IE4497 formula

$$\text{Gaussian Dist: } p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

$$\text{If } X \sim \mathcal{N}(\cdot|\mu,\sigma^2), \text{ then}$$

$$aX \sim \mathcal{N}(\cdot|a\mu, a^2\sigma^2); X + c \sim \mathcal{N}(\cdot|\mu+c,\sigma^2); \text{let } Z \sim \mathcal{N}(\cdot|\mu,\sigma^2), \text{ then} X = \sigma Z + \mu \sim \mathcal{N}(\cdot|\mu,\sigma^2)$$

$$\text{if X and Y} \sim \mathcal{N}(\cdot|0,v^2)\text{are independent, then } X+Y \sim \mathcal{N}(\cdot|0,\sigma^2+v^2)$$

$$\text{If} X \sim \mathcal{N}(\cdot|\mu,\sigma^2) Joint\ pdf: P((X,Y)\in A) = \int_A p(x,y)dxdy,\ \ p(x,y) = p(x)p(y|x)\ \ (= p(x)p(y)\ if\ independent)$$

$$p(x) = \int p(x,y)dy(marginal\ distribution),\ \ p(x|y) = \frac{p(x,y)}{p(y)} = \frac{p(y|x)p(x)}{p(y)}(Bayes'\ Theorem)$$

$$\frac{p(a|b,c)}{p(a|c)} = \frac{p(b|a,c)}{p(b|c)}, E[X] = \int xp(x)dx,$$

$$E[g(X)] = \int g(x)p(x)dx,\ E[X+Y] = E[X]+E[Y],\ E[X|Y=y] = \int xp(x|y)dx$$

$$E[E[X|Y]] = E[X],\ E[f(X)g(Y)|Y] = E[f(X)|Y]g(Y)$$

$$var(X) = E[(X-E[X])^2] = E[X^2]-E[X]^2,\ cov(X,Y) = E[(X-E[X])(Y-E[Y])]$$

$$var(X+Y) = var(X)+var(Y)+2cov(X,Y).\text{ If } X = AZ+\mu: E[x] = \mu, cov(X) = AA^T$$

$$E[Ax] = AE[x],\ cov(x) = \sum_{xx} = E[(x-E[x])(x-E[x])^T],\ cov(Ax) = A\ cov(x)A^T$$

$$cov(x,y) = \sum_{xy} = E[(x-E[x])(y-E[y])^T], \frac{\delta(a^Tx)}{\delta x} = a,\ \frac{\delta(x^TAx)}{\delta(x)} = (A+A^T)x,\ \frac{\delta a^TXb}{\delta X} = ab^T$$

$$\frac{\delta det(X)}{\delta X} = det(X)(X^{-1})^T,\ \frac{\delta a^TX^{-1}b}{\delta X} = -(X^{-1})^Tab^T(X^{-1})T$$

$$\underbrace{p(\theta|x)}_{posterior} = \frac{p(x,\theta)}{p(x)} = \frac{p(x|\theta)p(\theta)}{p(x)} \propto \underbrace{p(x|\theta)}_{likelihood}.\underbrace{p(\theta)}_{prior}$$

$$\text{Residual sum of squares } RSS = \sum_{i=1}^n (y_i-\hat{y}_i)^2, \text{Root mean squared error } RMSE = \sqrt{\frac{1}{n}RSS}.$$

$$\text{Coefficient of determination } R^2 = 1-\frac{RSS}{TSS} = 1-\frac{RSS}{\sum_{i=1}^n(y_i-\bar{y})^2},$$

$$\text{where } \bar{y} = \frac{1}{n}\sum_i y_i.\ \text{TSS is the total sum of squares or the empirical variance of the data.}$$

$$\text{- Binomial distribution: } Bin(x|\theta,n) = \binom{n}{x}\theta^x(1-\theta)^{n-x}, \binom{n}{x} = \frac{n!}{x!(n-x)!}$$

$$\text{- Beta distribution: } Beta(x|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}x^{a-1}(1-x)^{b-1}\ \text{(*Beta dist is a conjugate prior to Bin dist)}$$

$$\text{- Categorical distribution: } Cat(x|\theta_1,\ldots,\theta_K) = \theta_x$$

$$\text{- Multinomial distribution: } Mult(x|\theta,n) = \binom{n}{x_1,\ldots,x_K}\prod_{k=1}^K\theta_k^{x_k}, \binom{n}{x_1,\ldots,x_K} = \frac{n!}{x_1!\cdots x_K!}.$$

$$\text{- Dirichlet distribution: } Dir(x|\alpha_1,\ldots,\alpha_K) = \frac{\Gamma(\sum_{k=1}^K\alpha_k)}{\prod_{k=1}^K\Gamma(\alpha_k)}\prod_{k=1}^K x_k^{\alpha_k-1},\ \text{(*Dirich dist is a conjugate prior to Multi Nor dist)}$$

$$\text{- Joint Gaussian distribution: } N(x|\mu,\Sigma) = \frac{1}{(2\pi)^{K/2}(\det\Sigma)^{1/2}}exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

$$\text{- Uniform distribution: } Unif(x|a,b) = \begin{cases}\frac{1}{b-a} & \text{if } a\le x\le b,\\ 0 & \text{otherwise.}\end{cases}, \text{- Exponential distribution: } Exp(x|\lambda) = \begin{cases}\lambda e^{-\lambda x} & \text{if } x\ge 0,\\ 0 & \text{otherwise.}\end{cases}$$

$$\text{Bernoulli distribution:} p(D|\theta) = \theta^{N_1}(1-\theta)^{N_0}, \mathrm{N}_k = \sum_{i=1}^n 1\{x_i = k\}.\text{ Exponential distribution:} p(D|\lambda) = \lambda^n\exp(-\lambda\sum_{i=1}^n x_i)$$

$$\text{Normal distribution:} p(D|\theta) = N(\mu,\sigma^2)^n = \frac{1}{(2\pi)^{n/2}\sigma^n}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^n(x_i-\mu)^2\right)$$

$$\text{Linear model:} y = w^\top x+\epsilon, \text{Basis function expansion:} y = w^\top\phi(x)+\epsilon, \text{MLE of linear model:} w_{ML} = (\Phi^\top\Phi)^{-1}\Phi^\top y$$

$$\text{Kullback-Leibler divergence:} D(p_{\theta_0}\|p_\theta) = E_{\theta_0}\left[\log\frac{p(x|\theta_0)}{p(x|\theta)}\right], \text{MAP estimate:} \theta_{MAP} = \max_\theta p(\theta|D)$$

$$\Big[\quad p(x|\theta) \Big] \qquad \theta$$

$$\text{MMSE estimate:}\hat{\theta}(D) = \min_a E[(\theta - a)^2|D] = E[\theta|D]$$

$$\text{Mixture model:} \quad p(x|\theta) = \sum_{k=1}^{K} \pi[k] p_k(x|\theta), \text{GMM:} \quad p(x|\theta) = \sum_{k=1}^{K} \pi[k] \mathcal{N}(x|\mu_k, \Sigma_k), \quad \theta = (\pi[k], \mu_k, \Sigma_k)_{k=1}^{K}$$

$$\text{E step:} \quad Q(\theta|\theta^{(m)}) = E_{y\sim p(\cdot|x,\theta^{(m)})} \left[\log p(y|\theta)|x, \theta^{(m)} \right], \text{M step:} \quad \theta^{(m+1)} = \arg\max_{\theta\in\Theta} Q(\theta|\theta^{(m)})$$

$$\text{GMM E step:} \quad r_{ik}^{(m)} = p(z_i = k|x_i, \theta^{(m)}) = \frac{\pi^{(m)}[k]\mathcal{N}(x_i|\mu_k^{(m)}, \Sigma_k^{(m)})}{\sum_{k'} \pi^{(m)}[k']\mathcal{N}(x_i|\mu_{k'}^{(m)}, \Sigma_{k'}^{(m)})}, n_k^{(m)} = \sum_{i=1}^{n} r_{ik}^{(m)},$$

$$Q(\theta|\theta^{(m)}) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \|x_i - \mu_{k_i}^{(m)}\|^2 + const.$$

$$\text{GMM M step:} \quad \pi^{(m+1)}[k] = n_k^{(m)}/n, \mu_k^{(m+1)} = (1/n_k^{(m)}) \sum_{i=1}^{n} r_{ik}^{(m)} x_i; \;\; \Sigma_k^{(m+1)} = (1/n_k^{(m)}) \sum_{i=1}^{n} r_{ik}^{(m)} (x_i - \mu_k^{(m+1)})(x_i - \mu_k^{(m+1)})^\top.$$

$$\text{MAP estimate:} \quad \theta^{\text{MAP}} = \arg\max_{\theta\in\Theta}(\log p(x|\theta) + logp(\theta)). \;\; \text{EM for MAP:}Estep: Q(\theta|\theta^{(m)}) = E_{y\sim p(\cdot|x,\theta^{(m)})} \left[\log p(y|\theta)|x, \theta^{(m)} \right]$$

$$Mstep: \theta^{(m+1)} = argmax\theta \in \theta(Q(\theta|\theta(m)) + logp(\theta))$$

$$\text{Markov property: } p(x_t|x_1, \ldots, x_{t-1}) = p(x_t|x_{t-1}), \text{Transition probability: } T(i,j) = p_{x_t|x_{t-1}}(j|i)$$

$$\text{Unigram model: } p(x_t = x), \text{Bigram model: } p(x_t|x_{t-1}), \text{n-gram model: } p(x_t|x_{t-1}, x_{t-2}, \ldots, x_{t-n+1})$$

$$\text{PageRank score: } \pi_i = \sum_j T(j,i)\pi_j$$

$$\text{MLE for Markov model: } \log p(D|\pi, T) = \sum_{i=1}^{n} \log \pi(x_i, 0) + \sum_{i=1}^{n} \sum_{t=1}^{t_i} \log T(x_{i,t-1}, x_{i,t})$$

$$= \sum_{x=1}^{M} N_x \log \pi(x) + \sum_{x=1}^{M} \sum_{y=1}^{M} N_{xy} \log T(x,y)$$

$$N_x = \sum_{i=1}^{n} \mathbb{I}\{x_{i,0} = x\}, N_{xy} = \sum_{i=1}^{n} \sum_{t=1}^{t_i} \mathbb{I}\{x_{i,t-1} = x, x_{i,t} = y\}. \hat{\pi}(x) = \frac{N_x}{n}, \hat{T}(x,y) = \frac{N_{xy}}{\sum_z N_{xz}},$$

$$\text{HMM: } p(x_0, \ldots, x_T, z_0, \ldots, z_T|\theta) = \pi(z_0)p(x_0|z_0) \prod_{t=1}^{T} T(z_{t-1}, z_t)p(x_t|z_t)$$

$$\text{Baum-Welch algorithm: MLE - } \log p(D|\theta) = \sum_{i=1}^{n} \log \pi(z_{i,0}) + \sum_{i=1}^{n} \sum_{t=1}^{t_i} \log T(z_{i,t-1}, z_{i,t}) + \sum_{i=1}^{n} \sum_{t=0}^{t_i} \log p(x_{i,t}|\phi_{z_{i,t}})$$

Steps for BW Algo: (1)Initialize $\theta^{(0)}$. (2)E step: At iteration m, use Forward-Backward Algorithm to compute

$$\gamma_{i,t}(z) = p(z_{i,t} = z|x_{i,\cdot}, \theta^{(m)}) \propto \alpha_j(z)\beta_j(z), \;\; \xi_{i,t}(z, z') = p(z_{i,t-1} = z, z_{i,t} = z'|x_{i,\cdot}, \theta^{(m)})$$

$$\propto \alpha_{t-1}(z)p(x_{i,t}|z_{i,t} = z')\beta_t(z')p(z_{i,t} = z'|z_{i,t-1} = z). \;\; \text{(3)M step: Find } (m+1).$$

$$\pi^*(z) = \frac{\sum_{i=1}^{n} \gamma_{i,0}(z)}{n}, \;\; T^*(z, z') = \frac{\sum_{i=1}^{n} \sum_{t=1}^{t_i} \xi_{i,t}(z, z')}{\sum_u \sum_{i=1}^{n} \sum_{t=1}^{t_i} \xi_{i,t}(z, u)}, \hat{\phi}_z = \text{emission prob. model parameters.}$$

$$\text{Viterbi algorithm: } z_0^*, \ldots, z_T^* = \arg\max_{z_0,\ldots,z_T} p(z_0, \ldots, z_T|x_0, \ldots, x_T)$$

$$\text{Cumulative distribution function:} P(X \le x) = F(x), X = F^{-1}(U), \text{where } U \sim \text{Unif}([0,1]), u \le F(x) \Leftrightarrow F^{-1}(u) \le x$$

$$\text{Transformation method:} p_Y(y) = \sum_{k=1}^{K} \frac{p_X(x_k)}{|f'(x_k)|}, \text{where } x_1, x_2, \ldots, x_K \text{ are solutions to } f(x) = y$$

$$\text{Rejection sampling:} p(z) = \frac{1}{M}\tilde{p}(z), \text{where } M \text{ is unknown}, kq(z) \ge \tilde{p}(z) \text{ for all } z$$

$$\text{Accept } z \sim q(z) \text{ if } u \sim \text{Unif}([0, kq(z)]) \le \tilde{p}(z)$$

$$\text{Acceptance probability: } P(z \text{ accepted}) = \int P(z \text{ accepted}|z)q(z)dz = \int \frac{\tilde{p}(z)}{kq(z)} q(z)dz = \frac{M}{k}$$

$$\text{Rejection sampling for Bayesian Inference: } \tilde{p}(\theta) = p(D|\theta)p(\theta) \text{ and } q(\theta) = p(\theta) : k = \max_\theta \frac{\tilde{p}(\theta)}{q(\theta)} = \max_\theta p(D|\theta)$$

Importance sampling: sample z where $|f(z)|p(z)$ is large for better efficiency rather than from $p(z)$ directly.

$$E_p[f(z)] = E_q \left[\frac{p(z)}{q(z)} f(z) \right], \;\; \tilde{w} = \frac{p(z)}{q(z)}, w(z) = \frac{\tilde{w}(z)}{\sum_{i=1}^{n} \tilde{w}(z_i)}, \;\; E_p[f(z)] \approx \sum_{i=1}^{n} w(z_i)f(z_i)$$

$$\text{Tail sampling: } P(X > a) \approx \sum_{i=1}^{n} w(z_i), \text{where } z_1, z_2, \ldots \text{ are sampled from } q(z) \text{ with support } (a, \infty) \text{ and } w(z_i) = \frac{p(z_i)}{q(z_i)}$$

Sampling importance resampling (SIR): 1.Sample $z_1, \ldots, z_n$ from $q(z)$.

2.Compute weights $w(z_1), \ldots, w(z_n)$ where $w(z_i) = \dfrac{\tilde{w}(z_i)}{\sum_{j=1}^n \tilde{w}(z_j)}$.

3.Resample with replacement from $\{z_1, \ldots, z_n\}$ according to weights $(w(z_1), \ldots, w(z_n))$.

SIR for Bayesian inference: $w(z_i) = \dfrac{p(D|z_i)}{\sum_{j=1}^n p(D|z_j)}$; Sampling for EM: $Q(\theta|\theta^{(m)}) \approx \dfrac{1}{n}\sum_{i=1}^n \log p(x, z_i|\theta)$

Stationary distribution: $\sum_x \pi(x)T(x,y) = \pi(y)$, Reversible MC: $\pi(x)T(x,y) = \pi(y)T(y,x)$

Metropolis-Hastings algorithm: (1)Initialize $x = Z_0$. (2)For each $m = 1, 2, \ldots$ (3)Sample $y \sim q(x,y)$.

(4)Compute acceptance probability $A(x,y) = \min\left(1, \dfrac{\tilde{\pi}(y)q(y,x)}{\tilde{\pi}(x)q(x,y)}\right)$.

if $\pi(x) \propto \psi(x)h(x)$ and $q(x,y) = h(y)$, then $A(x,y) = \min\left(1, \dfrac{\psi(y)}{\psi(x)}\right)$.

(5)With probability $A(x,y)$, set $Z_m = y$; otherwise set $Z_m = x$ (6)Update $x = Z_m$.

- random walk MH: $q(x,y) = q(y-x)$. $y - x \sim \mathcal{N}(\cdot|0, \Sigma)$ - Gaussian centered at x.

$y - x \sim \text{Unif}[-\delta, \delta]^d$ - Uniform distribution centered at x.

Gibbs sampling: $p(z_i|z_{-i}) = \dfrac{p(z_1, \ldots, z_d)}{p(z_{-i})}$. For each i, let $z_{-i} = \{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_d\}$, i.e., z_i removed.

(1)Initialize $(z_1^{(0)}, \ldots, z_d^{(0)})$. (2)For each k : (3)sample $z_1^{(k)}$ from $p(\cdot|z_2^{(k-1)}, \ldots, z_d^{(k-1)})$

(4)sample $z_2^{(k)}$ from $p(\cdot|z_1^{(k)}, z_3^{(k-1)}, \ldots, z_d^{(k-1)})$ ... (5)sample $z_j^{(k)}$ from $p(\cdot|z_1^{(k)}, \ldots, z_{j-1}^{(k)}, z_{j+1}^{(k-1)}, \ldots, z_d^{(k-1)})$ ...

(6)sample $z_d^{(k)}$ from $p(\cdot|z_1^{(k)}, \ldots, z_{d-1}^{(k)})$

Ising model: Given a noisy image y, want to recover z.compute the posterior p(z | y).

$$Likelihood: p(y \mid z) = \prod_j p(y_j \mid z_j) = \prod_j \mathcal{N}(y_j \mid z_j, \sigma^2)$$

$p(z_j \mid z_{-j}) \propto \prod_{s \in N_j} \psi(z_s, z_j)$, where $N_j$is the neighorhood of pixel z_j, and

$\psi(u,v) = \exp(Juv)$with J $>$ 0 as the"coupling strength".

$(N + 2P - F) / \text{stride} + 1$

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

each filter has 5*5*3 + 1 = 76 params

=> 76*10 = **760**

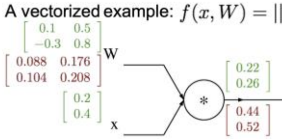
add gate: gradient distributor

mul gate: "swap multiplier"

copy gate: gradient adder

max gate: gradient router

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



$\nabla_W f = 2q \cdot x^T$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

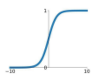
$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i} x_j$$

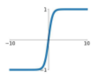
$$\frac{\partial f}{\partial W_{i,j}} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} = \sum_k (2q_k)(\mathbf{1}_{k=i} x_j) = 2q_i x_j$$

Activation Functions

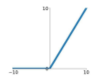
Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$



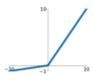
tanh $\tanh(x)$



ReLU $\max(0, x)$

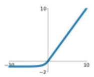


Leaky ReLU $\max(0.1x, x)$



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

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



- *Classification tasks*

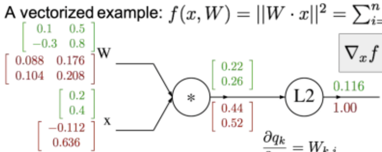
Training examples	Pairs of N inputs x_i and ground-truth class labels y_i	
Output Layer	Softmax Activations [maps to a probability distribution]	$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^T \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^T \mathbf{w}_k}}$
Loss function	*Cross-entropy*	$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\left[y_k^{(i)} \log \hat{y}_k^{(i)} + (1 - y_k^{(i)}) \log(1 - \hat{y}_k^{(i)})\right]$

Ground-truth class labels y_i and model predicted class labels $\hat{y}_i$

Regression tasks

Training examples	Pairs of N inputs x_i and ground-truth output values y_i	
Output Layer	Linear (Identity) or Sigmoid Activation	
Loss function	*Mean Squared Error*	$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$
	Mean Absolute Error	$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\lvert y^{(i)} - \hat{y}^{(i)}\rvert$

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



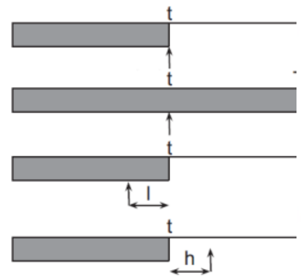
$\nabla_x f = 2W^T \cdot q$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

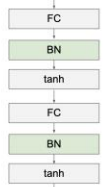
$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

$$\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i} = \sum_k 2q_k W_{k,i}$$



Batch Normalization

[Ioffe and Szegedy, 2015]



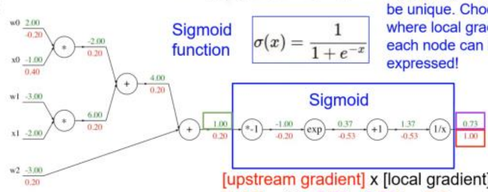
- Makes deep networks **much** easier to train!
- Improves gradient flow
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as regularization during training
- Zero overhead at test-time: can be fused with conv!
- Behaves differently during training and testing: this is a very common source of bugs!

Another example (cont'd):

$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

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

Computational graph representation may not be unique. Choose one where local gradients at each node can be easily expressed!



[upstream gradient] x [local gradient]

[1.00] x [(1 - 0.73) (0.73)] = 0.2

Sigmoid local gradient: $\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1 + e^{-x} - 1}{1 + e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1 - \sigma(x))\sigma(x)$

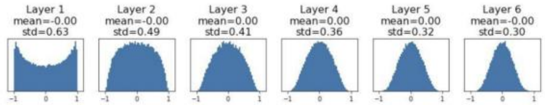
Weight Initialization: "Xavier" Initialization

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = np.random.randn(Din, Dout) / np.sqrt(Din)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

"Xavier" initialization: std = 1/sqrt(Din)

"Just right": Activations are nicely scaled for all layers!

For conv layers, Din is filter_size² * input_channels



- *Adaptive Moment Estimation (Adam)*
  - Adam combines insights from the momentum optimizers that accumulate the values of past gradients, and it also introduces new terms based on the second moment of the gradient
    - Similar to GD with momentum, Adam computes a **weighted average of past gradients** (first moment of the gradient), i.e., $V^t = \beta_1 V^{t-1} + (1 - \beta_1)\nabla\mathcal{L}(\theta^{t-1})$
    - Adam also computes a **weighted average of past squared gradients** (second moment of the gradient), , i.e., $U^t = \beta_2 U^{t-1} + (1 - \beta_2)(\nabla\mathcal{L}(\theta^{t-1}))^2$
  - The parameter update is: $\theta^t = \theta^{t-1} - \alpha \frac{\hat{V}^t}{\sqrt{\hat{U}^t + \epsilon}}$
    - Where: $\hat{V}^t = \frac{V^t}{1-\beta_1}$ and $\hat{U}^t = \frac{U^t}{1-\beta_2}$
    - The proposed default values are $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$
- Other commonly used optimization methods include:
  - Adagrad, Adadelta, RMSprop, Nadam, etc.
  - Most commonly used optimizers nowadays are Adam and SGD with momentum

AlexNet showed that you can use CNNs to train Computer Vision models.

ZFNet, **VGG** shows that bigger networks work better

GoogLeNet is one of the first to focus on efficiency using 1x1 bottleneck convolutions and global avg pool instead of FC layers

ResNet showed us how to train extremely deep networks

- Limited only by GPU & memory!
- Showed diminishing returns as networks got bigger

After ResNet: CNNs were better than the human metric and focus shifted to Efficient networks:

- Lots of tiny networks aimed at mobile devices: **MobileNet**, **ShuffleNet**

Neural Architecture Search can now automate architecture design

Backprop with Matrices

x: [N×D]
$\begin{bmatrix} 2 & \boxed{1} & -3 \\ -3 & 4 & 2 \end{bmatrix}$

w: [D×M]
$\begin{bmatrix} 3 & 2 & 1 & -1 \\ 2 & 1 & \boxed{3} & 2 \\ 3 & 2 & 1 & -2 \end{bmatrix}$

Matrix Multiply $y_{n,m} = \sum_d x_{n,d} w_{d,m}$

y: [N×M]
$\begin{bmatrix} \boxed{13 \quad 9 \quad -2 \quad -6} \\ 5 \quad 2 \quad 17 \quad 1 \end{bmatrix}$

dL/dy: [N×M]
$\begin{bmatrix} \boxed{2 \quad 3 \quad -3 \quad 9} \\ -8 \quad 1 \quad 4 \quad 6 \end{bmatrix}$

By similar logic:

[N×D] [N×M] [M×D]
$\frac{\partial L}{\partial x} = \left(\frac{\partial L}{\partial y}\right) w^T$

[D×M] [D×N] [N×M]
$\frac{\partial L}{\partial w} = x^T \left(\frac{\partial L}{\partial y}\right)$

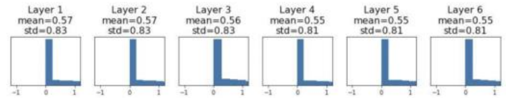
These formulas are easy to remember: they are the only way to make shapes match up!

Weight Initialization: He Initialization

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = np.random.randn(Din, Dout) * np.sqrt(2/Din)
    x = np.maximum(0, x.dot(W))
    hs.append(x)
```

ReLU correction: std = sqrt(2 / Din)

"Just right": Activations are nicely scaled for all layers!



Batch Normalization

[Ioffe and Szegedy, 2015]

Input: $x : N \times D$

Learnable scale and shift parameters:
$\gamma, \beta : D$

Learning $\gamma = \sigma$, $\beta = \mu$ will recover the identity function!

$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$ Per-channel mean, shape is D

$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$ Per-channel var, shape is D

$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$ Normalized x, Shape is N x D

$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$ Output, Shape is N x D