Neural Tangent Kernels (NTK)

☆ Looking back at classic kernels

$$x_i \in \mathbb{R}^d \longrightarrow \underbrace{\phi(x_i) \in \mathbb{R}^D}_{\text{Transforming our features to a higher dimension (D) using } \phi} \qquad D >> d$$

Non-linear

Ex - $d=3$ $\quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \longrightarrow \phi(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_1 x_2 \\ x_2 x_3 \\ \vdots \end{bmatrix}$

Now, $\boxed{f(w, x) = w^T \phi(x)}$

w: weight

x: features

$\phi(x)$: Transformed features

{ignore bias for simplicity}

This model is linear in w but not in x (due to non-linear Φ)

{We will still have a convex optimisation problem}

Gradient descent

$$w(t+1) = w(t) - \eta_t \nabla \mathcal{L}(w(t))$$

$$\sum_{i=1}^{n}(y_i - f(w, x_i))\underbrace{\nabla_w f(w(t), x_i)}_{(x_i)}$$

static (fixed)

As $\mathcal{L}(w) = \frac{1}{2}(y_i - \underbrace{f(w, x_i)}_{\hat{y}_i})^2$

Limitations

1. Φ is fixed

2. $\phi(x) \in \mathbb{R}^D$ $(D >> d)$

$\left\{\text{High computational cost}\right\}$

☆ Kernel trick

→ Usually we don't have to worry about computing $\phi(x)$s, rather we look at –

$$\langle \phi(x_i), \phi(x_j) \rangle$$

$\langle \rangle \rightarrow$ inner product (generalisation of dot product)

$V = [\quad]$, $U = [\quad]$

$\langle V, U \rangle = V^T U \rightarrow$ scalar quantity

$$\boxed{K(x_i, x_j) \triangleq \langle \phi(x_i), \phi(x_j) \rangle}$$

☆

Kernel value b/w x_i & x_j

$$\underbrace{\mathbb{K}}_{\text{Kernel matrix}} \in \mathbb{R}^{m \times n} = \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix} \geq 0$$

(i, j) element corresponds to $k(x_i, x_j)$

☆☆☆ $\mathbb{K} \geq 0$ & $\mathbb{K}$ is symmetric usually can be computed without "explicitly" computing $\Phi(x)$s

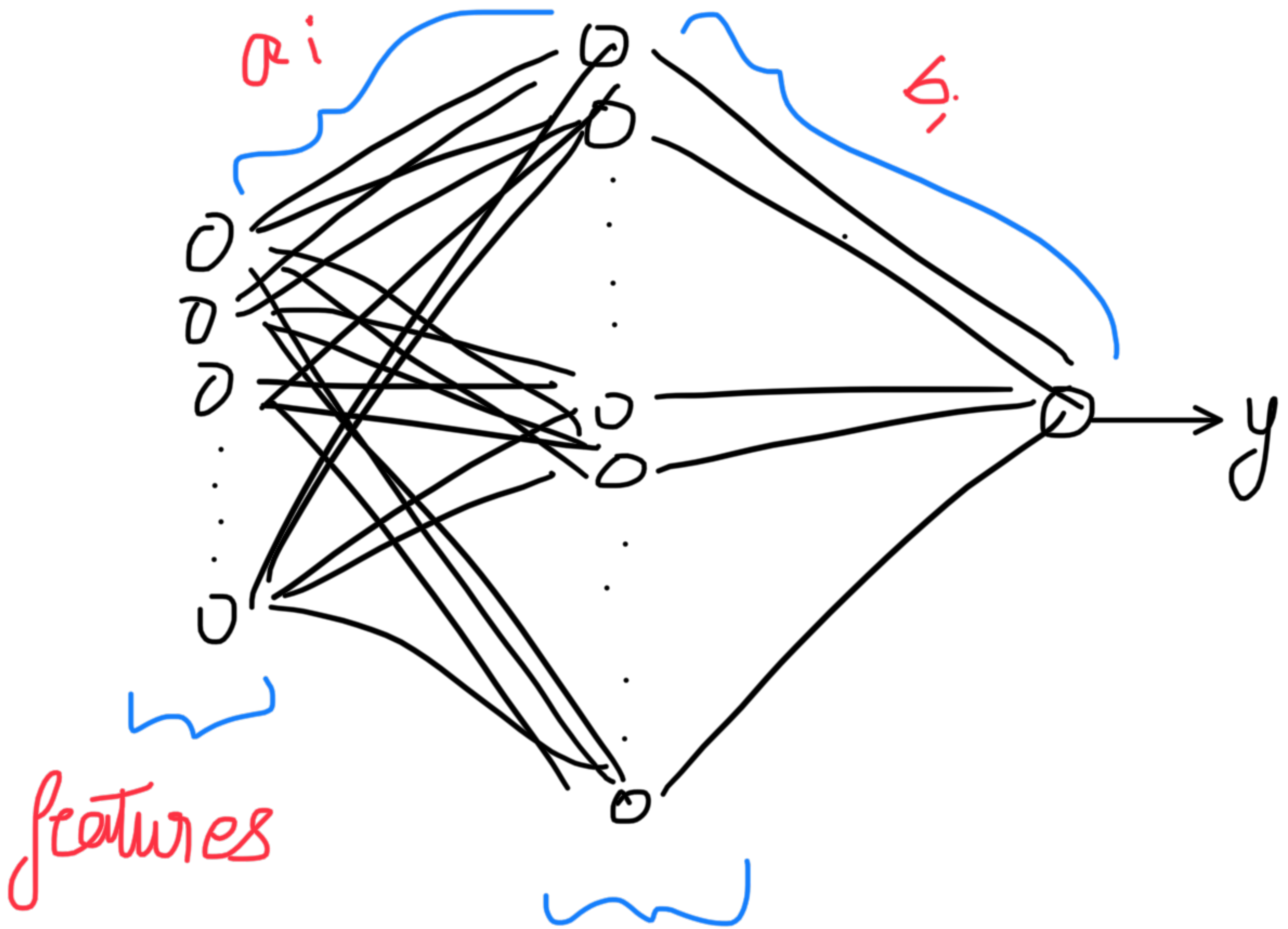
Example -

1. Polynomial Kernel

$$K(x_i, x_j) = (c + x_i^t x_j)^K$$

Neural Tangent Kernels (NTK)

Neural Tangent Kernels (NTKs)



only 1 hidden layer with

m neurons

Let $nn = y = f(w, x)$

for simplicity ignore bias Terms

$$y = f(w, x) = \sum_{i=1}^{m} b_i \, \sigma(a_i^t x)$$

$$y = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} b_i \, \sigma(a_i^t x)$$

Ahead (why?)

Now, taking SSE Loss

$$\mathcal{L}(w) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

Minimising $\mathcal{L}(w)$ using Gradient descent

$$w(t+1) = w(t) - \eta_t \nabla \mathcal{L}$$

Dynamic $(w(t))$

{Static in Linear regression}

$$\nabla L = \sum_{i=1}^{n} \left(f(w, x_i) - y_i\right) \underbrace{\nabla_w f(w(t), x_i)}_{\text{depends on } w(t) \text{ hence dynamic}}$$

Lets unitalize weights using

Normal distribution —

$$W_{init} \sim N(0, 1)$$

$$\underbrace{w_0 \rightarrow w_1 \rightarrow w_2 \rightarrow \cdots\cdots w_n}_{\text{Gradient descent}}$$

When $m \rightarrow$ very large

— [illegible] of Neural Network

$W_0, W_1, W_2 \ldots W_n$ remain almost static

This is known as "Lazy" Training
(Empirical observation)

Writing first order Taylor's series around W_0 —

$\phi(x)^T$

$$f(w,x) \sim f(w_0,x) + \overbrace{\nabla_w f(w_0,x)}^{T}(w-w_0)\ldots$$

$\rightarrow$ linear in w but not linear in x

$\Rightarrow$ Similar to the kernel method

$$\phi(x) \triangleq \nabla_w f(w_0,x)$$

$$\underbrace{K(x_i,x_j) = \langle \phi(x_i), \phi(x_j) \rangle}$$

Neural Tangent Kernel (NTK)

Finding NTK for our NeuralNet

$$f_m(w,x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} b_i \, \sigma(a_i^T x)$$

No. of neurons

$$\nabla_{a_i} f_m(w,x) = \frac{1}{\sqrt{m}} b_i \, \sigma'(a_i^T x)\, x$$

$$\nabla_{b_i} f_m(w,x) = 1 \;\; \sigma(a_i^T x)$$

$$b_i \ldots \frac{\quad}{\sqrt{m}}$$

$$K_m(x, x') = K_m^a(x, x') + K_m^b(x, x')$$

NTK

$$K_m(x, x') =$$

$$\sum_{i=1}^{m} \frac{1}{m} b_i^2 \, \sigma'(a_i^T x) \, \sigma'(a_i^T x) \, (x \cdot x')$$

☆ $$\boxed{K_m(x, x') = \frac{1}{m} \sum^{m} \sigma(a_i^T x) \, \sigma(a_i^T x') \ldots}$$

☆ $\boxed{\quad m \quad i=1 \quad}$

Let a_{i_s} and b_{i_s} be independent samples

$$K_m^a(x, x') \xrightarrow{m \to \infty} K^a(x, x') = E\left(b^2 \sigma(a^T x)\sigma(a^T x')(x \cdot x')\right)$$

Expectation

$$K_m^b(x, x') \xrightarrow{m \to \infty} K^b(x, x') = E\left(\sigma(a^T x)\sigma(a^T x')\right)$$

NTKs for NN with infinite width $(m \to \infty)$

☆ Gradient flow

$$w(t+1) = w(t) - \eta \nabla_w \mathcal{L}(w)$$

$\eta \to 0$ $\left\{ \text{Condition for gradient flow} \right\}$

$$\frac{w(t+1)-w(t)}{\eta} = -\nabla_w \mathcal{L}(w(t))$$

$\eta \to 0$

$$\underbrace{\frac{d\,w(t)}{dt} = -\nabla_w \mathcal{L}(w(t))}_{\text{ODE}}$$

ODE

Gradient flow dynamics equation

Let $\mathcal{L}(w) = \frac{1}{2} \| \hat{y}(w) - y \|^2$

$$\nabla_w \mathcal{L}(w) = \nabla_w \hat{y}(w) (\hat{y}(w) - y)$$

$$\frac{dw(t)}{dt} = - \nabla_w \hat{y}(w)(\hat{y}(w) - y)$$

$$\frac{d\hat{y}(w(t))}{dt} = \underbrace{-\nabla_w}_{} \hat{y}(w(t))^T \frac{dw(t)}{dt}$$

Chain Rule

$$\frac{d\hat{y}^n(w(t))}{dt} = -\nabla_w \hat{y}^n(w(t))^T X \nabla_w \hat{y}^n(w)\left(\hat{y}^n(w) - y\right)$$

$$\downarrow$$

$$\mathrm{NTK} \equiv K(w_0)$$

$$\star\star \quad \boxed{\frac{d\hat{y}^n}{dt} \approx -K(w_0)\left(\hat{y}^n(w(t)) - y\right)}$$

Let

Let $u = y - y$

$$\frac{du}{dt} \approx -k(w_0)\,u$$

$$\Rightarrow \boxed{u(t) = u(0)\,e^{-k(w_0)t}} \quad ☆$$