Note Sheet

Inner Product: $u^T A v = \langle u, v \rangle$. A is symmetric and positive definite. Dot Product is an inner product with $A = I$. Notice that a symmetric positive definite matrix has all positive eigenvalues and positive determinant. It is linear in each argument.

Orthogonal Projection: We have the projecton $\pi(x)$ onto some subspace W. We want to minimize $||x - \pi(x)||^2$. By the orthogonality condition, we have $\langle x - \pi(x), b \rangle = 0$. Since π is a linear map, we can write $\pi(x) = P_\pi x$ thus $\langle x - P_\pi x, b \rangle = 0 \implies \langle x - \lambda b, b \rangle = 0$. This results in $\lambda = \frac{\langle x,b \rangle}{\langle b,b \rangle}$. $\lambda = (B^T B)^{-1} B^T x$, $\pi(x) = B(B^T B)^{-1} B^T x$. and $P = B(B^T B)^{-1} B^T$.

SVD: Decompose $A = U \Sigma V^T$. Notice $A^T A = V \Sigma U^T U \Sigma V^T = V \Sigma^2 V^T$. And $AA^T = U \Sigma V^T V \Sigma U^T = U \Sigma^2 U^T$. $U =$ eigenvectors of AA^T. $V =$ eigenvectors of $A^T A$. $\Sigma =$ square root of eigenvalues of $A^T A$ in decending order. Notice that if A is $m \times n$ then U is $m \times m$, Σ is $m \times n$, and V is $n \times n$. A geometric intuition for SVD is that U is the ON basis of the column space of A, V is the ON basis of the row space of A, and Σ is the scaling factor. You can also Write $A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$ where r is the rank of A. Thus the SVD is a chnage of basis in the domain, then an independent scaling, and then a change of basis in the codomain.

Best Fit Line: With n points (x_i, y_i), minimize $\sum_{i=1}^{n} (y_i - \theta_1 x_i - \theta_0)^2$. We can see that the best fit line is the orthogonal projection of $(y_1, y_2, \ldots, y_n)$ onto the subspace spanned by $(\vec{1}, \vec{x})$

Reaching Line of best fit: for a 2 dimensional space, we can take the IPM of x_i as $\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}$. ie $B^T B$ where $B = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$. Solve $B^T B \Theta = B^T Y$ where $Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$. Using Cramer's rule for matrix inverses, we can find Θ as: $\theta_0 = \frac{(\sum y_i)(\sum x_i^2) - (\sum x_i)(\sum x_i y_i)}{n \sum x_i^2 - (\sum x_i)^2}$ and $\theta_1 = \frac{n \sum x_i y_i - (\sum x_i)(\sum y_i)}{n \sum x_i^2 - (\sum x_i)^2}$. In terms of IP it is $\theta_0 = \frac{\langle x,x \rangle \langle 1,y \rangle - \langle 1,x \rangle \langle x,y \rangle}{\langle 1,1 \rangle \langle x,x \rangle - \langle 1,x \rangle^2}$ and $\theta_1 = \frac{\langle 1,1 \rangle \langle x,y \rangle - \langle 1,x \rangle \langle 1,y \rangle}{\langle 1,1 \rangle \langle x,x \rangle - \langle 1,x \rangle^2}$. Note this abstracted version works with any inner product.

Differentiation Diff is a linear map. The gradient of a function is the vector of partial derivatives. $\nabla_x f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}$. The directional derivative is $\nabla_x f \cdot v$ where v is the direction. The Jacobian is the matrix of partial derivatives. $J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$. When we consider the line of best fit the derivate of our function $||Ax - b||^2$ is $2(Ax - b)^T A$. Setting this to 0 gives us the equation $A^T A x = A^T b$. Note that a jacobian of $f(x) = Ax$ is A.

Identities for Gradients: Short hand: $\frac{\partial}{\partial X} f = f_X$:

$$\begin{aligned}
(f(X)^T)_X &= (f(X)_X)^T \\
(\mathrm{tr} f(X))_X &= \mathrm{tr}(f(X)_X) \\
(\det f(X))_X &= \det(f(X))\mathrm{tr}(f(X)^{-1} f(X)_X) \\
(f(X)^{-1})_X &= -f(X)^{-1} f(X)_X f(X)^{-1} \\
(a^T X^{-1} b)_X &= -(X^{-1})^T ab^T (X^{-1})^T \\
(x^T a)_x &= a^T \\
(a^T x)_x &= a^T \\
(a^T X b)_X &= ab^T \\
(x^t B x)_x &= x^T (B + B^T) \\
((x - As)^T W (x - As))_s &= -2(x - As)^T WA
\end{aligned}$$

Chain Rule: $f(g(x))' = f'(g(x))g'(x)$. We can use this for multivariable functions as well. $\Delta_x f(g(x)) = \Delta_{g(x)} f \cdot \Delta_x g$.

Backpropgation: $f_0 = x$ and $f_i = \sigma(A_{i-1} f_{i-1} + b_{i-1})$. We also have a loss function $L(\theta) = ||y - f_K(\theta, x)||^2$ which we want to minimize. We can use the chain rule to find the gradient of L with respect to θ. We have:

$$\begin{aligned}
\Delta_\theta L &= \Delta_{f_K} L \cdot \Delta_\theta f_K \\
&= \Delta_{f_K} L \cdot \Delta_{f_{K-1}} f_K \cdot \Delta_\theta f_{K-1} \\
&= \Delta_{f_K} L \cdot A_{K-1}^T \sigma'(A_{K-1} f_{K-2} + b_{K-1}) \Delta_\theta f_{K-1}
\end{aligned}$$

Example (Steps of tuning a weight)**.** Simple steps to tune a weight in a neural network:

1. Initialize weights
2. Feed forward (compute the value of each neuron in the network)
3. Compute loss ($L(\theta) = ||y - f_K(\theta, x)||^2$)
4. Backpropagate (compute the gradient of the loss with respect to each weight)
5. Update weights ($\theta = \theta - \alpha \Delta_\theta L$ where α is the learning rate)
6. Repeat untill convergence