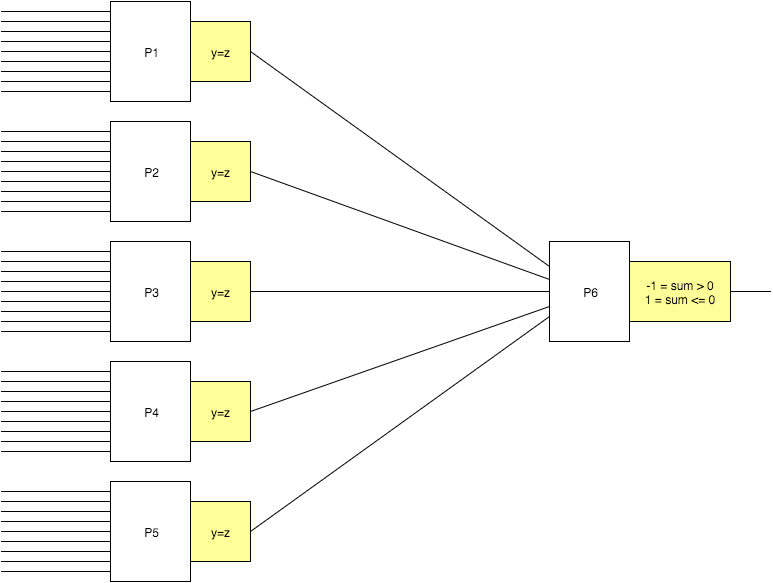
**Sieć neuronowa klasyfikująca nr. Indeksu**

Implementacja projektu składa się z dwóch części – głównej implementacji sieci neuronowej oraz mini programu do generowania danych wejściowych. Cały program powstał w języku Java przy użyciu IDE Idea IntelliJ i do jego uruchomienia potrzebna jest biblioteka log4j która była pomocna przy kompletowaniu wyników (dołączona w projekcie).

Moja sieć stworzona została z dwóch warstw. Pierwsza, składa się z 5 neuronów z 9 wejściami. Każdy z nich kolejno przyjmuje jedną cyfrę pod postacią matrycy 3x3. Jako funkcja aktywacyjna dla każdego z tych neuronów została użyta funkcja liniowa *y=z*. Druga warstwa, to jeden neuron który przyjmuje na wejściu wyjścia neuronów z poprzedniej warstwy. Funkcją aktywacyjną jest tutaj funkcja progowa bipolarna.



Dla każdego testu używany był zestaw 3 rodzajów danych:

1. Plik test\_data.txt – zawiera dane testowe – 100 poprawnych losowo wygenerowanych numerów indeksu w poniższym formacie:  
   *0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0*
2. Plik train\_data.txt – zawiera dane użyte do trenowania sieci. Jak wyżej zawiera 100 poprawnych numerów indeksu z tym, że na końcu każdej linii znajduje się oczekiwana wartość:  
   *0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0 1*
3. Plik wrong\_data.txt – zawiera dane użyte do testowania przypadkowych danych (nie będących numerem indeksu)

*0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 0 0 1 1 0 1 1 1 1 0*

Każdy test zaczynał się sprawdzeniem odpowiedzi sieci na wszystkie próbki z pliku test\_data.txt. Następnie program próbował wytrenować sieć jedną z próbek z pliku train\_data.txt i po każdej próbie uczenia ponownie sprawdzał odpowiedź sieci dla wszystkich testowych danych. Metoda ta w czytelny sposób dała możliwość sprawdzenia tempa uczenia się sieci.

**Zależność wyniku sieci od współczynnika kroku nauki**

Dla każdego z testów wagi startowe były losowane z zakresu <-1;1>. Na poniższych wykresach na osi X widać ilość (ze 100 testowych) poprawnie rozpoznanych próbek, na osi Y znajduje się natomiast liczba próbek na podstawie których sieć była uczona.

Za pierwszym podejściem parametr η miał wartość 0,9. Sieć rozpoznawała wszystkie testowe próbki już po 10 uczących próbkach, niestety, program napisany w Javie po nauczeniu sieci 65 razy nie był w stanie już modyfikować wag (przekroczony zakres float).

Po zmniejszeniu parametru η na 0,3 sieć sprawowała się już poprawnie. Po zaledwie 6 próbkach potrafiła poprawnie rozpoznać dane wejściowe.

Po kolejnym zmniejszeniu tego parametru ilość potrzebnych próbek do nauczenia sieci odpowiednio się zwiększała. W momencie kiedy parametr został ustawiony na wartość 0,001 sieć po 100 próbkach uczących nie była w stanie rozpoznać nawet połowy z testowych danych.

**Zależność wyniku sieci od parametrów funkcji aktywacji**

Do przetestowanie tej zależności użyłem stałej η = 0,0005 oraz jak wcześniej wag wejściowych z zakresu <-1;1>. Przy testowaniu funkcji liniowej w perceptonach warstwy pierwszej. W większości wypadków wynik znacząco różnił się od wartości wag na wejściach, jednak udało się zauważyć pewną zależność. Przy wykonywaniu mnożenia przez parametr wartości funkcji tempo nauki poprawiało się, a dzieląc go zaczęło znacząco maleć.