Regularizing your neural network

Friday, June 1, 2018 1:38 PM

_ff76ae8a7ee526eb18cc9b3acd0cbc3f_C2W1L03

deeplearning.ai

Regularizing your neural network

Regularization

prevent over-fitting
reduce variation

we usually $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

Logistic regression

$w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

λ = regularization parameter
lambda lambd

$$\min_{w,b} J(w,b)$$

$$J(w,b) = \underbrace{\frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})} + \frac{\lambda}{2m}\underbrace{\|w\|_2^2}$$

$\cancel{+\frac{\lambda}{2m} b^2}$ omit

→ why doing this? w high dim
b. one param. won't make much
difference. Don't bother to
include it.

L_2 regularization $\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$ ←

L_1 regularization $\frac{\lambda}{2m}\sum_{j=1}^{n_x} |w_j| = \frac{\lambda}{2m}\|w\|_1$

w will be sparse

if not for compress of model
won't be very helpful.

m or 2m only a normalizing const

Andrew Ng

Neural network

$$J(w^{[1]}, b^{[1]}, \dots, w^{[L]}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{n} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \|w^{[l]}\|_F^2$$

$$\|w^{[l]}\|_F^2 = \sum_{i=1}^{n^{[l]}}\sum_{j=1}^{n^{[l-1]}} (w_{ij}^{[l]})^2$$

$w^{[l]}: (n^{[l]}, n^{[l-1]})$

"Frobenius norm"

$\|\cdot\|_2^2$ $\|\cdot\|_F^2$

$$dw^{[l]} = \boxed{(\text{from backprop}) + \frac{\lambda}{m} w^{[l]}}$$

$$\frac{\partial J}{\partial w^{[l]}} = dw^{[l]}$$

back propagation.

$$w^{[l]} := w^{[l]} - \alpha\, dw^{[l]}$$

"Weight decay"

$$w^{[l]} := w^{[l]} - \alpha\left[(\text{from backprop}) + \frac{\lambda}{m} w^{[l]}\right]$$

$$= w^{[l]} - \frac{\alpha\lambda}{m} w^{[l]} - \alpha\,(\text{from backprop})$$

$$= \underbrace{\left(1 - \frac{\alpha\lambda}{m}\right)}_{<1} w^{[l]} - \alpha\,(\text{from backprop})$$

make it smaller.

L2 regularization = weight decay

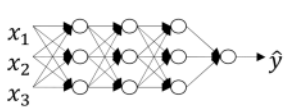
Andrew Ng

Neural network

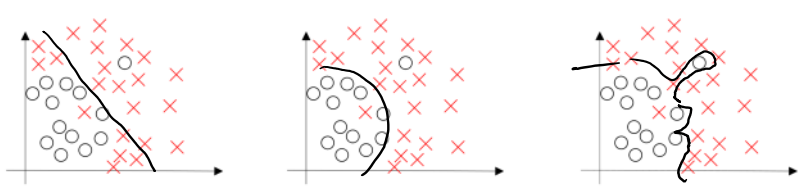
$$J(w^{[1]}, b^{[1]}, \dots, w^{[L]}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{n} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \|w^{[l]}\|^2$$

Andrew Ng

How does regularization prevent overfitting?



$$J(w^{[l]}, b^{[l]}) = \frac{1}{m}\sum_{i=1}^{n} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \|w^{[l]}\|_F^2$$

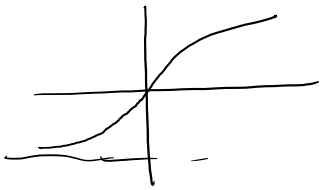


zero out lots of inputs of the hidden unit $\Rightarrow$ reduce impact of lots of hidden unit. $\Rightarrow$ $w^{[l]} = 0$.

not an accurate description though.

Andrew Ng

How does regularization prevent overfitting?



$\lambda \uparrow \Rightarrow w^{[l]} \downarrow \quad z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$

Every layer is roughly linear if regularized

$\Rightarrow$ simpler function

Andrew Ng

Implementation tip: add term to the loss function

plot cost function — gradient decent.

$\Rightarrow$ should $\downarrow$.

★ With regularization remember to use the new def of $J = \sum L(\,) + \boxed{\lambda / m \, \| w \|}$. remember to include this part in plotting.

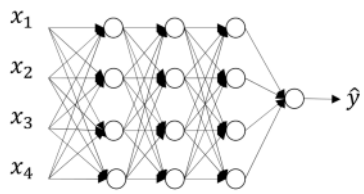
dropout

deeplearning.ai

Regularizing your neural network

Dropout regularization

Dropout regularization



0.5 0.5 0.5

Andrew Ng

Implementing dropout ("Inverted dropout")

Illustrate with layer $l=3$. keep-prob $= 0.8$ 0.2

$d3 = np.random.rand(a3.shape[0], a3.shape[1]) < keep\text{-}prob$

activation $a3 = np.multiply(a3, d3)$ # $a3 *= d3$.

$a3 /= \cancel{0.8}\ keep\text{-}prob$

true/false or 1/0.

50 units. ⇝ 10 units shut off example

$z^{[4]} = W^{[4]} \cdot a^{[3]} + b^{[4]}$

reduced by 20%

$/= 0.8$

scale it up.

Divide by keep-prob.

so it won't change the expected value of $z^{[4]}$

Andrew Ng

layer $l=3$

keep-prob. $=0.8$.

np.random.rand

$\mathbb{I}(<0.8)$

0.8 ⇒ prob 0.2 drop.

At test time, the inverted dropout is easier ⟶ has less scaling problem

Making predictions at test time

test time NO Drop out

$a^{[0]} = X$

No drop out.

$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

$a^{[1]} = g^{[1]}(z^{[1]})$

$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$

$a^{[2]} = \ldots$

$\hat{y}$

$/= keep\text{-}prob$ ⟶ the fact that we do this in training makes sure that there's no scaling problem even if we don't implement dropout at test time

Andrew Ng

deeplearning.ai

Regularizing your neural network

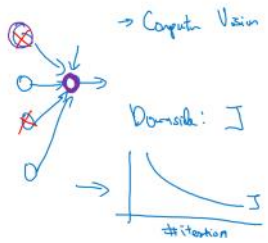
Understanding dropout

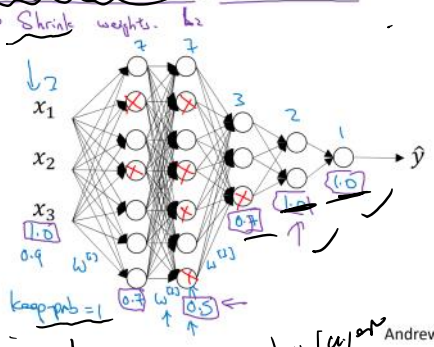
any feature could go away at random.
reluctant to put weight on any one particular input

Why does drop-out work?

Intuition: Can't rely on any one feature, so have to spread out weights. ⇝ Shrink weights. L_2

for each node toss a coin.
with some probability. remove it
"Diminished network"





Andrew Ng

$Z_3(12)$ 的 Z_3. param ...
⇒ keep-prob ?

more worry about overfitting ⇒ lower keep prop.

can vary keep-prop by layer

similar to playing with λ.

★ Occasionally, dropout on input feature
but this is not common practice

★ Many dropout has to do with CV
cuz so many input.

Unless the application has over-fitting problem ⇒ Don't use dropout

Another problem of Dropout: Loss function J not well defined.

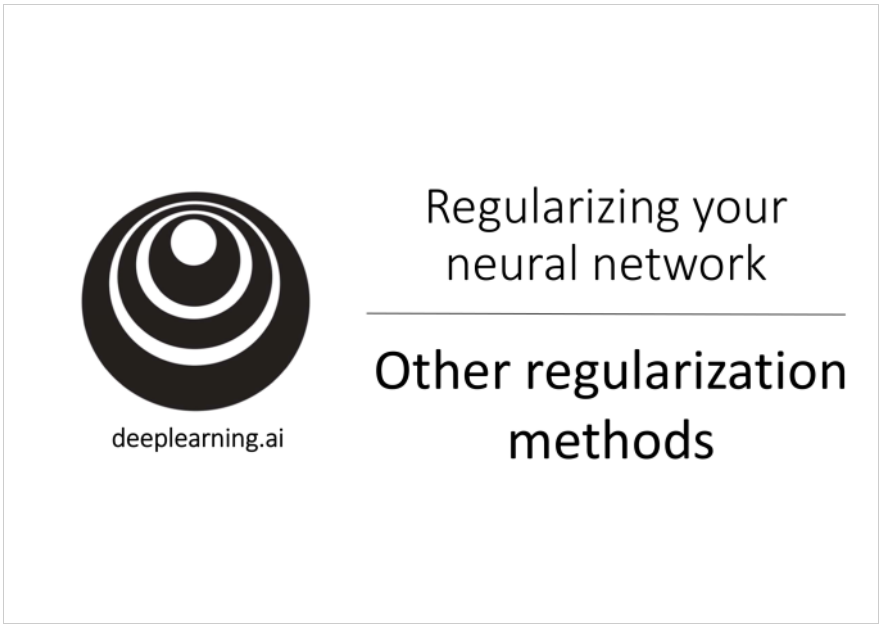
random J

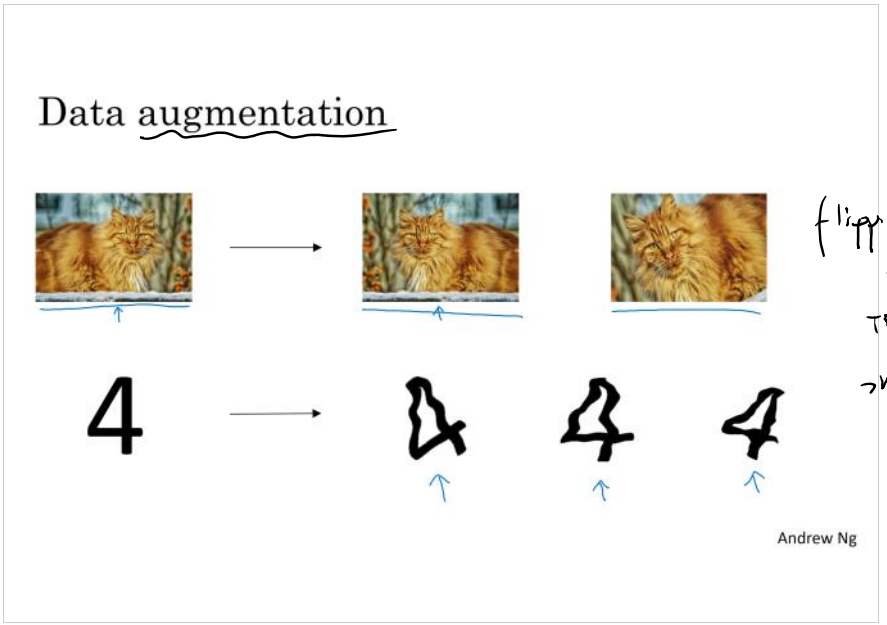
no longer monotonically decreasing.

(What to do?) Do it without dropout, if works then

⇒ ~~these~~ add dropout.

other regularization



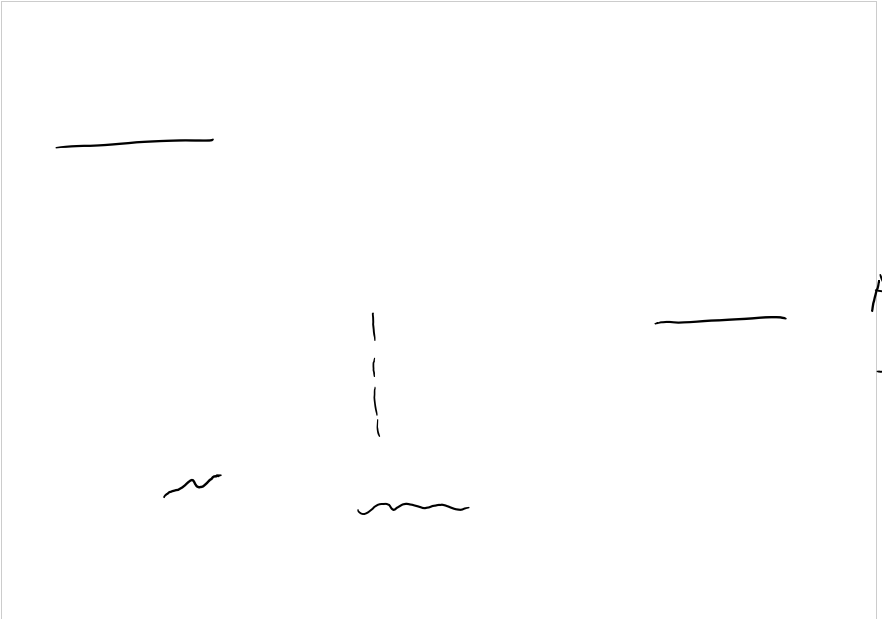


flipping images

transform

inexpensive way to get more data.

random rotation & distortion.



Also plot dev set error.

find: Dev set error ↓ => ↑.

stop halfway

Downside of early stopping [- Optimize J => J(w, b)
- avoid overfitting.] a different tool for this task
solo

Orthogonalization → 用同一个机制分别两个任务.

work on two tasks independently

不独立

→ alternative — L2 regularization

↳ Downside: computationally expensive

searching for λ.

will talk about it later.

7. With the inverted dropout technique, at test time:

- (selected) You do not apply dropout (do not randomly eliminate units), but keep the 1/keep_prob factor in the calculations used in training.

  This should not be selected

- You do not apply dropout (do not randomly eliminate units) and do not keep the 1/keep_prob factor in the calculations used in training
- You apply dropout (randomly eliminating units) and do not keep the 1/keep_prob factor in the calculations used in training
- You apply dropout (randomly eliminating units) but keep the 1/keep_prob factor in the calculations used in training.

4. You are working on an automated check-out kiosk for a supermarket, and are building a classifier for apples, bananas and oranges. Suppose your classifier obtains a training set error of 0.5%, and a dev set error of 7%. Which of the following are promising things to try to improve your classifier? (Check all that apply.)

- [x] Increase the regularization parameter lambda

Correct

- [ ] Decrease the regularization parameter lambda

Un-selected is correct

- [ ] Get more training data

This should be selected

- [ ] Use a bigger neural network

Un-selected is correct