Introduction

- quintessential example of a representation learning algorithm is the **auto-encoder**. An auto-encoder is the combination of an encoder function, which converts the input data into a diffeernt representation, and a decoder function, which converts the new representation back into the oirginal format.
- **factor of variation**: separate sources of influence. In speech analysis, e.g. speaker's age, sex, accent, words they are speaking.
- quintessential example of deep learning model is the **feedforward network or multi-layer perceptron (MLP)**. A MLP is just a mathematical function mapping some set of input values to output values.

Linear Algebra

tensor = array with more than two axes
element-wise or Hadamard product, denoted as $\mathbf{A} \circ \mathbf{B}$

Determinant:
maps matrices to real scalars
determinant = product of all eigenvalues
abs(det(A)) = a measure of how much multiplication by the matrix expands or contracts space

Probability and Information Theory

Marginal probability (Sum rule):

$P(\mathrm{x} = x) = \sum_y P(\mathrm{x} = x, \mathrm{y} = y)$, or $p(x) = \int p(x, y)dy$

Conditional probability

$P(\mathrm{y} = y|\mathrm{x} = x) = \frac{P(\mathrm{y}=y,\mathrm{x}=x)}{P(\mathrm{x}=x)}$

Chain/Product rule of conditional probabilites

$P(x^{(1)}, ..., x^{(n)}) = P(x^{(1)}) \prod_{i=1}^{n} P(x^{(i)}|x^{(1)}, ..., x^{(i-1)})$

Expected Value

The expected value of some function $f(x)$ with respect to a probability distribution $P(x)$ is the mean value that f takes on when x is drawn from P:
$E_{x\sim P}[f(x)] = \sum_x P(x)f(x)$ or $E_{x\sim p}[f(x)] = \int p(x)f(x)dx$

Variance

$Var(f(x)) = E[(f(x) - E[f(x)])^2]$

Important identities

$\sigma(x) = \frac{\exp(x)}{\exp(x)+\exp(0)}$
$\frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x))$
$1 - \sigma(x) = \sigma(-x)$
$\log(\sigma(x)) = \zeta(-x)$
$\frac{d}{dx}\zeta(x) = \sigma(x)$
$\sigma^{-1}(x) = \log\left(\frac{x}{1-x}\right)$
$\zeta^{-1}(x) = \log(\exp(x) - 1)$
$\zeta(x) = \int_{-\infty}^{x} \sigma(y)dy$
$\zeta(x) - \zeta(-x) = x$

Information Theory

Self-information: $I(x) = -\log P(x)$
Shannon entropy: $H(x) = E_{x\sim P}[I(x)] = -E_{x\sim P}[\log P(x)]$
The Shannon entropy of a distribution is the expected ammount of information in an event drawn from that distribution.
Kullback-Leibler (KL) divergence:
$D_{KL}(P||Q) = E_{x\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = E_{x\sim P}[\log P(x) - \log Q(x)]$
Measure how different two distributions $P(x)$ and $Q(x)$ (over the same random variable x) are.
Not symmetric: $D_{KL}(P||Q) \neq D_{KL}(Q||P)$
Cross-entropy:
$H(P, Q) = -E_{x\sim P} \log Q(x) = H(P) + D_{KL}(P||Q)$
Minimizing the cross-entropy wrt Q is equivalent to minimizing the KL divergence.

Numerical Computation

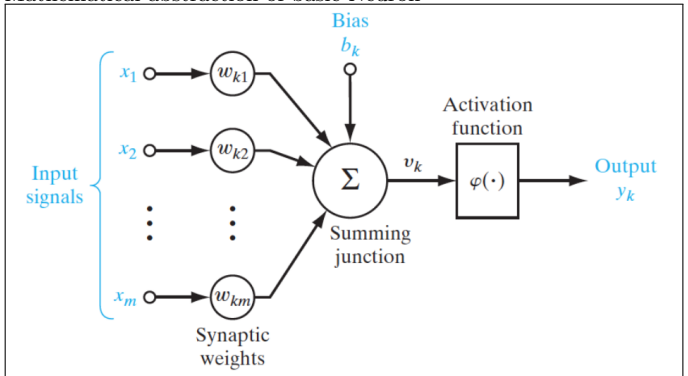
Condition number: $\max_{i,j} \left|\frac{\lambda_i}{\lambda_j}\right|$
When the Hessian has a poor condition number, gradient descent performs poorly. This is because in one direction, the derivative increases rapidly, while in another direction, it increases slowly. Gradient descent is unaware of this change in the derivative. Poor condition number also makes choosing a good step size difficult.
Hessian, Curvature: With negative curvature, the cost function actually decreases faster than the gradient predicts. With no curvature, the gradient predict the decrease correctly. With positive curvature, the function decreases more slowly than expected.
Second derivative test:
- Hessian positive definite (all EVs positive): local minimum
- Hessian negative definite (all Evs negative): local maximum
- At least one EV positive, at least one negative: saddle points
- All EVs same sign, at least one zero: inconclusive

Lipschitz continuous: $\forall \mathbf{x}, \forall \mathbf{y}, |f(\mathbf{x}) - f(\mathbf{y})| \leq \mathcal{L}\|\mathbf{x} - \mathbf{y}\|_2$ with Lipschitz constant $\mathcal{L}$.

Deep Feedforward Networks

1.4: Elements of Computation

Mathematical abstraction of basic Neuron



Linear function

Def.: A function $f : \mathbf{R}^n \to \mathbf{R}$ is a linear function if:
(1) $f(x + x') = f(x) + f(x'), (\forall x, x' \in \mathbf{R}^n)$
(2) $f(\alpha x) = \alpha f(x), (\forall \alpha \in \mathbf{R})$

Proposition: f linear $\Leftrightarrow f(x) = w^\top x$ for some $w \in \mathbf{R}^n$.
$\Leftarrow$ Properties of scalar product:
(1) $f(x + x') = ... = f(x) + f(x')$
(2) $f(\alpha x) = ... = \alpha f(x)$
$\Leftarrow$ Write $x = \sum_{i=1}^n x_i e_i$. Linearity implies:
$f(\mathbf{x}) = \sum_{i=1}^n x_i f(e_i)$ identify $w_i := f(e_i)$

Hyperplane

Def.: A hyperplane is an affine subspace of co-dimension 1.

Level set

Def.: The level sets of a fucntion $f : \mathbf{R}^n \to \mathbf{R}$ is a one-parametric family of sets defined as
$L_f(c) := \{x : f(x) = c\} = f^{-1}(c) \subseteq \mathbf{R}^n$

Level sets of linear functions

Def.: Let $f : \mathbf{R}^n \to \mathbf{R}$ be linear, $f(x) = w^\top x + b$, then
$L_f(c) = \{x : w^\top x = c - b\}$ = hyperplane $\perp w$

Linear (affine) maps

$F : \mathbf{R}^n \to \mathbf{R}^m$ with

$$F(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ ... \\ f_m(x) \end{pmatrix} = \begin{pmatrix} w_1^\top x + b_1 \\ w_2^\top x + b_2 \\ ... \\ w_m^\top x + b_m \end{pmatrix} = \begin{pmatrix} w_1^\top \\ w_2^\top \\ ... \\ w_m^\top \end{pmatrix} x + \begin{pmatrix} b_1 \\ b_2 \\ ... \\ b_m \end{pmatrix}$$

Composition of linear maps

Proposition: Let $F_1, .., F_L$ be linear maps, then $F = F_L \circ \cdots \circ F_1$ is also a linear map. Proof: $F(\mathbf{x}) = (\mathbf{W}_L ...(\mathbf{W}_2(\mathbf{W}_1\mathbf{x}))...) = (\mathbf{W}_L ...\mathbf{W}_2\mathbf{W}_1)\mathbf{x} = \mathbf{W}\mathbf{x}$
- every L-level hierarchy collapses to one level
- note that $rank(F) \equiv dim(im(F)) \leq \min_l rank(F_l)$

Conclusion: Need to move beyond linearity!

1.5: Approximation Theory

Ridge function

Def. (Ridge function): $f : \mathbf{R}^n \to \mathbf{R}$ is a ridge function, if it can be written as $f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$
- Limit set: $L_f(c) = \cup_{d\in\sigma^{-1}(c)} L_{\bar{f}}(d)$,
if linear part of f denoted by $\bar{f}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$
- If σ is differentiable at $z = \mathbf{w}^\top \mathbf{x} + b$ then
$\nabla_x f \overset{\text{chain rule}}{=} \sigma'(z)\nabla_x \bar{f} = \sigma'(z)\mathbf{w}$

Theorem Let $f : \mathbf{R}^n \to \mathbf{R}$ differentiable at $\mathbf{x}$. Then either $\nabla f(\mathbf{x}) = 0$ or $\nabla f(\mathbf{x}) \perp L_f(f(\mathbf{x}))$.

Def. (Dense Subsets): A function class $\mathcal{H} \subseteq C(\mathbf{R}^d)$ is dense in $C(\mathbf{R}^d)$ iff
$\forall f \in C(\mathbf{R}^d) \forall \epsilon > 0 \forall K \subset \mathbf{R}^d$, compact:
$\exists h \in \mathcal{H} s.t. \max_{\mathbf{x}\in K} |f(\mathbf{x}) - h(\mathbf{x})| = \|f - h\|_{\infty,K} < \epsilon$
Conclusion: We cann approximate any continuous f to arbitrary accuracy (on K) with a suitable member of $\mathcal{H}$.
- uniform approximation on compacta (i.e. use of ∞-norm)
- sup $\to$ max (Bolzano-Weierstrass)

Universal Approximation with Ridge Functions

Definitions: Let $\sigma : \mathbf{R} \to \mathbf{R}$ be a scalar function
$\mathcal{G}_\sigma^n := \{g : g(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$ for some $\mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R}\}$
$\mathcal{G}^n := \cup_{\sigma\in C(\mathbf{R})} \mathcal{G}_\sigma^n$, universe of continuous ridge functions
Theorem: Vostrecov and Kreines, 1961
$\mathcal{H}^n :=$ span$(\mathcal{G}^n)$ is dense in $C(\mathbf{R}^n)$.

Dimension Lifting Lemma (Pinkus)

Lemma (Pinkus 1999): The density of $\mathcal{H}_\sigma^1$ in $C(\mathbf{R})$ with
$\mathcal{H}_\sigma^1 := \text{span}(\mathcal{G}_\sigma^1) = \text{span}\{\sigma(\lambda t + \theta) : \lambda, \theta \in \mathbf{R}\}$
implies the density of
$\mathcal{H}_\sigma^n := \text{span}(\mathcal{G}_\sigma^1) = \text{span}\{\sigma(\mathbf{w}^\top \mathbf{x} + b) : \mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R}\}$
in $C(\mathbf{R}^n)$ for any $n \geq 1$.
Conclusion: We can lift density property of ridge function families from $C(\mathbf{R})$ to $C(\mathbf{R}^n)$.
Proof:...

- Continuous functions can be well approximated by linear combinations of ridge functions (universal function approximation).
- Justifies use of computational units which apply a scalar nonlinearity to a linear function of the inputs.

2.1: Rectification Networks

Def.: Rectified linear unit (ReLU)
$(x)_+ := \max(0, x)$,
$\partial(x)_+ = 1(x > 0), 0(x < 0), [0; 1](x = 0)$
Def.: Absolute value rectification (AVU)
$|x| := x(x \geq 0), -x(\text{otw.})$,
$\partial|x| = 1(x > 0), [-1; 1](x = 0), -1(x < 0)$

Shektman (1982)

Any $f \in C[0; 1]$ can be uniformly approximated to arbitrary precision by a polygonal line

Lebesgue (1898)

A polygonal line with m pieces can be written
$g(x) = ax + b + \sum_{i=1}^{m-1} c_i(x - x_i)_+$
$g(x) = a'x + b' + \sum_{i=1}^{m-1} c_i'|x - x_i|$
- knots: $0 = x_0 < x_1 < \cdots < x_{m-1} < x_m = 1$
- $m + 1$ parameters $a, b, c_i \in \mathbf{R}$
- ReLU function approximation in 1D

Proof (by induction over m):
$m = 1$: Linear function over $[0; 1] \Rightarrow$ fit line with a, b
$m \Rightarrow m+1$: Given $m+1$ knots and values (x_j, y_j). Eliminate knot x_m and choose a, b, and $c_i (i < m)$ to fit this function (induction hypothesis)
Now modify c_{m-1} and choose c_m to fit a wedge to the three points (x_{m-1}, y_{m-1}), (x_m, y_m), and $(x_{x+1} = 1, y_{m+1})$

- Weierstrass: $C[0; 1]$ functions can be uniformly approximated by polynomials
- Lebesgue: proof for Weierstrass theorem by showing that $|x|$ can be uniformly approximated on $[-1; 1]$ by polynomials

Theorem: Networks with one hidden layer of ReLU or AVU are universal function approximators

1. Universally approximate $C(K)$ functions (K, compact) by polygonal lines
2. Represent polygonal lines by (linear function +) linear combinations of $(\cdot)_+$ or $|\cdot|$ functions.
3. Apply dimension lifting lemma to show density of the linear span of resulting ridge function families $\mathcal{G}_{(\cdot)_+}^n$ and $\mathcal{G}_{|\cdot|}^n$

Linear Combinations of Rectified Units

By linearly combining m rectified units, into how many ($R(m)$) cells is $\mathbf{R}^n$ maximally partioned? (Zaslavsky, 1975)
$R(m) \leq \sum_{i=0}^{\min\{m,n\}} \binom{m}{i}$
- for $m \leq n$, $R(m) = 2^m$ (exponential growth)
- for given n, asymptotically, $R(m) \in \Omega(m^n)$ (bounded by m^n), i.e. there is a polynomial slow-down, which is induced by the limitation of the input space dimension.

Deep Combinations of Rectified Units

Process n inputs through L ReLU layers with widths $m_1, ..., m_L \in O(m)$. Into how many ($R(m, L)$) cells can $\mathbf{R}^n$ be maximally partitioned?
Theorem (Montufar, 2014): $R(m, L) \in \Omega\left((\frac{m}{n})^{n(L-1)} m^n\right)$
For any fixed n, exponential growth can be ensured by making layers sufficiently wide ($m > n$) and increasing the level of functional nesting (i.e. depth L).

Hinging Hyperplanes

Def.: Hinge function: If $f : \mathbf{R}^n \to \mathbf{R}$ can be written with parameters $w_1, w_2 \in \mathbf{R}^n$ and $b_1, b_2 \in \mathbf{R}$ as below it is called a hinge function:
$g(\mathbf{x}) = \max(\mathbf{w}_1^\top \mathbf{x} + b_1, \mathbf{w}_2^\top \mathbf{x} + b_2)$
- face: $(\mathbf{w}_1 - \mathbf{w}_2)^\top \mathbf{x} + (b_1 - b_2) = 0$
- representational power: $2\max(f, g) = f + g + |f - g|$
- k-Hinge function: $g(\mathbf{x}) = \max(\mathbf{w}_1^\top \mathbf{x} + b_1, ..., \mathbf{w}_k^\top \mathbf{x} + b_k)$
Theorem (Wang and Sun, 2004): Every continuous piecewise linear function from $\mathbf{R}^n \to \mathbf{R}$ can be written as a signed sum of k-Hinges with $k \leq n + 1$.
- exact representation (not approximation as ReLU, AVU).
- to represent k-Hinge with ReLU: need depth log. in k.

Polyhedral Function (Convex functions)

= convex and continuous piecewise linear functions
- f polyhedral $\leftrightarrow$ epi(f) is a polyhedral set
- epigraph of f (all points above the graph of f):
epi(f):= $\{(\mathbf{x}, t) \in \mathbf{R}^{n+1} : f(\mathbf{x}) \leq t\}$
- polyhedral set S: finite intersection of closed half-spaces
$S = \{\mathbf{x} \in \mathbf{R}^n : \mathbf{w}_j^\top \mathbf{x} + b_j \geq 0, j = 1, ...r\}$

Max-Representation of Polyhedral Functions

For every polyhedral f, there exists $\mathcal{A} \subset \mathbf{R}^{n+1}, |\mathcal{A}| = m$ s.t.
$f(x) = \max_{(w,b)\in\mathcal{A}}\{\mathbf{w}^\top \mathbf{x} + b\}$
- each polyhedral f can be repres. as max. of supp. hyperplanes
- linear functions in $\mathcal{A}$ describe supporting hyperplanes of epi(f).

Continuous Piecewise Linear Functions

Theorem (Wang, 2004): Every continuous piecewise linear function f can be written as the difference of two polyhedral functions; with finite $\mathcal{A}^+, \mathcal{A}^-$
$f(x) = \max_{(w,b)\in\mathcal{A}^+}\{\mathbf{w}^\top \mathbf{x} + b\} - \max_{(w,b)\in\mathcal{A}^-}\{\mathbf{w}^\top \mathbf{x} + b\}$

2 × Maxout = Allout

Theorem (Goodfellow, 2013): Maxout networks with two maxout units are universal function approximators.

1. Wang's theorem: linear network with two maxout units and a linear output unit (subtraction) can represent any continuous PWL function (exactly!).
2. Continuous PWL functions are dense in $C(\mathbf{R}^n)$.

2.2: Sigmoid Networks

Sigmoid functions

$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t} \in (0; 1), \sigma^{-1}(\mu) = \ln\left(\frac{\mu}{1-\mu}\right)$
$\tanh(t) = 2\sigma(2t) - 1 \in (-1; 1)$

Approximation Theorem

Theorem (Lencho, Lin, Pinkus, Schocken, 1993): Let $\sigma \in C^\infty(\mathbf{R})$, not a polynomial, then $\mathcal{H}_\sigma^1$ is dense in $C(\mathbf{R})$; i.e. results in dense function approximation.
Corollary: MLPs with one hidden layer and any non-polynomial, smooth activation function are universal function approximators.
Lemma: MLPs with one hidden layer and a polynomial activation function are **not** universal function approximators.

1. For all $h \neq 0$: $\frac{\sigma((\lambda+h)t+\theta)-\sigma(\lambda t+\theta)}{h} \in \mathcal{H}_\sigma$
2. It follows that (generalizing to all k-th derivatives):
$\frac{d^k}{d\lambda^k}\sigma(\lambda t + \theta)|_{\lambda=0} = t^k \sigma^{(k)}(\theta) \in cl(\mathcal{H}_\sigma)$
3. If we can show that there always is a θ_0 such that $\sigma^{(k)}(\theta_0) \neq 0$ then we are be guaranteed that $t^k \in cl(\mathcal{H}_\sigma)$ and hence all polynomials.
4. By the Weierstrass theorem this implies the result
Theorem (Donoghue, 1969): If σ is C^∞ on $(a; b)$ and it is not a polynomial thereon, then there exists a point $\theta_0 \in (a; b)$ such that $\sigma^{(k)}(\theta_0) \neq 0$ for $k = 0, 1, 2, ...$

Sigmoidal MLP: Approximation Guarantees

Theorem (Barron, 1993): For every $F : \mathbf{R}^n \to \mathbf{R}$ with absolutely continuous Fourier transform and for every m there is a function of the form $\tilde{f}_m$ such that
$\int_{B_r}(f(\mathbf{x} - \tilde{f}_m(\mathbf{x}))^2 \mu(d\mathbf{x}) \leq O(1/m)$
where $B_r = \{\mathbf{x} \in \mathbf{R}^n : \|\mathbf{x}\| \leq r\}$ and μ is any probability measure on B_r.
The residual bound does not depend on the input dimension n

2.3/3.1: Feedforward Networks

Def. (Feedforward networks): A set of computational units arranged in a DAG (directed acyclic graph). **Def. (Hidden layer):** A layer that is neither the input, nor the output layer.

2.4/3.2: Output Units and Objectives

Loss function

Def. (Loss function): A non-negative function
$l : \mathcal{Y} \times \mathcal{Y} \to \mathbf{R}_{\geq 0}$, $(y^*, y) \to l(\mathbf{y}^*, \mathbf{y})$, output space: $\mathcal{Y}$
squared error: $\mathcal{Y} = \mathbf{R}^m, l(\mathbf{y}^*, \mathbf{y}) = \|\mathbf{y}^* - \mathbf{y}\|_2^2 = \sum_{i=1}^m (y_i^* - y_i)^2$
claissification error: $\mathcal{Y} = [1 : m], l(\mathbf{y}^*, \mathbf{y}) = 1 - \delta_{\mathbf{y}^*\mathbf{y}}$

Def. (Expected risk (expected loss)): Assume inputs and outputs are governed by a distribution $p(\mathbf{x}, \mathbf{y})$ over $\mathcal{X} \times \mathcal{Y}, \mathcal{X} \subset \mathbf{R}^n$. The expected risk of F is given by $J^*(F) = \mathbf{E}_{x,y}[l(\mathbf{y}, F(\mathbf{x})]$

Def. (Training/Empirical risk): Assume we have a random sample of N input-output pairs $\mathcal{S}_N := \{(\mathbf{x}_i, \mathbf{y}_i)$ iid distr. $\{p : i = 1, ..., N\}$.
The training risk of F on a training sample is
$J(F; \mathcal{S}_N) = \frac{1}{N}\sum_{i=1}^N l(y_i, F(x_i))$
- training risk is the expected risk under the empirial distribution induced by the sample $\mathcal{S}_N$.

Def. (Empirical risk minimizer):
$\hat{F}(\mathcal{S}_N) = \arg\min_{F\in\mathcal{F}} J(F; \mathcal{S}_N)$ with parameter $\hat{\theta}(\mathcal{S}_N)$.

Def. (Generalized linear models): predict the mean of the output distribution: $\mathbf{E}[y|x] = \sigma(\mathbf{w}^\top \mathbf{x})$, σ is invertible, σ^{-1} is the link function.

Log Likelihood

$J(\theta; (\mathbf{x}, \mathbf{y})) = -\log p(\mathbf{y}|\mathbf{x}; \theta)$

Logistic Log Likelihood

$J(F;(x,y)) = -\log p(y|z) = -\log\sigma((2y-1)z) = \zeta((1-2y)z)$
with $z := \bar{F}(\mathbf{x}) \in \mathbf{R}$, $\zeta = \log(1+\exp(\cdot))$ (soft-plus)

Likelihood for logistic regression

$L = \prod_{i=1}^n p(x_i)^{y_i}(1-p(x_i))^{1-y_i}$

Multinomial Log Likelihood

$J(F;(\mathbf{x},y)) = -\log p(y|\mathbf{x};F) = -\log\left[\frac{e^{z_y}}{\sum_{i=1}^m e^{z_i}}\right]$
$= -z_y + \log\sum_{i=1}^m \exp[z_i]$ with $\mathbf{z} := \bar{F}_i(\mathbf{x}) = \mathbf{w}_i^\top \mathbf{x} \in \mathbf{R}^m$

3.3/4.1: Backpropagation: Exploit compositional structure

1. perform a forward pass (for given training example $(\mathbf{x},\mathbf{y})$) to compute activations for all units (use $\mathbf{x}$, apply F_1, get first hidden layer...)
2. compute gradient of J wrt. output layer activations
3. iteratively propagate activation gradient information from outputs to inputs ($\rightarrow$ backpropagation)
4. compute local gradients of activations wrt. weights

Chain rule: $(f\circ g)' = (f'\circ g)\cdot g'$
$\frac{d(f\circ g)}{dx}|_{x=x_0} = \frac{df}{dz}|_{z=g(x_0)} \cdot \frac{dg}{dx}|_{x=x_0}$

Jacobi matrix

$$\mathbf{J}_F := \begin{bmatrix}\nabla^\top F_1\\ \nabla^\top F_2\\ \dots\\ \nabla^\top F_m\end{bmatrix} = \begin{bmatrix}\frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \dots & \frac{\partial F_1}{\partial x_n}\\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \dots & \frac{\partial F_2}{\partial x_n}\\ \dots & \dots & \dots & \dots\\ \frac{\partial F_m}{\partial x_1} & \frac{\partial F_m}{\partial x_2} & \dots & \frac{\partial F_m}{\partial x_n}\end{bmatrix} \in \mathbf{R}^{m\times n}$$

for vector-valued function (map) $F : \mathbf{R}^n \to \mathbf{R}^m$

Jacobi Matrix Chain Rule

Vector-valued funtions $G : \mathbf{R}^n \to \mathbf{R}^q, F : \mathbf{R}^q \to \mathbf{R}^m$
componentwise rule:
$\frac{\partial(F\circ G)}{\partial x_i}|_{x=x_0} = \sum_{k=1}^q \frac{\partial F_j}{\partial z_k}|_{\mathbf{z}=G(\mathbf{x}_0)} \cdot \frac{\partial G_k}{\partial x_i}|_{x=x_0}$
Jacobi matrix chain rule (do not commute!)
$\mathbf{J}_{F\circ G}|_{x=x_0} = \mathbf{J}_F|_{z=G(x_0)} \cdot \mathbf{J}_G|_{x=x_0}$

Function Composition

$G : \mathbf{R}^n \to \mathbf{R}^m, f : \mathbf{R}^m \to \mathbf{R}, f\circ G : \mathbf{R}^n \to \mathbf{R}$
$\mathbf{R}^n \ni \mathbf{x} \xrightarrow{G} \mathbf{y} \xrightarrow{f} z \in \mathbf{R}$
Lemma(Chain rule for "activations"):
$\boxed{\nabla_{\mathbf{x}} z = \nabla_{\mathbf{y}}^\top z \cdot \mathbf{J}_G}$, $\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial y_j}{\partial x_i}\frac{\partial z}{\partial y_j}$
z: output, x:input, for f don't need Jacobian

Activity Backpropagation

$F = F^L \circ \dots \circ F^1 : \mathbf{R}^n \to \mathbf{R}^m$
$\mathbf{x} = \mathbf{x}^0 \xrightarrow{F^1} \mathbf{x}^1 \xrightarrow{F^2} \mathbf{x}^2 \to \dots \xrightarrow{F^L} \mathbf{x}^L = \mathbf{y} \xrightarrow{\mathcal{R}} \mathcal{R}(\theta;\mathbf{y})$
$\nabla_{\mathbf{x}} J = \mathbf{J}_{F^1}^\top \cdots \mathbf{J}_{F^L}^\top \nabla_{\mathbf{y}} J$
Can compute all activity gradients $\nabla_{\mathbf{x}^l}$ in backward order via successive matrix multiplication with (transposed) Jacobians.
$\mathbf{e}^0 := \nabla_{\mathbf{y}}^\top \mathcal{R}$,
$\mathbf{e}^l := \nabla_{\mathbf{x}^l}^\top \mathcal{R} = \mathbf{e}^0 \cdot \mathbf{J}_{F^l} \cdots \mathbf{J}_{F^{l+1}} = \mathbf{e}^{l+1}\mathbf{J}_{F^{l+1}}$

Proof (by induction over depth L)
Base case ($L=0$): $\mathbf{x} = \mathbf{y} \Rightarrow \nabla_{\mathbf{x}} J = \nabla_{\mathbf{y}} J$
Induction case: $F = (F^L \circ \dots \circ F^2)\circ F^1$
$\nabla_{\mathbf{x}} J \overset{\text{lemma}}{=} \mathbf{J}_{F^1}^\top \nabla_{\mathbf{x}^1} J \overset{IH}{=} \mathbf{J}_{F^1}^\top (\mathbf{J}_{F^2}^\top \dots \mathbf{J}_{F^L}^\top \nabla_{\mathbf{y}} J)$

Jacobian for ridge function

$\mathbf{x}^l = F^l(\mathbf{x}^{l-1}) = \sigma(\mathbf{W}^l\mathbf{x}^{l-1} + \mathbf{b}^l)$
$\frac{\partial x_i^l}{\partial x_j^{l-1}} = \sigma'(\langle \mathbf{w}_i^l, \mathbf{x}^{l-1}\rangle + b_i^l) W_{ij}^l := \bar{W}_{ij}^l$
thus: $\mathbf{J}_{F^l} = \tilde{\mathbf{W}}^l$

Multinomial Logistic Regression
$\mathbf{z} := \bar{F}_i(\mathbf{x}) \in \mathbf{R}^m$
$J(F;(\mathbf{x},y)) = -\log p(y|\mathbf{x};F) = -\log\left[\frac{e^{z_y}}{\sum_{i=1}^m e^{z_i}}\right]$
$= -z_y + \log\sum_{i=1}^m \exp[z_i] = \log\left[1 + \sum_{i\neq y}\exp[z_i - z_y]\right]$
Multivariate logistic loss
$-\frac{\partial J(x,y^*)}{\partial z_y} = \frac{\partial}{\partial z_y}\left[z_{y^*} - \log\sum_i \exp[z_i]\right]$
$= \delta_{yy^*} - \frac{\exp[z_y]}{\sum_i \exp[z_i]} = \delta_{yy^*} - p(y|x)$

Quadratic loss (neg. gradient: in what direction want to move)
$-\nabla_{\mathbf{y}} J(\mathbf{x},\mathbf{y}^*) = -\nabla_{\mathbf{y}} \frac{1}{2}\|\mathbf{y}^* - \mathbf{y}\|^2 = \mathbf{y}^* - \mathbf{y}$

From Activations to Weights

$\frac{\partial \mathcal{R}}{\partial W_{ij}^l} = \frac{\partial \mathcal{R}}{\partial x_i^l}\frac{\partial x_i^l}{\partial W_{ij}^l} = \frac{\partial \mathcal{R}}{\partial x_i^l} \cdot \sigma'(\langle \mathbf{w}_i^l, \mathbf{x}^{l-1}\rangle + b_i^l) \cdot x_j^{l-1}$
$\frac{\partial \mathcal{R}}{\partial b_i^l} = \frac{\partial \mathcal{R}}{\partial x_i^l}\frac{\partial x_i^l}{\partial b_i^l} = \frac{\partial \mathcal{R}}{\partial x_i^l} \cdot \sigma'(\langle \mathbf{w}_i^l, \mathbf{x}^{l-1}\rangle + b_i^l) \cdot 1$

4.2: Optimization for Deep Network's

Gradient Descent

$\theta(t+1) \leftarrow \theta(t) - \eta\nabla_\theta J(\mathcal{S})$
$\mathcal{S}$ = all training data $\Rightarrow$ steepest descent
$\mathcal{S}$ = mini batch of data $\Rightarrow$ SGD

$\theta(t+1) = \theta(t) - \eta\nabla_\theta \mathcal{R}$, cont.: $\dot\theta = -\nabla_\theta \mathcal{R}$ (Euler's method)

Grad Descent Analysis (Convex objective $\mathcal{R}$)

$\mathcal{R}$ has L-Lipschitz-continuous gradients:
$\mathcal{R}(\theta(t)) - \mathcal{R}^* \le \frac{2L}{t+1}\|\theta(0) - \theta^*\|^2 \in \mathbf{O}(t^{-1})$

$\mathcal{R}$ is μ-strongly convex in θ:
$\mathcal{R}(\theta(t)) - \mathcal{R}^* \le \left(1 - \frac{\mu}{L}\right)^t \mathcal{R}(\theta(t)) - \mathcal{R}^*)$
- exponential convergence ("linear rate")
- rate depends adversely on condition number L/μ.

Lower bound (general case): $\mathbf{O}(t^{-2})$, achieved by Neterov acceleration.

Curvature of objective function

$\mathcal{R}(\theta - \eta\nabla\mathcal{R}) \overset{Taylor}{\approx} \mathcal{R}(\theta) - \eta\|\nabla\mathcal{R}\|^2 + \frac{\eta^2}{2}\nabla\mathcal{R}^\top\mathbf{H}\nabla\mathcal{R}$
with $\nabla\mathcal{R}^\top\mathbf{H}\nabla\mathcal{R} = \|\nabla\mathcal{R}\|_{\mathbf{H}}^2$, $\mathbf{H} = \nabla^2\mathcal{R}$
ill-conditioning: $\frac{\eta}{2}\|\nabla\mathcal{R}\|_{\mathbf{H}}^2 \gtrsim \|\nabla\mathcal{R}\|^2$

Least-Squares: Single Layer Linear Network

Objective:
$\mathcal{R}(\mathbf{A}) = \mathbf{E}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2 = \operatorname{Tr}\mathbf{E}[(\mathbf{y}-\mathbf{A}\mathbf{x})(\mathbf{y}-\mathbf{A}\mathbf{x})^\top]$
$= \operatorname{Tr}\mathbf{E}[\mathbf{y}\mathbf{y}^\top] + \operatorname{Tr}(\mathbf{A}\mathbf{E}[\mathbf{x}\mathbf{x}^\top]\mathbf{A}^\top) - 2\operatorname{Tr}(\mathbf{A}\mathbf{E}[\mathbf{x}\mathbf{y}^\top])$
Gradient of objective:
$\nabla_{\mathbf{A}}\mathcal{R} = \nabla\mathbf{A}\operatorname{Tr}(\mathbf{A}\mathbf{A}^\top) - 2\nabla\mathbf{A}\operatorname{Tr}(\mathbf{A}\mathbf{\Gamma}^\top) = 2(\mathbf{A} - \mathbf{\Gamma})$

Least-Squares: Two Layer Linear Network ($\mathbf{A} = \mathbf{Q}\mathbf{W}$)

Objective:
$\mathcal{R}(\mathbf{Q},\mathbf{W}) = \text{const.} + \operatorname{Tr}(\mathbf{Q}\mathbf{W}\cdot(\mathbf{Q}\mathbf{W})^\top) - 2\operatorname{Tr}(\mathbf{Q}\mathbf{W}\cdot\mathbf{\Gamma}^\top)$
$\frac{1}{2}\nabla_{\mathbf{Q}}\mathcal{R} = (\mathbf{Q}\mathbf{W})\mathbf{Q}^\top - \mathbf{\Gamma}\mathbf{W}^\top = (\mathbf{A}-\mathbf{\Gamma})\mathbf{W}^\top \in \mathbf{R}^{m\times k}$
$\frac{1}{2}\nabla_{\mathbf{W}}\mathcal{R} = \mathbf{Q}^\top(\mathbf{A}-\mathbf{\Gamma}) \in \mathbf{R}^{k\times n}$
SVD of $\mathbf{\Gamma} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, lin. transform: $\tilde{\mathbf{Q}} = \mathbf{U}^\top\mathbf{Q}$, $\tilde{\mathbf{W}} = \mathbf{W}\mathbf{V}$
$\frac{1}{2}\nabla_{\tilde{\mathbf{Q}}}\mathcal{R} = \mathbf{U}\mathbf{U}^\top(\tilde{\mathbf{Q}}\tilde{\mathbf{W}} - \mathbf{\Sigma})\mathbf{V}\mathbf{V}^\top\tilde{\mathbf{W}}^\top = (\tilde{\mathbf{Q}}\tilde{\mathbf{W}} - \mathbf{\Sigma})\tilde{\mathbf{W}}^\top$
$\frac{1}{2}\nabla_{\tilde{\mathbf{W}}}\mathcal{R} = \tilde{\mathbf{Q}}^\top(\tilde{\mathbf{Q}}\tilde{\mathbf{W}} - \mathbf{\Sigma})$
$\frac{1}{2}\nabla_{\mathbf{q}_r}\mathcal{R} = (\mathbf{q}_\mathbf{r}^\top\mathbf{w}_\mathbf{r} - \sigma_\mathbf{r})\mathbf{w}_\mathbf{r} + \sum_{\mathbf{s}\neq\mathbf{r}}(\mathbf{q}_\mathbf{r}^\top\mathbf{w}_\mathbf{s})\mathbf{w}_\mathbf{s}$
$\frac{1}{2}\nabla_{\mathbf{w}_r}\mathcal{R} = (\mathbf{q}_\mathbf{r}^\top\mathbf{w}_\mathbf{r} - \sigma_\mathbf{r})\mathbf{q}_\mathbf{r} + \sum_{\mathbf{s}\neq\mathbf{r}}(\mathbf{q}_\mathbf{s}^\top\mathbf{w}_\mathbf{r})\mathbf{q}_\mathbf{s}$
Equivalent energy function (guessed)
$\tilde{\mathcal{R}}(\tilde{\mathbf{Q}},\tilde{\mathbf{W}}) = \sum_\mathbf{r}(\mathbf{q}_\mathbf{r}^\top\mathbf{w}_\mathbf{r} - \sigma_\mathbf{r})^\mathbf{2} + \sum_{\mathbf{s}\neq\mathbf{r}}(\mathbf{q}_\mathbf{s}^\top\mathbf{w}_\mathbf{r})^\mathbf{2}$
cooperation: same input-output mode weight vector align
competition: different mode weight vectors are decoupled

4.3: Stochastic Gradient Descent

Choose update direction $\mathbf{v}$ at random such that $\mathbf{E}[\mathbf{v}] = -\nabla\mathcal{R}$
$\mathcal{S}_K \subseteq \mathcal{S}_N, K \le N$
$\mathbf{E}\mathcal{R}(\mathcal{S}_K) = \mathcal{R}(\mathcal{S}_N) \Rightarrow \mathbf{E}\nabla\mathcal{R}(\mathcal{S}_K) = \nabla\mathcal{R}(\mathcal{S}_N)$
Update step: $\theta(t+1) = \theta(t) - \eta\nabla\mathcal{R}(t), \mathcal{R} := \mathcal{R}(\mathcal{S}_K(t))$
Convergence to optimum: convex or strongly convex objective, Lipschitz gradients, decaying learning rate $\sum_{t=1}^\infty \eta^2(t) < \infty, \sum_{t=1}^\infty \eta(t) = \infty$, e.g. $\eta(t) = Ct^{-\alpha}$, $\frac{1}{2} < \alpha \le 1$, iterate (Polyak) averaging
Convergence rates:
- strongly-convex case: $\mathcal{O}(1/t)$
- non-strongly convex: $\mathcal{O}(1/\sqrt{t})$

Heavy Ball Method (accelerate learning)

Update: $\theta(t+1) = \theta(t) - \eta\nabla\mathcal{R} + \alpha(\theta(t) - \theta(t-1)), \alpha \in [0;1)$
Gradients are constant $\Rightarrow$ update steps are boosted by $1/(1-\alpha)$:
$\eta\|\nabla J\|(1 + \alpha + \alpha^2 + \alpha^3 + \dots) \to \frac{\eta\|\nabla J\|}{1-\alpha}$, $\alpha = 0.9 \Rightarrow 10\times$
- accelerate for high curvature, small but consistent gradient, or noisy gradients.
- aims to solve poor conditioning of Hessian matrix and variance in stochastic gradient.

AdaGrad

Consider the entire history of gradients: gradient matrix:
$\theta \in \mathbf{R}^d, \mathbf{G} \in \mathbf{R}^{d\times t_{max}}, g_{it} = \frac{\partial\mathcal{R}(t)}{\partial\theta_i}|_{\theta=\theta(t)}$
Learning rate decays faster for weights that have seen significant updates. Compute (partial) row sums of $\mathbf{G}$: $\gamma_i^2(t) := \sum_{s=1}^t g_{is}^2$
Adapt learning rate per parameter:
$\theta_i(t+1) = \theta_i(t) - \frac{\eta}{\delta+\gamma_i(t)}\nabla\mathcal{R}(t), \delta > 0$ (small)
Non-convex variant: RMSprop (moving average, expon. weighted):
$\gamma_i^2(t) := \sum_{s=1}^t \rho^{t-s} g_{is}^2, \rho < 1$

BFGS/LBFGS (advantages of Newton, without comp. burden)

Newton method: $\theta(t+1) = \theta(t) - (\nabla^2\mathcal{R})^{-1}\nabla\mathcal{R}|_{\theta=\theta(t)}$
BFGS: $(\nabla^2\mathcal{R})^{-1} \approx \mathbf{M}(t)$: $\theta(t+1) = \theta(t) - \eta(t)M(t)\nabla\mathcal{R}|_{\theta=\theta(t)}$
where $\mathbf{M}(t+1) = \mathbf{M}(t)$ + rank one update with $\nabla\mathcal{R}$, $\eta(t)$ via line search. LBFGS: Reduce memory footprint with $\tilde{\mathbf{M}} \approx \mathbf{M}(t)$ with $k \approx 30$ rank one matrices (pairs of vectors), mini-batch

4.4/6.1: Optimization Heuristics

Polyak averaging (Average over iterates, reduce fluctuation)

Linear (convex case): $\bar\theta(t) = \frac{1}{t}\sum_{s=1}^t \theta(s)$
Running (non-convex): $\bar\theta(t) = \alpha\theta(t-1) + (1-\alpha)\theta(t), \alpha \in [0;1)$

Batch normalization

Hard to find suitable learning rate for all layers (strong dependencies between weights in layers exist) $\Rightarrow$ normalize the layer activations + backpropagate through normalizations

Fix layer l, fix set of example $I \subseteq [1:N]$

$\mu_j^l := \frac{1}{|I|}\sum_{i\in I}(F_j^l \circ ... \circ F^1)(\mathbf{x}[i]) \in \mathbf{R}^{m_l}$

$\sigma_j^l := \sqrt{\delta + \frac{1}{|I|}\sum_i (F_j^l \circ ... \circ F^1)(\mathbf{x}[i]) - \mu_j)^2}, \delta > 0$

Normalized activities: $\tilde{\mathbf{x}}_j^l := \frac{\mathbf{x}_j^l - \mu_j}{\sigma_j}$

Regain representational power: $\tilde{\tilde{\mathbf{x}}}_j^l = \alpha_j \tilde{\mathbf{x}}_j^l + \beta_j$

Batch normalization (simplified)

1) **Input**: mini-batch of real values $X = (x_1, .., x_n) \in \mathbf{R}^n$
2) **Learnable parameters**: $\gamma, \beta \in \mathbf{R}$
3) **Output**: $Y = (y_1, .., y_n) \in \mathbf{R}^n$, where we have
(a) Mini-batch mean: $\mu := \frac{1}{n}\sum_i x_i$
(b) Mini-batch variance: $\sigma^2 := \frac{1}{n}\sum_i (x_i - \mu)^2$
(c) Normalized mini-batch (matrix form): $\hat{X} := \frac{X-\mu}{\sqrt{\sigma^2+\epsilon}}$
(d) Output: $Y = BN_{\gamma,\beta}(X) := \gamma\hat{X} + \beta$

4.4/6.2: Norm-based Regularization

Regularization = Any aspect of a learning algorithm that is intended to lower the generalization error but not the training error. E.g.: Informed regularization: encode specific prior knowledge. simplicity bias: preference for simpler models (Occam's razor). Data augmentation and cross-task learning. Model averaging, e.g. ensemble methods, drop-out.

Standard regularization: $\mathcal{R}_\Omega(\theta; \mathcal{S}) = \mathcal{R}(\theta; \mathcal{S}) + \Omega(\theta)$
Deep networks: $\Omega(\theta) = \frac{1}{2}\sum_{l=1}^L \mu^l \|\mathbf{W}^l\|_F^2, \mu^l \geq 0$

Weight decay

Regularization based on L_2-norm is also called weight decay: $\frac{\partial\Omega}{\partial W_{ij}^l} = \mu^l w_{ij}^l$
weights in l-th layer get pulled towards zero with "gain" μ^l
naturally favors weights of small magnitude
GD update: $\theta(t+1) = (1-\mu)\cdot\theta(t) - \eta\cdot\nabla_\theta\mathcal{R}$

Weight decay (Analysis)

Taylor: $\mathcal{R}(\theta) \approx \mathcal{R}(\theta^*) + \frac{1}{2}(\theta-\theta^*)^\top \mathbf{H}(\theta-\theta^*)$, where $\mathbf{H}_\mathcal{R}$ is the Hessian of $\mathcal{R}$: $\mathbf{H}_\mathcal{R} = \left(\frac{\partial^2\mathcal{R}}{\partial\theta_i\partial\theta_j}\right)$, and $\mathbf{H} := \mathbf{H}_\mathcal{R}|_{\theta=\theta^*}$

$\nabla_\theta\mathcal{R}_\Omega \stackrel{!}{=} 0$ with $\mathbf{H} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top, \mathbf{\Lambda} = diag(\lambda_1, ..., \lambda_d)$
$\Rightarrow \theta = \mathbf{Q}(\Lambda + \mu I)^{-1}\Lambda Q^\top\theta^*$
- Along directions in parameter space with large eigenvalues of $\mathbf{H}$ (i.e. $\lambda_i \gg \mu$): vanishing effect
- Along directions in parameter space with small eigenvalues of $\mathbf{H}$ (i.e. $\lambda_i \ll \mu$): shrunk to nearly zero magnitude
Linear regression: $\mathcal{R}_\Omega(\theta) = \frac{1}{2}(\mathbf{X}\theta - y)^\top(\mathbf{X}\theta - y) + \frac{\mu}{2}\|\theta\|^2 \Rightarrow \theta = (\mathbf{X}^\top\mathbf{X} + \mu\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$

Regularization via Constrained Optimization

$\min_{\theta:\|\theta\|\leq r}\mathcal{R}(\theta)$
Optimization approach: Projected gradient descent:
$\theta(t+1) = \Pi_r(\theta(t) - \eta\nabla\mathcal{R}), \Pi_r(\mathbf{v}) := \min\left\{1, \frac{r}{\|\mathbf{v}\|}\right\}\mathbf{v}$
Only active when weights are (too) large

Early Stopping

Stop learning after finite (small) number of iterations. E.g. use validation data to estimate risk. Stop when flat or worsening. Keep best solution.

Early Stopping (Analysis)

Taylor: $\nabla_\theta\mathcal{R}|_{\theta_0} \approx \nabla_\theta\mathcal{R}|_{\theta^*} + \mathbf{H}_{\nabla\mathcal{R}}|_{\theta^*}(\theta_0 - \theta^*) = \mathbf{H}(\theta_0 - \theta^*)$
$(\mathbf{I} - \eta\mathbf{\Lambda})^t \stackrel{!}{=} \mu(\mathbf{\Lambda} + \mu\mathbf{I})^{-1}$ which for $\eta\lambda_i \ll 1, \lambda_i \ll \mu$ can be achieved approximately via performing $t = \frac{1}{\eta\mu}$ steps. Early stopping = approximate L_2 regularizer.

6.3: Dataset Augmentation

- Generate virtual examples by applying transformations τ to each training example $(\mathbf{x}, \mathbf{y})$ to get $(\tau(\mathbf{x}), \mathbf{y})$): e.g. crop, resize, rotate, reflect, add transformations through PCA. - Inject noise: to inputs, to weights (regularizing effect), to targets (soft targets, robustness wrt. label errors)

Semi-supervised training (more unlabeled data)

- define generative model with corresponding log-likelihood
- Opt. additive combination of supervised and unsupervised risk, sharing parameters

Multi-Task Learning

Share representations across tasks and learn jointly (i.e. minimize combined objective); typically: share low level representations, learn high level representations per task.

Ensemble Methods: Bagging

Ensemble method that combines model trained on bootstrap samples (BS); BS $\tilde{\mathcal{S}}_N^k$: sample N times from $\mathcal{S}_N$ with replacement for $k = 1, .., K$; train model on $\tilde{\mathcal{S}}_N^k \to \theta^k$.
Prediction: average model output probabilities $p(\mathbf{y}|\mathbf{x}; \theta^\mathbf{k})$:
$p(\mathbf{y}|\mathbf{x}) = \frac{1}{K}\sum_{k=1}^K p(\mathbf{y}|\mathbf{x}; \theta^k)$

6.4: Dropout

Randomly "drop" subsets of units in network; keep probability π_i^l for unit i in layer l. Typically: $\pi_i^0 = 0.8, \pi_i^{l\geq 1} = 0.5$

Dropout Ensembles

Dropout realizes an ensemble $p(\mathbf{y}|\mathbf{x}) = \sum_\mathbf{Z} p(\mathbf{Z})p(\mathbf{y}|\mathbf{x}; \mathbf{Z})$, where $\mathbf{Z}$ denotes the binary "zeroing" mask.

Weight Rescaling

Approximation to geometrically averaged ensemble, to avoid 10-20× sampling blowup: Scale each weight w_{ij}^l by probability of unit j being active: $\tilde{w}_{ij}^l \leftarrow \pi_j^{l-1} w_{ij}^l$
Make sure, net input to unit i is calibrated, i.e.
$\sum_j \tilde{w}_{ij}^l x_j \stackrel{!}{=} \mathbf{E}_\mathbf{Z}\sum_j z_j^{l-1} w_{ij}^l x_j = \sum_j \pi_j^{l-1} w_{ij}^l x_j$

7.1: Convolutional Layers

Continuous Convolution
$(f * h)(u) := \int_{-\infty}^{\infty} h(u-t)f(t)dt = \int_{-\infty}^{\infty} f(u-t)h(t)dt$
Discrete Convolution
$(f * h)[u] := \sum_{t=-\infty}^{\infty} f[t]h[u-t]$
$(F * G)[i,j] = \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} F[i-k, j-l]\cdot G[k,l]$
Theorem: Any linear, translation-invariant transformation T can be written as a convolution with a suitable h.
Discrete Cross-Correlation (sliding inner product)
$(f \star h)[u] := \sum_{t=-\infty}^{\infty} f[t]h[u+t]$
- Border handling: same padding, valid padding
- "Kernels" (across channels) form a linear map: $h : \mathbf{R}^{r^2\times d} \to \mathbf{R}^k$, where $r \times r$ is the window size (of convolution) and d is the depth (RGB).
- Sub-sampling (strides) to reduce temporal/spatial resolution
- Learn multiple convolution kernels (or filters) = multiple channels

Toeplitz matrix

A matrix $\mathbf{H} \in \mathbf{R}^{k\times n}$ is a Toeplitz matrix, if there exists $n+k-1$ numbers $c_l (l \in [-(n-1) : (k-1)] \cup \mathbf{Z})$ s.t. $H_{i,j} = c_{i-j}$

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \dots & \dots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \dots & \dots & a_2 & a_1 & a_0 \end{bmatrix}$$

That is: $A_{i,j} = A_{i+1,j+1} = a_{i-j}$

$$y = h * x = \begin{bmatrix} h_1 & 0 & \dots & 0 & 0 \\ h_2 & h_1 & \dots & \vdots & \vdots \\ h_3 & h_2 & \dots & 0 & 0 \\ \vdots & h_3 & \dots & h_1 & 0 \\ h_{m-1} & \vdots & \dots & h_2 & h_1 \\ h_m & h_{m-1} & \vdots & \vdots & h_2 \\ 0 & h_m & \dots & h_{m-2} & \vdots \\ 0 & 0 & \dots & h_{m-1} & h_{m-2} \\ \vdots & \vdots & \vdots & h_m & h_{m-1} \\ 0 & 0 & 0 & \dots & h_m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

$\mathbf{H}_n^h \in \mathbf{R}^{(n+m-1)\times n}$.

Backpropagation

Exploit structural sparseness in computing $\frac{\partial x_i^l}{\partial x_j^{l-1}}$
Receptive field of x_i^l: $\mathcal{I}_i^l := \{j : W_{ij}^l \neq 0\}$, where $\mathbf{W}^l$ is the Toeplitz matrix of the convolution; also $\frac{\partial x_i^l}{\partial x_j^{l-1}} = 0$ for $j \notin \mathcal{I}_i^l$
Weight sharing in computing $\frac{\mathcal{R}}{\partial h_j^l}$ where h_j^l is a kernel weight:
$\frac{\mathcal{R}}{\partial h_j^l} = \sum_i \frac{\mathcal{R}}{\partial x_i^l}\frac{\partial x_i^l}{\partial h_i^l}$, weight is re-used for every unit within target layer $\Rightarrow$ additive combination

CNNs (dimension)

CNN input: $H_1 \times W_1 \times C_1$, output of conv. layer with N filters, kernel size K, stride S and zero padding P:
$H2 = (H_1 - K + 2P)/S + 1, W_2 = (W_1 - K + 2P)/S + 1, C_2 = N$
$H2 = (H_1 - K)/S + 1, W_2 = (W_1 - K)/S + 1, C_2 = C_1$

BProp: Single input channel, single output channel

input $x \in \mathbf{R}^{d\times d}$, weights $w \in \mathbf{R}^{k\times k}$
output (before nonlinearity) $y = x * w$
$\frac{\partial\mathcal{L}}{\partial w_{uv}} = \sum_i\sum_j \frac{\partial\mathcal{L}}{\partial y_{ij}}\frac{\partial y_{ij}}{\partial w_{uv}}$
$= \sum_i\sum_j \partial\delta_{ij}\frac{\partial}{\partial w_{uv}}\sum_a\sum_b x_{i-a,j-b}w_{ab}$
$= \sum_i\sum_j \partial\delta_{ij}x_{i-u,j-v}$
$= \sum_i\sum_j \mathrm{rot}_{180}(x_{u-i,v-j})\delta_{ij} = (\mathrm{rot}_{180}(x) * \delta)_{u,v}$
$\frac{\partial\mathcal{L}}{\partial x_{uv}} = \sum_i\sum_j \frac{\partial\mathcal{L}}{\partial y_{ij}}\frac{\partial y_{ij}}{\partial x_{ij}} = \ldots = (\mathrm{rot}_{180}(w) * \delta)_{u,v}$

FFT (compute convolutions faster, $\mathbf{O}(n\log n)$)

$(f * h) = \mathcal{F}^{-1}((\mathcal{F}f))\cdot(\mathcal{F}h))$; pays off if many channels; small kernels ($m < \log n$): favor time/space domain

Convolutional Layers: Stages

Input to layer → Convolution stage: affine transform → Detector stage: nonlinearity (e.g. rectified linear) → pooling stage (locally combine activities) → next layer

Max Pooling

Maximum over a small "patch" of units:
$1D: x_i^{max} = \max\{x_{i+k} : 0 \le k < r\}$
$2D: x_{ij}^{max} = \max\{x_{i+k,j+l} : 0 \le k,l < r\}$
$\mathcal{T}$-invariance through maximization $f_{\mathcal{T}}(\mathbf{x}) := \max_{\tau\in\mathcal{T}} f(\tau\mathbf{x})$
$f_{\mathcal{T}}$ is invariant under $\tau \in \mathcal{T}$: $f_{\mathcal{T}}(\tau\mathbf{x}) = \max_{\rho\in\mathcal{T}} f(\rho(\tau\mathbf{x})) = \max_{\rho\in\mathcal{T}} f((\rho\circ\tau)\mathbf{x}) = \max_{\sigma\in\mathcal{T}} f(\sigma\mathbf{x})$, as $\forall\sigma,\sigma = \rho\circ\tau$ with $\rho = \sigma\circ\tau^{-1}$

8.1: Conv. Networks for Natural Language

Point-wise mutual information (pmi)

$\text{pmi}(v,w) = \log\frac{p(v,w)}{p(v)p(w)} = \log\frac{p(v|w}{p(v)} = \mathbf{x}\top\mathbf{x}_w + \text{const.}$

Skip-gram objective

$\mathcal{L}(\theta;\mathbf{w}) = \sum_{(i,j)\in\mathcal{C}_{\mathcal{R}}} \log\left[\frac{p_\theta(w_i|w_j)}{p(w_i)}\right]$ with
co-occurence index set $\mathcal{C}_R := \{(i,j)\in[1:T]^2 : 1 \le |i-j| \le R\}$

Skip-gram model (soft-max)

$\log p_\theta(v|w) = \mathbf{x}_v^\top\mathbf{z}_w - \log\sum_{u\in\mathcal{V}}\exp[\mathbf{x}_u^\top\mathbf{z}_w]$

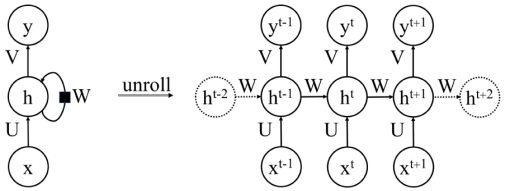
Skip-gram model (negative sampling, logistic regression)

$\mathcal{L}(\theta;\mathbf{w}) = \sum_{(i,j)\in\mathcal{C}_{\mathcal{R}}}\left[\log\sigma(\mathbf{x}_{w_i}^\top\mathbf{z}_{w_j}) + k\mathbf{E}_{v\sim p_n}[\log\sigma(-\mathbf{x}_{w_i}^\top\mathbf{z}_{w_j})]\right]$

8.2: Recurrent Networks

- Markov property
- Time-invariance, share weights

$\bar{F}(h,x;\theta) := Wh + Ux + b$
$y = H(h;\theta), H(h;\theta) := \sigma(Vh + c)$



Backpropagation
$\frac{\partial\mathcal{R}}{\partial w_{ij}} = \sum_{t=1}^T \frac{\partial\mathcal{R}}{\partial h_i^t}\frac{\partial h_i^t}{\partial w_{ij}} = \sum_{t=1}^T \frac{\partial\mathcal{R}}{\partial h_i^t}\cdot\dot{\sigma}_i^t\cdot h_j^{t-1}$
$\frac{\partial\mathcal{R}}{\partial u_{ik}} = \sum_{t=1}^T \frac{\partial\mathcal{R}}{\partial h_i^t}\frac{\partial h_i^t}{\partial u_{ik}} = \sum_{t=1}^T \frac{\partial\mathcal{R}}{\partial h_i^t}\cdot\dot{\sigma}_i^t\cdot x_k^t$
with $\dot{\sigma}_i^t := \sigma'(\bar{F}_i(h^{t-1},x^t))$
MLP: $\nabla_{\mathbf{x}}\mathcal{R} = \mathbf{J}_{F^1}....\mathbf{J}_{F^L}\nabla_{\mathbf{y}}\mathcal{R}$
RNN ($F^t = F$): $\nabla_{\mathbf{x}^t}\mathcal{R} = \left[\prod_{s=t+1}^T \mathbf{W}^\top\mathbf{S}(\mathbf{h}^s)\right]\cdot\mathbf{J}_H\nabla_{\mathbf{y}}\mathcal{R}$
where $\mathbf{S}(\mathbf{h}^s) = diag(\dot{\sigma}_1^s,...,\dot{\sigma}_n^s)$

Loss depends on all outputs

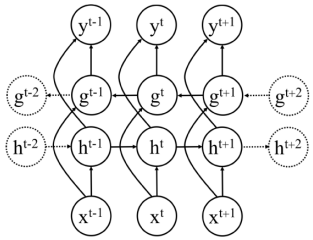
loss $L = \sum_{t=1}^T L_t$, input $\mathbf{x}^t$, state $\mathbf{h}^t$:
$\mathbf{h}^t = F(\mathbf{h}^{t-1},\mathbf{x}^t,\theta) = \alpha(\mathbf{W}\mathbf{h}^{t-1} + \mathbf{U}\mathbf{x}^t + \mathbf{b})$
$\frac{\partial L}{\partial\theta} = \sum_{t=1}^T \frac{\partial}{\partial\theta}L_t$
Sum over all the paths in the (unfolded) network leading from the parameters to the loss: $\frac{\partial L_t}{\partial\theta} = \sum_{k=1}^t \frac{\partial L_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial\theta}$
Expansion along a single path:
$\frac{\partial h_t}{\partial h_k} = \prod_{i=k}^t \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k}^t W^\top\text{diag}(\alpha'(\cdot))$

Loss depends only on last output

$\bar{h}_t = F(x_t, x_{t-1};\theta)$
$h_t = \sigma(t)$
$y_t = G(h_t;\kappa)$
$L_T := L(y_T) + \frac{\lambda}{2}\|\theta\|_2^2$
$\frac{\partial L_T}{\partial\theta} = \frac{\partial L(y_T)}{\partial\theta} + \lambda$
$= \frac{\partial L(y_T)}{\partial y_T}\frac{\partial G(h_T;\kappa)}{\partial h_T}\sum_{k=t}^T \frac{\partial h_T}{\partial h_t}\frac{\partial h_t}{\partial\theta}$
$= \frac{\partial L(y_T)}{\partial y_T}\frac{\partial G(h_T;\kappa)}{\partial h_T}\sum_{k=t}^T \prod\frac{\partial h_T}{\partial h_t}\frac{\partial h_t}{\partial\theta}$

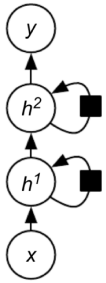
Bi-Directional Recurrent Networks

$g^t = G(x^t, g^{t+1};\theta)$



Deep Recurrent Networks

$h^{t,1} = F^1(h^{t-1,1}, x^t;\theta)$
$h^{t,l} = F^l(h^{t-1,l}, h^{t,l-1};\theta), l = 2,...,L$
$y^t = H(h^{t,L};\theta)$

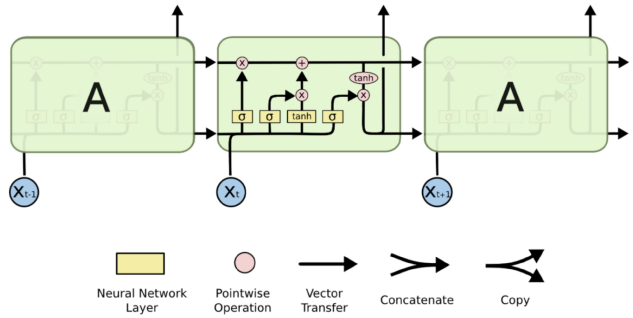


9.1: Memory Units

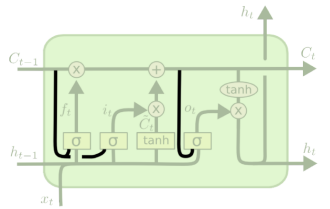
LSTM

input processing (1), input g. (2), forget g. (3), output g. (4)
$F^\kappa = \sigma\circ\bar{F}^\kappa$, $\bar{F}^\kappa = W^\kappa h^{t-1} + U^\kappa x^t + b^\kappa, \kappa\in\{1,2,3,4\}$
Next state: $h^t = F^3(...)\circ h^{t-1} + F^2(...)\circ F^1(...)$
Output: $y^t = F^4(...)\circ\tanh(h^t)$

= building unit for RNN. A common LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate.
$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \in R^h$ (forget gate)
$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \in R^h$ (input gate)
$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \in R^h$ (output gate)
$c_t = f_t\circ c_{t-1} + i_t\circ\sigma_c(W_c x_t + U_c h_{t-1} + b_c)$ (cell state)
$h_t = o_t\circ\sigma_h(c_t)$ (output vector)
$W\in R^{h\times d}$, $U\in R^{h\times h}$ and $\in R^h$: weight matrices and bias vector parameters which need to be learned during training
$x_t\in R^d$: input vector to the LSTM unit

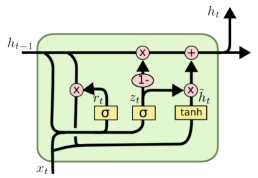


Peephole connections allow the gates to access the constant error carousel (CEC), whose activation is the cell state. h_{t-1} is not used, c_{t-1} is used instead in most places.
$f_t = \sigma_g(W_f x_t + U_f c_{t-1} + b_f)$
$i_t = \sigma_g(W_i x_t + U_i c_{t-1} + b_i)$
$o_t = \sigma_g(W_o x_t + U_o c_{t-1} + b_o)$
$c_t = f_t\circ c_{t-1} + i_t\circ\sigma_c(W_c x_t + b_c)$
$h_t = o_t\circ\sigma_h(c_t)$



Gated Memory Units

Memory state = output (lack output gate)
$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$
$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$
$h_t = z_t\circ h_{t-1} + (1 - z_t)\circ\sigma_h(W_h x_t + U_h(r_t\circ h_{t-1}) + b_h)$



9.2: Differentiable Memory/Neural Turing Machine

- Able to learn to read from and write arbitrary content to memory cells.
- To read, they take a weighted average of many cells.
$r \leftarrow \sum_i \alpha_i M_i, \alpha \geq 0, \sum_i \alpha_i = 1$
- To write, they modify multiple cells by different amounts.
$M_i \leftarrow (1-\beta_i)M_i + \beta_i w, \beta_i \in [0;1]$
- Weights with nonzero derivatives (softmax) enables the functions controlling access to the memory to be optimized using GD.

9.3: Attention

Selectively attend to inputs or feature representations computed from inputs; select what is relevant from the past in hindsight

9.4: Recursive Networks

For a sequence of length τ, the depth can be reduced from τ to $O(\log \tau)$.

10.1: Autoencoders

Linear auto-encoding (hidden layer $\mathbf{z} \in \mathbf{R}^m$, input dimension n, data points $i = 1, ..., k$)
$\mathbf{x} \in \mathbf{R}^n \overset{\mathbf{C}}{\to} \mathbf{z} \in \mathbf{R}^m (m \leq n) \overset{\mathbf{D}}{\to} \hat{\mathbf{x}} \in \mathbf{R}^n \overset{\mathcal{R}}{\to} \frac{1}{2}\|\mathbf{x} - \hat{\mathbf{x}}\|^2$
Optimal choice of $\mathbf{C} \in \mathbf{R}^{n \times m}$ and $\mathbf{D} \in \mathbf{R}^{m \times n}$ s.t.
$\frac{1}{2k}\sum_{k=1}^k \|\mathbf{x}_i - \mathbf{DC}\mathbf{x}_i\|^2 \leftarrow \min$

Eckart-Young Theorem (for $m \leq \min(n,k)$)

$\arg\min_{\hat{\mathbf{X}}: rank(\hat{\mathbf{X}})=m} \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 = \mathbf{U}_m \cdot diag(\sigma_1, ..., \sigma_m) \cdot \mathbf{V}_m^\top$
No linear auto-encoder with m hidden units can improve on SVD as $\text{rank}(CD) \leq m$
Given data $\mathbf{X} = \mathbf{U} diag(\sigma_1, .., \sigma_n)\mathbf{V}^\top$. The choice $\mathbf{C} = \mathbf{U}_m^\top$ and $\mathbf{D} = \mathbf{U}_m$ minimizes the squared reconstruction error of a two layer linear eauto-encoder with m hidden units.
$\tilde{D}\tilde{C} = (U_m A^{-1}) \cdot (A U_m^\top) = U_m U_m^\top$
Solutions restricted to $\mathbf{D} = \mathbf{C}^\top$ (weight-sharing)
$\Rightarrow A^{-1} = A^\top$ (orthogonal)
$\Rightarrow$ mapping $x \to z$ only determined up to rotations.

Non-linear auto-encoder

$\min \mathbf{E}_\mathbf{x}[l(\mathbf{x}, (H \circ G)(\mathbf{x})]$, e.g. $l(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{2}\|\mathbf{x} - \hat{\mathbf{x}}\|^2$
Encoder: $G = F_l \circ \cdots \circ F_1 : \mathbf{R}^n \to \mathbf{R}^m, \mathbf{x} \to \mathbf{z} := \mathbf{x}^l$
Decoder: $H = F_L \circ \cdots \circ F_{l+1} : \mathbf{R}^m \to \mathbf{R}^n, \mathbf{z} \to \mathbf{y} := \hat{\mathbf{x}}$

Denoising non-linear auto-encoder

$\min \mathbf{E}_\mathbf{x}\mathbf{E}_\eta[l(\mathbf{x}, (H \circ G)(\mathbf{x}_\eta)]$ with $\mathbf{x}_\eta = \mathbf{x} + \eta$, $\eta \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$

Denoising non-linear auto-encoder

code sparseness (sparse activity vector): $\Omega(\mathbf{z}) = \lambda\|\mathbf{z}\|_1$
contractive AE (stable wrt. changes in input): $\Omega(\mathbf{z}) = \lambda\|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\|_F^2$

10.2: Factor Analysis

$x = \mu + Wz + \eta$, $z \sim \mathcal{N}(0, I), \eta \sim \mathcal{N}(0, \Sigma)$ then
$x \sim \mathcal{N}(\mu, WW^\top + \Sigma)$, posterior $p(z|x) = \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$
$\mu_{z|x} = (W^\top(WW^\top + \Sigma)^{-1}(x - \mu)$
$\Sigma_{z|x} = I - W\top(WW^\top + \Sigma)^{-1}W$
Pseudo-inverse: $W^\dagger := W^\top(WW^\top + \sigma^2 I)^{-1}$ for $\sigma^2 \to 0$
$W^\dagger = W^\top$ if W orthogonal columns

Moment generating functions

MGF of random vector $\mathbf{x}$: $M_\mathbf{x} : \mathbf{R}^n \to \mathbf{R}, M_\mathbf{x} := \mathbf{E}_x \exp[\mathbf{t}^T\mathbf{x}]$
Uniqueness thereom: If M_x, M_y exist for RVs $\mathbf{x}, \mathbf{y}$ and $M_x = M_y$ then (essentially) $p(\mathbf{x}) = p(\mathbf{y})$.
$\mathbf{E}[x_1^{k_1} \cdot x_n^{k_n}] = \frac{\partial^k}{\partial t_1^{k_1} ... \partial t_n^{k_n}} M_\mathbf{x}|_{t=0}$
$\mathbf{Ex} = \mu$, $\Sigma = \mathbf{E}(\mathbf{x} - \mu)(\mathbf{x} - \mu)^\top$
PDF: $p(\mathbf{x}; \mu, \mathbf{\Sigma}) = \frac{\exp[-\frac{1}{2}(\mathbf{x}-\mu)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)]}{\sqrt{(2\pi)^n \cdot \det \mathbf{\Sigma}}}$,
MGF: $M_x(\mathbf{t}) = \exp[\mathbf{t}^\top\mu + \frac{1}{2}\mathbf{t}^\top\mathbf{\Sigma}\mathbf{t}]$

10.3: Deep Latent Gaussian Models (DLGMs)

Noise variables $\mathbf{z}^l \overset{iid}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}), l = 1, ...L$
Hidden activities (top-down: $\mathbf{h}^L \to \mathbf{h}^1$)
$\mathbf{h}^L = \mathbf{W}^L\mathbf{z}^L$, $\mathbf{h}^l = F^l(\mathbf{h}^{l+1}) + \mathbf{W}^l\mathbf{z}^l$
Hidden layer (conditional) distribution
$h|h^+ \sim \mathcal{N}(F(\mathbf{h}^+), \mathbf{W}\mathbf{W}^\top)$
Recognition model: $\mathbf{x} \overset{\vartheta}{\to} (\mu^l, \mathbf{\Sigma}^l)_{l=1}^L \to q \sim \mathcal{N}(...)$

Jensen's inequality

If g is a real-valued function that is μ-integrable, and if φ is a convex function, then: $\varphi(\int_\Omega g d\mu) \leq \int_\Omega \varphi \circ g d\mu$

ELBO: Evidence lower BOund

$-\log p_\theta(\mathbf{x}) = -\log \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$
$= -\log \int q(\mathbf{x}) \left[p_\theta(\mathbf{x}|\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}\right] d\mathbf{z}$
$\leq -\int q(\mathbf{z}) \log p_\theta(\mathbf{x}|\mathbf{z})d\mathbf{z} + \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})}d\mathbf{z} =: \mathcal{F}(\theta, q; x)$
$D_{KL}(q||p) = \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})} d\mathbf{z}$

$\log p_\theta(\mathbf{x}) \geq \mathbf{E}_q[\log p_\theta(\mathbf{x}, \mathbf{z})] + KL(q(\mathbf{z})||p_\theta(\mathbf{z})) \overset{\max_\theta \mathbf{E}}{\leq} \theta, q$
q variational distr. approx. true intractable posterior $p(\mathbf{z}|\mathbf{x})$
optimal $q(\mathbf{z}; \mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$(posterior)

Stochastic Backpropagation

$\mathbf{z} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$, f: smooth and integrable, then
$\nabla_\mu \mathbf{E} f(\mathbf{z})] = \mathbf{E}[\nabla_\mathbf{z} f(\mathbf{z})]$, $\nabla_\mathbf{\Sigma}\mathbf{E}[f(\mathbf{z})] = \frac{1}{2}\mathbf{E}[\nabla_\mathbf{z}^2 f(\mathbf{z})]$

11.1: Density Estimation

Prescribed model: Use observer likelihoods and assume observation noise
Implicit models: Likelihood-free models

Partition Function

$p(x; \theta) = \frac{1}{Z(\theta)}\tilde{p}(x; \theta) = \frac{1}{\sum_x \tilde{p}(x)}\tilde{p}(x; \theta)$
$\nabla_\theta \log p(x; \theta) = \nabla_\theta \log \tilde{p}(x; \theta) - \nabla_\theta \log Z(\theta)$
$\nabla_\theta \log Z(\theta) = \mathbf{E}_{x \sim p(x)} \nabla_\theta \log \tilde{p}(x)$

Score Matching (Alternative to MLE)

avoids computing quantities related to the partition function
score $= \nabla_x \log p(x)$
Minimize the expected squared difference between the derivatives of the model's log density wrt the input and the derivatives of the data's log density wrt the input:
$\psi_\theta := \nabla \log \bar{p}_\theta, \psi = \nabla \log p$, minimize $J(\theta) = \mathbf{E}\|\psi_\theta - \psi\|^2$
$J(\theta) \overset{\pm c}{=} \mathbf{E}\left[\sum_i \partial_i \psi_{\theta,i} - \frac{1}{2}\psi_{\theta,i}^2\right]$
+ partition function Z is not a function of $x \Rightarrow \nabla_\mathbf{x} Z = 0$
- not applicable to models of discrete data
- need to evaluate $\log \tilde{p}(x)$ and its derivatives
- not compatible if only lower bound available

Noise Constrastive Estimation (NCE)

The probability distribution estimated by the model is represented explicitly as $\log p_{\text{model}}(x) = \log \tilde{p}_{\text{model}}(x; \theta) + c$ where c approximates $-\log Z(\theta)$.
reduces density estimation to binary classification
$\tilde{p}(\mathbf{x}, y = 1) = \frac{1}{2}p_{\text{model}}(x), \tilde{p}(\mathbf{x}, y = 0) = \frac{1}{2}p_{\text{noise}}(\mathbf{x})$
y is a switch variable that determines whether we will generate x from the model or from the noise distribution
prob. classifier: $q_\theta = \frac{\alpha\bar{p}_\theta}{\alpha\bar{p}_\theta + p_n}$, $\alpha > 0$, p_n: constrastive distr.
Bayes optimal if $\alpha\bar{p}_\theta = p$
- does not work with lower bound
+ estimator for θ consistent as long as p_n is dominating p
- Generally not statistically efficient; much worse thatn Cramer-Rao bound if p_n very different from p.

11.2: Generative Adversarial Models

Generator: gen. samples that are indistinguishable from real data.
Train by minimizing logistic likelihood:
$l^*(\theta) := \mathbf{E}_{\tilde{p}_\theta}[y \ln q_\theta(\mathbf{x}) + (1-y)\ln(1 - q_\theta(x))]$
Classification model: $q_\phi : \mathbf{x} \to [0;1], \phi \in \mathbf{\Phi}$
$l^*(\theta) \geq \sup_{\phi \in \Phi} l(\theta, \phi)$
$l(\theta, \phi) := \mathbf{E}_{\tilde{p}_\theta}[y \ln q_\phi(\mathbf{x}) + (1-y)\ln(1 - q_\phi(\mathbf{x}))]$
Optimizing GANs: saddle-point problem:
$\theta^* := \arg\min_{\theta \in \Theta}\{\sup_{\phi \in \Phi} l(\theta, \phi)\}$
explicitly performing inner sup is impractical, iteratively update θ, ϕ with SGD, but may diverge.
$\max_D \min_G V(G, D)$ with
$V(G, D) = \mathbf{E}_{p_{data}(\mathbf{x})} \log D(\mathbf{x}) + \mathbf{E}_{p_g(\mathbf{x})} \log(1 - D(\mathbf{x}))$
$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$
G is optimal when $p_g(x) = p_{data}(x)$, equivalent to optimal discriminator producing 0.5 for all samples drawn from x. In other words, the generator is optimal when the discriminator is maximally confused and cannot distinguish real samples from fake ones.

Trace

$\mathbf{v}^\top\mathbf{w} = \sum_i v_i w_i = \text{Tr}(\mathbf{v}\mathbf{w}^\top)$
$\text{Tr}(\mathbf{A} + \mathbf{B}) = \text{Tr}(\mathbf{A}) + \text{Tr}(\mathbf{B})$
$\mathbf{E}\,\text{Tr}(\mathbf{X}) = \text{Tr}\,\mathbf{E}(\mathbf{X})$
Invariance to transpose operator:
$\text{Tr}(\mathbf{A}) = \text{Tr}(\mathbf{A}^\top)$
Invariance to cyclic permutation:
$\text{Tr}(\mathbf{ABC}) = \text{Tr}(\mathbf{CAB}) = \text{Tr}(\mathbf{BCA})$

Derivatives of Traces

$\nabla_\mathbf{A}\,\text{Tr}(\mathbf{A}\mathbf{A}^\top) = 2\mathbf{A}$
$\nabla_\mathbf{A}\,\text{Tr}(\mathbf{AB}) = \mathbf{B}^\top$
$\nabla_\mathbf{A}\,\text{Tr}(\mathbf{S}\mathbf{A}^{-1}) = -\mathbf{A}^{-1}\mathbf{S}\mathbf{A}^{-1}$

$\nabla_\mathbf{A} \log\det\mathbf{A} = \mathbf{A}^{-1}$

Differentiation rules

power: $\frac{d}{dx}x^n = nx^{n-1}$
product: $\frac{d}{dx}[f(x) \cdot g(x)] = f(x) \cdot \frac{d}{dx}g(x) + g(x) \cdot \frac{d}{dx}f(x)$
quotient: $\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{g(x)\frac{d}{dx}f(x) - f(x)\frac{d}{dx}g(x)}{(g(x))^2}$
chain:
$(f \circ g)' = (f' \circ g) \cdot g'$ or $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = f'(y)g'(x) = f'(g(x))g'(x)$
$\frac{d}{dx}[f(g(x))] = \frac{d}{dx}f(g(x)) \cdot \frac{d}{dx}g(x)$
Schwarz-Theorem: $\frac{\partial^2 f(x,y)}{\partial x \partial y} = \frac{\partial^2 f(x,y)}{\partial y \partial x}$
Leibniz integral rule: $\frac{d}{dx}(\int_a^b f(x,t)dt) = \int_a^b \frac{\partial}{\partial x}f(x,t)dt$

Important functions

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

Tan. hyperbolicus: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Softmax: $\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_k e^{x_k}}$

Softplus: $\zeta(x) = \log(1 + \exp(x))$

Differences

Derivative: $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

Finite differences: $\nabla f(w) = \frac{f(w+\epsilon) - f(w)}{\epsilon} + \mathcal{O}(\epsilon)$

Finite differences (2nd): $f''(x) = \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2}$

Central differences: $\nabla f(w) = \frac{f(w+\epsilon) - f(w-\epsilon)}{2\epsilon}$

Central differences (2nd): $f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$

Taylor series: $f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \frac{f'''(x)}{3!}h^3 ..$

Taylor expansion:

$f(x) \approx f(a) + (x-a)^\top \nabla f(a) + \frac{1}{2!}(x-a)^\top \nabla^2 f(a)(x-a)$

Norms

$\|x\|_p = \sqrt{\sum_i x_i^p} = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$

$\langle x, y \rangle = y^\top x = \sum x_i y_i$

$\|v\| = \sqrt{\langle v, v \rangle}$

$\|\mathbf{A}\|_F := \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2} = \sqrt{\text{Tr}\left(\mathbf{A}\mathbf{A}^\top\right)}$

Probabilities

$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$ (Bayes)

$p(x) = \sum_z p(x|z)p(z)$

$p(b, c) = p(b|c)p(c)$

Expected Value

$\mu = \mathbf{E}(X)$

$Var(X) = \mathbf{E}[(X-\mu)^2 = \mathbf{E}[X^2] - \mathbf{E}[X]^2$

$\mu = \int x f(x) dx$

$Var(X) = \int (x-\mu)^2 f(x) dx$

Inverse (2×2)

$$A^{-1} = \frac{1}{\det} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Hessian

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Entropy

$H(X) = -\sum_i P(x_i) \log_2 P(x_i)$

Conditional entropy

$H(X|Y) = -\sum p(x,y) \log \frac{p(x,y)}{p(y)}$

$= \sum p(x,y) \log \left(\frac{p(x)}{p(x,y)}\right)$

$= -\sum p(x,y) \log(p(x,y)) + \sum p(x,y) \log(p(x))$

$= H(X,Y) + \sum p(x) \log(p(x)) = H(X,Y) - H(X)$

$H(Y|X) = 0$ iff Y is completely determined by X.

$H(Y|X) = H(Y)$ iff Y and X are indep. RVs.

Bayes rule: $H(Y|X) = H(X|Y) - H(X) + H(Y)$

Proof:

- $H(Y|X) = H(X,Y) - H(X)$
- $H(X|Y) = H(Y,X) - H(Y)$
- symmetry implies: $H(X,Y) = H(Y,X)$

Cross entropy

$H(p,q) = E_p[-\log q] = H(p) + D_{KL}(p\|q)$

Discrete p, q: $H(p,q) = -\sum_x p(x) \log q(x) = \sum_x p(x) \log \frac{1}{q(x)}$

logistic function: $g(z) = 1/(1+e^{-z})$

$q_{y=1} = \hat{y} = g(\mathbf{w} \cdot \mathbf{x}) = 1/(1+e^{-\mathbf{w} \cdot \mathbf{x}})$

$q_{y=0} = 1 - \hat{y}$

$H(p,q) = -\sum_i p_i \log q_i = -y \log \hat{y} - (1-y) \log(1-\hat{y})$

Kullback-Leibler divergence

Measure of how one probablity distribution diverges from a second, expected probability distribution.

$D_{KL} = 0$: expect same/similar behavior of two distributions.

$D_{KL} = -\sum_i P(i) \log \frac{Q(i)}{P(i)} = \sum_i P(i) \log \frac{P(i)}{Q(i)}$

$= -\sum p(x) \log q(x) + \sum p(x) \log p(x) = H(P,Q) - H(P)$

Jensen-Shannon divergence

Method of measuring the similarity beteen two probability distributions.

$\text{JSD}(P\|Q) = \frac{1}{2} D_{KL}(P\|M) + \frac{1}{2} D_{KL}(Q\|M)$, $M = \frac{1}{2}(P+Q)$

Minimizing Jensen-Shannon divergence

0 Given: $p(y) = \frac{1}{2}$

1. Entropy = minimizer of cross entropy:

$\min_q \{H(p,q) := -(p \ln q + (1-p) \ln(1-q))\} = H(p)$

ELBO

$\log p(x) = \log \int_z p(x,z) = \log \int_z p(x,z) \frac{q(z)}{q(z)}$

$= \log \left(\mathbf{E}_q \left[\frac{p(x,z)}{q(z)}\right]\right) \geq \mathbf{E}_q \left[\log \frac{p(x,z)}{q(z)}\right] = \mathbf{E}_q[\log p(x,z)] + H(z)$

Jensen's inequality

$f(\mathbf{E}[x]) \leq \mathbf{E}[f(x)]$

Jensen's inequality

$\frac{\partial \mathbf{x}^\top \mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^\top \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}$

$\frac{\partial \mathbf{a}^\top \mathbf{X} \mathbf{b}}{\partial \mathbf{X}} = \mathbf{a}\mathbf{b}^\top$

$\frac{\partial \mathbf{a}^\top \mathbf{X}^\top \mathbf{b}}{\partial \mathbf{X}} = \mathbf{b}\mathbf{a}^\top$

$\frac{\partial \mathbf{b}^\top \mathbf{X}^\top \mathbf{X} \mathbf{c}}{\partial \mathbf{X}} = \mathbf{X}(\mathbf{b}\mathbf{c}^\top + \mathbf{c}\mathbf{b}^\top)$

$\frac{\partial (\mathbf{B}\mathbf{x} + \mathbf{b})^\top \mathbf{C} (\mathbf{D}\mathbf{x} + \mathbf{d})}{\partial \mathbf{x}} = \mathbf{B}^\top \mathbf{C}(\mathbf{D}\mathbf{x} + \mathbf{d}) + \mathbf{D}^\top \mathbf{C}^\top (\mathbf{B}\mathbf{x} + \mathbf{b})$

$\frac{\partial \mathbf{x}^\top \mathbf{B} \mathbf{x}}{\partial \mathbf{x}} = (\mathbf{B} + \mathbf{B}^\top)\mathbf{x}$

$\frac{\partial \mathbf{b}^\top \mathbf{X}^\top \mathbf{D} \mathbf{X} \mathbf{c}}{\partial \mathbf{X}} = \mathbf{D}^\top \mathbf{X} \mathbf{b}\mathbf{c}^\top + \mathbf{D}\mathbf{X}\mathbf{c}\mathbf{b}^\top$