value function: $v_\pi : S \to \mathbb{R}$, $v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$

Bellman equation:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$$
$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$$
$$= \sum_a \pi(a|s) \sum_{s',r} \sum p(s', r | s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']]$$
$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a)[r + \gamma v_\pi(s')], \text{ for all } s \in S$$

- Action-value function: $q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$
- Greedy policy: $\pi'(s) \doteq \arg\max_a q_\pi(s,a) = \arg\max_a [r(s,a) + \gamma v_\pi(p(s,a))]$
- Discounted function: $G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

TD(0): $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$ (choose A_t acc. to policy)

SARSA: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$

Q-learning: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$ (greedily choose a)

Expected reward $J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}[r(\tau)] = \int_\tau r(\tau) p(\tau;\theta) d\tau$

REINFORCE trick: $\nabla_\theta J(\theta) = \int_\tau r(\tau) \nabla_\theta p(\tau;\theta) d\tau = \int_\tau r(\tau) p(\tau;\theta) \frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} d\tau$

$= \int_\tau r(\tau) p(\tau;\theta) \nabla_\theta (\log p(\tau;\theta)) d\tau$ (RL)

we can estimate $\nabla_\theta J(\theta) = \int_\tau r(\tau) p(\tau;\theta) \nabla_\theta \log p(\tau;\theta) d\tau$

$= \mathbb{E}_{\tau \sim p(\tau;\theta)}[r(\tau) \nabla_\theta \log p(\tau;\theta)]$ with MonteCarlo sampling

independence from transition probabilities can be seen as below

$$p(\tau;\theta) = \prod_{t \geq 0} p(s_{t+1} | s_t, a_t) \pi_\theta(a_t | s_t)$$

$$\log p(\tau;\theta) = \sum_{t \geq 0} \log p(s_{t+1} | s_t, a_t) + \log \pi_\theta(a_t | s_t)$$

$$\nabla_\theta \log p(\tau;\theta) = \sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t | s_t)$$

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} A^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t), \quad A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$$

- bootstrapping: not waiting for the final outcome
- both sarsa and q-learning use ε-greedy methods
- actor-critic combines policy gradients and q-learning
  actor: policy, critic: q-function → tells how good the actor's decides which action to take was and how actor should adjust

TRPO (trust region policy opt.): DRL of old and new policies, hard constraint, idea is to control the step length (don't change too much)
PPO: (proximal policy opt.): TRPO with 1st order optimization and soft constraint, we can do better using clipped objective rather than DRL
DDPG: (deep deterministic policy gradients): actor-critic off-policy DRL, combines deep q-learning and DPG, random noise when selecting a action

AR models: $p(x)$ is tractable, easy to train, easy (but slower) to sample, no natural latent variable repr (but doable: CRWN, STCN)
- major downside is sequential (slow) generation
- work for both cont. and disc. data (GANs hard to learn disc.)
- training more stable than GANs
- because we train via NLL, we have a direct measure for comparison. This also makes it straightforward to apply in domains such as compression and prob. planning/exploration

Dynamic programming
+ exact methods
+ Guaranteed to converge (certain iters.)
- need to know transition matrix
+ easy to implement
- need to iterate over whole state space
- requires memory prop. to size of statespace
- value iteration typically more efficient
- policy eval expensive as it may be also iterative over state-space
- bootstraps, do not sample
- uses value of neighboring states to update current state (given π)

Monte Carlo sampling (RL)
+ unbiased estimate
+ no need to know system dynamics (experience-based)
- high variance
- exploration/exploitation dilemma
- need terminating state
- slow for long episodes (mc must wait until the final outcome to update)
+ easy to focus on small subset of states
- first-visit and every-visit MC both converge to $v_\pi(s)$ as visits → infty
- does not bootstrap, samples

Temporal differences
+ less variance than MC due to bootstrap
+ more sample efficient
+ no need to know prob. matrix
- biased due to bootstrapping
- exploration/exploitation dilemma
- can behave poorly in stochastic envs
- combination of DP and MC ideas
- bootstraps and samples
- exploitation may lead to local minima

Policy optimization
- directly optimizes desired quality
- more compatible with modern ML (RNNs recurrence)
- more versatile and flexible
- more compatible with aux. objectives

Dynamic programming
- indirect, exploits hidden structure and self-consistency
- more sample efficient (when they work)
- more compatible with off-policy and exploration

Issues in DRL research
- Explexp dilemma: agents can get stuck in local minima (exploit behaviour that is irreversible)
- Reward hacking: ill shaped reward may lead to undesirable behavior (needs careful design)
- Domain knowledge: even small implem. differences may have a big impact
- Sample efficiency: require enormous amount of samples, if we update and improve policies we collect better data and throw away the rest

Softmax: $\text{softmax}(a)_k = \frac{e^{a_k}}{\sum_i^K e^{a_i}} \to \hat{y}_k = \frac{e^{a_k}}{\sum_j^K e^{a_j}}$ (softmax grad)

$$\frac{\partial \hat{y}_k}{\partial a_i} = \frac{\partial}{\partial a_i}\left(\frac{e^{a_k}}{\sum_j^K e^{a_j}}\right) = \frac{-e^{a_i} \cdot e^{a_k}}{(\sum_j^K e^{a_j})^2} = -\hat{y}_i \cdot \hat{y}_k \quad \} \text{ (for } k \neq i\text{)}$$

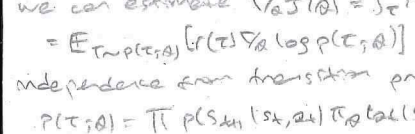
$$\frac{\partial \hat{y}_k}{\partial a_i} = \frac{e^{a_i} \sum_j^K e^{a_j} - e^{a_i} e^{a_i}}{(\sum_j^K e^{a_j})^2} = \frac{e^{a_i}(\sum_j^K e^{a_j} - e^{a_i})}{(\sum_j^K e^{a_j})^2}$$
$$= \frac{e^{a_i}}{\sum_j^K e^{a_j}} \cdot \left(\frac{\sum_j^K e^{a_j}}{\sum_j^K e^{a_i}} - \frac{e^{a_i}}{\sum_j^K e^{a_j}}\right) = \hat{y}_i(1 - \hat{y}_i) \quad \} \text{ (for } k = i\text{)}$$

$\delta_i^{(l)}$: gradient respect to the ith unit in the l layer
$\delta^{(l)}$: gradient that flows from layer $l+1$ to l

$$\delta_i^{(l)} = \frac{\partial C}{\partial z_i^{(l)}} = \sum_{j=1}^N \frac{\partial C}{\partial z_j^{(l+1)}} \frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}} = \sum_{j=1}^N \delta_j^{(l+1)} \frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}}$$

$$\delta^{(l)} = \frac{\partial C}{\partial z^{(l)}} = \sum_{j=1}^N \delta_j^{(l+1)} \frac{\partial z_j^{(l+1)}}{\partial z^{(l)}} = \frac{\partial C}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial z^{(l)}}$$

$z_j^{(l+1)} = \sigma(\theta_j^{(l+1)}, z^{(l)})$ $\quad \leftrightarrow \quad z^{(l+1)} = \sigma(\theta^{(l+1)}, z^{(l)})$
$\langle 1 \rangle \quad \langle 1, M \rangle \quad \langle M, 1 \rangle \qquad \langle N, 1 \rangle \quad \langle N, M \rangle \quad \langle M, 1 \rangle$



Backprop

$$\frac{\partial C}{\partial \theta_i^{(l)}} = \frac{\partial C}{\partial z_i^{(l)}} \frac{\partial z_i^{(l)}}{\partial \theta_i^{(l)}} \quad \text{(ith unit in the l layer)}$$

$$\frac{\partial C}{\partial \theta^{(l)}} = \sum_{i=1}^N \frac{\partial C}{\partial z_i^{(l)}} \frac{\partial z_i^{(l)}}{\partial \theta^{(l)}} = \sum_{i=1}^N \delta_i^{(l)} \frac{\partial z_i^{(l)}}{\partial \theta^{(l)}}$$

$\tau = 1$: $\begin{bmatrix} \alpha_1' & \cdots & \alpha_m' \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$ $\tau = 2$: $\begin{bmatrix} 0 & \cdots & 0 \\ \alpha_1^2 & \cdots & \alpha_m^2 \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$ $\Big\} = \begin{bmatrix} \alpha_1' & \cdots & \alpha_m' \\ \vdots & & \vdots \\ \alpha_n^\wedge & \cdots & \alpha_m^\wedge \end{bmatrix}$

$$z^{(l)} = w^{(l)} + z_{i,j}^{(l-1)} + b^{(l)} = \sum_m \sum_n w_{m,n}^{(l)} z_{i-m,j-n}^{(l-1)} + b^{(l)} \quad \text{(fwd pass)}$$

$$\delta_{i,j}^{(l-1)} = \frac{\partial C}{\partial z_{i,j}^{(l-1)}} = \sum_{i'} \sum_{j'} \frac{\partial C}{\partial z_{i',j'}^{(l)}} \frac{\partial z_{i',j'}^{(l)}}{\partial z_{i,j}^{(l-1)}} \quad \text{(bwd wrt inputs)}$$

CNN backprop

$$= \sum_{i'} \sum_{j'} \delta_{i',j'}^{(l)} \frac{\partial}{\partial z_{i,j}^{(l-1)}} \sum_m \sum_n w_{m,n}^{(l)} z_{i-m,j-n}^{(l-1)} + b$$

from above

$$= \sum_{i'} \sum_{j'} \delta_{i',j'}^{(l)} w_{m,n}^{(l)} \quad \text{(change of vars: } \tau = i' - m \to m = i' - \tau,\ j = j' - n \to n = j' - j\text{)}$$

$$= \sum_{i'} \sum_{j'} \delta_{i',j'}^{(l)} w_{[-i,j']}^{(l)} = \delta^{(l)} * w_{-\tau,-\tau} = \delta^{(l)} * \text{rot}_{180}(w_{i,j}^{(l)})$$

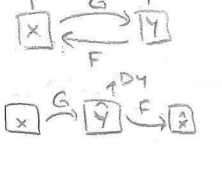
$$\frac{\partial C}{\partial w_{m,n}^{(l)}} = \sum_i \sum_j \frac{\partial C}{\partial z_{i,j}^{(l)}} \frac{\partial z_{i,j}^{(l)}}{\partial w_{m,n}^{(l)}} = \sum_i \sum_j \delta_{i,j}^{(l)} \frac{\partial}{\partial w_{m,n}^{(l)}} \left(\sum_m \sum_n w_{m,n}^{(l)} z_{i-m,j-n}^{(l-1)} + b^{(l)}\right) \quad \text{(weight update}$$

$$= \sum_i \sum_j \delta_{i,j}^{(l)} z_{i-m,j-n}^{(l-1)} = \delta^{(l)} * z_{-m,-n}^{(l-1)} = \delta^{(l)} * \text{rot}_{180}(z_{m,n}^{(l-1)})$$

Pix2pix: $h(G,D) = h_{cGAN}(G,D) + \lambda h_L(G)$
- $h_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[1 - \log D(x, G(x,z))]$
- $h_L(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1]$

CycleGAN: $h(G, F, D_X, D_Y)$
$= h_{GAN}(G, D_Y, X, Y) + h_{GAN}(F, D_X, Y, X) + \lambda h_{cyc}(G,F)$
$h_{cyc} = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1]$

CycleGAN:



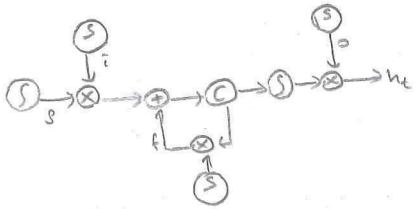
- Vanilla RNN: $h^t = \tanh(W_{hh}h^{t-1} + W_{xh}x^t)$, $y^t = W_{hy}h^t$

in matrix form $\begin{cases} h_t^l = \tanh W^l \begin{bmatrix} h_t^{l-1} \\ h_{t-1}^l \end{bmatrix} \end{cases}$ **RNN**

- LSTM: $h_t^l = o \odot \tanh c_t^l$, $c_t^l = f \odot c_{t-1}^l + i \odot g$

where gates are $\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{bmatrix} W^l \begin{bmatrix} h_t^{l-1} \\ h_{t-1}^l \end{bmatrix}$

- representing recurrence
  1- $h^t = g^t(x^{t-1}, \dots, x^1)$ has var. seq.len.
  2- $h^t = f(h^{t-1}, x^t; \theta)$ same params and transition func for all, one model for all timesteps + (also some input seq)



BPTT: $h^t = f(h^{t-1}, x^t; W)$, $\hat{y}^t = W_{hy}h^t$, $L^t = \|\hat{y}^t - y^t\|^2$, $\frac{\partial L}{\partial W} = \sum_{t=1}^{S} \frac{\partial L^t}{\partial W}$

$\frac{\partial L^t}{\partial W} = \sum_{k=1}^{t} \frac{\partial L^t}{\partial y^t} \frac{\partial y^t}{\partial h^t} \frac{\partial h^t}{\partial h^k} \frac{\partial^+ h^k}{\partial W}$ **BPTT**

$\frac{\partial h^t}{\partial h^k} = \prod_{i=k+1}^{t} \frac{\partial h^i}{\partial h^{i-1}} = \prod_{i=k+1}^{t} W_{hh}^T \operatorname{diag}[f'(h^{i-1})]$

$\forall i: \left\| \frac{\partial h^i}{\partial h^{i-1}} \right\| \le \|W_{hh}^T\| \cdot \|\operatorname{diag}[f'(h^{i-1})]\| < \frac{1}{\gamma} \gamma < 1$

$\left\| \frac{\partial h^t}{\partial h^k} \right\| < (\eta)^{t-k}$ where $\left\| \frac{\partial h^i}{\partial h^{i-1}} \right\| \le \eta < 1$

λ_1 is the largest (abs) singular value of W_{hh}

- if $\lambda_1 < 1/\gamma$ → vanishing
- if $\lambda_1 > 1/\gamma$ → exploding

Grad in the early layers is the prod. of terms from all the later layers

KL-divergence: $D_{KL}(p\|q) = \int_x \log \frac{p(x)}{q(x)} dx$

JS-divergence: $D_{JS}(p\|q) = \frac{1}{2} D_{KL}\left(p \,\|\, \frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q \,\|\, \frac{p+q}{2}\right)$ **DKL**

Generative models → Direct: GAN
↳ Implicit density → Markov chain: GSN
↳ Explicit density → Tractable: Belief nets
↳ NADE, MADE, PixelRNN/CNN
↳ Approximate → variational: VAE
→ Markov chain: Boltzmann mach.

VAE data likelihood: $p_\theta(x) = \int p_\theta(x|z) p_\theta(z) dz$

- we approx $p_\theta(z|x)$ through $q_\phi(z|x)$ **VAE**

ELBO: $\log p_\theta(x^{(i)}) = \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log(p_\theta(x^{(i)}))]$

$= \mathbb{E}_z\left[\log \frac{p_\theta(x^{(i)}|z) p_\theta(z)}{p_\theta(z|x^{(i)})} \frac{q_\phi(z|x^{(i)})}{q_\phi(z|x^{(i)})}\right]$

$= \mathbb{E}_z[\log p_\theta(x^{(i)}|z)] - \mathbb{E}_z\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z|x^{(i)})}\right]$

$= \underbrace{\mathbb{E}_z[\log p_\theta(x^{(i)}|z)] - D_{KL}(q_\phi(z|x^{(i)}) \| p_\theta(z))}_{} + D_{KL}(q_\phi(z|x^{(i)}) \| p_\theta(z|x^{(i)}))$

ELBO: $L = \mathbb{E}_{z\sim q_\phi}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x^{(i)})}\right]$, training: $\theta^*, \phi^* = \operatorname{argmax} \sum_i^N L(x^{(i)}; \theta, \phi)$

Gradient computation:

$\nabla_{\theta,\phi} \mathbb{E}_{z\sim q_\phi}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x^{(i)})}\right] = \nabla_{\theta,\phi} \mathbb{E}_{\epsilon\sim N(0,1)}\left[\log \frac{p_\theta(x, f(x,\epsilon,\theta))}{q_\phi(f(x,\epsilon,\phi)|x)}\right]$

$= \mathbb{E}_\epsilon\left[\nabla_{\theta,\phi} \log \frac{p_\theta(x,f)}{q_\phi(f|x)}\right] \simeq \frac{1}{k} \sum_{i=1}^{k} \nabla_{\theta,\phi} \log w(x,f,\theta,\phi)$

Reparametrization: $z \sim N(\mu, \sigma^2)$ → value changes every sample
- assume $\exists$ underlying r.v. $\epsilon \sim N(0,1)$, $z = \mu + \sigma\epsilon$, $z = f(x,\epsilon,\theta)$
that's not a func of par θ
- we can now take deriv. wrt μ, σ. it tells us how an infinitesimal change in μ and σ affects the output if we keep ϵ fixed

- Generator $G: \mathbb{R}^n \to X$
- Discriminator $D: X \to [0,1]$

-ve explicitly chang variables by replacing $G(z)$ with x, this will never hold in practice

consider only D:

$L(D) = -\frac{1}{2}(\mathbb{E}_{x\sim p_d} \log D(x)) + \mathbb{E}_{z\sim p_z} \log(1 - D(G(z)))$

$= -\frac{1}{2}(\mathbb{E}_{x\sim p_d} \log D(x)) + \mathbb{E}_{x\sim p_g} \log(1 - D(x))$ will fool any discriminator D

$V(G,D) = \mathbb{E}_{x\sim p_d} \log D(x) + \mathbb{E}_{x\sim p_g} \log(1 - D(x))$ $G^* = \operatorname{argmin}_G \max_D V(G,D)$

Let $D^*_G = \operatorname{argmax}_D V(G,D)$, then objective becomes $G^* = \operatorname{argmin}_G V(G, D^*_G)$

we give value func. $V(G,D) = \int_x p_d(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx$

↳ $f(y) = a \log y + b \log(1-y) \to f'(y) = 0 \Rightarrow y = a/(a+b)$, $f''(y = a/(a+b)) < 0$ for $a,b \in (0,1)$ means $D^*_G = p_d/(p_d + p_g) \Leftrightarrow p_d = p_g$, optimum is maximum

D^*_G is unique but can't be calculated in practice

Finding the best G: plug D^*_G in objective:

$V(D^*_G, G) = \mathbb{E}_{x\sim p_d}\left[\log \frac{p_d}{p_d+p_g}\right] + \mathbb{E}_{x\sim p_g}\left[\log\left(1 - \frac{p_d}{p_d+p_g}\right)\right]$

$= \mathbb{E}_{x\sim p_d}\left[\log \frac{p_d}{p_d+p_g}\right] + \mathbb{E}_{x\sim p_g}\left[\log \frac{p_g}{p_d+p_g}\right]$

GAN

$= -\log 2 + \mathbb{E}_{x\sim p_d}\left[\log \frac{2p_d}{p_d+p_g}\right] - \log 2 + \mathbb{E}_{x\sim p_g}\left[\log \frac{2p_g}{p_d+p_g}\right]$

$= -\log 4 + D_{KL}\left(p_d \,\|\, \frac{p_d+p_g}{2}\right) + D_{KL}\left(p_g \,\|\, \frac{p_d+p_g}{2}\right)$

$= -\log 4 + 2D_{JS}(p_d \| p_g)$

→ obj. minimizes JS-divergence, minimum is at $p_d = p_g$ with $V(\cdot,\cdot) = -\log 4$
$-2D_{JS}(p_d\|p_g) \ge 0$ (0 for $p_d = p_g$)

D is maximally confused and attains $\frac{1}{2}$ uniformly everywhere

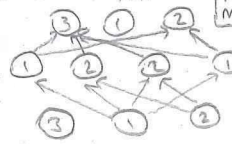
Training GANs
- Gradient ascent on D: $\max_{\theta_d} \mathbb{E}_{x\sim p_d(x)}\left[\log D_{\theta_d}(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$
- Gradient ascent on G: $\max_{\theta_g} \mathbb{E}_{z\sim p_z(z)}\left[\log(D_{\theta_d}(G_{\theta_g}(z)))\right]$ → maximizing D being wrong
↳ we do this instead of $\min_{\theta_g} \mathbb{E}_{z\sim p_z(z)}\left[\log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$ → minimizing D being right

tabular approach: $2^n - 1$ params, fully visible belief nets: $O(n^2)$ via logreg
NADE: computation takes $O(nD)$ time (regular times # of samples)

$h_k(x) = \text{sigm}(b + W_{\cdot,<k} x_{<k})$, $\hat{x}_k = \text{sigm}(c_k + V_{k,\cdot} h^{(k)})$

most of the time random order works fine, in training use teacher forcing
we can leverage: $(b + W_{\cdot,<k+1} x_{<k+1}) - (b + W_{\cdot,<k} x_{<k}) = W_{\cdot,k} x_{k+1}$

MADE: sample k from $(1, n-1)$ for each hidden unit, for timesteps $t = 1, \dots, n-1$ if the sampled value is smaller than the preceding one, drop (mask) the connection



NADE MADE

Temporal causal networks (TCNs): long-term temporal dependencies, much higher dimensionality (audio vs images), use dilated convs to reach larger receptive fields, cannot use strided convs due to the need to preserve resolution, inference speed

VRNN (variational RNN): one VAE per timestep (stochastic latent vars)
C-VRNN (conditional VRNN): more latent vars to decouple style and content
STCN (stochastic TCNs):

correlation: $I'(i,j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} k(m,n) I(i+m, j+n)$ ↳ if $k(i,j) = k(-i,-j)$, convolution equals correlation

convolution: $I'(i,j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} k(m,n) I(i-m, j-n) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} k(-m,-n) I(i+m, j+n)$

- linearity: $T(\alpha U + \beta V) = \alpha T(U) + \beta T(V)$
- T invariant to f: $T(f(U)) = T(U)$
- T equivariant to f: $T(f(U)) = f(T(U))$

any linear, shift-equivariant function can be written as a convolution

$X * K = Y$ (fwd)

$\frac{\partial E}{\partial K} = \frac{\partial E}{\partial Y} * \text{rot}_{180}(K) = \frac{\partial E}{\partial Y} \star K$ ↙ apply zero padding where necessary

$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y} * \text{rot}_{180}(X) = \frac{\partial E}{\partial Y} \star X = \text{rot}_{180}\left(X \star \frac{\partial E}{\partial Y}\right)$

$X = z^{(l-1)}$, $Y = z^{(l)}$, $K = w^{(l)}$, $\frac{\partial E}{\partial Y} = \delta^{(l)}$, $\frac{\partial E}{\partial Y} = \delta^{(l)}$, $\frac{\partial E}{\partial K} = \frac{\partial C}{\partial w^{(l)}}$, $E = C$ **CNN**

Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$, $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ **Basics**

tanh: $\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1$, $\tanh'(x) = 1 - \tanh^2(x)$

relu: $\text{ReLU}(x) = \max(0,x)$, $\text{ReLU}'(x) = \begin{cases} 0 & \text{if } x \le 0 \\ 1 & \text{if } x > 0 \end{cases}$ hinge-loss $= \max(0, 1 - y \cdot \hat{y})$

leaky relu: $\text{leaky}(x) = \mathbb{1}(x < 0)(\alpha x) + \mathbb{1}(x \ge 0)(x)$

softmax: $\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}$, gradient checking: $\frac{df(x)}{d} = \frac{(f(x+h) - f(x-h)}{2h}$

binary cross-entropy: $-y \log \hat{y} - (1-y)\log(1-\hat{y})$, multiclass: $-\sum_{k=1}^{K} y_{i,k} \log(\hat{y}_{i,k})$

quot. rule: $f(x) = g(x)/h(x) \Rightarrow f'(x) = \frac{g'(x)h(x) - h(x)g(x)}{[h(x)]^2}$

$\frac{\partial |x|}{\partial x} = \text{sgn}(x) = (-1)^{\mathbb{1}[x<0]}$