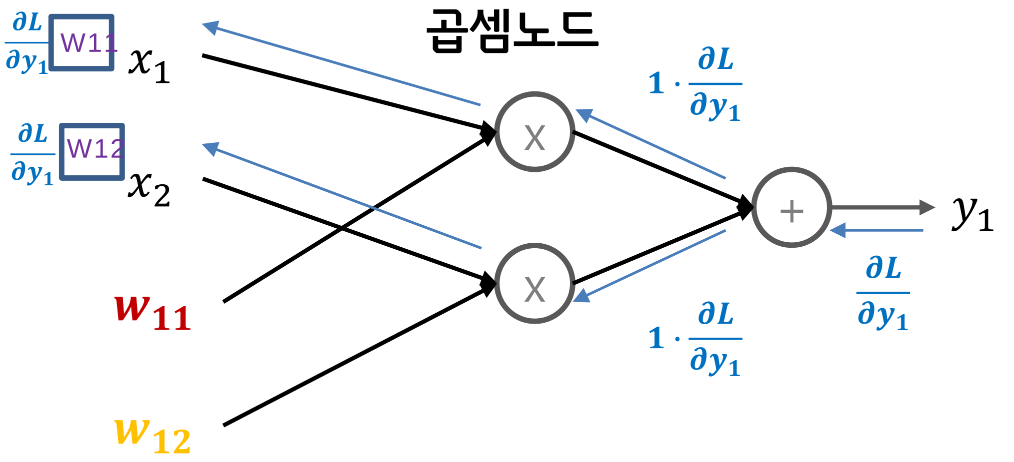
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



복습 : Computational Graph

모두의연구소

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \;\; x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$\frac{\partial L}{\partial \mathrm{x}} = \left[\frac{\partial L}{\partial y_1} w_{11} \quad \frac{\partial L}{\partial y_1} w_{12}\right] = \frac{\partial L}{\partial y_1} \cdot [w_{11} \quad w_{12}] = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

Gradient **w**

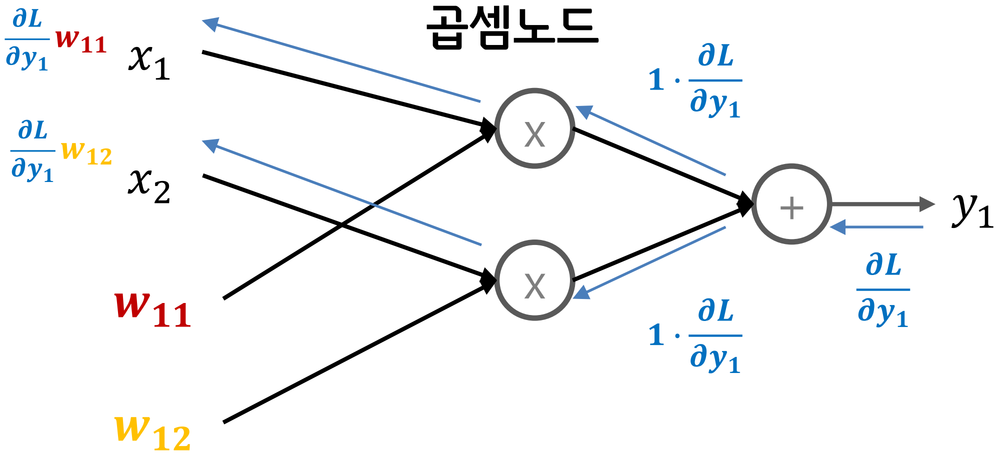
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



복습 : Computational Graph

모두의연구소

$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$

같은형태로 나오게 $\boxed{[x_1 \;\; x_2]} \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$

$w_{11}x_1 + w_{12}x_2 = y_1$

$$\frac{\partial L}{\partial \mathrm{x}} = \boxed{\left[\frac{\partial L}{\partial y_1} w_{11} \quad \frac{\partial L}{\partial y_1} w_{12}\right]} = \frac{\partial L}{\partial y_1} \cdot [w_{11} \quad w_{12}] = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

Gradient **w**

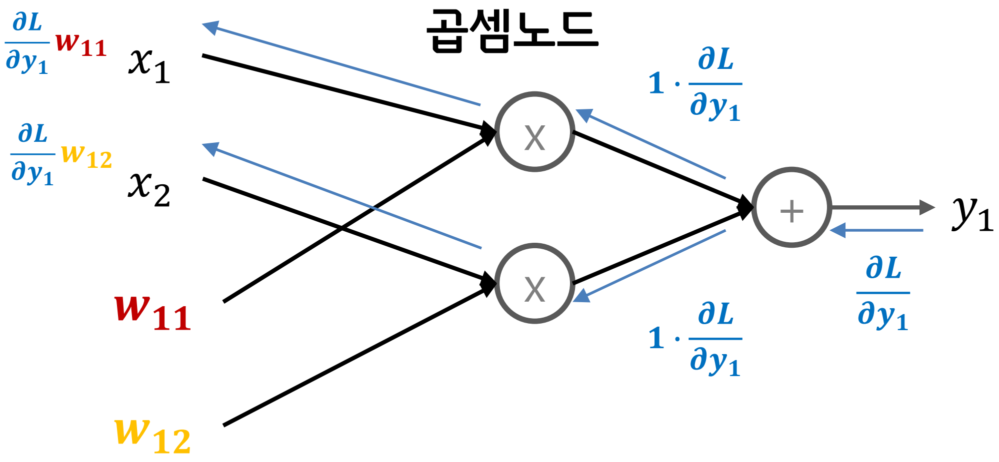
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



복습 : Computational Graph

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \;\; x_2] \times \boxed{\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix}} = y_1$$ 같은형태로 나오게

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$\frac{\partial L}{\partial \mathrm{x}} = \left[\frac{\partial L}{\partial y_1} w_{11} \quad \frac{\partial L}{\partial y_1} w_{12}\right] = \frac{\partial L}{\partial y_1} \cdot \boxed{[w_{11} \quad w_{12}]} = \frac{\partial L}{\partial y_1} \cdot \boxed{\mathbf{w}^{\mathrm{T}}}$$

Gradient **w**

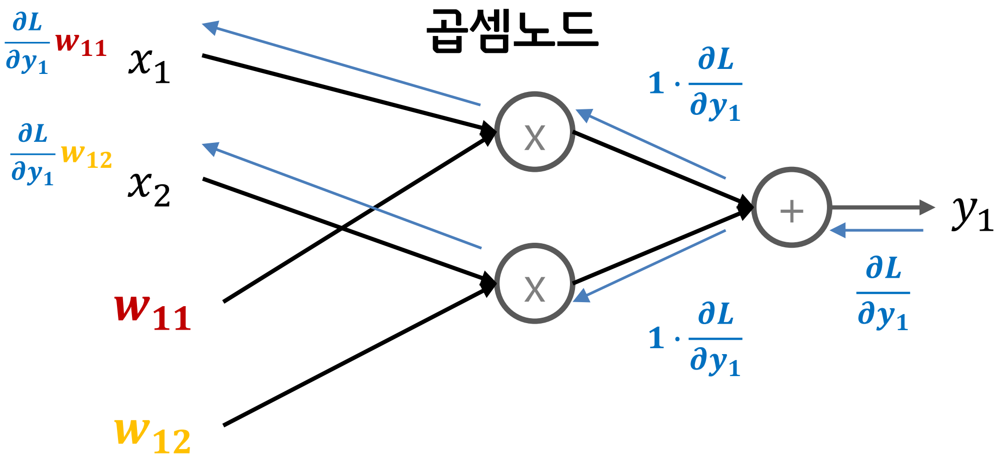
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



복습 : Computational Graph

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \ \ x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$\boxed{\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}}$$

Gradient **w**

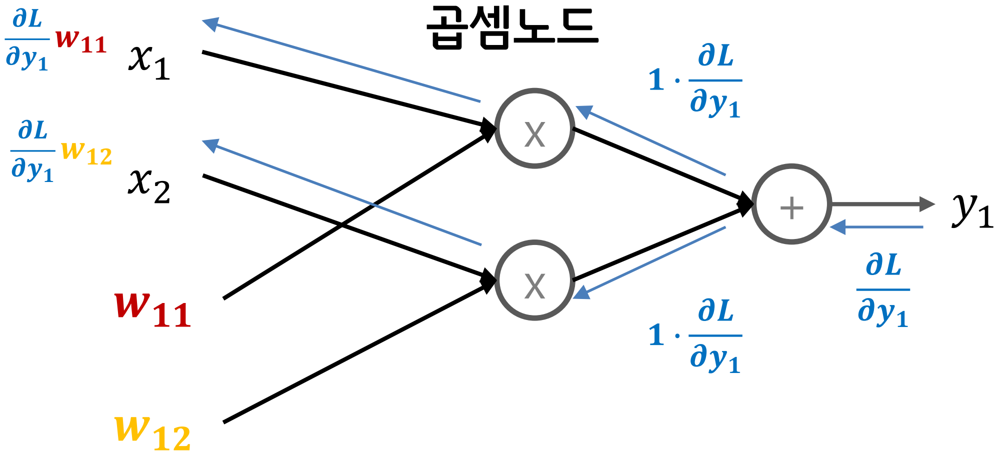
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



- Affine 계층

$$\mathbf{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$[x_1\ x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

Gradient **w**

$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

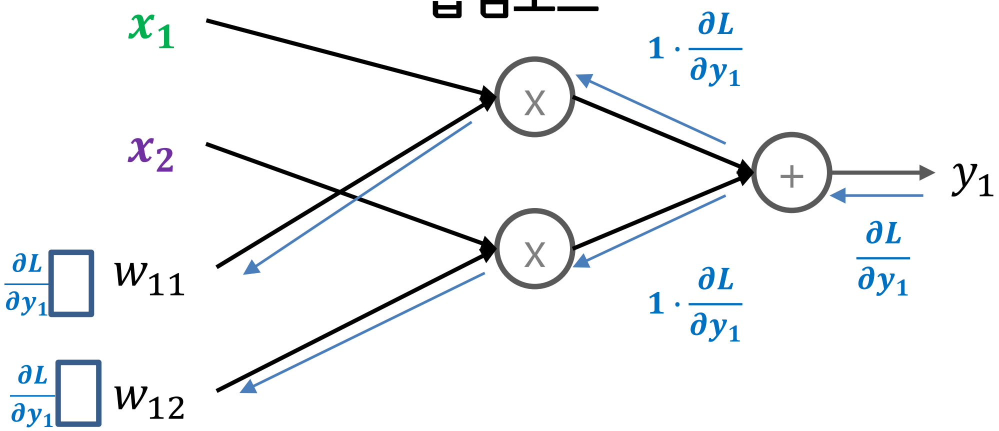
$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$

곱셈노드



복습 : Computational Graph

$$\frac{\partial L}{\partial \mathrm{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \;\; x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$\frac{\partial L}{\partial \mathrm{w}} = \begin{bmatrix} \frac{\partial L}{\partial y_1} x_1 \\ \frac{\partial L}{\partial y_1} x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \frac{\partial L}{\partial y_1} = \mathrm{x}^{\mathrm{T}} \cdot \frac{\partial L}{\partial y_1}$$

Gradient **w**

$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

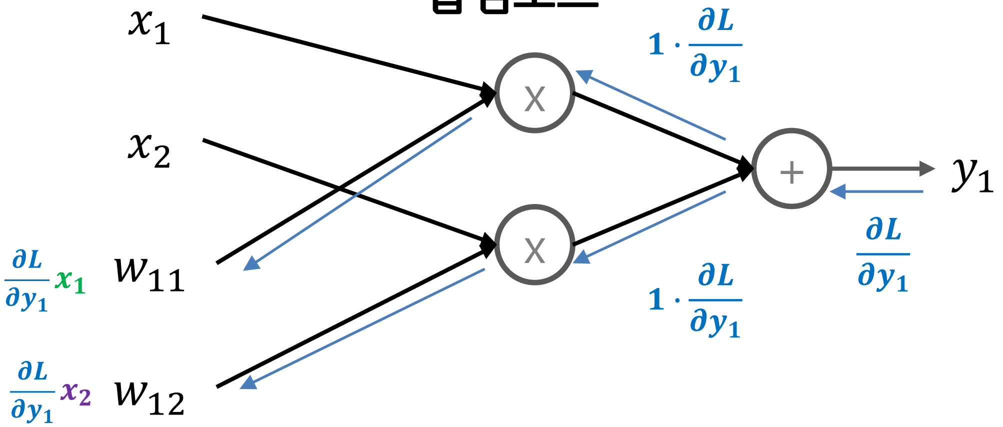
$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$

곱셈노드



복습 : Computational Graph

$$\frac{\partial L}{\partial \mathrm{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

같은형태로 나오게

$$[x_1 \;\; x_2] \times \boxed{\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix}} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

$$\frac{\partial L}{\partial \mathrm{w}} = \boxed{\begin{bmatrix} \frac{\partial L}{\partial y_1} x_1 \\ \frac{\partial L}{\partial y_1} x_2 \end{bmatrix}} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \frac{\partial L}{\partial y_1} = \mathbf{x}^{\mathrm{T}} \cdot \frac{\partial L}{\partial y_1}$$

Gradient **w**

$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

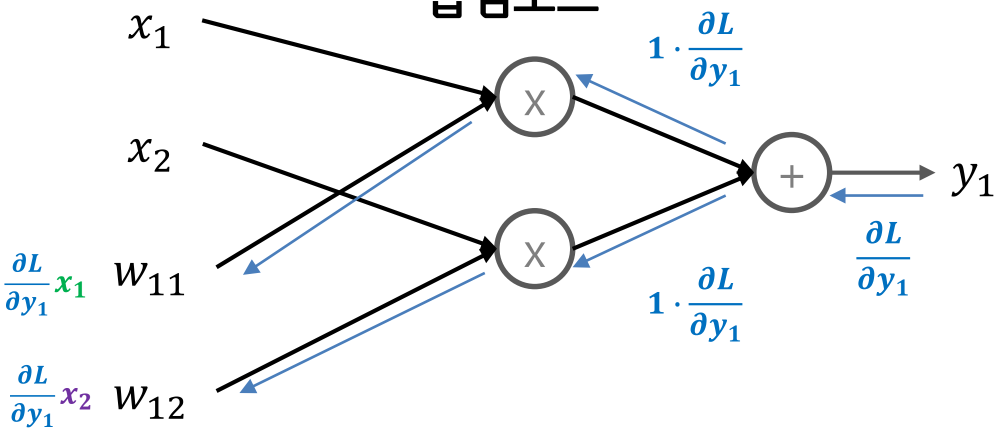
$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$

곱셈노드



복습 : Computational Graph

$$\frac{\partial L}{\partial \mathrm{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$\boxed{[x_1 \ \ x_2]} \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

같은 표현으로

$$\frac{\partial L}{\partial \mathrm{w}} = \begin{bmatrix} \frac{\partial L}{\partial y_1} x_1 \\ \frac{\partial L}{\partial y_1} x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \frac{\partial L}{\partial y_1} = \boxed{\mathrm{x}^{\mathrm{T}}} \cdot \frac{\partial L}{\partial y_1}$$

Gradient **w**

$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

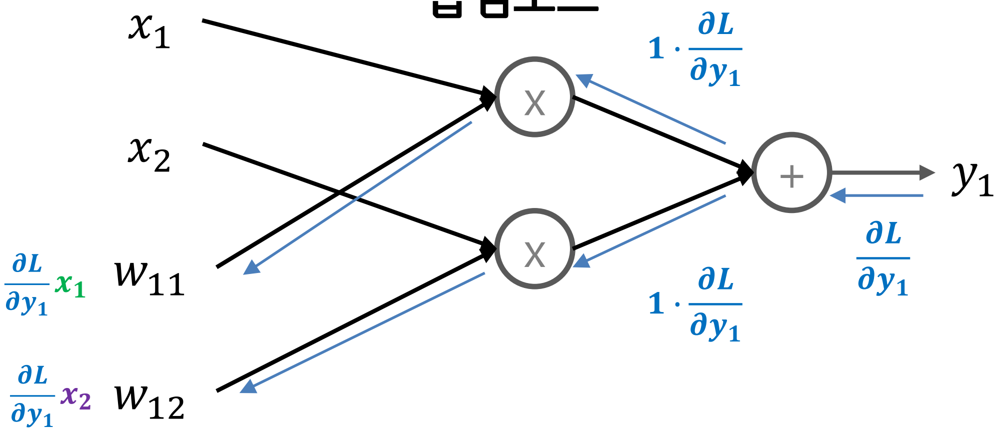
$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient **x**

$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$

곱셈노드



복습 : Computational Graph

$$\frac{\partial L}{\partial \mathrm{x}} = \frac{\partial L}{\partial y_1} \cdot \mathbf{w}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{x}^{\mathrm{T}} \cdot \frac{\partial L}{\partial y_1}$$

$$\mathrm{x}^{\mathrm{T}}\mathbf{w} = \mathbf{y}$$

$$[x_1 \ \ x_2] \times \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = y_1$$

$$w_{11}x_1 + w_{12}x_2 = y_1$$

Gradient w

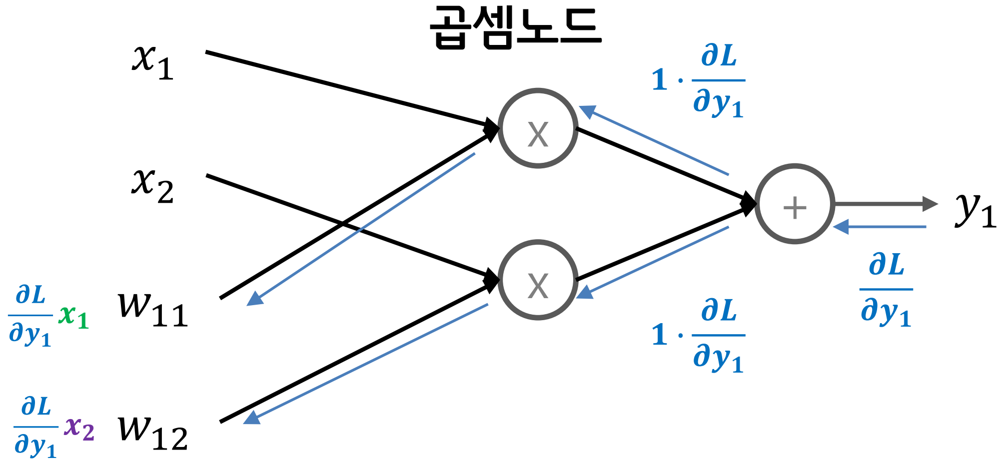
$$\frac{\partial y_1}{\partial w_{11}} = x_1$$

$$\frac{\partial y_1}{\partial w_{12}} = x_2$$

Gradient x

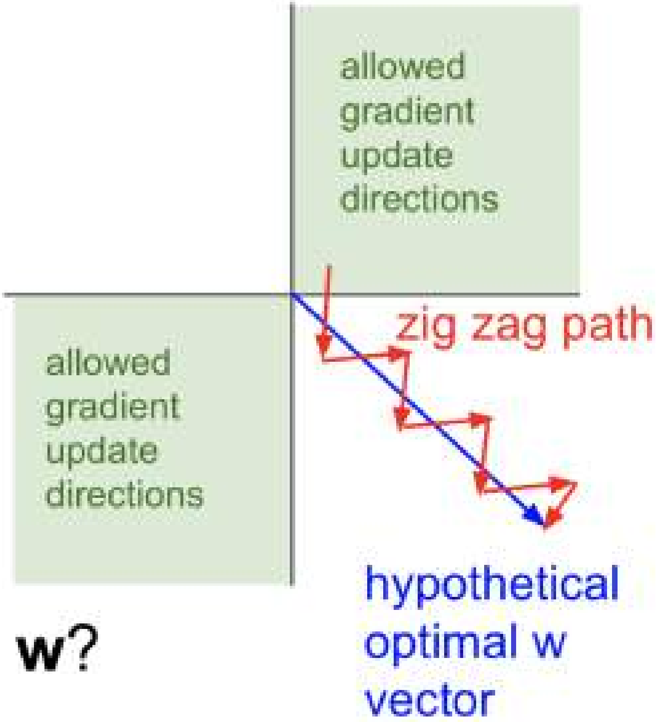
$$\frac{\partial y_1}{\partial x_1} = w_{11}$$

$$\frac{\partial y_1}{\partial x_2} = w_{12}$$



Consider what happens when the input to a neuron is always positive...

$$f\left(\sum_i w_i x_i + b\right)$$



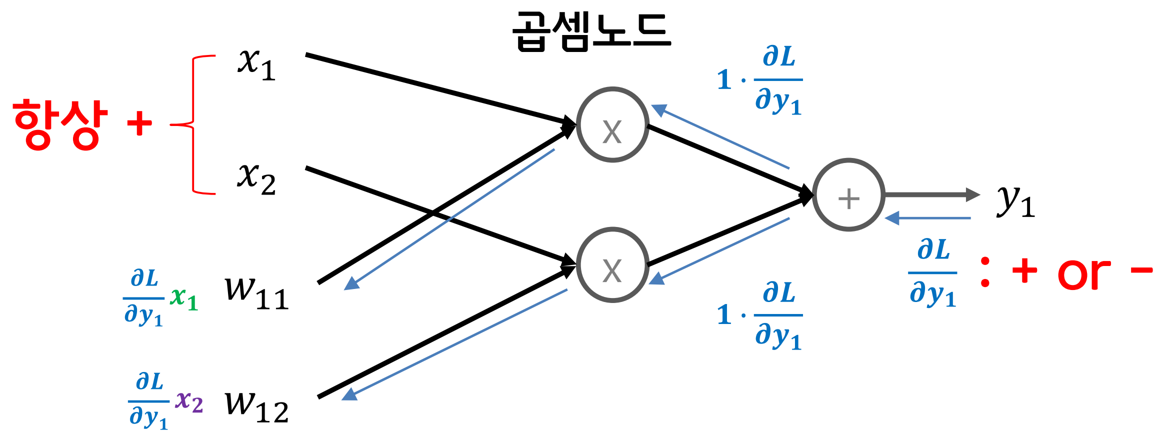
What can we say about the gradients on **w**?

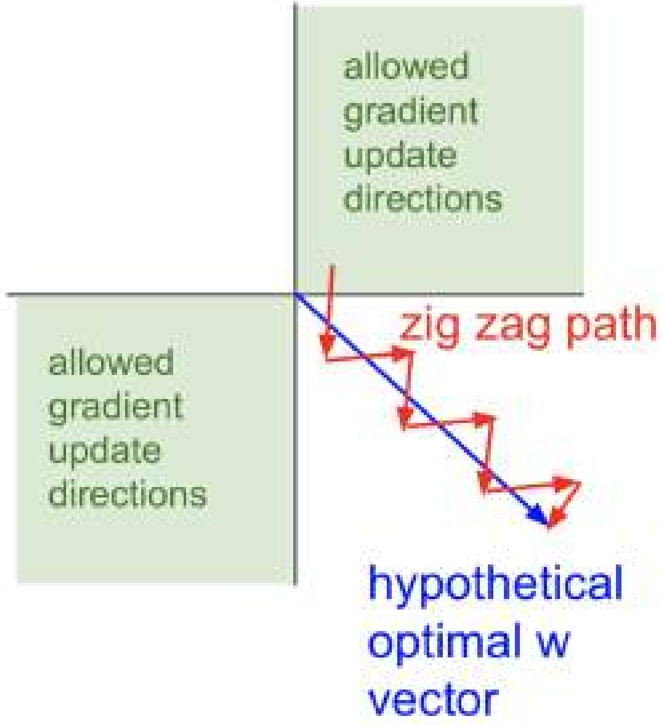
Always all positive or all negative :(

(this is also why you want zero-mean data!)

Consider what happens when the input to a neuron is always positive...

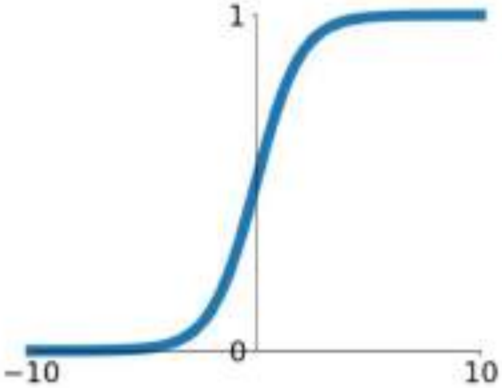
$$f\left(\sum_i w_i x_i + b\right)$$





Activation Functions

$$\sigma(x) = 1/(1 + e^{-x})$$



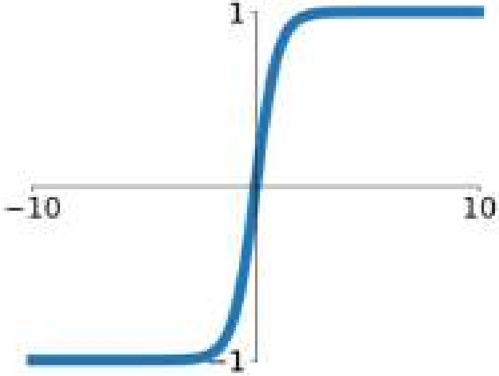
Sigmoid

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

3 problems:

1. Saturated neurons "kill" the gradients
2. Sigmoid outputs are not zero-centered
3. exp() is a bit compute expensive

Activation Functions

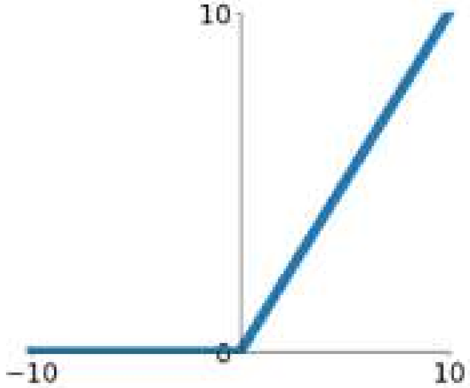


tanh(x)

- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated :(

[LeCun et al., 1991]

Activation Functions



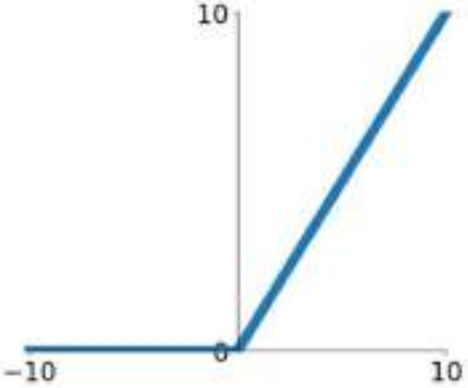
ReLU
(Rectified Linear Unit)

- Computes **f(x) = max(0,x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)
- Actually more biologically plausible than sigmoid

[Krizhevsky et al., 2012]

Activation Functions



ReLU
(Rectified Linear Unit)

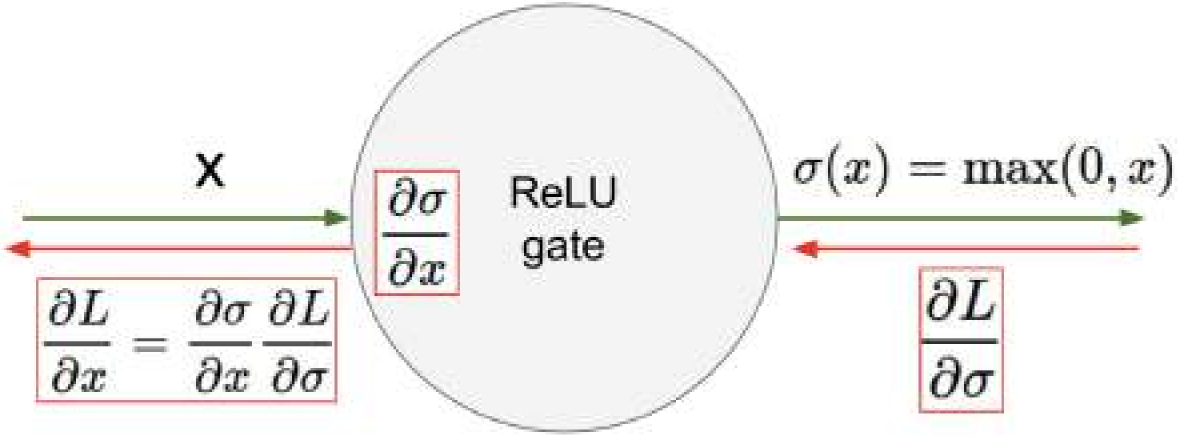
- Computes **f(x) = max(0,x)**

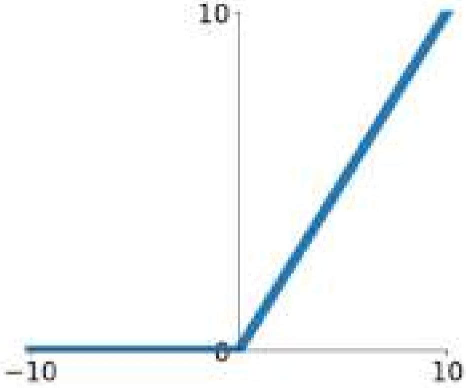
- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)
- Actually more biologically plausible than sigmoid

- Not zero-centered output
- An annoyance:

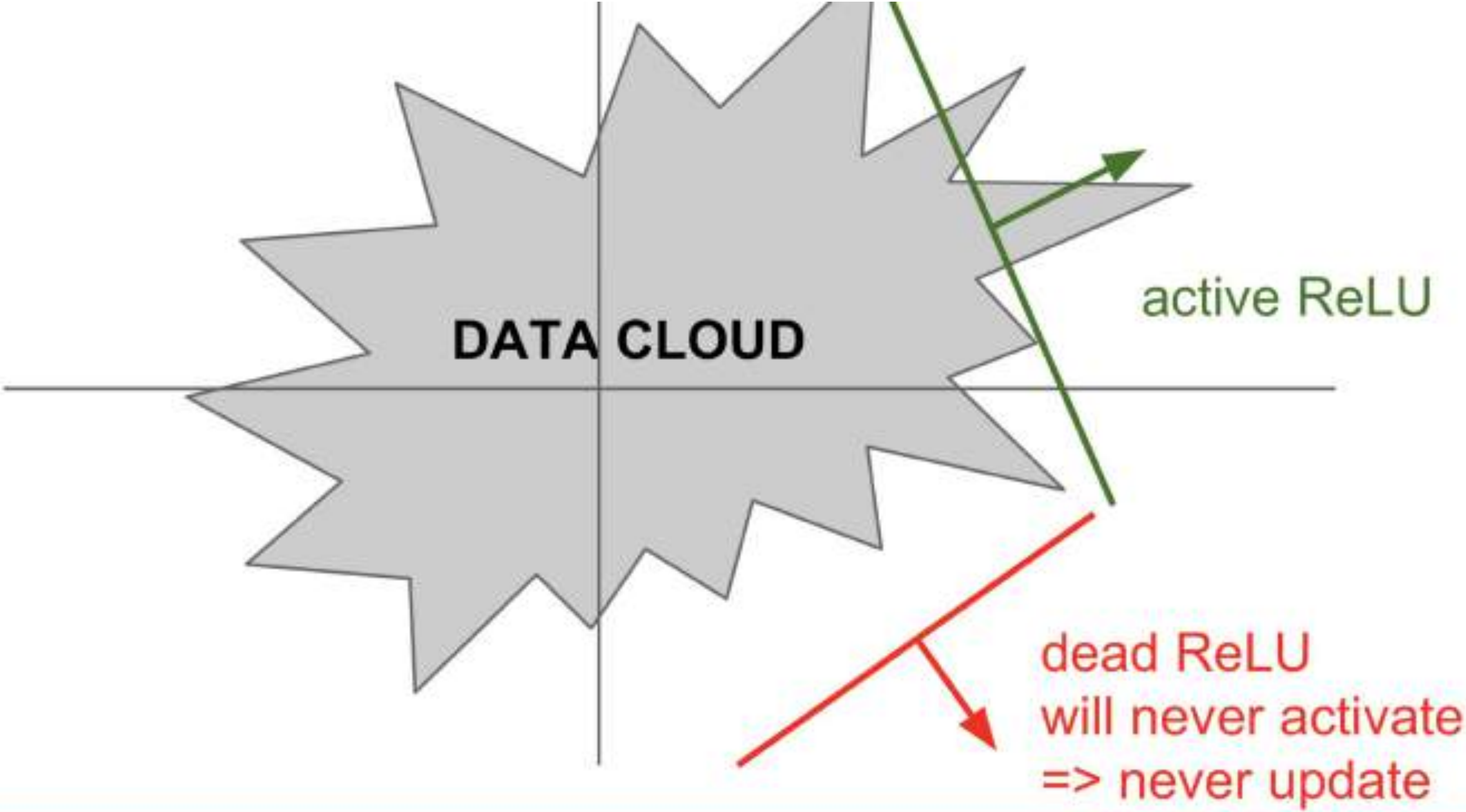
hint: what is the gradient when x < 0?

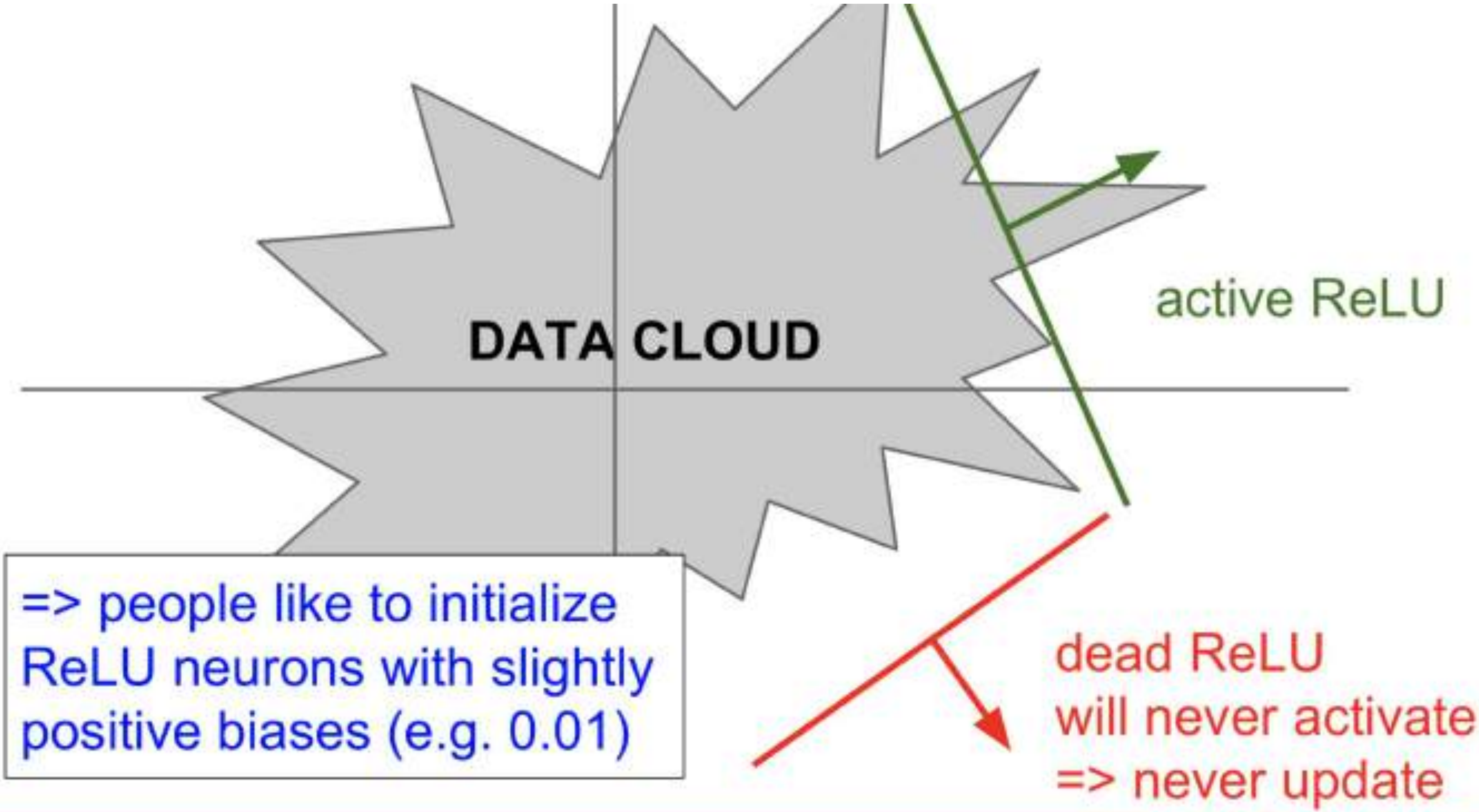
x <0 :: gradient가 항상 0





What happens when x = -10?
What happens when x = 0?
What happens when x = 10?

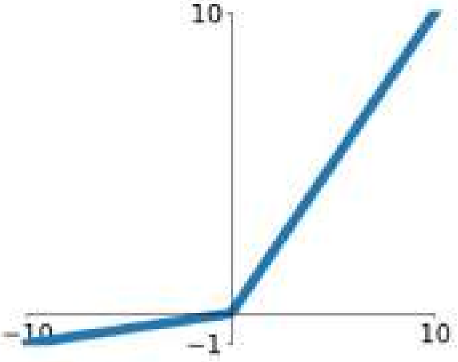

DATA CLOUD
active ReLU
dead ReLU
will never activate
=> never update



=> people like to initialize ReLU neurons with slightly positive biases (e.g. 0.01)

Activation Functions

[Mass et al., 2013]
[He et al., 2015]



- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not "die".**

Leaky ReLU

$$f(x) = \max(0.01x, x)$$

Gradient 가 0
: 업데이트가 발생하지 않음

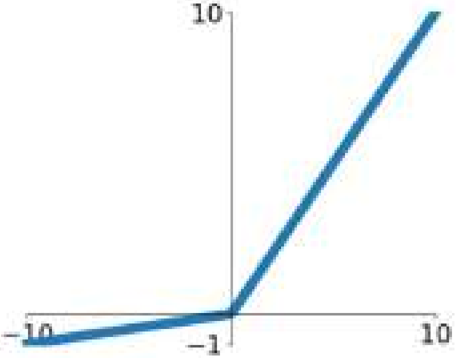
모두의연구소

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{x}^{\mathrm{T}} \cdot \frac{\partial L}{\partial y_1}$$

– Update : $$\mathbf{w} = \mathbf{w} - \eta \frac{\partial L}{\partial \mathbf{w}}$$

Activation Functions

[Mass et al., 2013]
[He et al., 2015]



- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not "die".**

Leaky ReLU

$$f(x) = \max(0.01x, x)$$

Parametric Rectifier (PReLU)

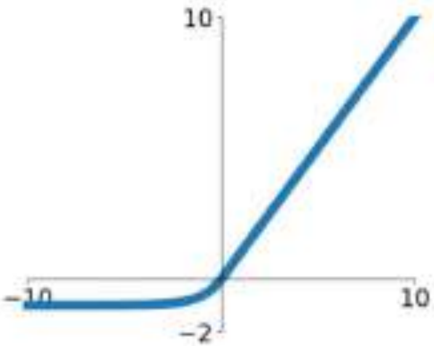
$$f(x) = \max(\alpha x, x)$$

backprop into \alpha
(parameter)

Activation Functions

[Clevert et al., 2015]

Exponential Linear Units (ELU)



- All benefits of ReLU
- Closer to zero mean outputs
- Negative saturation regime compared with Leaky ReLU adds some robustness to noise

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \le 0 \end{cases}$$

- Computation requires exp()

Maxout "Neuron"

[Goodfellow et al., 2013]

- Does not have the basic form of dot product -> nonlinearity
- Generalizes ReLU and Leaky ReLU
- Linear Regime! Does not saturate! Does not die!

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

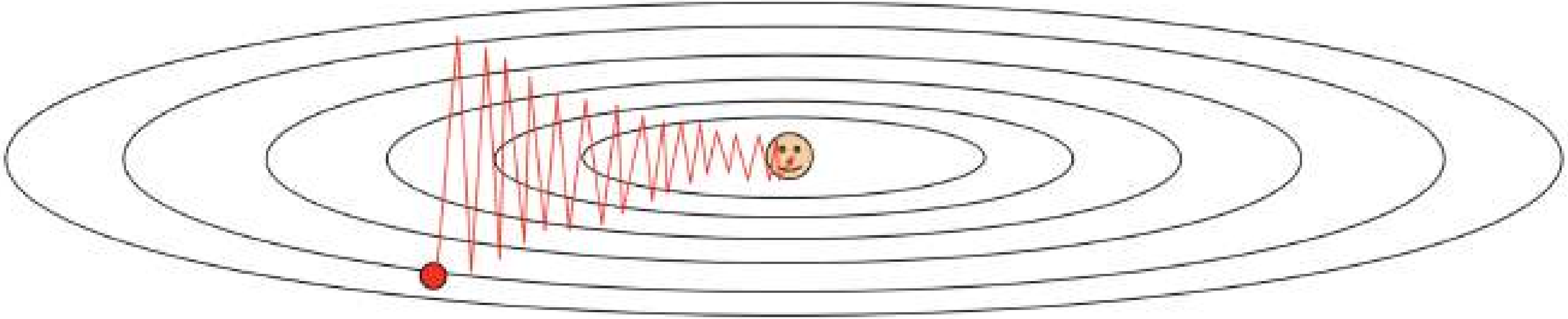
Problem: doubles the number of parameters/neuron :(

TLDR: In practice:

- Use ReLU. Be careful with your learning rates
- Try out Leaky ReLU / Maxout / ELU
- Try out tanh but don't expect much
- Don't use sigmoid

Optimization: Problems with SGD

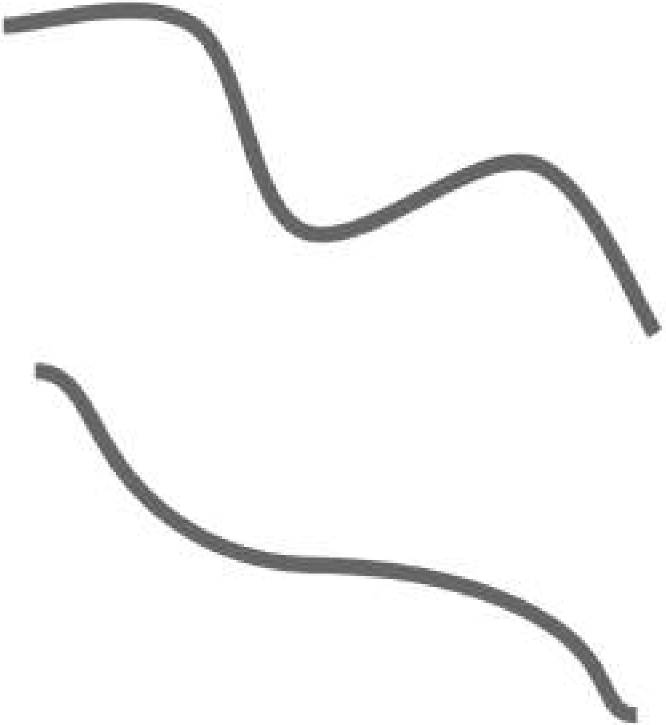
What if loss changes quickly in one direction and slowly in another?
What does gradient descent do?
Very slow progress along shallow dimension, jitter along steep direction



Loss function has high **condition number**: ratio of largest to smallest singular value of the Hessian matrix is large

Optimization: Problems with SGD

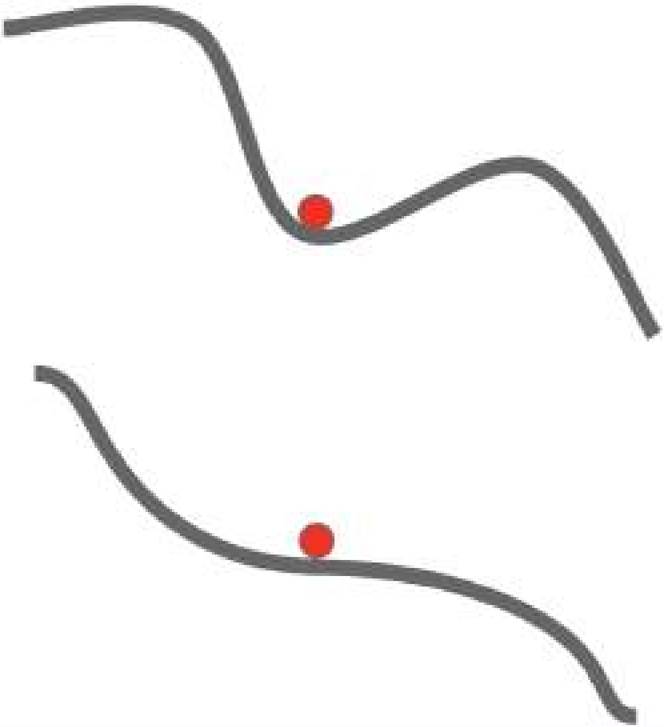
What if the loss function has a **local minima** or **saddle point**?



Optimization: Problems with SGD

What if the loss function has a **local minima** or **saddle point**?

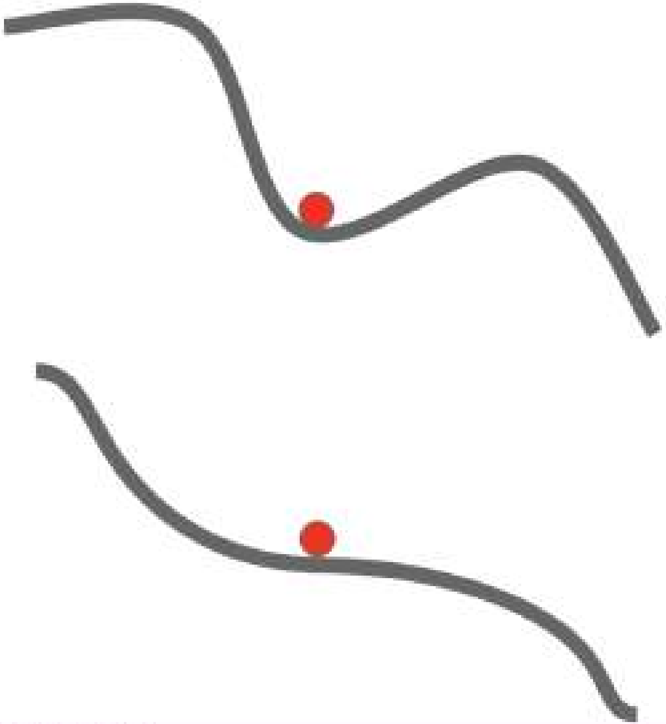
Zero gradient, gradient descent gets stuck



Optimization: Problems with SGD

What if the loss function has a **local minima** or **saddle point**?

Saddle points much more common in high dimension



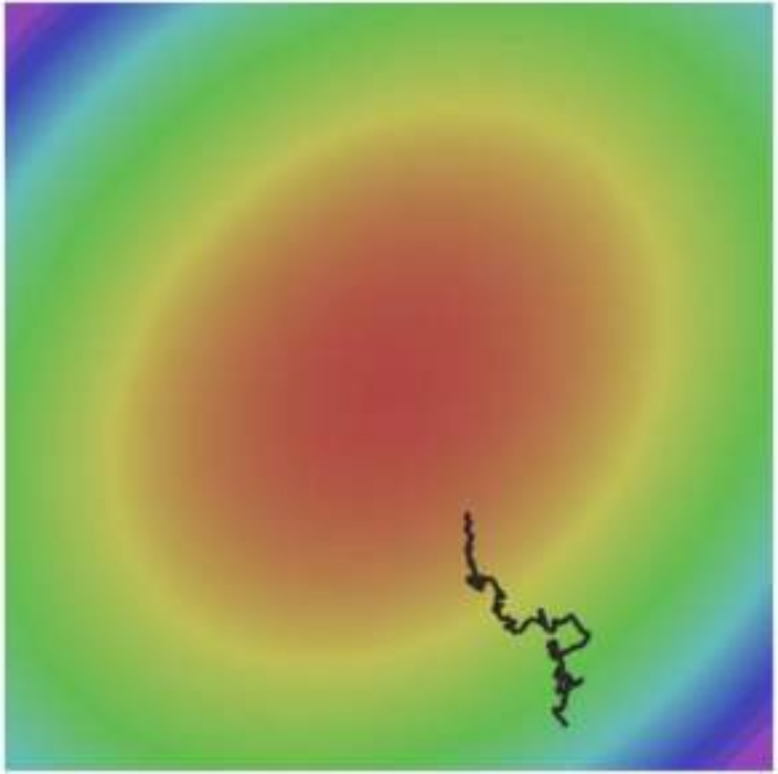
Dauphin et al, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization", NIPS 2014

Optimization: Problems with SGD

Our gradients come from minibatches so they can be noisy!

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(x_i, y_i, W)$$

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W)$$



SGD + Momentum

SGD

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

```
while True:
  dx = compute_gradient(x)
  x += learning_rate * dx
```

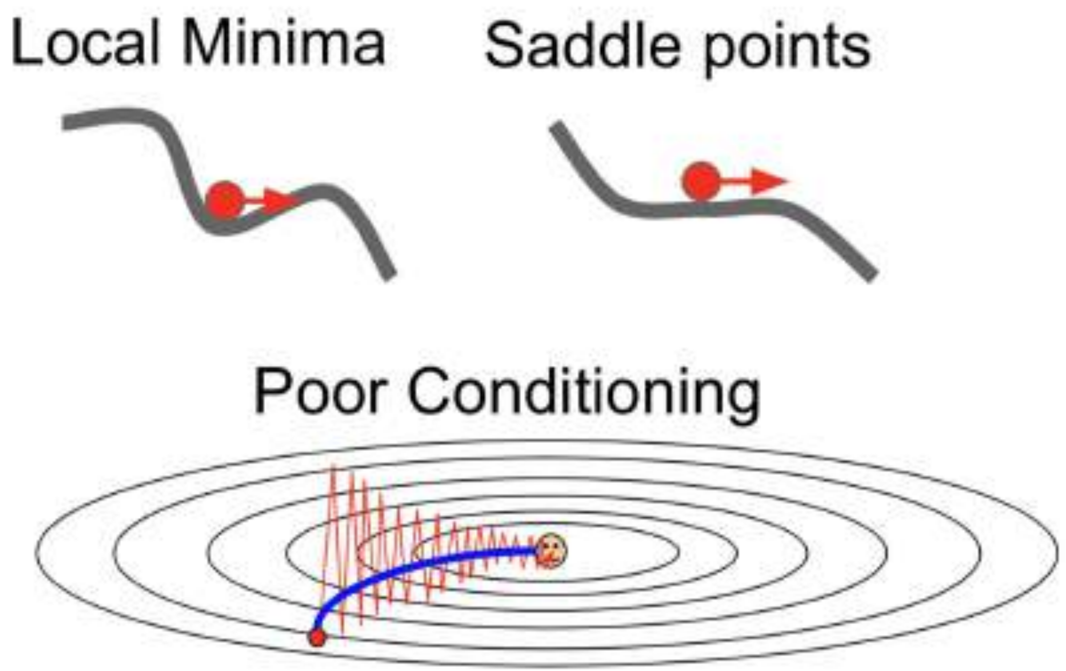
SGD+Momentum

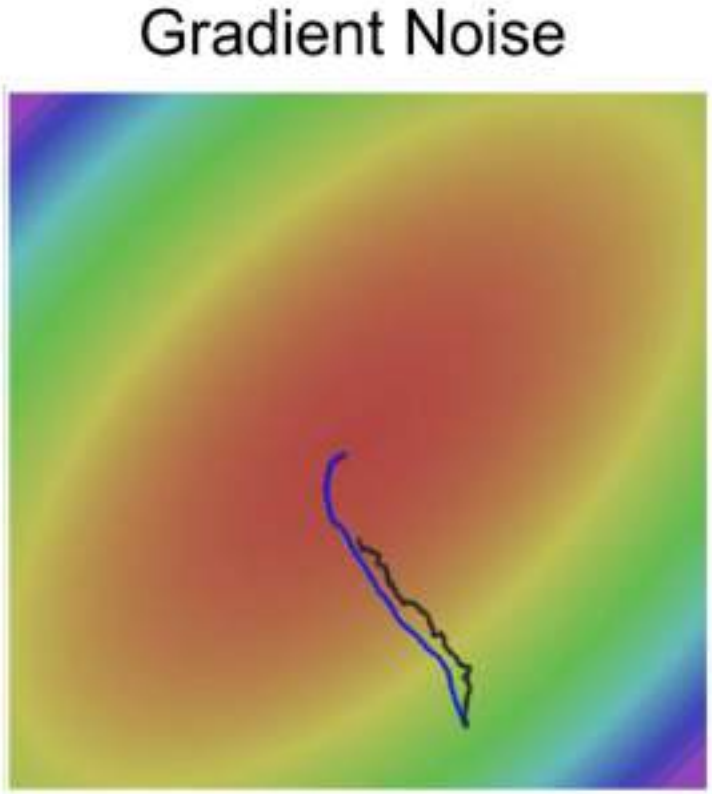
$$v_{t+1} = \rho v_t + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha v_{t+1}$$

```
vx = 0
while True:
  dx = compute_gradient(x)
  vx = rho * vx + dx
  x += learning_rate * vx
```

- Build up "velocity" as a running mean of gradients
- Rho gives "friction"; typically rho=0.9 or 0.99

SGD + Momentum





지난시간 돌아보기 ...

모두의연구소

- Momentum

$$v \leftarrow \alpha v - \eta \frac{\partial L}{\partial W}$$

$$W \leftarrow W + v$$

업데이트 1) $v_1 \leftarrow \alpha * 0 - K_o : -K_o$
업데이트 2) $v_2 \leftarrow \alpha v_1 - K_1 : -\alpha K_o - K_1$
업데이트 3) $v_3 \leftarrow \alpha v_2 - K_2 : -\alpha^2 K_o - \alpha K_1 - K_2$
업데이트 4) $v_4 \leftarrow \alpha v_3 - K_3 : -\alpha^3 K_o - \alpha^2 K_1 - \alpha K_2 - K_3$

- 1) $W \leftarrow W + v_1$
- 2) $W \leftarrow W + v_2$
- 3) $W \leftarrow W + v_3$
- 4) $W \leftarrow W + v_4$

- Adagrad

$$h \leftarrow h + \frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W}$$

$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

업데이트 1) $\frac{1}{\sqrt{\boxed{K_0^2}}} K_o$

업데이트 2) $\frac{1}{\sqrt{\boxed{K_1^2+K_0^2}}} K_1$

업데이트 3) $\frac{1}{\sqrt{\boxed{K_3^2+K_1^2+K_0^2}}} K_3$

업데이트 4) $\frac{1}{\sqrt{\boxed{K_4^2+K_3^2+K_1^2+K_0^2}}} K_4$

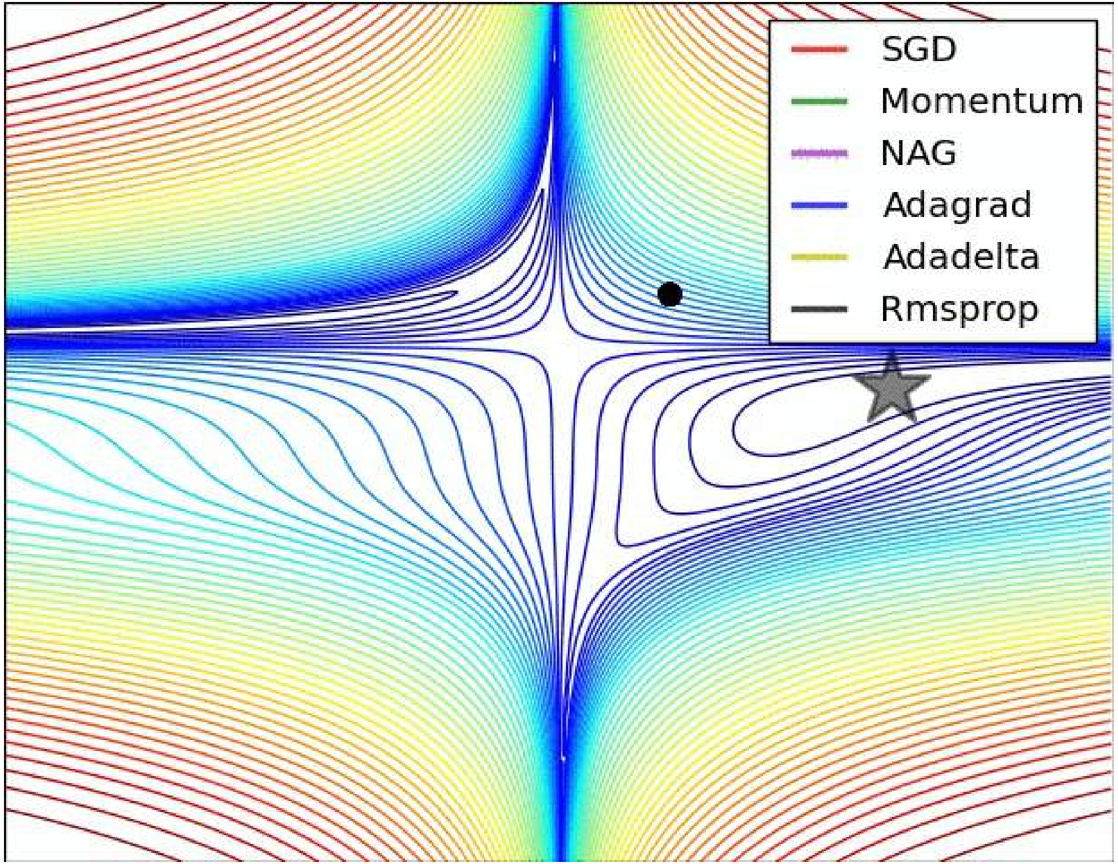
두 방법의 같이쓰자
Adam

- RMSprop

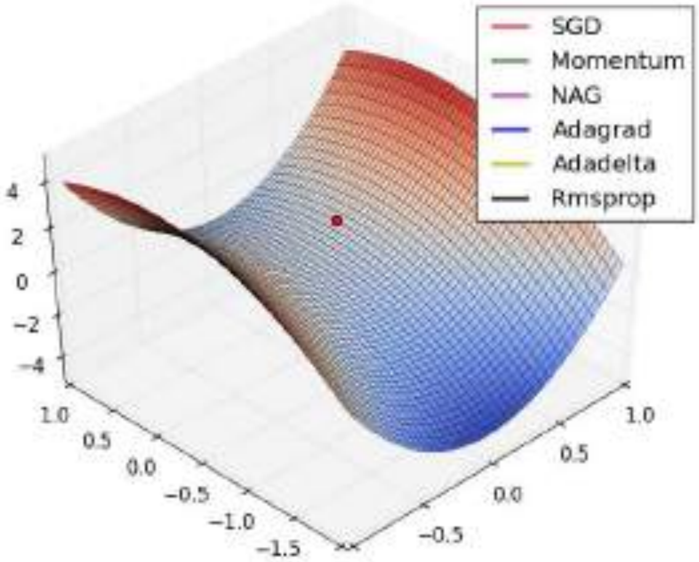
$$h \leftarrow \alpha h + (1-\alpha)(\frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W})$$

$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

업데이트 1) $h_1 = (1-a)K_1^2$
업데이트 2) $h_2 = a(1-a)K_1^2 + (1-a)K_2^2$
업데이트 3) $h_3 = a^2(1-a)K_1^2 + a(1-a)K_2^2 + (1-a)K_3^2$
업데이트 4) $h_4 = a^3(1-a)K_1^2 + a^2(1-a)K_2^2 + a(1-a)K_3^2 + (1-a)K_4^2$

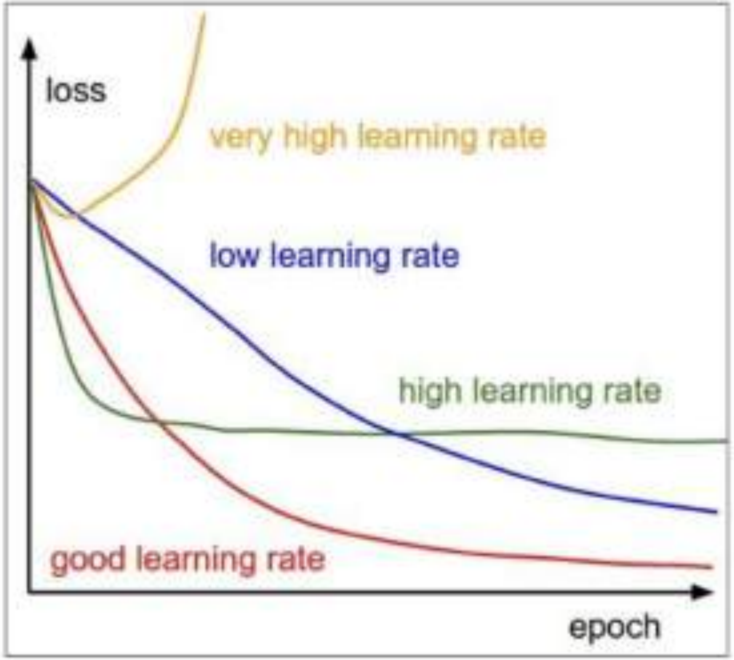


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop



SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

SGD, SGD+Momentum, Adagrad, RMSProp, Adam all have **learning rate** as a hyperparameter.



=> Learning rate decay over time!

step decay:
e.g. decay learning rate by half every few epochs.

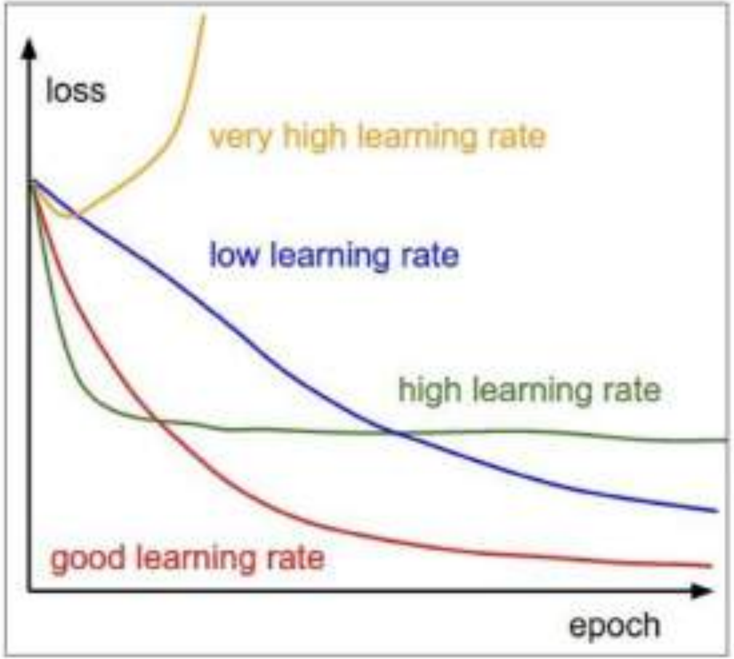
exponential decay:

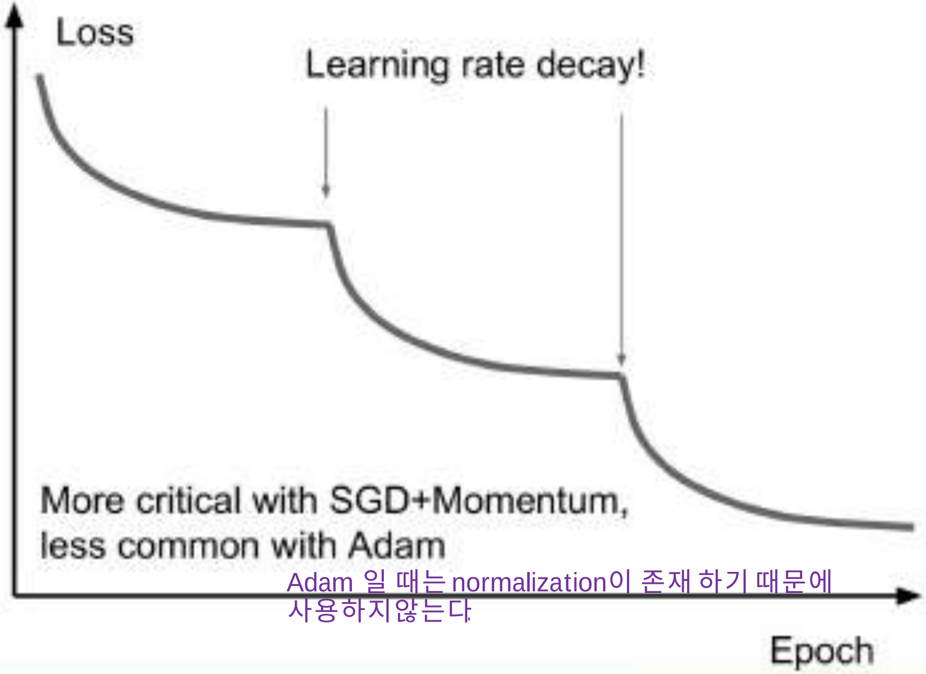
$$\alpha = \alpha_0 e^{-kt}$$

1/t decay:

$$\alpha = \alpha_0 / (1 + kt)$$

SGD, SGD+Momentum, Adagrad, RMSProp, Adam all have **learning rate** as a hyperparameter.





Adam 일 때는 normalization이 존재 하기 때문에 사용하지않는다.

지난 시간엔 ...

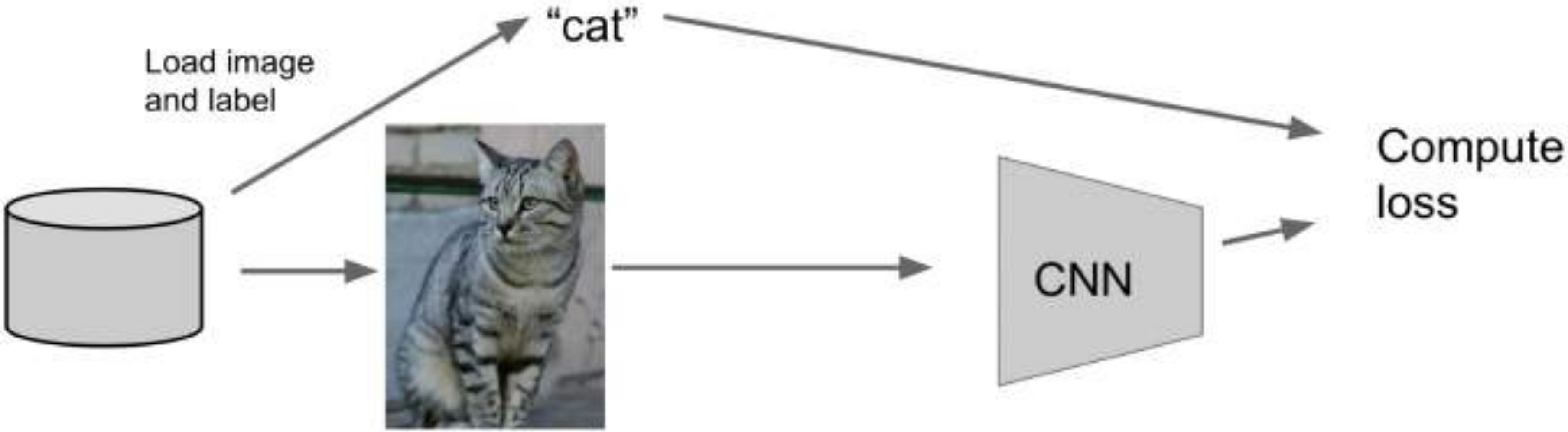
- Regularization
  - Weight Decay
  - Batch normalization
  - Dropout

지난 시간엔 ...

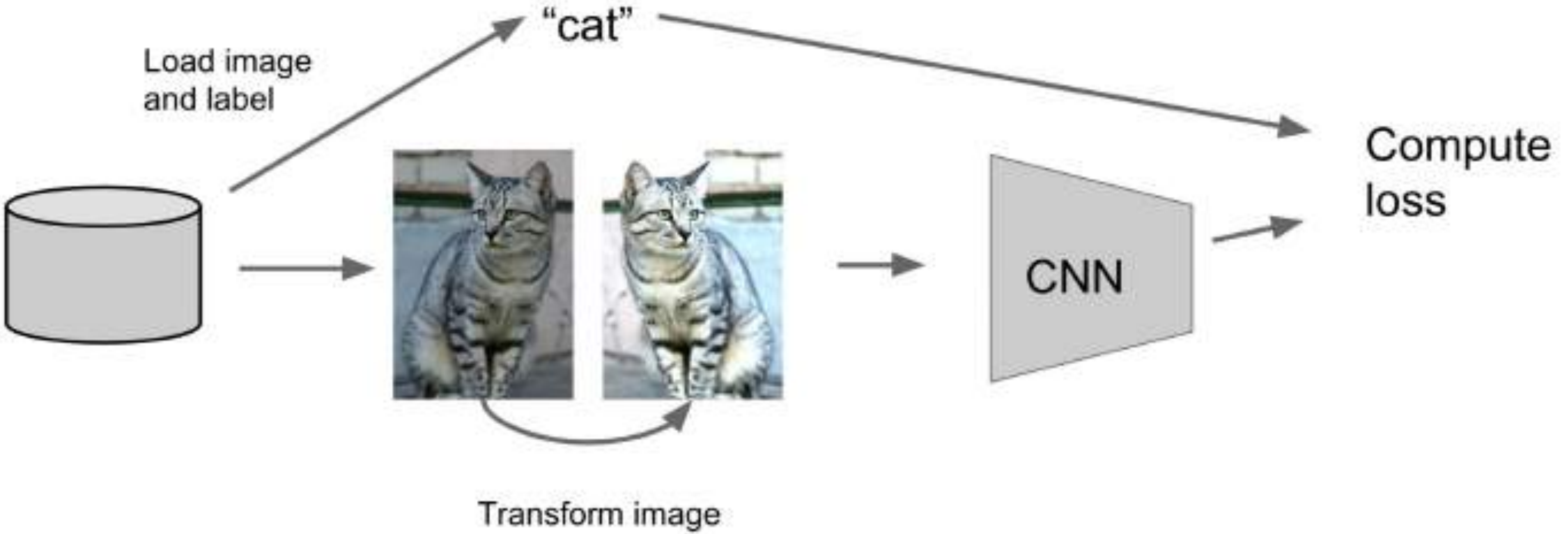
모두의연구소

- Regularization
  - Weight Decay
  - Batch normalization
  - Dropout
  - Data augmentation

Regularization: Data Augmentation

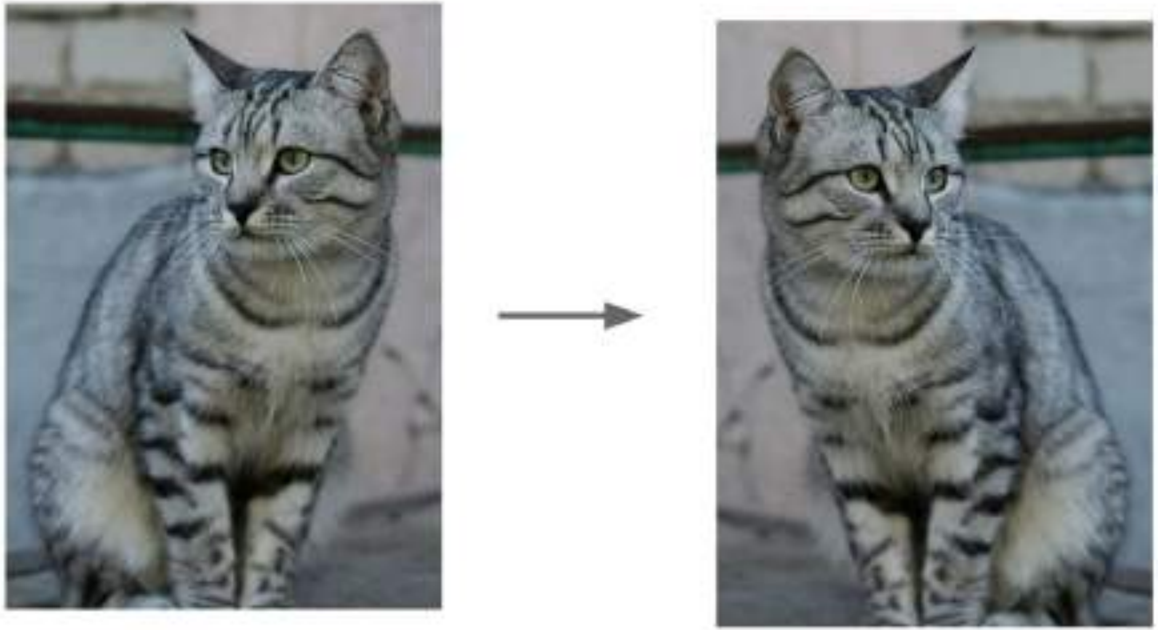


Regularization: Data Augmentation



Data Augmentation

Horizontal Flips



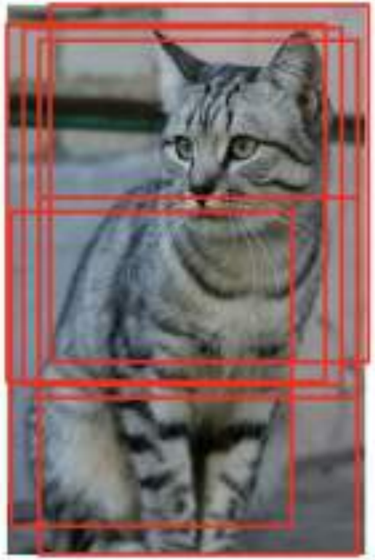
Data Augmentation

Random crops and scales

Training: sample random crops / scales

ResNet:

1. Pick random L in range [256, 480]
2. Resize training image, short side = L
3. Sample random 224 x 224 patch



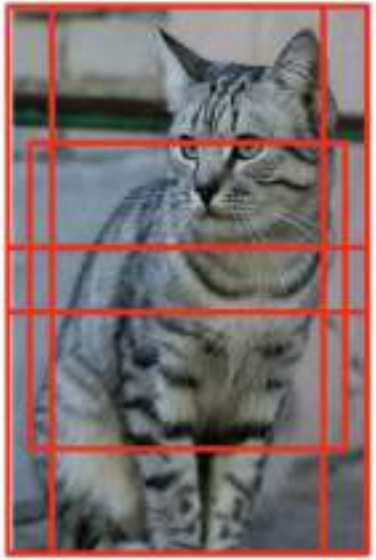
Data Augmentation

Random crops and scales

Training: sample random crops / scales

ResNet:

1. Pick random L in range [256, 480]
2. Resize training image, short side = L
3. Sample random 224 x 224 patch



Testing: average a fixed set of crops

ResNet:

1. Resize image at 5 scales: {224, 256, 384, 480, 640}
2. For each size, use 10 224 x 224 crops: 4 corners + center, + flips

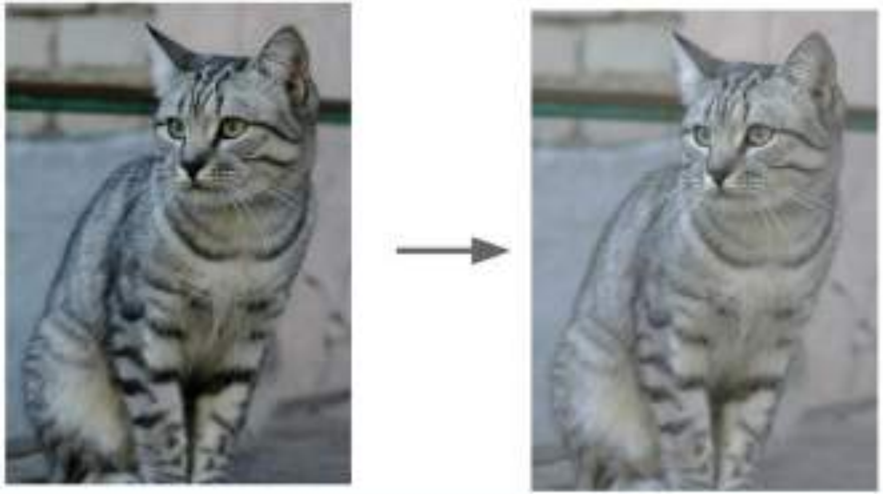
horizontal flip

Data Augmentation
Color Jitter

Simple: Randomize contrast and brightness

Data Augmentation
Color Jitter

Simple: Randomize contrast and brightness



More Complex:

1. Apply PCA to all [R, G, B] pixels in training set
2. Sample a “color offset” along principal component directions
3. Add offset to all pixels of a training image

(As seen in *[Krizhevsky et al. 2012]*, ResNet, etc)

Data Augmentation

Get creative for your problem!

Random mix/combinations of :

- translation
- rotation
- stretching
- shearing,
- lens distortions, … (go crazy)

Regularization: A common pattern

Training: Add random noise

Testing: Marginalize over the noise

Examples:

Dropout

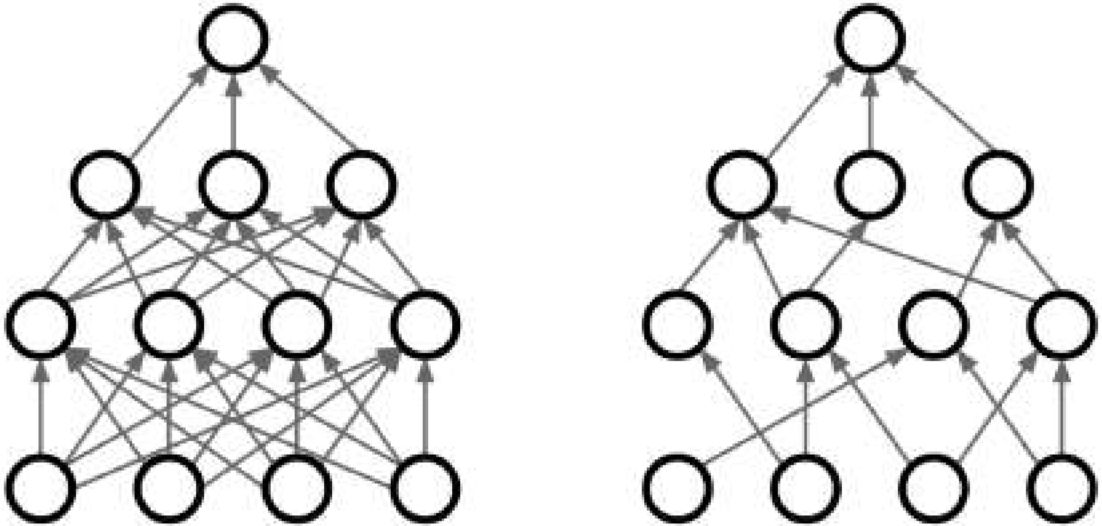
random noise를 추가하여 ...

Batch Normalization

Data Augmentation

DropConnect

>> over fitting 방지

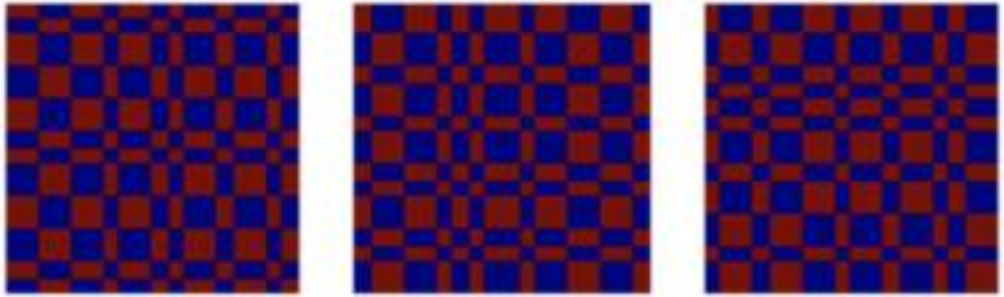


Wan et al, "Regularization of Neural Networks using DropConnect", ICML 2013

Regularization: A common pattern

Training: Add random noise
Testing: Marginalize over the noise

Examples:
Dropout
Batch Normalization
Data Augmentation
DropConnect
Fractional Max Pooling



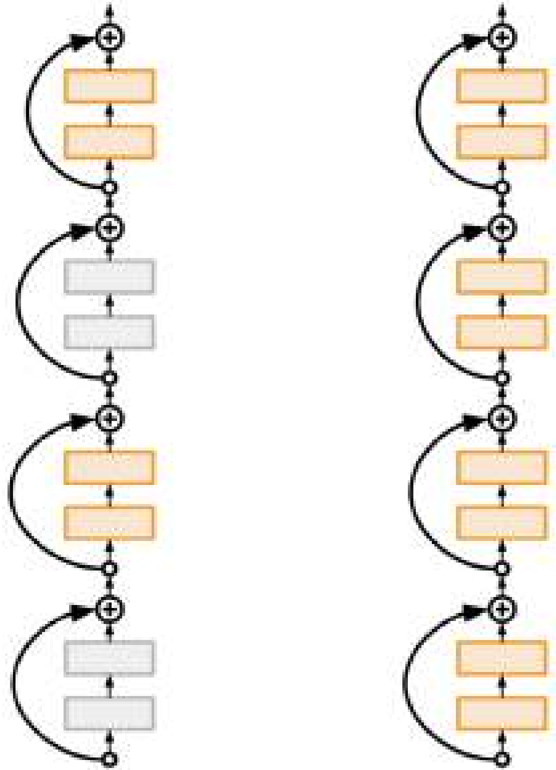
CNN 에서 사용 되는 Image size down 기법

Graham, "Fractional Max Pooling", arXiv 2014

Regularization: A common pattern

Training: Add random noise
Testing: Marginalize over the noise

Examples:
Dropout
Batch Normalization
Data Augmentation
DropConnect
Fractional Max Pooling
Stochastic Depth



Huang et al, "Deep Networks with Stochastic Depth", ECCV 2016

Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

BUSTED

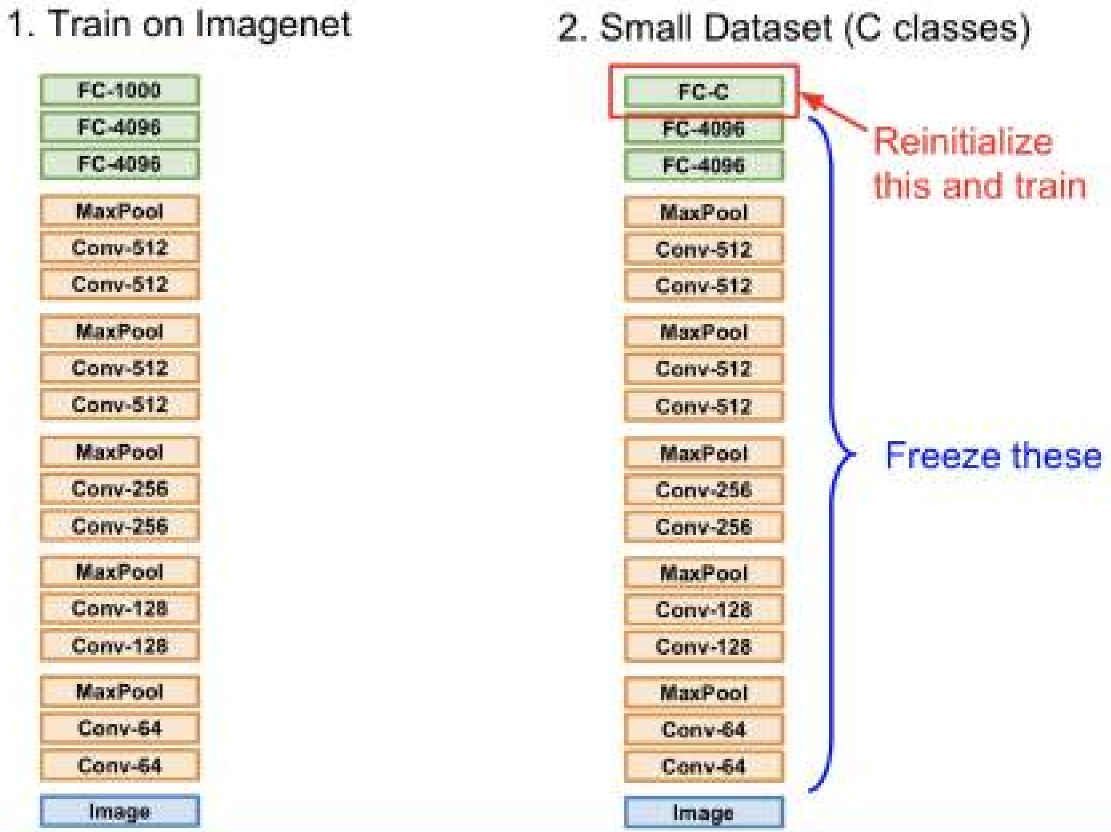
Transfer Learning with CNNs

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

1. Train on Imagenet

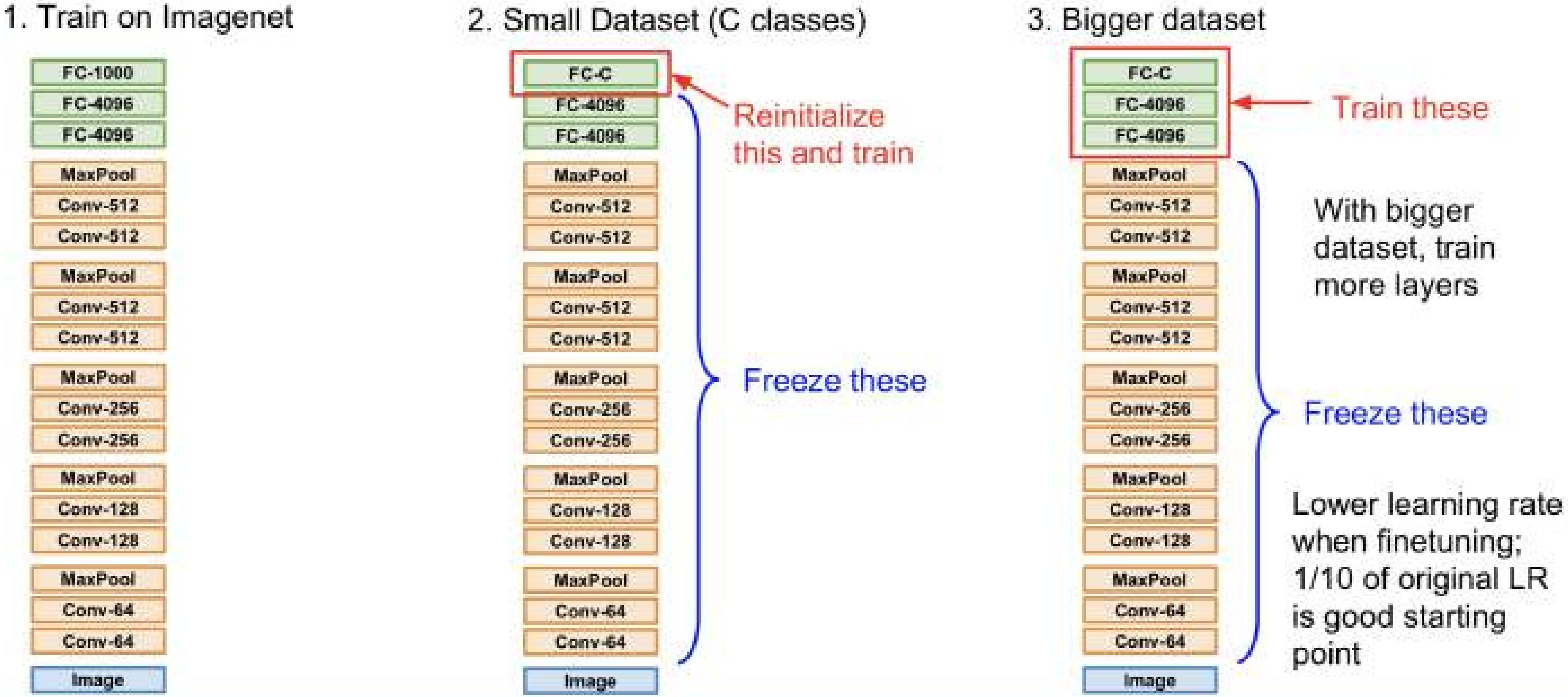
Transfer Learning with CNNs

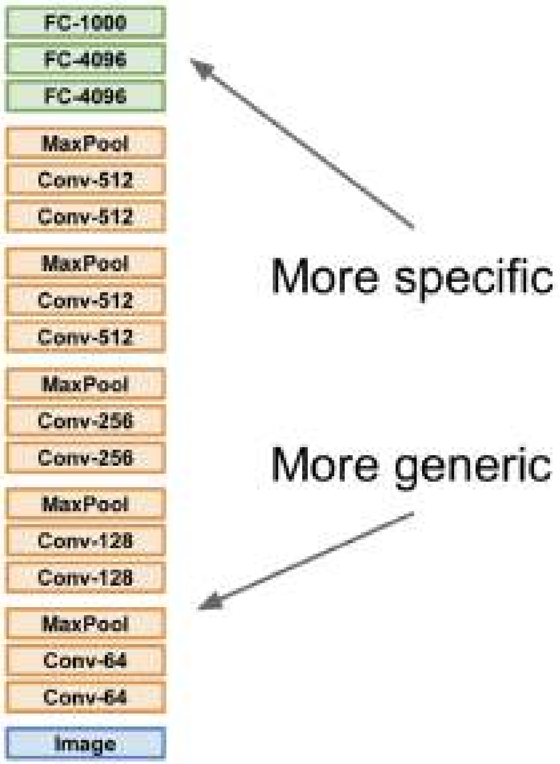
Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014



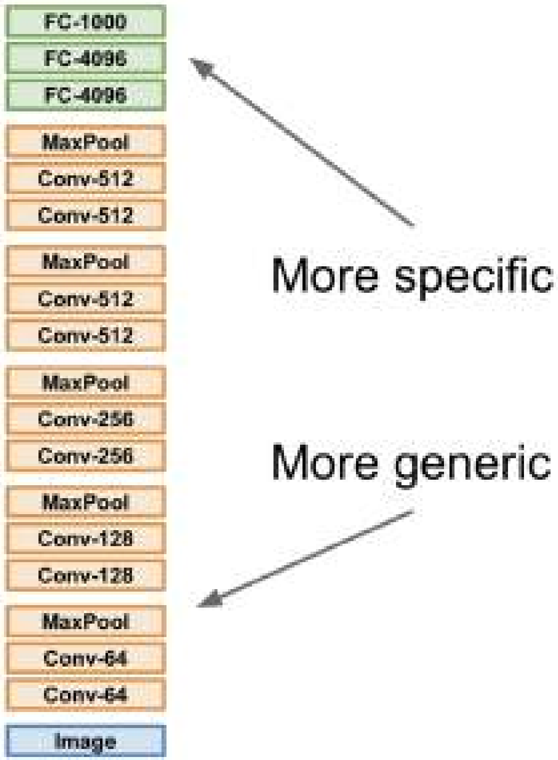
Transfer Learning with CNNs

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

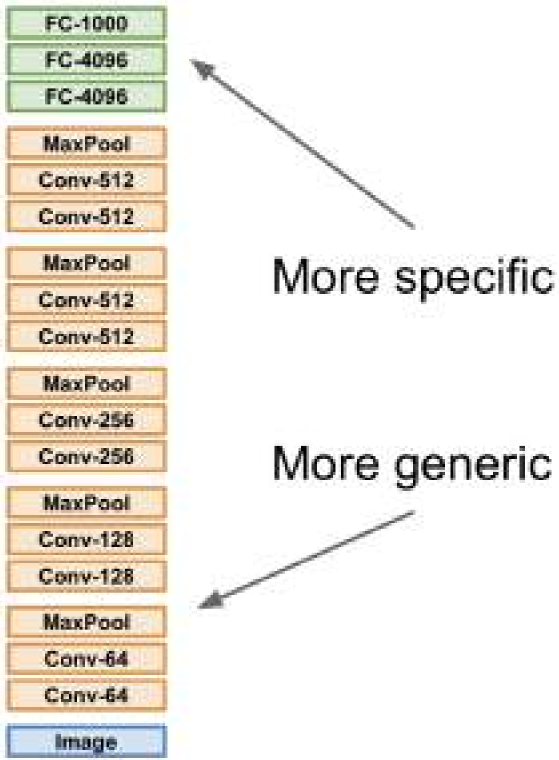




	very similar dataset	**very different dataset**
very little data	?	?
quite a lot of data	?	?



	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	?
quite a lot of data	Finetune a few layers	?



	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

Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)

Object Detection
(Fast R-CNN)

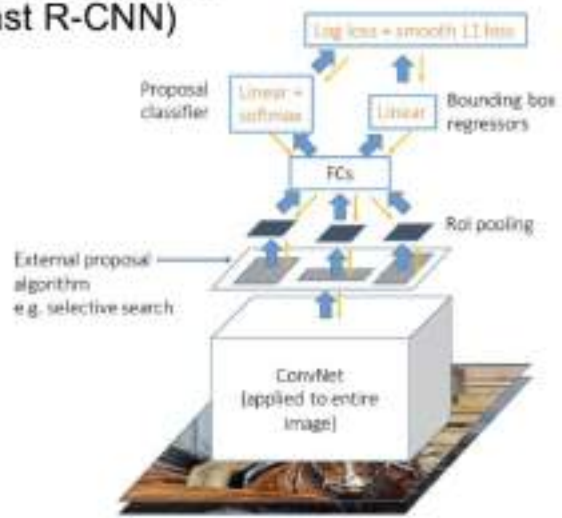
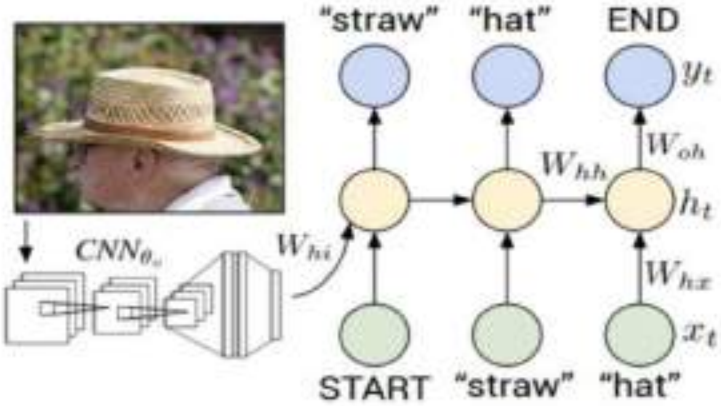


Image Captioning: CNN + RNN



Girshick, "Fast R-CNN", ICCV 2015

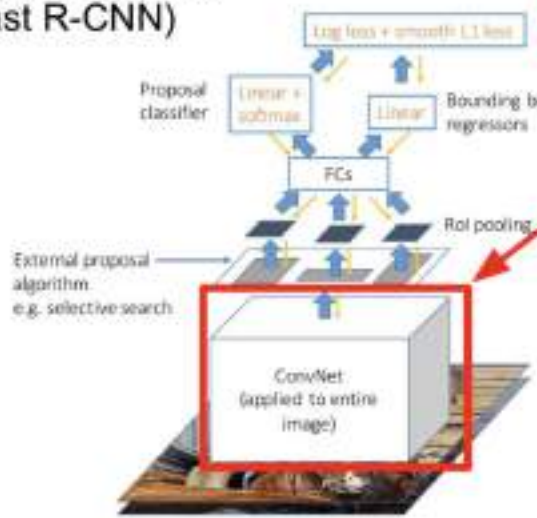
Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

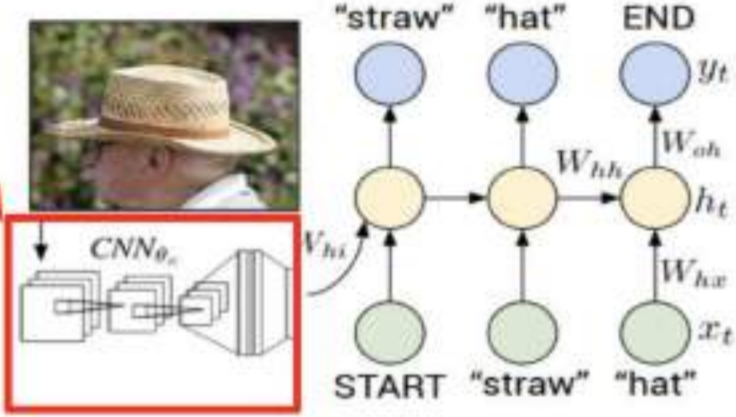
Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)

Object Detection
(Fast R-CNN)

CNN pretrained on ImageNet

Image Captioning: CNN + RNN



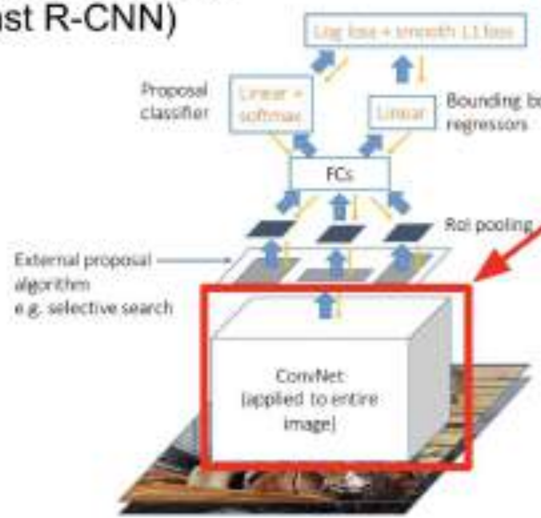


Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

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

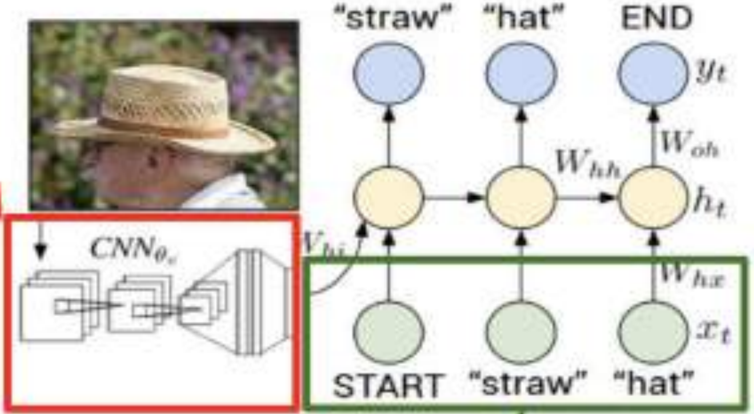
Transfer learning with CNNs is pervasive...
(it's the norm, not an exception)

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.

Takeaway for your projects and beyond:

Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big ConvNet there
2. Transfer learn to your dataset

Deep learning frameworks provide a "Model Zoo" of pretrained models so you don't need to train your own

Caffe: https://github.com/BVLC/caffe/wiki/Model-Zoo
TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision

Summary

- Optimization
  - Momentum, RMSProp, Adam, etc
- Regularization
  - Dropout, etc
- Transfer learning
  - Use this for your projects!

모두의연구소

박 은 수 Research Director

E-mail : es.park@modulabs.co.kr