1 Math

$$\sigma^s = \mathbb{E}[X^2] - \mathbb{E}[X]^2 \quad (1)$$

$$\frac{\partial}{\partial q_k} soft(q)_i = soft(q)_i(\delta_{i,k} - soft(q)_k) \quad (2)$$

$$KL(p||q) = \int p(x)\log(\frac{p(x)}{q(x)})dx \quad (3)$$

$$(4)$$

2 NN and Optimization

RMSProp and **ADAM** are actly worse for simple landscapes. **Adam**: $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \ \wedge\ v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2 \ \wedge\ \hat{m}_t = \frac{m_t}{1-\beta_1^t} \ \wedge\ \hat{v}_t = \frac{v_t}{1-\beta_2^t}$ init correction for sec momentum necessary (RMSProp+Mom is worse). **RMSProp**: $r_t = \alpha r_{t-1} + (1-\alpha)g_t^2 \ \wedge\ u_t = -\frac{\eta}{\sqrt{r_t+\epsilon}}g^t \ \wedge\ w_{t+1} = w_t + \eta_t u_t$
2ND-Order Optim $w_{t+1} = w_t - H_{\mathcal{L}}^{-1}\eta_t g_t$ weight updates by hessian but to big to compute :(.
Activation functions around 0, better not saturating, not bounded, center of it should be mean of inputs
Batchnorm whitening for activation functions, regularizes inference. Have linear function after batchnorm: at beginning its centered but can unlearn this.

3 RNN

$c_t = \tanh(x_{t-1}) + Ux_t + b, \quad \mathcal{L} = \sum_t \mathcal{L}_t(c_t)$. Grad is chain Jacobs: $\frac{\partial c_t}{\partial c_k} = \prod_{j=k+1}^t \frac{\partial c_j}{\partial c_{j-1}}$ and $\frac{\partial \mathcal{L}}{\partial W} = \sum_{\tau=1}^t \frac{\partial \mathcal{L}_t}{\partial c_t}\frac{\partial c_t}{\partial c_\tau}\frac{\partial c_\tau}{\partial W}$ with restr: $||\frac{\partial c_{t+1}}{\partial c_t}|| \leq ||W^T|| \cdot ||\text{diag}(\sigma'(c_t))||$
If $||\frac{\partial c_k}{\partial c_{k-1}}|| \leq \frac{1}{\lambda_{max}}||\text{diag}(\max(\sigma'(\cdot)))|| < 1$ then $\pi_{k=1}^\tau \frac{\partial c_k}{\partial c_{k-1}}$ goes zero exp, van grads. Opposite is expld grads.

4 GNN

DeepWalk: randomwalk + LSTM with skip-gram, works not good new nodes need retrain.

5 Deep Generative Models

Boltzman dist: $p(x) = \frac{1}{Z}\exp(-E(x))$. Comp. of normal. Const. Z difficult. **Boltzmann machine** $E(x) = -x^T Wx - b^T x$ x is 256^2 big. Instead **RBM** $E(x) = -x^T Wh - b^T x - c^T h$ with latent h.

5.1 GAN

implicit density, sampling from PDF $\mathcal{L} = -12\mathbb{E}_{x\sim p_{data}}\log D(x) - \frac{1}{2}\mathbb{E}_{z\sim p_z}\log(1-D(G(z)))$
For better learning train for G the opposite, for approx ML estimate:
$J^G = -\frac{1}{2}\mathbb{E}_z\exp(\sigma^{-1}(D(G(z))))$ (one opt, Goodfellow). Normal object resembles minimizing Jesnon-Shannon divergence:
$D_{JS}(a||b) = 0.5D_{KL}(a||(a+b)/2) + 0.5D_{KL}(b||(a+b)/2)$
GAN problems: minimax instability, van grads, mode collapse (not due to divergence probably)
Improvements: Wasserstein dist, cBN, cGAN, label smoothing

5.2 Variational Inference

How est. posterior: MCMC or var. infer.: $\phi^* = \arg\min_\phi KL(q(\theta|\phi)||p(\theta|x))$, rev divergence (underestimate var, overest. with forward). **ELBO** $\mathbb{E}_{q_\phi(\theta)}[\log p(x|\theta)] - KL(q_\phi(\theta)||p(\theta)) = \mathbb{E}_{q_\phi(\theta)}[\log p(x|\theta)] + \mathbb{E}_{q_\phi(\theta)}[\log p(\theta)] - \mathbb{E}_{q_\phi(\theta)}[\log q(\theta)]$ with that $\log p(x) = ELBO_{\theta,\phi}(x) + KL(q_\phi(\theta)||p(\theta|x))$.
ELBO is vari. free Enrgy. Backprop in VAE: use REINFORCE to approx grad (high var grards slow down) or reparam trick

5.3 Normalizing Flows

$\log p(x) = \log \pi_o(z_o - \sum_i^K |\det \frac{df_i}{dz_{i-1}}|)$
requirements: f_i must be easily invertible and the Jacobian must be computable

6 Bayesian Deep Learning

Benefits of Bayesian: ensemble makes better accuracies, uncertainty estimates, sparsity makes model compression, active learning, distributed learning. **Epistemnic uncertainty** ignorance which model generated the data. More data reduces this. For safety critical stuff, small datasets. **Aleatoric uncertainty** ignorance about the nature of the data. *Heteroscedastic* uncertainty about specific data $\mathcal{L} = \frac{||y_i - \hat{y}_i||^2}{s\sigma_i^2} + \log\sigma_i$, *homoscedastic* uncertainty about the task, we might reduce by combining tasks. $\mathcal{L}$ same but without idx. **MC Dropout** have d. during inference (by Bernoulli as vari. dist.) Then model prec. $\tau = \frac{l^2 p}{2N\lambda}$.

7 Deep Sequential models

7.1 Autoregressive models

With sequential data we have:
$x = [x_1, \ldots, x_k] \implies p(x) = \prod_{k=1}^D p(x_k|x_{j<k})$ thus no param sharing and no ∞ chains $\implies p(x)$ is tractable.
NADE: fixed masks, conditionals modeled as MoG. **MADE**: masked conv on an autoencoder. **PixelRNN** seq. order over rows and channel R,G and B. Conditionals modeled with LSTM. Slow train and gen, but good gen. **PixelCNN** model conds with masked convs. Is worse than RNN cause blind spot. Fix by having convs for left row and everything above cascading. Output 8bit softmax **GatedPixelCNN** use two conv stacks, horiz and vart to not have blind sport
PixelCNN++ dropout/whole pixels/discr log mix likelihood (from continuos output).
PixelCNN is too powerful **PixelVAE** VAE+PixelCNN as the networks

8 DeepRL

Goal: max fut rewards (Q-func, value).
$Q^\pi(s_t, a_t) = \mathbb{E}(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3}, \ldots |s_t, a_t) = \mathbb{E}_{s',a'}(r + \gamma Q^\pi(s', a')|s_t, a_t)$ (Bellman eq).
Approaches, -based, policy,value,model.
Optimal Value Function
$Q^*(s,a) = t_{t+1} + \gamma\max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) = \mathbb{E}_{s'}(t + \gamma\max_{a'} Q^*(s', a')|s, a)$
Value based: **Q-learning**: minimize $min(r + \gamma\max_{a'} Q_t(s', a') - Q_t(s,a))^2$ $Q_t(s,a)$ is your prev est Q-Value for state s and action a. The other stuff is your new estimate at new state, action. You want to minimize this to get better.
Gradient ist then $\frac{\partial\mathcal{L}}{\partial\theta} = \mathbb{E}[-2\cdot(r + \gamma\max_{a'} Q(s', a', \theta) - Q(s, a, \theta))\frac{\partial Q(s,a,\theta)}{\partial\theta}]$
Is unstable because target depends on Q, also seq breaks independence assump, highly correlated samples break SGD. Solut1: exp replay, play random steps from other history. Solt2: have a sec network which is updated once in a while to calc targets so that dies not interfere with grad calc stability. *Other tricks:* clip rewards to -1,1, skip frames
Policy Optimization: q-func often too expensive, must account for all states/actions. Instead directly learn policy $\pi_\theta(a|s)$:
$\frac{\partial\mathcal{L}}{\partial w} = \mathbb{E}[\frac{\partial\log\pi(a|s,w)}{\partial w}Q^\pi(s,a)]$(deterministic) or
$\frac{\partial\mathcal{L}}{\partial w} = \mathbb{E}[\frac{\partial Q^\pi(s,a)}{\partial a}\frac{\partial a}{\partial w}]$(stochastic $a = \pi(s)$) compute gradients with log-derivative trick, REINFORCE:
$\nabla_\theta \log p(x;\theta) = \frac{\nabla_\theta p(x;\theta)}{p(x;\theta)}$