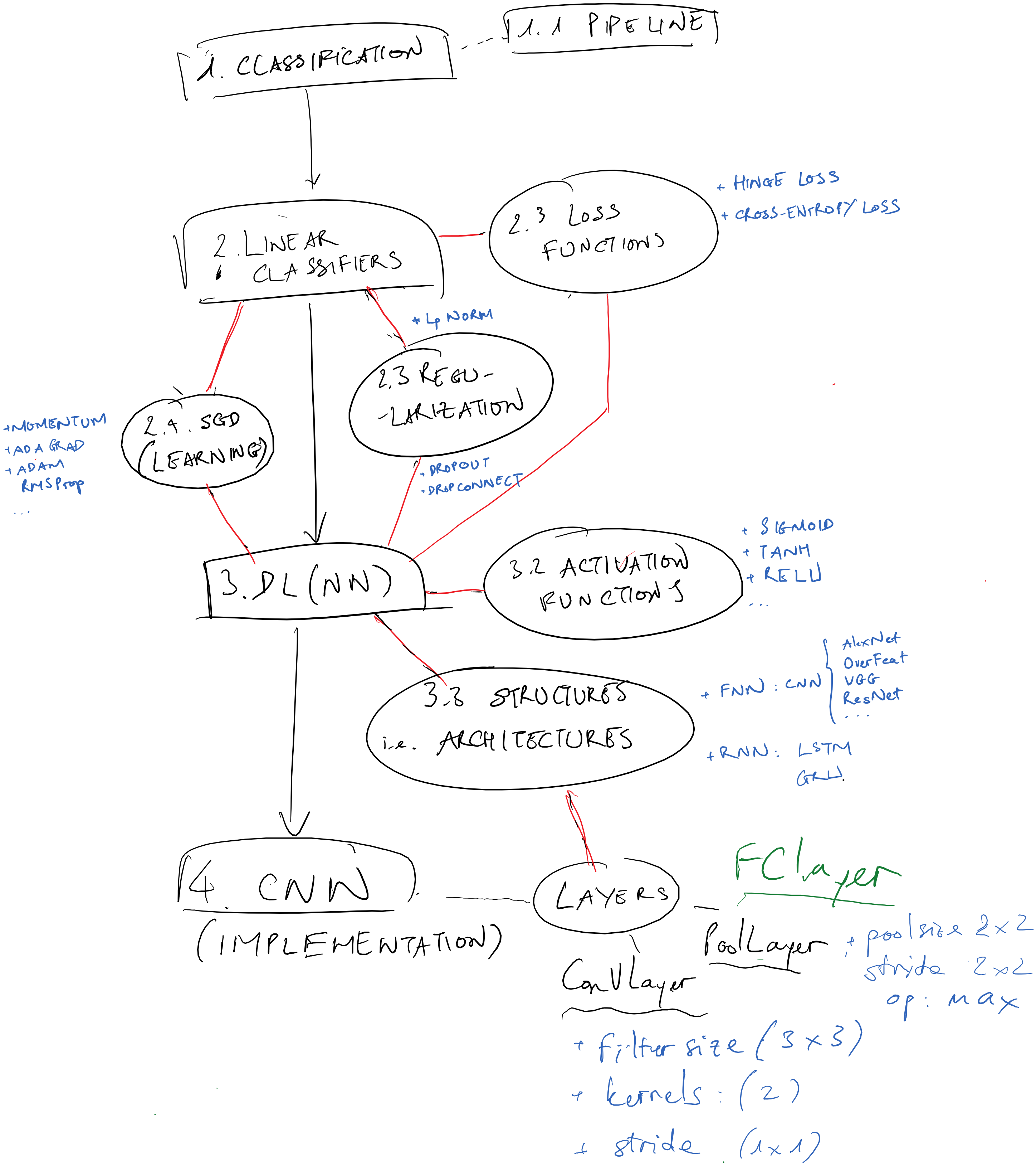
DL tutorial sketch

Thursday, March 24, 2016 10:01 AM

OVERVIEW



+ zero Padding (1)

(Single-label) Classification, ANN

Sunday, September 25, 2016 10:50 PM

x -> $\hat{y}$: need **feature extraction f(.)** and (linear) **classifier c**

- Linear classifier c: *assume feature representation z given*

	2-class	K-class
Example of c	2-way softmax classifier ≡ logistic regression	K-way softmax classifier
Loss function J(.)	ideal: $\sum_n \mathbb{I}(\hat{y}^{(n)} \neq y^{(n)})$	<---
- approximation	binary cross-entropy	multinomial CE
score function s(.)	sigmoid	softmax
Decision d(.)	$\hat{y}^{(n)} = \mathbb{I}(s(z^{(n)}) > 0.5)$	$\hat{y}^{(n)} = \underset{k}{\text{argmax}}\, s_k(z^{(n)})$

Learning: minimize J(.) with respect to each {W,b} by (stochastic) gradient descent

- Feature extractor f(.): z is actually not given
  - z = f(x) <--- f(.) ≡ NN
  - $z = f^{(2)}(f^{(1)}(x))$ <--- $f^{(2)} \circ f^{(1)}$ ≡ DNN
  - activation function $\sigma^{(l)}(\cdot)$

=> Learning: minimize J(.) w.r.t. each $\{W, b, W_{f(1)}, b_{f(1)}, W_{f(2)}, b_{f(2)}\}$

Single label

ANN

N^{224} $f(\cdot)$ $\mathbb{R}^{4096}$ c

$x^{(n)} \rightarrow z^{(n)} \rightarrow \hat{y}^{(n)} \rightarrow \{+,-\}$, or $\{1, 2, \ldots, K\}$

RGB feature

$900 \times 900 \times 3$ $f(x, \Theta)$

1-of-K representation

$y^{(n)} = 5 \rightarrow y^{(n)} = [0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$ ← 5th

$y^{(n)} = \text{cat} \rightarrow y^{(n)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ cat dog

multi-label

$y^{(n)} = [1\ 1\ 0\ 0\ 1]^T$

$\mathbb{I}(y_n \neq \hat{y}_n)$

binary cross-entropy

loss function argmin w

$J(y, z, w) = \sum_n y_n \log(\pi(z_n^{(1)})) + (1 - y_n) \log(1 - \pi(z_n^{(1)}))$ $\mathcal{D} = \{y_n\}_{n=1}^N$

$Y_n \sim \text{Bernoulli}(\pi)$ 2-class

$\boxed{\text{Score function}}$ ≡ inverse link function in GLM — Generalized Linear Model

K-class

$J(y, z, w) = \sum_n \sum_k y_{nk} \log \pi_k(z_n)$

multinomial CE

$Y_n \sim \text{Cat}(\pi = [\pi_1, \pi_2, \ldots, \pi_K])$

N^{224} K

$x \xrightarrow{f^{(1)}} z^{(1)} \xrightarrow{f^{(2)}} z^{(2)} \xrightarrow{s} s = \pi(z) = s(z)$

$s_1 = \sigma(w_1^{(3)} z_1 + w_2^{(3)} z_2 + \ldots + w_D^{(3)} z_D + b)$

$s_2 = \sigma(w_{12} z_1 + w_{22} z_2 + \ldots + w_{D,2} z_D + b_2)$

$s_K = \sigma(\underbrace{w_{1,K} z_1 + w_{2,K} z_2 + \ldots + w_{D,K} z_D + b_K}_{w_k^T z + b})$

K=10

(2-class)

$\hat{y} = +$ if $s > 0.5$

$\hat{y} = -$ otherwise

K-class

$\hat{y} = k$ if $s_k = \max_{k'} \{s_{k'}\}_{k'=1}^K$

$\hat{y} = w_1 z_1 + w_2 \log(z_2) + w_0$

"linear" w.r.t w, not z

$z_1 = \sigma(w_1^{(1)'} x + b_1^{(1)})$

$z_2 = \sigma(w_2^{(1)'} x + b_2^{(1)})$

$\vdots$

$z_{4096} = \sigma(w_{4096}^{(1)'} x + b_{4096}^{(1)})$

$\sigma^{(2)}$ = sigmoid = tanh = ReLU

$\Theta = \{W_{D_1 D_2}^{(1)}, b_{D_1}^{(1)}\}$

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

Deep Neural Network

2-class : $s(\alpha) = \frac{e^{w^T \alpha + b}}{1 + e^{w^T \alpha + b}}$ (sigmoid)

K-class : $s_k(\alpha) = \frac{e^{w_k^T \alpha + b_k}}{\sum_{k'} e^{w_{k'}^T \alpha + b_{k'}}}$ (softmax)

$s = c(z^{(2)}) = c(f^{(2)}(f^{(1)}(x)))$

$s = \sigma^{(3)}(W^{(3)} \sigma^{(2)}(W^{(2)} \sigma^{(1)}(W^{(1)} x + b^{(1)}) + b^{(2)}) + b^{(3)})$

learning $\{\hat{w}_{dk}\}^{D,K}$ $\{\hat{b}_k\}^K$

$w^T x + b = \underbrace{\begin{bmatrix} w_1 \\ w_0 \\ b \end{bmatrix}}_{w}^T \underbrace{[x_1\ x_2\ \ldots\ x_D\ 1]}_{x}$

Random init: Glorot Uniform, Glorot Normal : "gaussian", pretrained model

optim algo : SGD (minibatch gradient descent, GD)

SGD :

1, Init $\{\hat{w}_{dk}^{(0)}\}$, $\{\hat{b}_k^{(0)}\}$

2, Loop until convergence (eg. $|J^{(i+1)} - J^{(i)}| < \varepsilon$)

(b) $\hat{w}_{dk}^{(i+1)} \leftarrow \hat{w}_{dk}^{(i)} - \eta \frac{1}{N^{(i)}} \frac{\partial J}{\partial w_{dk}}\Big|_{w_{dk} = \hat{w}_{dk}^{(i)}} - \lambda \hat{w}_{dk}^{(i)}$

$\hat{b}_k^{(i+1)} \leftarrow \hat{b}_k^{(i)} - \eta \frac{1}{N^{(i)}} \frac{\partial J}{\partial b_k}\Big|_{b_k = \hat{b}_k^{(i)}} - \underbrace{\lambda b_k^{(i)}}_{\text{weight decay}}$

update expression

η : learning rate

$\sigma(w, z) \equiv$ sigmoid

$\frac{\partial \sigma(w, z)}{\partial w_d} = z_d \sigma(1 - \sigma)|_{w = w_0}$

"back prop"

$\frac{\partial J}{\partial w_{dj}^{(1)}} = \underbrace{\frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{dj}^{(1)}}}$

exploding / vanishing gradient

Stochastic (mB)GD $N^{(i)} << N$

eg : $N^{(i)} \simeq 100, 200$

Nesterov Momentum : $+\mu$

Adam : $+\alpha, \beta, \ldots$

Adagrad : $+\alpha, \beta$

RMSprop : $+\alpha, \beta$

...

(adaptive learning rate methods)

Learning-Evaluation procedure

Monday, September 26, 2016 4:21 PM

$|\mathcal{D}_{test}| = 500$

$\underbrace{|\mathcal{D}_{train} + \mathcal{D}_{val}|}_{\text{train set}} = 4500$ (3000, 1500)

update (backward pass)

$\hat{\Theta}$

$J(\tilde{X}^{train}, Y^{train})$

F1 Precision Recall $(\tilde{X}^{train}, Y^{train})$

Accuracy $(\tilde{X}^{train}, Y^{train})$

(fwd pass inference)

$J(\tilde{X}^{valid}, Y^{valid})$

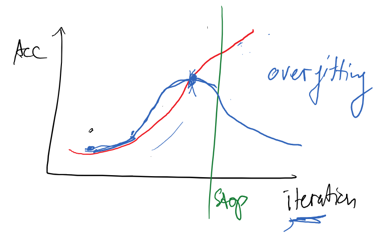
$1 - \frac{1}{N}\sum_n \mathbb{I}(y_n \neq \hat{y}_n)$ ⟶ Accuracy $(\tilde{X}^{valid}_{,}, Y^{valid})$

F1 $(\tilde{X}^{valid}_{,}, Y^{valid})$

terminate training — N / Y

fight overfit by Early Stopping

Y ⟶ F1 / Accuracy $(\tilde{X}^{test}, Y^{test})$ Prec/Rec/...



Data preprocessing

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

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

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

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

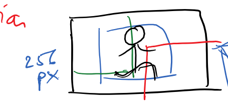
$\text{mean}(\tilde{w}) \approx 0$

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

cropsize: 227×227

X^{train}

Augmentation



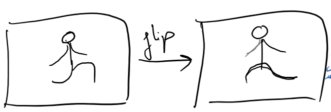
$X_{central}$

$X_{top\,left}$

$X_{bottom\,right}$

$X_{top\,right}$

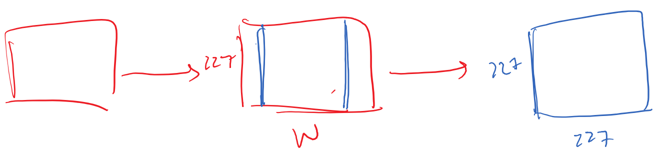
$X_{bottom\,left}$



translational invariance

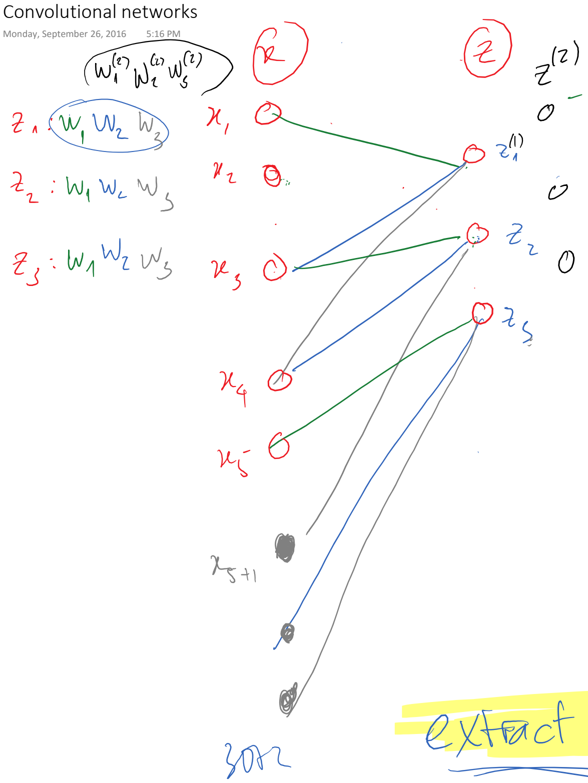
test

valid



Convolutional networks

Monday, September 26, 2016 5:16 PM



Convolutional layer

+ locally connected >< fully-connected (FC)
+ weight sharing
+ hyperparam of the structure

eg 2 (cs231n)

- stride	: 2	2×2
- zero-padding	: 3	1
- filter size	: 3	$3 \times 3 \times (3)$
- # filters (# kernels)	: 2	2

kernel width of the input

extract invariant features w.r.t translation

due to { maxpooling / weight sharing

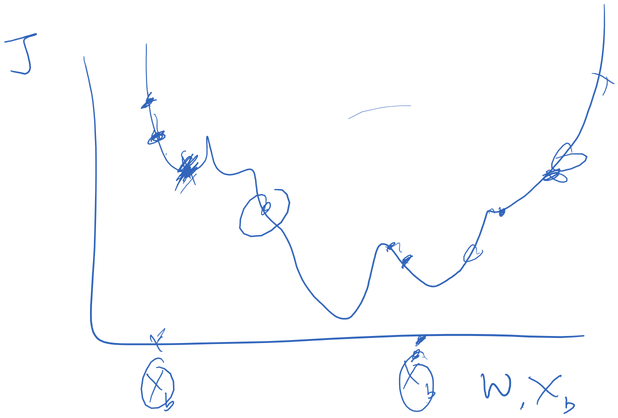
POOL layer

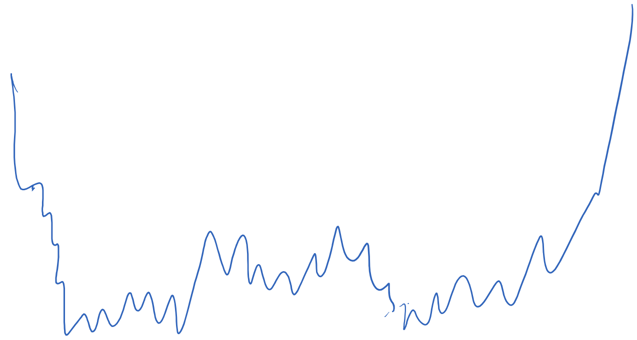
z_{conv} z_{pool}

stochastic optimization

Monday, September 26, 2016 6:33 PM

SGD

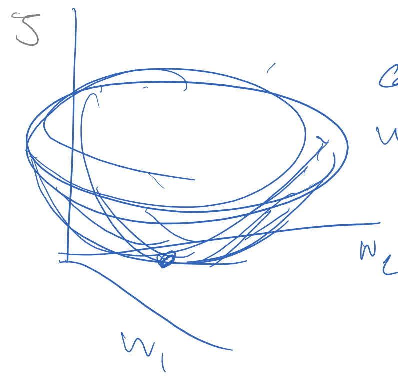




Stochastic optim

W

non-convex loss fn



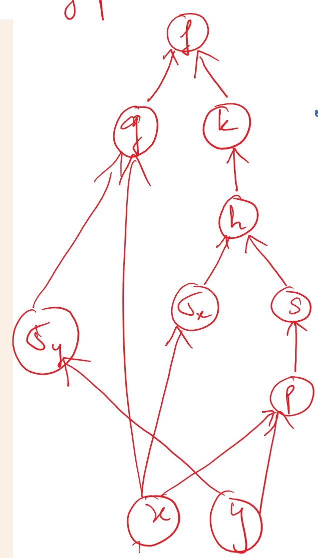
Convex wrt. w_i

backprop

Monday, May 2, 2016 5:55 PM

ALGO IS THE SAME FOR ALL f

STEP 1 have a variable dependency graph



ANALOGY TO FNN:

- $f(\cdot) \equiv$ loss function
- other variables $\equiv W, b$ at each layer

eg: compute $\left.\frac{\partial f}{\partial x}\right|_{x=x_o}$ and $\left.\frac{\partial f}{\partial x}\right|_{y=y_o}$ when $f(x,y) = \frac{x+\sigma(y)}{\sigma(x)+(x+y)^2}$

forward		backward	
		DONE	
1	$\sigma_y = \sigma(y) = \frac{1}{1+e^{-y}}$	8 (*)	$dy \leftarrow dy + d\sigma_y \cdot \frac{\partial \sigma_y}{\partial y} = dy + (1-\sigma_y)\sigma_y$
2	$g(x,\sigma_y) = x + \sigma_y$	7 (*)	$dx \leftarrow dx + dg \cdot \frac{\partial g}{\partial x} = dx + dg$ $d\sigma_y = \frac{\partial f}{\partial \sigma_y} = dg \cdot \frac{\partial g}{\partial x} = dg$
3 (*)	$\sigma_x = \sigma(x) = \frac{1}{1+e^{-x}}$	6 (*)	$dx \leftarrow dx + d\sigma_x \cdot \frac{\partial \sigma_x}{\partial x} = dx + (1-\sigma_x)\sigma_x$
4 (*)	$p(x,y) = x+y$	5 (*)	$dx = \frac{\partial f}{\partial x} \leftarrow dp \cdot \frac{\partial p}{\partial x} = dp$ $dy = \frac{\partial f}{\partial y} \leftarrow dp \cdot \frac{\partial p}{\partial y} = dp$
5	$s(p) = p^2$	4	$dp = \frac{\partial f}{\partial p} = ds \cdot \frac{\partial s}{\partial p} = ds \cdot 2p$
6	$h(\sigma_x, s) = \sigma_x + s$	3	$d\sigma_x = \frac{\partial f}{\partial \sigma_x} = dh \cdot \frac{\partial h}{\partial \sigma_x} = dh$ $ds = \frac{\partial f}{\partial s} = dh \cdot \frac{\partial h}{\partial s} = dh$
7	$k(h) = \frac{1}{h}$	2	$dh = \frac{\partial f}{\partial h} = dk \cdot \frac{\partial k}{\partial h} = dk \cdot \left(-\frac{1}{h^2}\right)$
8	$f(g,k) = g \cdot k$	1	$dg = \frac{\partial f}{\partial g} = k$ $dk = \frac{\partial f}{\partial k} = g$
DONE			

Note

Backprop intuitions: passing the messages
http://cs231n.github.io/optimization-2/

STEP 2 compute derivative $\frac{\partial f}{\partial \eta}$ (denoted as $d\eta$) w.r.t. direct child θ_o of η

$$\frac{\partial f}{\partial \eta} := \frac{\partial f}{\partial \theta_o} \cdot \frac{\partial \theta_o}{\partial \eta} \quad \text{i.e. } d\theta_o$$

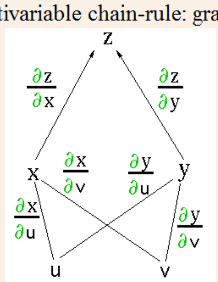
STEP 3: if a parent η has ≥ 2 children say $\theta_1, \theta_2, \ldots$, ADD UP

$$\frac{\partial f}{\partial \eta} \mathrel{+}= d\theta_1 \frac{\partial \theta_1}{\partial \eta}$$

$$\frac{\partial f}{\partial \eta} \mathrel{+}= d\theta_2 \frac{\partial \theta_2}{\partial \eta}$$

(multivariate calculus chain-rule: add up at fork (parent who has ≥ 2 children))

multivariable chain-rule: gradients add up at forks (*)



$$\frac{\partial z}{\partial u} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial u} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial u}$$

$$\frac{\partial z}{\partial v} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial v} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial v}$$

Multi-label learning strategy

Monday, September 26, 2016 6:47 PM

MLMC : OvA