

(Single-label) Classification, ANN

Sunday, September 25, 2016 10:50 PM

$x \Rightarrow \Phi$: need feature extraction $f(\cdot)$ and (linear) classifier c

Linear classifier c : assume feature representation z given

	2-class	K-class
Example of c	2-way softmax classifier	K-way softmax classifier
Loss function $J(\cdot)$	ideal: $\sum_n \mathbb{1}(y^{(n)} \neq \hat{y}^{(n)})$ approximation: binary cross-entropy	logistic regression multinomial CE
score function $s(\cdot)$	sigmoid	softmax
Decision $d(\cdot)$	$\hat{y}^{(n)} = \mathbb{1}(s(x^{(n)}) > 0.5)$	$\hat{y}^{(n)} = \arg \max_k s_k(x^{(n)})$

Learning: minimize $J(\cdot)$ with respect to each (W, b) by (stochastic) gradient descent

- Feature extractor $f(\cdot)$: z is actually not given
 - $z = f(x) \leftarrow f(\cdot) \equiv \text{NN}$
 - $z = f^{(2)}(f^{(1)}(x)) \leftarrow f^{(2)} \circ f^{(1)} \equiv \text{DNN}$
 - activation function $\sigma^{(l)}(\cdot)$
- \Rightarrow Learning: minimize $J(\cdot)$ w.r.t. each (W, b) by (stochastic) gradient descent

$\sigma^{(2)} = \text{sigmoid}$
 $\hat{y} = \text{ReLU}$
 $\hat{y} = K \cdot \text{ReLU}$

$z_1 = \sigma(W_{11}^{(1)}x + b_1^{(1)})$
 $z_2 = \sigma(W_{21}^{(1)}x + b_2^{(1)})$
 \vdots
 $z_{d_{\text{max}}} = \sigma(W_{d_{\text{max}}1}^{(1)}x + b_{d_{\text{max}}}^{(1)})$

$\hat{y} = \{W_{11}^{(1)}, b_{11}^{(1)}\}$
 $z = f(x) = \sigma^{(2)}(W^{(1)}x + b^{(1)})$

$S = c(\hat{z}) = c(f^{(2)}(f^{(1)}(x)))$
 $S = \sigma^{(2)}(W^{(2)}f^{(1)}(x) + b^{(2)})$

learning $\{W_{dk}^{(2)}\}^{D,K}$
 $\{b_k^{(2)}\}^{1,K}$

optim algo: SGD
 minibatch gradient descent
 GD

SGD : 1. Init $\{W_{dk}^{(0)}\}, \{b_k^{(0)}\}$
 2. Loop until convergence (e.g. $|J^{(i+1)} - J^{(i)}| < \epsilon$)

(b) $\hat{W}_{dk}^{(i+1)} \leftarrow \hat{W}_{dk}^{(i)} - \eta \frac{\partial J}{\partial W_{dk}} \bigg|_{W=W_{dk}^{(i)}} = \hat{W}_{dk}^{(i)} - \lambda \hat{W}_{dk}^{(i)}$
 $\hat{b}_k^{(i+1)} \leftarrow \hat{b}_k^{(i)} - \eta \frac{\partial J}{\partial b_k} \bigg|_{b=b_k^{(i)}} = \hat{b}_k^{(i)} - \lambda \hat{b}_k^{(i)}$

update expressions

- Nesterov Momentum: $+ \alpha, \mu$
 - Adam: $+ \alpha, \beta, \dots$
 - Ada grad: $+ \alpha, \beta$
 - RMS prop: $+ \alpha, \beta$
 - ...
- (adaptive learning rate methods)

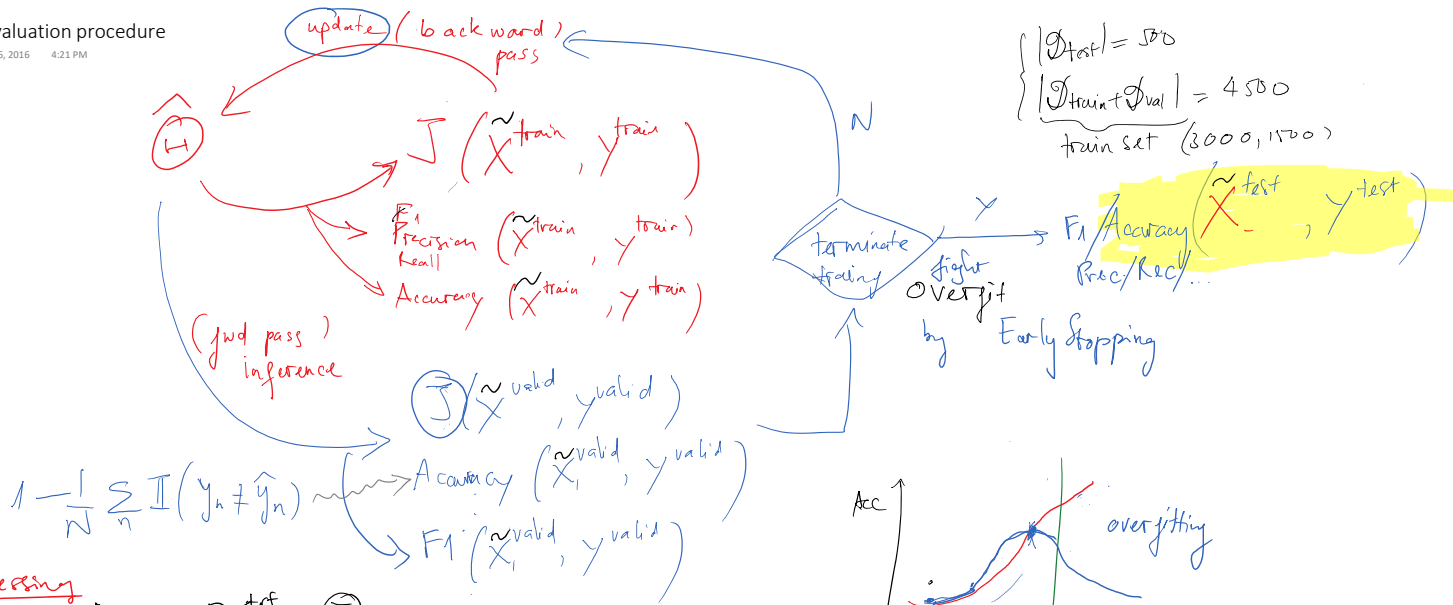
stochastic (MB)GD $N^{(1)} \ll N$
 e.g. $N^{(1)} = 100, 200$

$W^T x + b = \begin{bmatrix} W_1 \\ W_0 \end{bmatrix}^T \begin{bmatrix} x_1, \dots, x_D, 1 \end{bmatrix}$
 x

$\sigma(w, z) \equiv \text{sigmoid}$
 $\frac{\partial \sigma(w, z)}{\partial w_1} = z \sigma(1 - \sigma) |_{w=w_1}$ "back prop"

η : learning rate

$\frac{\partial J}{\partial W_{d1}^{(2)}} = \frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial z_1} \cdot \frac{\partial z_1}{\partial W_{d1}^{(2)}}$
 exploding / vanishing gradient



Data preprocessing

Normalization $X_n^{test} \rightarrow \tilde{X}_n^{test} = X_n^{test} - \bar{X}$

$c \times w \times h \rightarrow c \times \tilde{w} \times \tilde{h}$

$\bar{X} \leftarrow \{X^{train}\}$

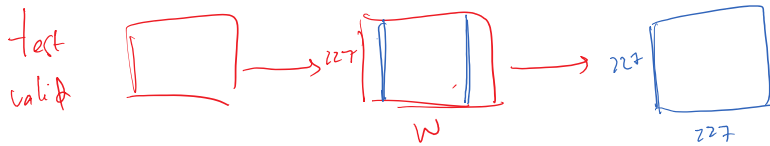
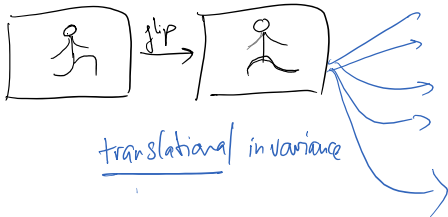
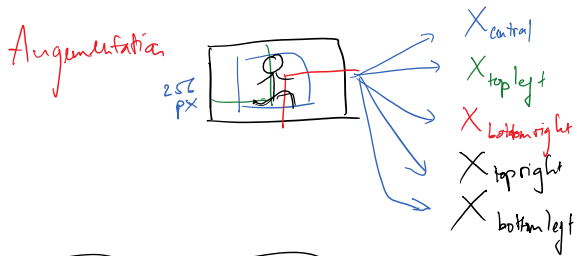
$w, h \in [0, 255]$

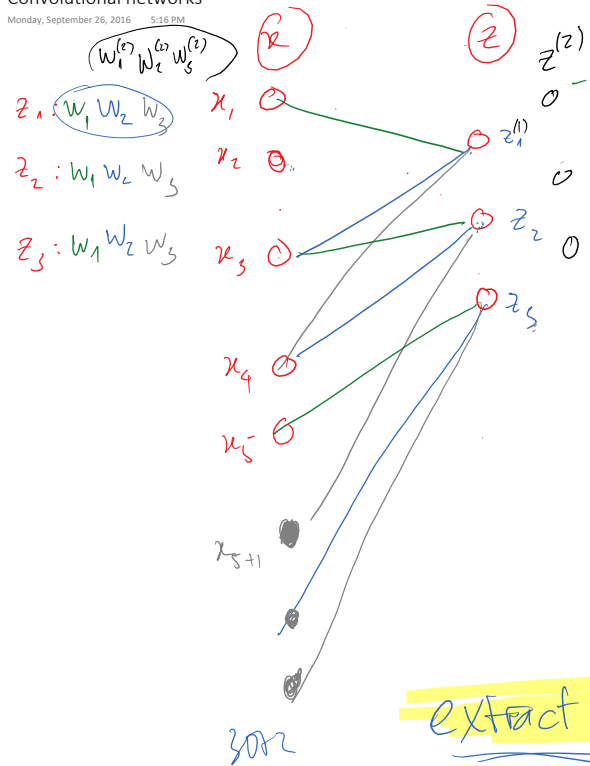
$\text{mean}(\tilde{w}) \approx 0$

$\text{mean}(\tilde{h}) \approx 0$

crop size: 227×227

Augmentation





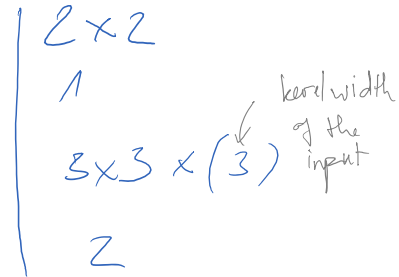
Convolutional layer

- + locally connected \gg fully-connected (FC)
- + weight sharing

+ hyperparam of the structure

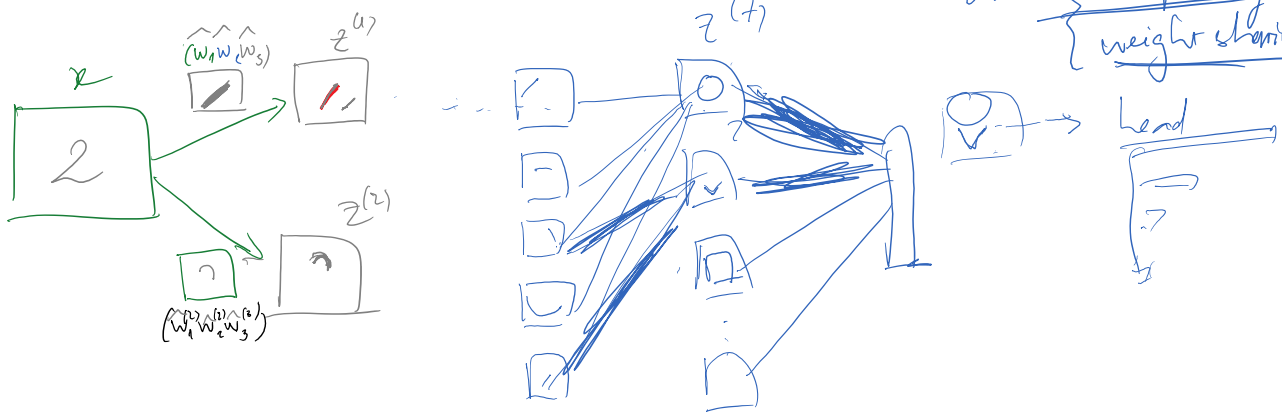
- stride: 2
- zero-padding: 3
- filter size: 3
- # filters (# kernels): 2

eg $2 \times 2 \times 3 \ln$



extract invariant features w.r.t translation

due to maxpooling
weight sharing

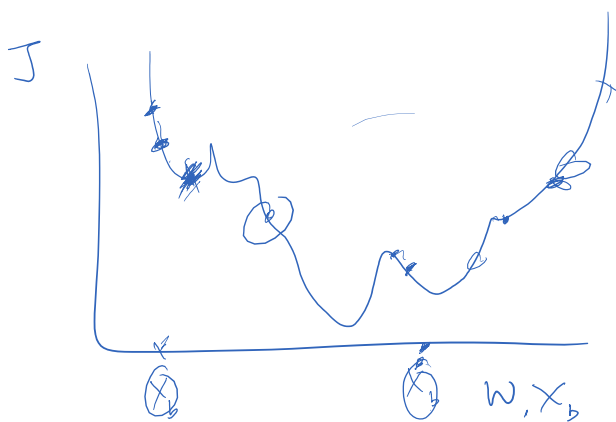


POOL layer

z_{conv}

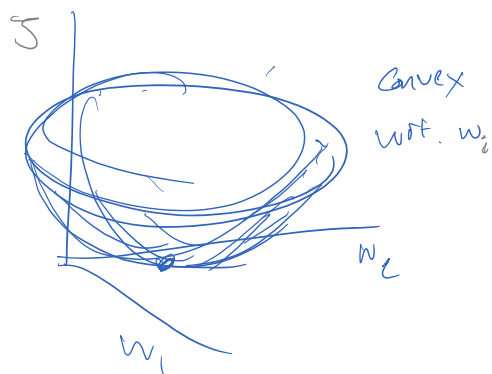
z_{pool}

SGD



Stochastic optian (w)

non-convex loss fn



Multi-label learning strategy

Monday, September 26, 2016 6:47 PM

