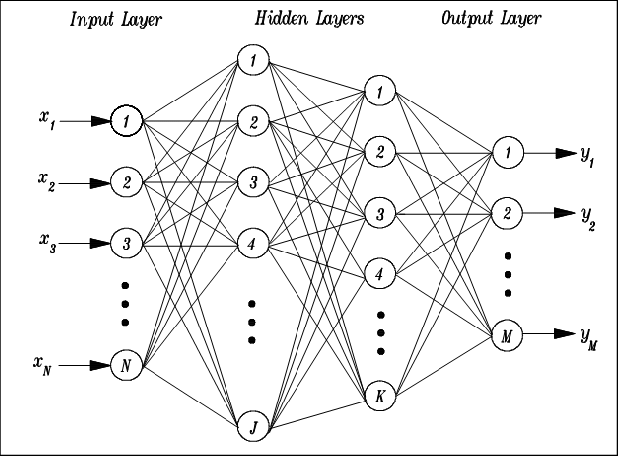
**4. Vícevrstvý perceptron**

**4.1 Inspirace a historický kontext**

*Vícevrstvý perceptron* (MLP – *multilayer perceptron*) je typ hluboké neuronové sítě, která se skládá z neuronů uspořádaných ve vrstvách. Po představení MCP modelu neuronu a následné první implementaci Rosenblattova perceptronu v padesátých letech začal rychle upadat zájem o neuronové sítě. Nikdo totiž nedokázal vymyslet efektivní způsob, jakým trénovat více perceptronu ve vrstvách. V roce 1986 však D.E.Rumelhart, G.E. Hinton a R.J. Wiliams ve své práci zpopularizovali algoritmus *zpětného šíření* (*backpropagation*)1 a obnovili tak ztracený zájem nejen většiny vědecké komunity, ale později i mnoha amatérů. V posledních několika letech došlo k obrovským pokrokům ve výzkumu hlubokých neuronových sítí obecně, hlavně díky posunu ve výrobě výkonných grafických karet, které jsou klíčové pro náročné výpočty. Se složitějšími architekturami tak dnes mohou pracovat i naprostí amatéři pouze za použití svého počítače.

**4.2 Architektura MLP**



Obr. 4.1 Příkladné schéma architektury vícevrstvého perceptronu (MLP)

První dvě kapitoly této práce se týkaly jednovrstvých neuronových sítí (perceptron a ADALINE). MLP je na rozdíl od těchto algoritmů typově *hluboká neuronová síť* (DNN – *Deep Neural Network*), jelikož se skládá z více vrstev. Zároveň se jedná o *hustě propojenou* (*dense*) síť, takže každý neuron jedné vrstvy bude propojen se všemi neurony druhé vrstvy. Hluboké sítě, které nejsou cyklické, označujeme jako *dopředné* (*feedforward*).

**4.2.1 Vrstvy sítě**

Vrstvy neuronových sítí slouží k předávání a transformaci vstupních dat pomocí aktivačních funkcí. Existuje mnoho způsobů, jakými se dají tyto vrstvy nastavit podle zamýšleného využití (například rozpoznávání visuálních nebo zvukových dat, klasifikace, či generování nových dat). Vrstvou se rozumí soubor neuronů se stejnými aktivačními funkcemi, které jsou zároveň propojeny s neurony vedlejších vrstev (v případě husté sítě plně). MLP obsahuje (ostatně jako všechny hluboké sítě) *vstupní vrstvu* (*input layer*), *výstupní vrstvu* (*output layer*) a jednu nebo více *skrytých vrstev* (*hidden layers*). Neuronová síť je označována za hlubokou právě tehdy, když má alespoň dvě skryté vrstvy. Vstupní vrstva slouží k přebrání vstupních dat, tedy příznaků, zatímco výstupní vrstva naopak transformuje vstupní data na požadované hodnoty (například můžeme chtít jako výstup pravděpodobnosti cílových tříd). Skryté vrstvy jsou přidávány právě kvůli nelinearitě a jejich přesnou funkci vysvětlím později. Na obrázku 4.1 je vidět schéma MLP se dvěma skrytými vrstvami. Opět zdůrazňuji, že se jedná o síť hustou, takže každý neuron jedné vrstvy je propojen se všemi neurony druhé vrstvy.

**4.2.2 Neurony a synapse**

Neurony v jednotlivých vrstvách MLP fungují naprosto stejně jako perceptron (odsud název vícevrstvý perceptron). Vstupní data jsou sítí předávána jako příznaky daného příkladu, takže počet neuronů ve vstupní vrstvě odpovídá počtu příznaků. Na obrázku 4.2 je vidět hustá síť s jednou skrytou vrstvou (vstupní vrstva je plně propojena se skrytou vrstva, ta je zase plně propojena s vrstvou výstupní). Počet neuronů ve skrytých vrstvách je volitelný, počet neuronů ve výstupní vrstvě je obvykle určen počtem tříd (cílová třída je poté určena neuronem s nejvyšší hodnotou). Synapse, kterými jsou jednotlivé neurony propojeny, představují váhy, které se budeme Diagram

Description automatically generatedsnažit optimalizovat během učení sítě (stejně jako u ADALINE a perceptronu).

Obr. 4.2 Hustě propojená síť s jednou skrytou vrstvou

**4.3 Dopředené šíření**

Proces trénování sítě ve smyslu předávání vstupních dat je obvykle označován jako *dopředné šíření* (*forward propagation*). Jedná se postupné transformování dat pomocí jednotlivých vrstev. Pro zjednodušení budu celý tento proces vysvětlovat na MLP s jednou skrytou vrstvou, avšak postup je v tomto případě obecný a lze ho aplikovat i na hluboké sítě tohoto typu.

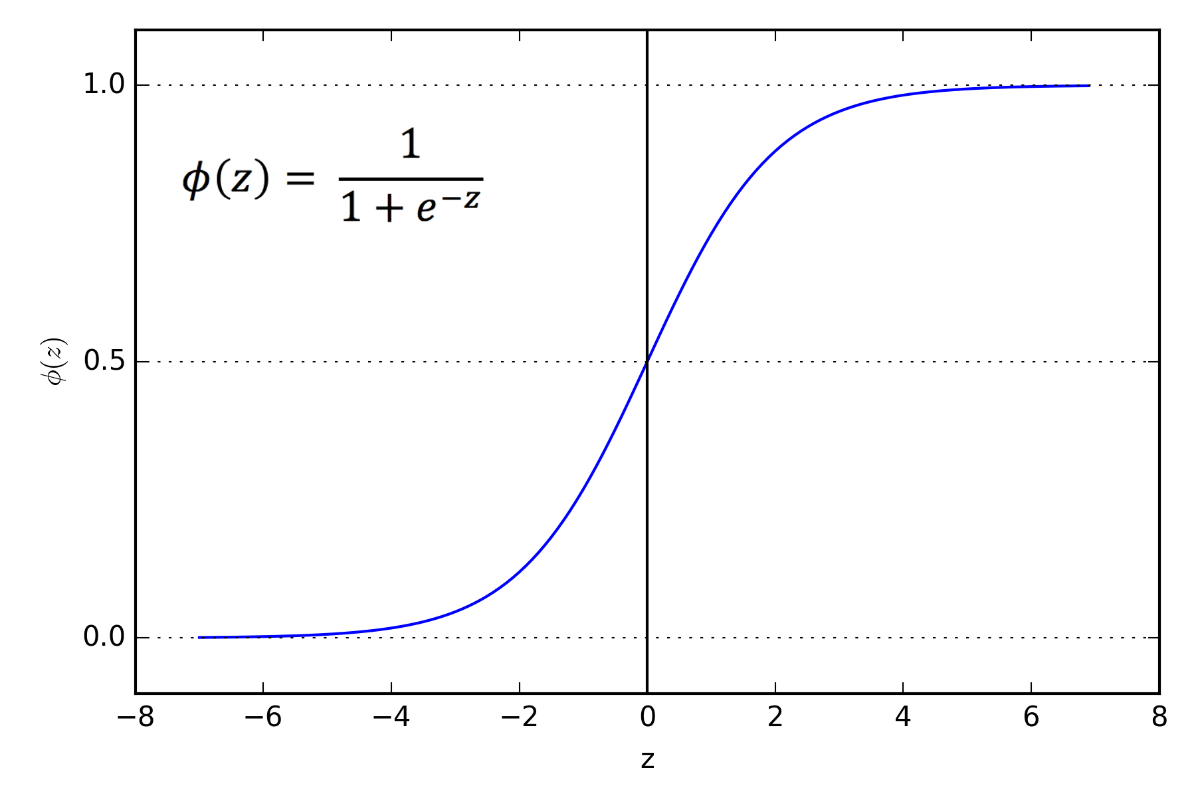
**4.3.1 Aktivace skryté vrstvy**

Hodnoty neuronů **A(in)** ve vstupní vrstvě jsou identické s hodnotami příznaků **X(i)**:

Jedinou výjimkou je první neuron **a0(in)**, který obvykle představuje bias. Počet neuronů **A(in)** ve vstupní vrstvě tak odpovídá počtu příznaků **X(i) + 1.** Prvním krokem při dopředném šíření je výpočet hodnot neuronů ve skryté vrstvě. Stejně jako u perceptronu musíme nejdříve vypočítat lineární vstup daného neuronu, a poté na něj aplikovat aktivační funkci. Stejně jako vstupní vrstva, skrytá též obsahuje jednotku biasu **a0(h)**. Jelikož se jedná o síť hustou, k výpočtu lineárního vstupu **zj(h)** libovolného neuronu **aj(h)** ve skryté vrstvě sečteme násobky všech neuronů **A(in)** vstupní vrstvy (včetně biasu **a0(in)**) a jejich příslušných vah **Wi,j(h)** (příslušnou váhou se myslí konkrétní synapse mezi oběma neurony):

nebo:

V dalším kroku aplikujeme na lineární vstup aktivační funkci . Často používanou aktivační funkcí v případě MLP je například logistická aktivační funkce (sigmoida), která transformuje vstup na hodnotu od 0 do 1 (jedná se tzv. logistickou regresi):

****

Obr. 4.3 Graf logistické aktivační funkce (sigmoidy)

Hodnotu libovolného neuronu skryté vrstvy můžeme tedy zaspat jako:

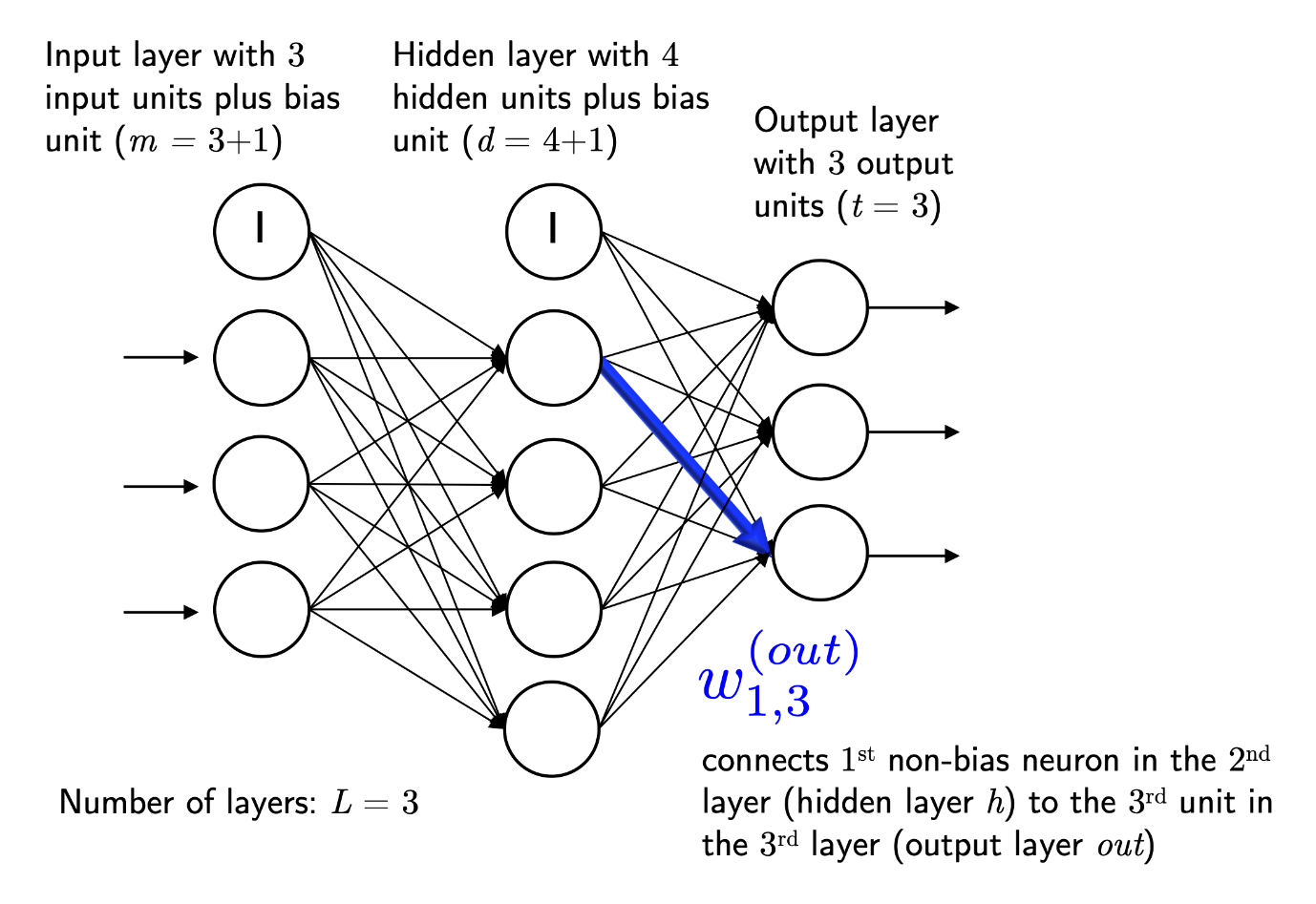
**4.3.2 Aktivace výstupní vrstvy**

Výpočet hodnot neuronů **A(out)** výstupní vrstvy je naprosto stejný jako u skryté vrstvy. Výstupní vrstva samozřejmě na rozdíl od vstupní a skryté neobsahuje bias, ale pouze výstupní neurony **A(out)**. Opět připomínám, že MLP je hustá síť, takže všechny neurony **A(out)** výstupní vrstvy jsou propojeny se všemi neurony **A(h)** předchozí skryté vrstvy. Hodnota libovolného výstupního neuronu je tedy opět:

Je důležité zvolit správnou aktivační funkci . Její výběr je určen tvarem výstupu, který od sítě požadujeme. Můžeme například opět použít logistickou aktivační funkci – výstupní neuron s nejvyšší hodnotou bude poté představovat predikovanou třídu. Pokud bychom však chtěli vidět procentní pravděpodobnosti jednotlivých neuronů **A(out)** výstupní vrstvy (tento přístup je obvykle označován jako *pravděpodobnostní hluboké učení* – *probabilistic deep learning*), bylo by výhodnější použít funkci *softmax*. Ta sice stejně jako sigmoida transformuje vstup na hodnotu od 0 do 1, ale zároveň bere do úvahy všechny neurony (v případě výstupní vrstvy se tak jedná o ostatní třídy). Součet hodnot všech neuronů ve vrstvě s aktivační funkcí softmax bude 1, takže použijeme-li tuto funkci ve výstupní vrstvě místo sigmoidy, můžeme hodnoty neuronů interpretovat jako pravděpodobnosti:

Proměnná ***K***ve jmenovateli vzorce představuje počet tříd. Celý jmenovatel je tzv. *normalizační termín* (*normalization term*), který je zodpovědný za onu pravděpodobnostní distribuci (součet výsledných hodnot vyjde jako 1). V případě určování více než 2 tříd se s použitím softmax jedná o *mnohonásobnou logistickou regresi* (*multinomial logistic regression*).9

Poté, co data projdou všemi vrstvami sítě, je potřeba vypočítat ztrátové skóre a následně posílit nebo oslabit synapse mezi neurony (optimalizovat váhové koeficienty).



Obr. 4.4 Schéma MLP s jednou skrytou vrstvou se zvýrazněnými biasy

**4.4 Logistická ztrátová funkce**

Jak jsem již vysvětloval ve 2. kapitole, jedním z hlavních pilířů učení neuronových sítí je ztrátová funkce. Ta slouží k výpočtu ztrátového skóre, které se následně snažíme minimalizovat různými metodami optimalizace pro co nejlepší výkon sítě – v případě ADALINE se jednalo o jednoduchou kvadratickou sumu odchylek. Samozřejmě nejsme vázání pouze jednou ztrátovou funkcí na každý typ sítě, v praxi se s výběrem hodně experimentuje. Při MLP můžeme použít například logistickou ztrátovou funkci **:**

Výraz představuje logistickou sigmoid aktivaci *i*-tého příkladu1, o které jsem mluvil v podkapitole 4.3.1. Tuto funkci ještě musíme upravit tak, aby brala v úvahu všechny aktivační jednotky **t** naší sítě:

**4.4.1 Regularizace ztrátové funkce**

Abychom snížili riziko přetrénování sítě, můžeme na ztrátovou funkci aplikovat takzvanou L2 regularizaci:

Neboli:

Logistická ztrátová funkce s L2 regularizací bude tedy vypadat takto:

Neboli:

**+**

Zjednodušeně se jedná pouze o přičtení součtu všech váhových koeficientů vrstvy ***l***. Použité L2 regularizace není zásadní, avšak výkon modelu se může zlepšit (jak jsem již zmínil, L2 značně snižuje šanci na přetrénování).

**4.5 Zpětné šíření chyby**