Sprawozdanie z zagadnienia nr4.

Opis syntetyczny:

Reguła Hebba jest jedną z popularnych metod samouczenia sieci neuronowych.
Zasada działania polega na tym, że sieci przedstawia się kolejne przykłady sygnałów wejściowych , jednak nie mówi się informacji co z tymi sygnałami zrobić. Sieć odbiera różne sygnały i obserwuje otoczenie. Na podstawie napływających danych sieć dedukuje jakie mają one znaczenie i ustala zachodzące między nimi zależności.

Ogólna reguła Hebba mówi, że zmiany wag powinny odbywać się według reguły

$$\Delta \mathbf{w}(k) = F(\mathbf{x}(k), y(k)),$$

czyli ogólnie, że przyrost wag $\Delta \mathbf{w}(k)$ powinien zależeć zarówno od wielkości wzorca presynaptycznego $\mathbf{x}(k)$ jak i od wytworzonego wzorca postsynaptycznego y(k).

Prosta reguła Hebba: (stała η oznacza tu współczynnik proporcjonalności):

$$\Delta \mathbf{w}(k) = \eta \cdot \mathbf{x}(k) \cdot y(k)\,.$$

Jednak ta reguła była zbyt uproszczona. Przy wielokrotnej prezentacji tego samego wzorca wagi wzrastały wykładniczo. Po każdej prezentacji wzorca $\mathbf{x}(k)$ wektor wag był przesuwany w kierunku tego wzorca, mogło to powodować raptowne i dość znaczne zmiany wyuczonego dotychczas wektora wag.

Modyfikacja (1) - wprowadzenie współczynnika zapominania b

$$\Delta w_{ij}(k) = c\, x_j(k)\, y_i(k) - b\, w_{ij}(k)\, y_i(k)$$

- dobór współczynnika b jest kluczowy dla zapewnienia stabilności procesu i zależy od konkretnego problemu, powstrzymuje niekontrolowany wzrost wartości wag.

Modyfikacja 2 (Oja)

$$\Delta w_{ij}(k) = c\, y_i(k)[x_j(k) - y_i(k)]w_{ij}(k)$$

- wprowadza samonormalizację wektora wag w każdym kroku uczącym
- w stanie ustalonym ||W||=1

Dane uczące:

Wygenerowane zostały matryce w rozmiarze 15x15 dla 4 emotikon: śmiech,płacz,miłość,całus.
Przykładowa matryca:

```
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1  1  1  1  1  1 -1 -1 -1 -1 -1
-1 -1 -1  1  1 -1 -1 -1 -1 -1  1  1 -1 -1 -1
-1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1
-1 -1  1 -1  1 -1 -1 -1 -1 -1  1 -1  1 -1 -1
-1  1 -1  1 -1 -1 -1 -1 -1 -1 -1  1 -1  1 -1
-1  1 -1 -1  1  1  1 -1  1  1  1 -1 -1  1 -1
-1  1 -1 -1  1 -1 -1 -1 -1 -1  1 -1 -1  1 -1
-1  1 -1  1  1 -1 -1 -1 -1 -1  1  1 -1  1 -1
-1  1  1  1 -1  1  1  1  1  1 -1  1  1  1 -1
-1 -1  1  1 -1  1  1  1  1  1 -1  1  1 -1 -1
-1 -1  1 -1 -1 -1  1  1  1 -1 -1 -1  1 -1 -1
-1 -1 -1  1  1 -1 -1 -1 -1 -1  1  1 -1 -1 -1
-1 -1 -1 -1 -1  1  1  1  1  1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Do testów używano zaszumionych matryc gdzie poprawność pojedynczych pikseli była na poziomie 60% i 80% względem oryginału.

Zestawienie danych + przykładowe wydruki:

```
 MSE: 2.66676e+144      MAPE: 1.63302e+73%
 MSE: 4.1149e+144       MAPE: 2.02852e+73%
 MSE: 4.95428e+144      MAPE: 2.22582e+73%
epoka: 201
 MSE: 1.11882e+145      MAPE: 3.34487e+73%
 MSE: 1.35005e+145      MAPE: 3.6743e+73%
 MSE: 2.08317e+145      MAPE: 4.56418e+73%
 MSE: 2.5081e+145       MAPE: 5.0081e+73%
epoka: 202
 MSE: 5.66401e+145      MAPE: 7.52596e+73%
 MSE: 6.83462e+145      MAPE: 8.26717e+73%
 MSE: 1.05461e+146      MAPE: 1.02694e+74%
 MSE: 1.26973e+146      MAPE: 1.12682e+74%
epoka: 203
 MSE: 2.86741e+146      MAPE: 1.69334e+74%
 MSE: 3.46002e+146      MAPE: 1.86011e+74%
 MSE: 5.33894e+146      MAPE: 2.31061e+74%
 MSE: 6.42799e+146      MAPE: 2.53535e+74%
epoka: 204
 MSE: 1.45162e+147      MAPE: 3.81002e+74%
 MSE: 1.75164e+147      MAPE: 4.18526e+74%
 MSE: 2.70284e+147      MAPE: 5.19888e+74%
 MSE: 3.25417e+147      MAPE: 5.70454e+74%
```

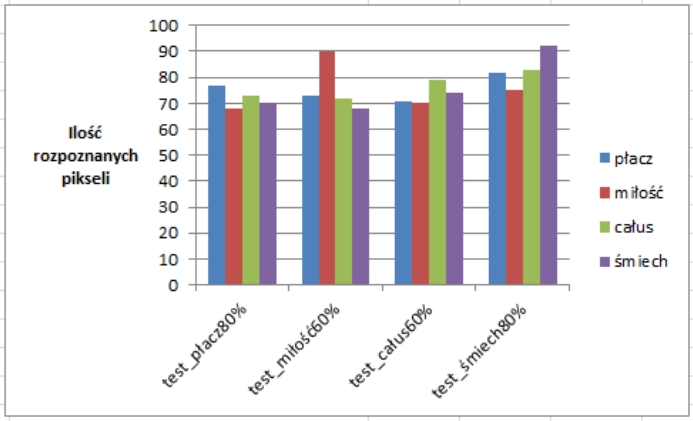
Przykładowy zrzut wykonany w trakcie wykonywania procesu uczenia, na którym widnieje błąd MSE i MAPE.

Dane uzyskane :

1) **Bez współczynnika zapominania**

Bez współczynnika zapominania

		Emotikona			
rodzaj i %podobienstwa do wzorca	Nr Epoki	płacz	miłość	całus	śmiech
test_płacz80%	230	77	68	73	70
test_miłość60%	231	73	90	72	68
test_całus60%	238	71	70	79	74
test_śmiech80%	231	82	75	83	92



Przy współczynniku uczenia = 0,5 proces uczenia trwał około 233 epoki. Dla zniekształconych emotikon testowych, siec rozpoznała wszystkie obrazy poprawnie.

Bez współczynnika zapominania

l.r=0,75

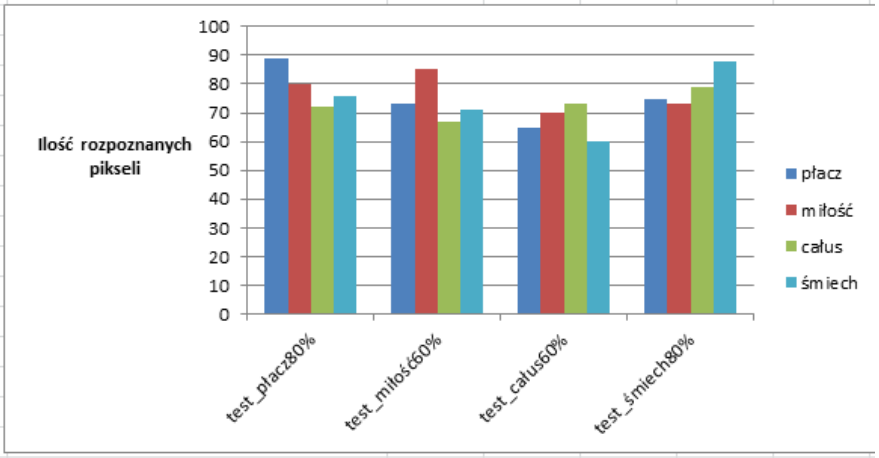
		Emotikona			
rodzaj i %podobien	Nr Epoki	płacz	miłość	całus	śmiech
test_płacz80%	208	78	70	73	72
test_miłość60%	213	75	90	72	64
test_całus60%	209	73	75	78	74
test_śmiech80%	215	85	78	82	90
	211,25				

Dla porównania zwiększony współczynnik uczenia =0,75, który przyśpieszył proces uczenia o około 22 epoki.

2) Ze współczynnikiem zapominania

Ze współczynnikiem zapominania

		Emotikona			
rodzaj i %podobienstwa do wzorca	Nr Epoki	płacz	miłość	całus	śmiech
test_płacz80%	173	89	80	72	76
test_miłość60%	171	73	85	67	71
test_całus60%	172	65	70	73	60
test_śmiech80%	172	75	73	79	88



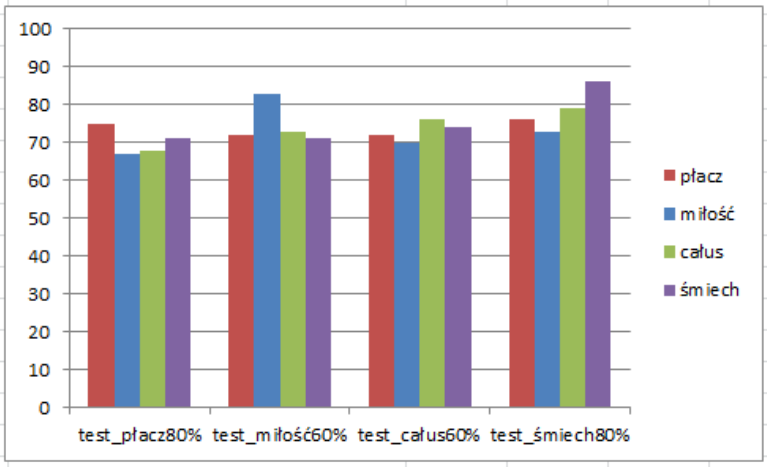
Arkusz1 Arkusz2 Arkusz3

Przy współczynniku uczenia=0,5 i współczynniku zapominania =0,05 czas uczenia trwał ok. 172 epoki. Rozpoznano wszystkie obrazy poprawnie, choć jak widać z inną dokładnością.

3) Uczenie metodą Oji

Metoda Oji

		Emotikona			
rodzaj i %podobienstwa do wzorca	Nr Epoki	płacz	miłość	całus	śmiech
test_płacz80%	10	75	67	68	71
test_miłość60%	10	72	83	73	71
test_całus60%	11	72	70	76	74
test_śmiech80%	8	76	73	79	86
	9,75				

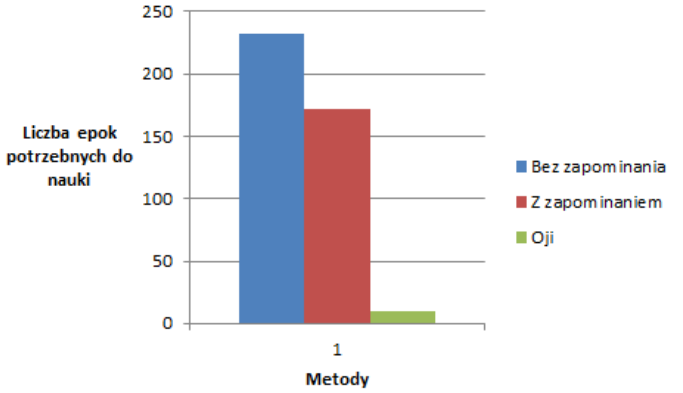


Przy współczynniku uczenia =0,5 sieć metoda Oji uczyła się w ok. 10 epok. Obrazy zostały rozpoznane poprawnie szybciej lecz z mniejszą precyzją.

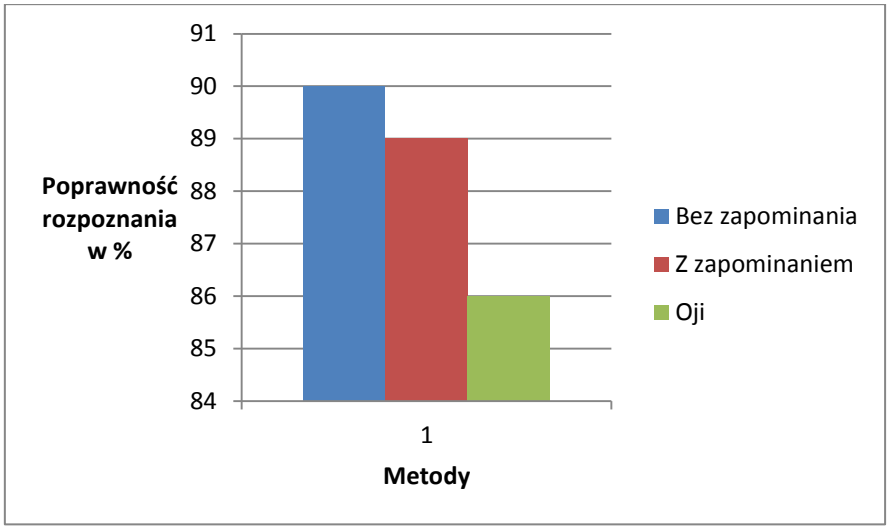
Zestawienie wyników:

Porównanie prędkości uczenia sieci w zależności od wybranej metody uczenia.

Metoda	l.epok
Bez zapominania	232,5
Z zapominaniem	172
Oji	9,75



Zestawienie dokładności rozpoznawania obrazów w zależności od wybranej metody.



Wnioski:

- Metoda Hebba polega na wzmacnianiu połączeń między źródłami silnymi i osłabieniu tych słabszych

-Modyfikacja tej metody w metodę Oji w znacznym stopniu przyśpiesza proces uczenia się

-współczynnik zapominania ogranicza wzrost wektora wag (pozwala mieć nad nim kontrole)

- wielkości współczynników uczenia i zapominania mają wpływ na szybkość i dokładność uczenia

-testy polegały na podaniu sieci zniekształconego obrazu i próbie jego rozpoznania

-metodą najszybszą okazała się być metoda Oji ,lecz najdokładniejszą metoda Hebba bez współczynnika zapominania

Listing kodu:

```
void
generateInput(){
	srand(time(NULL));
	int i, j, tmp;
	for (i = 0; i<SIZE; i++){
		tmp = rand() % 2 - 1;                                          //
generowanie losowych danych uczacych
		if (tmp == 0) input(i) = 1;
		else input(i) = -1;
	}

	for (i = 0; i<SIZE; i++){                                          //
generowanie zdeformowanych emotikon
		inputSmiech80pr(i) = vec_emoticons(i, 0);
		inputCalus60pr(i) = vec_emoticons(i, 1);
		inputMilosc80pr(i) = vec_emoticons(i, 2);
		inputPlacz60pr(i) = vec_emoticons(i, 3);
	}

	for (i = 0; i<45; i++){
		j = rand() % 225;
		inputMilosc80pr(j) = !inputMilosc80pr(j);
		j = rand() % 225;
		inputSmiech80pr(j) = !inputSmiech80pr(j);
	}

	for (i = 0; i<90; i++){
		j = rand() % 225;
		inputCalus60pr(j) = !inputCalus60pr(j);
		j = rand() % 225;
		inputPlacz60pr(j) = !inputPlacz60pr(j);
	}

}

void
generateWeights(){
	srand(time(NULL));
	int i;
	for (i = 0; i<SIZE; i++){
		w(i, 0) = (float)rand() / (float)RAND_MAX;                     //
losowanie wag
		w(i, 1) = (float)rand() / (float)RAND_MAX;                     //
losowanie wag
		w(i, 2) = (float)rand() / (float)RAND_MAX;                     //
losowanie wag
		w(i, 3) = (float)rand() / (float)RAND_MAX;                     //
losowanie wag
	}
}


int
main(){
```

```
	int i, j;
	int epoka = 0;                                  // liczba epok

	const double LEARNING_RATE = 0.5;        /* wspolczynnik uczenia */
	const double FORGET_RATE = 0.05;         /* wspolczynnik zapominania */

	double globalError = 0.;                 /* bledy */
	double localError = 0.;
	double MSE = 0.;
	double MAPE = 0.;

	double pom = 0.;                         /* zmienne pomocnicze */
	double pomt = 0.;

	zeruj();                                 /* ustawienie wartosci domyslnych dla
wektora a  */

	//UCZENIE WG HEBBA
	generateWeights();                       /* wygenerowanie wag */
	setLearnVectors();                       /* ustawienie wektorow uczacych */

	do{
		cout << "epoka: " << epoka << endl;

		/* BEZ WSPOLCZYNNIKA ZAPOMINANIA
		for (i = 0; i<4; ++i){
			globalError = 0.;
			for (j = 0; j<SIZE; ++j){
				pom = a(j);
				a(j) = (w(j, i)*vec_emoticons(j, i));
				w(j, i) = w(j, i) + LEARNING_RATE*a(j)*vec_emoticons(j, i);

				if (localError == abs(pom - a(j))) break;
				localError = abs(pom - a(j));
				globalError = globalError + pow(localError, 2);

			}

			MSE = pow(globalError, 2) / SIZE;
			MAPE = (globalError * 10) / SIZE;
			cout << " MSE: " << MSE << "\tMAPE: " << MAPE << "%\n";
		}*/

		// ZE WSPOLCZYNNIKIEM ZAPOMINANIA
		for(i=0;i<4;++i){
		globalError = 0.;
		for(j=0;j<SIZE;++j){
		pom = a(j);
		a(j) = (w(j,i)*vec_emoticons(j,i));
		w(j,i) = w(j,i)*FORGET_RATE + LEARNING_RATE*a(j)*vec_emoticons(j,i);

		if(localError==abs(pom-a(j))) break;
		localError = abs(pom - a(j));
		globalError = globalError + pow(localError,2);
		}

		MSE = pow(globalError,2)/(SIZE);
		MAPE = (globalError*10/SIZE);
		cout << "i:" << i << " MSE: " << MSE << "\tMAPE: " << MAPE << "%\n";
		}

		// REGUŁA OJI
```

```
            /*for(i=0;i<4;++i){
            globalError = 0.;
            for(j=0;j<SIZE;++j){
            pom = a(j);
            a(j) = (w(j,i)*vec_emoticons(j,i));
            w(j,i) = w(j,i) + (LEARNING_RATE*a(j)*(vec_emoticons(j,i)-a(j)*w(j,i)));

            if(localError==abs(pom-a(j))) break;
            localError = abs(pom - a(j));
            globalError = globalError + pow(localError,2);
            }

            MSE = pow(globalError,2)/(SIZE);
            MAPE = (globalError*10/SIZE);
            cout << "i:" << i << " MSE: " << MSE << "\tMAPE: " << MAPE << "%\n";
            }*/

            epoka++;

        } while (globalError != 0 && epoka<1000);

        cout << "\nliczba epok: " << epoka << endl;


        test();

        return 0;
}
```