AI Manifesto: Appreciation of Mathematics Underlying Machine Learning Concepts

Batuhan Karaca

May 25, 2024

Contents

1 Introduction

This book aims to give theoretical background of the machine learning concepts. It is not mandatory, but recommended to have basic knowledge in linear algebra and probability theory and some familiarity with the concepts (e.g. through courses).

Unless stated otherwise, these math notations apply. Capital letters represent matrices (or hypermatrices in general) and random variables. Lowercase symbols denote vectors and scalars. Scalar (or a row/column vector) of a matrix is denoted with corresponding lowercase symbol. Pre-superscript on a symbol of a matrix denotes the shape of the matrix. Subscript on a scalar variable symbol denotes the index of that symbol in its corresponding matrix.

2 Linear Algebra Used In Learning and Optimization

2.1 Proving that a matrix of the form X^TX is positive semi-definite

Let A be a symmetric square matrix, and also $A = X^TX$. For any vector z,

$$z^TAz = z^TX^TXz = (Xz)^T(Xz) \geq 0 \tag{2.1.1}$$

Since squared norm is non-negative, $A = X^TX$ is positive semi-definite.

2.2 Proving the Existence of a (Trivially Unique) Orthonormal Basis of Eigenvectors for a Symmetric Diagonalizable Matrix

Let A be a diagonalizable symmetric square matrix. Let P be a square matrix such that it's ith column v_i is the ith eigenvector of A. Let Λ be the diagonal matrix whose ith diagonal λ_i is the eigenvalue corresponding to v_i. According to definition of eigenvalues/eigenvectors.

$$AP = P\Lambda \tag{2.2.1}$$
$$A = P\Lambda P^{-1} \tag{2.2.2}$$
$$\Lambda = P^{-1}AP \tag{2.2.3}$$
$$\lambda_i v_i = Av_i \tag{2.2.4}$$
$$\lambda_i v_i^T = v_i^TA^T \tag{2.2.5}$$
$$\lambda_i v_i^T v_j = v_i^TA^Tv_j \tag{2.2.6}$$
$$= v_i^TAv_j \tag{2.2.7}$$
$$= \lambda_j v_i^T v_j \tag{2.2.8}$$
$$(\lambda_i - \lambda_j)v_i^Tv_j = 0 \tag{2.2.9}$$

Since A is diagonalizable, $\lambda_i = \lambda_j$ if and only if $i = j$. Hence if $i \neq j$, $v_i^Tv_j = 0$ and P is orthogonal, hence $PP^T = D$ such that

$$D_{ij} = \begin{cases} |v_i|^2 & i = j \\ 0 & otherwise \end{cases} \tag{2.2.10}$$

We can have $P' = PD^{-\frac{1}{2}}$, and $\Lambda' = \Lambda D = D^{\frac{1}{2}}\Lambda D^{\frac{1}{2}}$. Then

$$A = P'\Lambda' P'^T \tag{2.2.11}$$

such that P' is orthonormal. We are going to reference P' and Λ' as P and Λ respectively. We have shown that any symmetric diagonalizable matrix has orthonormal basis of eigenvectors.

2.3 Relation Between Definiteness and Sign of the Eigenvectors

$$z^TAz = z^TP\Lambda P^Tz \tag{2.3.1}$$
$$= (P^Tz)^T\Lambda(P^Tz) \tag{2.3.2}$$
$$= \sum_i (P^Tz)_i^2\lambda_i \tag{2.3.3}$$
$$= \sum_i z_i'^2\lambda_i \tag{2.3.4}$$

The term is actually sum of eigenvalues weighted by values of z', which has the same norm as z due to the fact that P is orthonormal. For all z (hence z'), the weighted sum is non-negative if and only if A is semi-positive definite. Non-negativity in this case is guaranteed when all eigenvectors are non-negative. This approach also works for other types of definite matrices (we cannot say anything about indefinite matrices).

2.4 Relation Between Invertability and Definiteness

For any diagonalizable matrix A,

$$det(A) = det(P)det(\Lambda)det(P^{-1}) \tag{2.4.1}$$
$$= det(P)det(\Lambda)\frac{1}{det(P)} \tag{2.4.2}$$
$$= det(\Lambda) \tag{2.4.3}$$
$$= \prod_i \lambda_i \tag{2.4.4}$$

We see that the semi-definite matrices are not invertible since they have at least one zero eigenvalue. We have shown that semi-definite matrices are not invertible and definite matrices are invertible.

2.5 Definiteness of Hessian Matrix to Determine Curvature Near Critical Points

2.5.1 Quadratic Function

Let $q(x)$ be a quadratic function with ${}^{d\times d}A, {}^{d\times 1}x, {}^{d\times 1}b$ and a scalar c

$$q(x) = x^T Ax + x^T b + c \tag{2.5.1}$$

of the form where A is symmetric.

$$\nabla q(x) = (A + A^T)x + b \tag{2.5.2}$$
$$= 2Ax + b \tag{2.5.3}$$

The critical point $x^* = -\frac{1}{2}A^{-1}b$ where $\nabla q(x^*) = 0$.

$$q(x^* + \Delta x) = q(x^*) + \Delta x^T A\Delta x + 2\Delta x^T Ax^* + \Delta x^T b \tag{2.5.4}$$
$$= q(x^*) + \Delta x^T A\Delta x \tag{2.5.5}$$

We can deduce from previous results,

$$\forall \Delta x[q(x^*) < q(x^* + \Delta x)] \text{ if } A \text{ is positive definite } (x^* \text{ is minimum, minimization problem}) \tag{2.5.6}$$
$$\forall \Delta x[q(x^*) > q(x^* + \Delta x)] \text{ if } A \text{ is negative definite } (x^* \text{ is maximum, maximization problem}) \tag{2.5.7}$$

If A is indefinite, then depending on Δx, $q(x^*)$ can be less or greater than $q(x^* + \Delta x)$, which means x^* is a saddle point. Since semi-definite matrices are not invertible we cannot use $x^* = -\frac{1}{2}A^{-1}b$. If A is semi-definite, for some x, we have $q(x) = x^T b + c$ (a hyperplane with a $d+1$-dimensional normal having scalars of b and having 1 in the dimension of a reference axis, i.e. inclined-plane). If b is zero, then we have infinitely many critical points (a hyperplane with a normal along the reference axis, i.e. ground-plane).

Let d be an arbitrary vector.

$$\nabla q(x^* + Pd) = 2A(Pd + x^*) + b \tag{2.5.8}$$
$$= 2(P\Lambda P^T)(Pd + x^*) + b \tag{2.5.9}$$
$$= 2P\Lambda P^T Pd + 2Ax^* + b \tag{2.5.10}$$
$$= 2P\Lambda d \tag{2.5.11}$$
$$= \sum_i 2d_i\lambda_i v_i \tag{2.5.12}$$
$$\nabla q(x^* + Pd)^T \nabla q(x^* + Pd) = (2P\Lambda d)^T 2P\Lambda d \tag{2.5.13}$$
$$= 4d^T \Lambda P^T P\Lambda d \tag{2.5.14}$$
$$= 4d^T \Lambda^2 d \tag{2.5.15}$$
$$= \sum_i 4d_i^2\lambda_i^2 \tag{2.5.16}$$
$$= \tilde{c}^2 \quad \text{For an arbitary scalar } \tilde{c} \tag{2.5.17}$$
$$\sum_i \frac{d_i^2}{\frac{\tilde{c}^2}{4\lambda_i^2}} = 1 \tag{2.5.18}$$

We have shown that contours (curves where gradient magnitudes/norms are equal) are ellipsoids.

$$\nabla q(x^* + \frac{\tilde{c}}{2\lambda_i} v_i) = \tilde{c} v_i \tag{2.5.19}$$

in a contour with multiplier $\tilde{c}$. The eigenvector v_i is a unit direction along a principal semi-axis of the ellipsoid. Note that in order to reach to a contour with multiplier $\tilde{c}$ from x^*, the magnitude/norm of the vector is $\frac{\tilde{c}}{2\lambda_i}$. As λ_i increases, the magnitude/norm decreases, and vice-versa.

For any function, $f(x)$ its second order Tailor expansion approximates it in the neighbor of x.

$$f(x + \Delta x) \sim f(x) + \Delta x^T \nabla f(x) + \Delta x^T H(x) \Delta x \tag{2.5.20}$$

where $H(x) = \nabla^2 f(x)$ is Hessian of $f(x)$. Substituting A, b and c, this approximation can be used to determine the behavior of f near x in optimization algorithms such as gradient descent (where Δx is learning rate times the gradient). For example if $H(x)$ is semi-definite, this time we have a plateau, as plateau is a plane near x (ground or inclined).

2.5.2 Arbitrary Gaussian Function

Let $g(x) = he^{q(x)}$, with an arbitrary scalar h.

$$\nabla g(x) = \nabla q(x) g(x) \tag{2.5.21}$$

We see that the critical points of $g(x)$ are the same as of $q(x)$.

$$g(x^* + \Delta x) = he^{q(x^* + \Delta x)} \tag{2.5.22}$$
$$= he^{q(x^*) + \Delta x^T A \Delta x} \tag{2.5.23}$$
$$= e^{\Delta x^T A \Delta x} g(x^*) \tag{2.5.24}$$

Similarly from previous subsection

$$\forall \Delta x [g(x^*) < g(x^* + \Delta x)] \text{ if } A \text{ is positive definite } (x^* \text{ is minimum, minimization problem}) \tag{2.5.25}$$
$$\forall \Delta x [g(x^*) > g(x^* + \Delta x)] \text{ if } A \text{ is negative definite } (x^* \text{ is maximum, maximization problem}) \tag{2.5.26}$$

Same as the previous subsection, If A is indefinite, then depending on Δx, $g(x^*)$ can be less or greater (saddle point). If A is semi-sefinite, for some infinitely many x, we have $g(x) = e^{x^T b + c}$ (inclined-plane). If

b is zero, then we have infinitely many critical points (ground-plane). Otherwise, we have no critical points at all.

$$\nabla g(x^* + Pd) = \nabla q(x^* + Pd) g(x^* + Pd) \tag{2.5.27}$$
$$= (2P\Lambda d) e^{(Pd)^T A (Pd)} g(x^*) \tag{2.5.28}$$
$$= (2P\Lambda d) e^{d^T P^T A P d} g(x^*) \tag{2.5.29}$$
$$= (2P\Lambda d) e^{d^T \Lambda d} g(x^*) \tag{2.5.30}$$
$$\nabla g(x^* + Pd)^T \nabla g(x^* + Pd) = (\sum_i 4 d_i^2 \lambda_i^2) e^{2 d^T \Lambda d} g(x^*)^2 \tag{2.5.31}$$
$$= (\sum_i 4 d_i^2 \lambda_i^2) e^{\sum_i 2 d_i^2 \lambda_i} g(x^*)^2 \tag{2.5.32}$$
$$(\sum_i 4 d_i^2 \lambda_i^2) e^{\sum_i 2 d_i^2 \lambda_i} g(x^*)^2 = \tilde{c}^2 \tag{2.5.33}$$

Note that after this point, I could not come with a rigorous proof, but used a computer. If we use density function of normal distribution which is a special case of $g(x)$ with $b = -2A\mu$ (x^* becomes μ), $c = \mu^T A \mu$, $h = \frac{1}{\sqrt{(2\pi)^n det(A)}} = \frac{1}{\sqrt{(2\pi)^n \prod_j \lambda_j}}$ (to have infinite integral equal 1 by definition of density functions), and inputting the values to a calculator (I used Desmos), the function behaves like an ellipsoid having eigenvectors along its principal semi-axes. please note that $A = -\frac{1}{2}\Sigma^{-1}$, where covariance matrix Σ needs to be positive definite (it can trivially be semi-positive definite in the limit for generalization). Therefore, A is negative definite, having all negative eigenvalues. As I increased the eigenvalues, the length of the principal semi-axis corresponding to the eigenvector has decreased. This analysis gives ellipsidicity information of the Gaussian that is fitted to a normally distributed sample.

3 Probability Theory Concepts Used In Learning

3.1 Distinction Between Correlation and Dependence

We have seen the relationship between eigenvectors/values and the covariance matrix. Assume that we fitted a continuous normal distribution to an elliptic sample of data in order to estimate the most probable regions of occurence, the properties of the continuous distribution will also approximate the properties of the sample. If we squish the sample more, the data will be better approximated with a more squished Gaussian. Remember from the previous parts along the axis of compression, the eigenvalue corresponding to that axis will become larger. At some point it will be the largest eigenvalue and the data will be approximated by a hyperplane with the normal being its eigenvector (1D hyperplane is a line). We say, such data has a *linear dependence*. However, dependence -relation between axes/dimensions/features in the data- is a more general concept and the covariance matrix only gives information about linear dependence. There can be infinitely many other types such as quadratic, cubic and so on. Hence, correlation is a subset of dependence.

3.1.1 Arbitrary Gaussian Function

By definition, if

$$p(A|B,C) = p(A|C) \tag{3.1.1}$$

then A and B are conditionally independent given C. We also know that

$$p(D|E)P(E) = p(D,E) \tag{3.1.2}$$

for any E, F. Then given previous conditions

$$p(A|B,C)p(B|C) = p(A,B|C) \tag{3.1.3}$$
$$p(A|C)p(B|C) = p(A,B|C) \tag{3.1.4}$$

Now assume that for a vector of input x and vector output (label) y, for $i \neq j$, y_i, y_j are conditionally independent given x (1), and x_i, y_i are conditionally independent given x_j (2). You can think of the vector as the data and pairs (x, y) as points such that the point y_i is label of x_i.

$$p(y|x) = \prod_i p(y_i|x) \tag{3.1.5}$$

$$= \prod_i p(y_i|x_i) \tag{3.1.6}$$

If we use natural language, then we can say (as also proven above) y_i depends only on x_i. This is the foundational assumption that people build their learning models on. Note that the input x_i can be further divided into features x_ij that could be correlated among each other. Then we can further remove those features in the equation above, still having $p(y|x)$.

People use dimension reduction algorithms such as PCA to find those correlated features for removal. PCA simply finds the line having eigenvector with the largest eigenvalue as its normal. Than merges the features affected by this eigenvector. One may iteratively drop the features until the desired dimension.

3.2 Derivation of Cross-Entropy and KL-Divergence via Maximum-Likelihood

We know that when training with given parameters θ and input x, learning models choose the combination of parameters that yields the maximum output y_{max}, the process known as the maximum likelihood estimation. We assume that y_i can only be a positive integer in the range $[1, C]$, where C is the maximum number of classes. In this case, y_i represents the class of the input x_i. For $p(y_i|x_i, \theta)$ we come up with the most generalized discrete generalized distribution, categorical distribution. It is actually generalized Bernoulli distribution including non-binary random variables. Mathematically

$$p(y_i) = \prod_j p(y_i = j)^{1\{y_i=j\}} \tag{3.2.1}$$

$$1\{y_i = j\} = \begin{cases} 1 & y_i = j \\ 0 & otherwise \end{cases} \tag{3.2.2}$$

$$\hat{\theta}_{MLE} = \arg_\theta \max p(y|x, \theta) \tag{3.2.3}$$

$$= \arg_\theta \max \prod_i p(y_i|x_i, \theta) \tag{3.2.4}$$

$$= \arg_\theta \max \prod_i \prod_j p(y_i = j|x_i, \theta)^{1\{y_i=j\}} \tag{3.2.5}$$

$$-\log\hat{\theta}_{MLE} = \arg_\theta \min \sum_i \sum_j -\log p(y_i = j|x_i, \theta)^{1\{y_i=j\}} \tag{3.2.6}$$

$$= \arg_\theta \min \sum_i \sum_j -(1\{y_i = j\}) \log p(y_i = j|x_i, \theta) \tag{3.2.7}$$

We assume there is an underlying data distribution that we try to approximate with a model distribution. Simple substitution gives

$$-\log\hat{\theta}_{MLE} = \arg_\theta \min \sum_i \sum_j -p_{data}(y_i = j|x_i) \log p_{model}(y_i = j|x_i, \theta) \tag{3.2.8}$$

$$= \arg_\theta \min[H(p_{data}(y|x), p_{model}(y|x, \theta))] \tag{3.2.9}$$

$$= \arg_\theta \min[\sum_i \sum_j p_{data}(y_i = j|x_i) \log \frac{p_{data}(y_i = j|x_i)}{p_{model}(y_i = j|x_i, \theta)} + \sum_i \sum_j -p_{data}(y_i = j|x_i) \log p_{data}(y_i = j|x_i)] \tag{3.2.10}$$

$$= \arg_\theta \min[KL(p_{data}(y_i = j|x_i) \;||\; p_{model}(y_i = j|x_i, \theta)) + H(p_{data}(y_i = j|x_i))] \tag{3.2.11}$$

$$= \arg_\theta \min[KL(p_{data}(y_i = j|x_i) \;||\; p_{model}(y_i = j|x_i, \theta))] \tag{3.2.12}$$

$H(p,q), H(p)$ and $KL(p \mid\mid q)$ denote cross-entropy, Shannon's entropy and KL-divergence respectively. Shannon's entropy cancels out since it does not depend on parameters. We have shown that maximizing likelihood will minimize the cross-entropy, which minimizes KL-divergence between data and model distributions. In this regard, the model distribution tries to approximate the data distribution when training.

3.3 Derivation of Mean Squared Error via Maximum-Likelihood

We again use maximum likelihood. However, this time we assume y_i is a real number with noise $\epsilon_i \sim \mathcal{N}(0, I)$ around the real output intended (by the nature :D).

$$y_i = f(x_i, \theta) + \epsilon_i \tag{3.3.1}$$

$$\hat{\theta}_{MLE} = \arg_\theta \max \prod_i p(y_i|x_i, \theta) \tag{3.3.2}$$

$$= \arg_\theta \max \prod_i \frac{1}{2\pi} e^{-\frac{\epsilon_i^2}{2}} \tag{3.3.3}$$

$$= \arg_\theta \max \prod_i \frac{1}{2\pi} e^{-\frac{1}{2}(f(x_i,\theta)-y_i)^2} \tag{3.3.4}$$

$$-\log\hat{\theta}_{MLE} = \arg_\theta \min [\sum_i \log 2\pi + \sum_i \frac{1}{2}(f(x_i, \theta) - y_i)^2] \tag{3.3.5}$$

$$= \arg_\theta \min \sum_i \frac{1}{2}(f(x_i, \theta) - y_i)^2 \tag{3.3.6}$$

In this case, maximizing likelihood will minimize the mean-squared error. Deriving the formulae for both of the important functions in learning, we see a pattern in which we try to minimize a loss function.

$$\hat{\theta}_{MLE} = \arg_\theta \min \sum_i \mathcal{L}(\hat{y_i}, y_i) \tag{3.3.7}$$

3.4 Preference of Mean over Summation

It is easy to prove that the mean (expected value) has the super-position property

$$\mu_{aX+bY} = \int\int p(X,Y)(aX + bY)dXdY \tag{3.4.1}$$

$$= a\int(\int p(X,Y)dY)X)dX + b\int(\int p(X,Y)dX)Y)dY \tag{3.4.2}$$

$$= a\int p(X)XdX + b\int p(Y)YdY \tag{3.4.3}$$

$$= a\mu_X + b\mu_Y \tag{3.4.4}$$

$$Cov(aX + bY, cZ + dT) = \int\int\int\int p(X,Y,Z,T)(aX + bY - \mu_{aX+bY})(cZ + dT - \mu_{cZ+dT})dXdYdZdT \tag{3.4.5}$$

$$= \int\int\int\int p(X,Y,Z,T)(a(X - \mu_X) + b(Y - \mu_Y))(c(Z - \mu_Z) + d(T - \mu_T))dXdYdZdT \tag{3.4.6}$$

$$\begin{aligned} = \int\int\int\int [&p(X,Y,Z,T)ac(X - \mu_X)(Z - \mu_Z) + ad(X - \mu_X)(T - \mu_T) + \\ &bc(Y - \mu_Y)(Z - \mu_Z) + bd(Y - \mu_Y)(T - \mu_T)]dXdYdZdT \end{aligned} \tag{3.4.7}$$

$$
\begin{aligned}
&= ac\int\int(\int\int p(X,Y,Z,T)dYdT)(X-\mu_X)(Z-\mu_Z)dXdZ+\\
&\quad ad\int\int(\int\int p(X,Y,Z,T)dYdZ)(X-\mu_X)(T-\mu_T)dXdT+\\
&\quad bc\int\int(\int\int p(X,Y,Z,T)dXdT)(Y-\mu_Y)(Z-\mu_Z)dYdZ+\\
&\quad bd\int\int(\int\int p(X,Y,Z,T)dXdZ)(Y-\mu_Y)(T-\mu_T)dYdT
\end{aligned}
\tag{3.4.8}
$$

$$
\begin{aligned}
&= ac\int\int p(X,Z)(X-\mu_X)(Z-\mu_Z)dXdZ+\\
&\quad ad\int\int p(X,T)(X-\mu_X)(T-\mu_T)dXdT+\\
&\quad bc\int\int p(Y,Z)(Y-\mu_Y)(Z-\mu_Z)dYdZ+\\
&\quad bd\int\int p(Y,T)(Y-\mu_Y)(T-\mu_T)dYdT
\end{aligned}
\tag{3.4.9}
$$

$$= acCov(X,Z) + adCov(X,T) + bcCov(Y,Z) + bdCov(Y,T) \tag{3.4.10}$$

We assume the equation

$$Cov(\sum_{i=1}^{m-1} a_iX_i, \sum_{i=1}^{n-1} b_iY_i) = \sum_{i=1}^{m-1}\sum_{j=1}^{n-1} a_ib_jCov(X_i,Y_j) \tag{3.4.11}$$

holds.

$$Cov(\sum_{i=1}^{m} a_iX_i, \sum_{i=1}^{n} b_iY_i) = Cov(\sum_{i=1}^{m-1} a_iX_i + a_mX_m, \sum_{i=1}^{n-1} b_iY_i + b_nY_n) \tag{3.4.12}$$

$$= Cov(\sum_{i=1}^{m-1} a_iX_i, \sum_{i=1}^{n-1} b_iY_i) + b_nCov(\sum_{i=1}^{m-1} a_iX_i, Y_n) + a_mCov(X_m, \sum_{i=1}^{n-1} b_iY_i) + a_nb_mCov(X_m,Y_n) \tag{3.4.13}$$

$$= \sum_{i=1}^{m-1}\sum_{j=1}^{n-1} a_ib_jCov(X_i,Y_j) + b_n\sum_{i=1}^{m-1} a_iCov(X_i,Y_n) + a_m\sum_{j=1}^{n-1} b_jCov(X_m,Y_j) + a_nb_mCov(X_m,Y_n) \tag{3.4.14}$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{n} a_ib_jCov(X_i,Y_j) \tag{3.4.15}$$

If we assume all the values (including zero) that the coefficients can take, then using mathematical induction, we prove that the equation (3) holds.

$$Var(\sum_{i=1}^{n} a_iX_i) = Cov(\sum_{i=1}^{n} a_iX_i, \sum_{i=1}^{n} a_iX_i) \tag{3.4.16}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} a_ia_jCov(X_i,X_j) \tag{3.4.17}$$

$$= \sum_{i=1}^{n} a_i^2Cov(X_i,X_i) + \sum_{i=1}^{n}\sum_{j\neq i}^{n} a_ia_jCov(X_i,X_j) \tag{3.4.18}$$

$$= \sum_{i=1}^{n} a_i^2Var(X_i) + \sum_{i=1}^{n}\sum_{j\neq i}^{n} a_ia_jCov(X_i,X_j) \tag{3.4.19}$$

$$= \sum_{i=1}^{n} a_i^2Var(X_i) \quad \text{(Since we assumed } X \text{ is uncorrelated)} \tag{3.4.20}$$

Then for a sample with constant variance σ^2 the variance of the sample mean,

$$Var(\overline{X}) = Var(\frac{1}{n}\sum_{i=1}^{n} X_i) \tag{3.4.21}$$

$$= \frac{1}{n^2}\sum_{i=1}^{n} Var(X_i) \tag{3.4.22}$$

$$= \frac{1}{n^2}\sum_{i=1}^{n} \sigma^2 \tag{3.4.23}$$

$$= \frac{1}{n^2} n\sigma^2 \tag{3.4.24}$$

$$= \frac{\sigma^2}{n} \tag{3.4.25}$$

$$Std(\overline{X}) = \sqrt{Var(\overline{X})} \tag{3.4.26}$$

$$= \frac{\sigma}{\sqrt{n}} \tag{3.4.27}$$

We realize that as number of samples increases, the mean of the sample approximates the true mean of the distribution better. That is why people usually use mean instead of sum. Hence new expression for the loss function becomes

$$\hat{\theta}_{MLE} = \frac{1}{n}\arg_\theta\min\sum_{i=1}^{n} \mathcal{L}(\hat{y}_i, y_i) \tag{3.4.28}$$

As n is a constant relative to the parameters, we are still maximizing the likelihood.

4 Gradients of Functions in Backpropagation

4.1 FC (Fully Connected) Layers

An FC layer implements the function on a set of matrices

$$^{N\times M}Y = {}^{N\times D}X\,{}^{D\times M}W + {}^{N\times 1}1\,{}^{1\times M}b \tag{4.1.1}$$

Using the definition of matrix multiplication, we have

$$y_{nm} = \sum_{d=1}^{D} x_{nd}w_{dm} + b_m \tag{4.1.2}$$

Looking at this definition, we can find partial derivatives

$$\frac{\delta y_{nm}}{\delta x_{ij}} = \begin{cases} w_{jm} & if\ n = i \\ 0 & otherwise \end{cases} \tag{4.1.3}$$

$$\frac{\delta y_{nm}}{\delta w_{ij}} = \begin{cases} x_{ni} & if\ m = j \\ 0 & otherwise \end{cases} \tag{4.1.4}$$

$$\frac{\delta y_{nm}}{\delta b_j} = \begin{cases} 1 & if\ m = j \\ 0 & otherwise \end{cases} \tag{4.1.5}$$

Using the chain rule with a loss function gives

$$L_{x_{ij}} = \sum_{n=1}^{N}\sum_{m=1}^{M} L_{y_{nm}} \frac{\delta y_{nm}}{\delta x_{ij}} \tag{4.1.6}$$

$$= \sum_{m=1}^{M} L_{y_{nm}} w_{jm} \tag{4.1.7}$$

$${}^{N\times D}L_X = {}^{N\times M}L_Y \, {}^{M\times D}W^T \tag{4.1.8}$$

$$L_{w_{ij}} = \sum_{n=1}^{N}\sum_{m=1}^{M} L_{y_{nm}} \frac{\delta y_{nm}}{\delta w_{ij}} \tag{4.1.9}$$

$$= \sum_{n=1}^{N} L_{y_{nm}} x_{ni} \tag{4.1.10}$$

$${}^{D\times M}L_W = {}^{D\times N}X_T \, {}^{N\times M}L_Y \tag{4.1.11}$$

$$L_{b_j} = \sum_{n=1}^{N}\sum_{m=1}^{M} L_{y_{nm}} \frac{\delta y_{nm}}{\delta b_j} \tag{4.1.12}$$

$$= \sum_{n=1}^{N} L_{y_{nm}} \tag{4.1.13}$$

$${}^{1\times M}L_b = {}^{1\times N}1 \, {}^{N\times M}L_Y \tag{4.1.14}$$

5 Diffusion Models

5.1 What is a Diffusion Model

Diffusion models require two processes that adds noise to data for a number of steps T, (*Sampling steps* parameter in AUTOMATIC1111/stable-diffusion-webui and alike). The first one, *forward (diffusion) process*, makes data noisier by adding Gaussian noise T times, whereas the second, *reverse process*, tries to recover the original image from the result by adding noise again T times.

5.2 Solving for the Forward Process

Let *forward process* be defined as a Markov Chain as in the original paper:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{5.2.1}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{5.2.2}$$

where $\mathcal{N}(x; \mu, \Sigma)$ is the Gaussian with mean μ and Σ is the covariance matrix (notice when $\Sigma = \sigma^2 I$, then every dimension has the same variance $sigma^2$). Using the reparameterization trick as in the VAE (Variational Autoencoder), paper and substituting $\alpha = 1 - \beta_t$ we have

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \tag{5.2.3}$$

where $\epsilon_\tau \sim \mathcal{N}(0, I)$ is sampled from the standard normal distribution. Then

$$x_t = (\sqrt{\prod_{\tau=1}^{t}\alpha_\tau})x_0 + \sum_{\tau=0}^{t-2}(\sqrt{\prod_{i=\tau+2}^{t}\alpha_i - \prod_{i=\tau+1}^{t}\alpha_i})\epsilon_\tau + (\sqrt{1-\alpha_t})\epsilon_{t-1} \tag{5.2.4}$$

Let us prove this argument using mathematical induction. When $t = 1$, it is easy to see equations (1) and (2) are the same. For an arbitrary t

$$x_{t-1} = (\sqrt{\prod_{\tau=1}^{t-1} \alpha_\tau})x_0 + \sum_{\tau=0}^{t-3}(\sqrt{\prod_{i=\tau+2}^{t-1} \alpha_i - \prod_{i=\tau+1}^{t-1} \alpha_i})\epsilon_\tau + (\sqrt{1-\alpha_{t-1}})\epsilon_{t-2} \tag{5.2.5}$$

$$(\sqrt{\alpha_t})x_{t-1} = (\sqrt{\prod_{\tau=1}^{t} \alpha_\tau})x_0 + \sum_{\tau=0}^{t-3}(\sqrt{\prod_{i=\tau+2}^{t} \alpha_i - \prod_{i=\tau+1}^{t} \alpha_i})\epsilon_\tau + (\sqrt{\alpha_t - \alpha_t\alpha_{t-1}})\epsilon_{t-2} \tag{5.2.6}$$

$$= (\sqrt{\prod_{\tau=1}^{t} \alpha_\tau})x_0 + \sum_{\tau=0}^{t-2}(\sqrt{\prod_{i=\tau+2}^{t} \alpha_i - \prod_{i=\tau+1}^{t} \alpha_i})\epsilon_\tau \tag{5.2.7}$$

$$(\sqrt{\alpha_t})x_{t-1} + (\sqrt{1-\alpha_t})\epsilon_{t-1} = (\sqrt{\prod_{\tau=1}^{t} \alpha_\tau})x_0 + \sum_{\tau=0}^{t-2}(\sqrt{\prod_{i=\tau+2}^{t} \alpha_i - \prod_{i=\tau+1}^{t} \alpha_i})\epsilon_\tau + (\sqrt{1-\alpha_t})\epsilon_{t-1} = x_t \tag{5.2.8}$$

Therefore, by induction we have proven that the equation (2) is true. We should have an equation of the form

$$x_t = \mu + \sigma \cdot \epsilon \tag{5.2.9}$$

We found that $\mu = (\sqrt{\prod_{\tau=1}^{t} \alpha_\tau})x_0 = \sqrt{\tilde{\alpha}_t}x_0$. By the property of *sum of normally distributed random variables* if

$$X_i \sim \mathcal{N}(x_i; \mu_i, \sigma_i^2 I) \tag{5.2.10}$$

and

$$Y = \sum_{i=1}^{n} c_i X_i \tag{5.2.11}$$

then

$$Y \sim N(y; \sum_{i=1}^{n} c_i\mu_i, \sum_{i=1}^{n} c_i^2\sigma_i^2) \tag{5.2.12}$$

We know ϵ is normally distributed. Substituting $X_i = \epsilon_i$ and the corresponding coefficients

$$\sum_{\tau=0}^{t-2}(\prod_{i=\tau+2}^{t} \alpha_i - \prod_{i=\tau+1}^{t} \alpha_i) + (1-\alpha_t) = \prod_{i=2}^{t} \alpha_i - \prod_{i=1}^{t} \alpha_i + \prod_{i=3}^{t} \alpha_i - \prod_{i=2}^{t} \alpha_i + ... + \prod_{i=t-1}^{t} \alpha_i - \prod_{i=t-2}^{t} \alpha_i + \prod_{i=t}^{t} \alpha_i - \prod_{i=t-1}^{t} \alpha_i + (1-\alpha_t) \tag{5.2.13}$$

$$= \prod_{i=2}^{t} \alpha_i - \prod_{i=1}^{t} \alpha_i + \prod_{i=3}^{t} \alpha_i - \prod_{i=2}^{t} \alpha_i + ... + \prod_{i=t-1}^{t} \alpha_i - \prod_{i=t-2}^{t} \alpha_i + \alpha_t - \prod_{i=t-1}^{t} \alpha_i + 1 - \alpha_t \tag{5.2.14}$$

Note that similar elements cancel each other and we find $\sigma^2 = 1 - \prod_{i=1}^{t} \alpha_i = 1 - \tilde{\alpha}_t$, finding

$$x_t = \sqrt{\tilde{\alpha}_t}x_0 + \sqrt{1-\tilde{\alpha}_t} \cdot \epsilon \tag{5.2.15}$$

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\tilde{\alpha}_t}x_0, (1-\tilde{\alpha}_t)I) \tag{5.2.16}$$

as given in the paper.

5.3 Solving for the Reverse Process and the Loss Function

We found a closed form solution for the forward process function. However, reverse process will be learnt by a machine learning algorithm. The authors decided to use a type of CNN (Convolutional Neural Network), U-Net.

Let *reverse process* be defined as another Markov Chain (backward in time) as in the original paper:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{5.3.1}$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{5.3.2}$$