Streszczenie

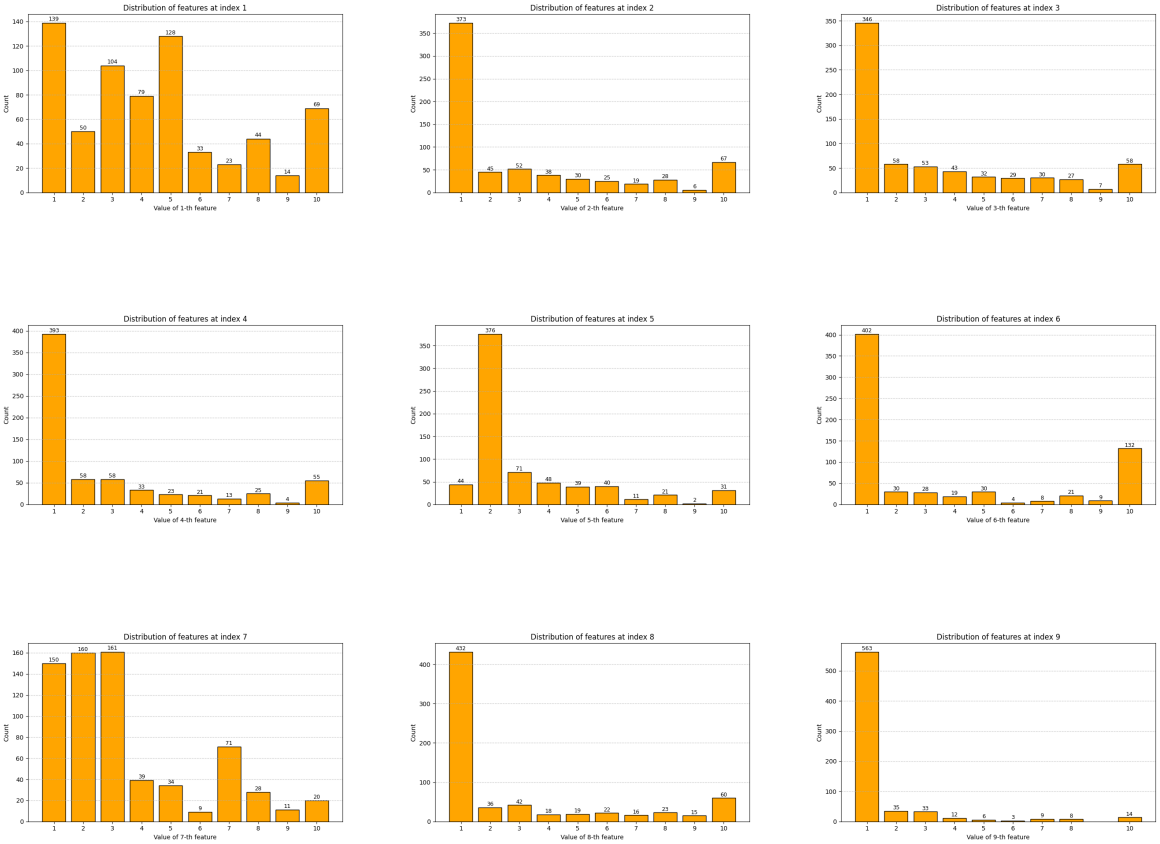
W poniższym dokumencie przedstawiam wnioski wyciągnięte podczas pisania projektu dot. klasyfikacji z Metod Probabilistycznych w Uczeniu Maszynowym. Celem projektu jest porównanie naiwnego klasyfikatora Bayesowskiego i regresji logistycznej.

Przygotowanie danych

Na wejściu otrzymujemy zbiór danych postaci $D = \{(x_1^{(i)}, x_2^{(i)}, \ldots, x_9^{(i)}, y^{(i)}) : i = 1, 2, \ldots, m\}$, zatem mamy 9 niezależnych cech opisujących badanie oraz informację, czy badany rak jest łagodny ($y = 2$), czy złośliwy ($y = 4$). Dla ułatwienia będę oznaczał $y = 2$ jako klasę 0, zaś $y = 4$ jako klasę 1 W celu podziału tych danych na zbiory treningowy i testowy, dzielę D na D_0 i D_1 dla danych należących do klas odpowiednio 0 i 1, następnie dzielę obydwa zbiory w proporcji $2 : 1$ i łącząc te części tworzę zbiór treningowy S i testowy T. W ten sposób losowo wybrana dana z S ma takie same prawdopodobieństwo bycia w klasie 0, co losowo wybrana dana z T.

Wstępna analiza danych

Każda z cech jest pewną liczbą całkowitą ze zbioru $\{1, 2, \ldots, 10\}$. Dla naszych danych nie stosuję standaryzacji, gdyż przy **naiwnym klasyfikatorze bayesowskim** standaryzacja danych które chcemy podzielić na klasy i tak nie miałaby sensu, a dla **regresji logistycznej** zakresy tych danych są na tyle małe, że nie utrudnia wyboru współczynnika `learning rate`, ani nie psuje regularyzacji. Popatrzmy teraz na rozkład poszczególnych cech.



Widzimy, że w niektórych przypadkach cechy mają wartość wokół której się koncentrują, a w innych występują równomiernie różne ich wartości.

Ocena skuteczności modeli

Sposób w jaki sprawdzam skuteczność modeli jest następujący: najpierw trenuję model na zbiorze treningowym, a następnie tak obliczone parametry używam do klasyfikacji danych ze zbioru testowego. Następnie porównuję przewidywania z faktycznymi wynikami na zbiorze testowym i liczę ile jest przypadków `True Positive`, `False Positive`, `True Negative` oraz `False Negative`. Na ich podstawie wyliczam statystyki, takie jak precyzja i czułość.

Miarą, której używam do oceny klasyfikatorów to miara F_1, gdyż zależy mi obecnie na modelu który zarówno wykrywa jak najwięcej przypadków i przy okazji unika fałszywych alarmów.

$$F_1(\text{precision}, \text{sensitivity}) = \frac{2 \cdot \text{precision} \cdot \text{sensitivity}}{\text{precision} + \text{sensitivity}}$$

Możemy teraz przejść do porównania wyników obydwu modeli.

Regresja logistyczna

Na potrzeby tego projektu zaimplementowałem regresję logistyczną używając do tego klasycznego algorytmu spadku wzdłuż gradientu. Nie zdecydowałem się na żadne optymalizacje, np. **mini−batch**, gdyż model trenował się względnie bardzo szybko: dla $n = 15000$ iteracji (dla tylu epok wyniki już były satysfakcjonujące), na pełnym zbiorze treningowym przebieg algorytmu zajmował ok. 5 sekund. Za `learning rate` wziąłem współczynnik $\eta = 0.0025$

Do regresji logistycznej wykorzystałem następującą `funkcję błędu` z wykorzystaniem regularyzacji L_2:

$$L(\theta) = \frac{-1}{m} \sum_{i=1}^{m} (1 - y^{(i)}) \cdot \ln(1 - h_\theta(x^{(i)})) + (y^{(i)}) \cdot \ln(h_\theta(x^{(i)})) + \lambda \cdot \theta^T \theta$$

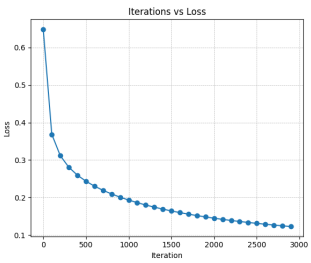
Gdzie za naszą hipotezę bierzemy funkcję sigmoid

$$h_\theta(x) = \sigma(\theta^T x)$$

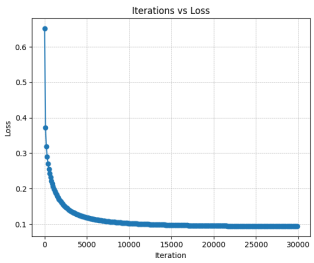
Licząc pochodne cząstkowe, dostajemy

$$\frac{\partial}{\partial \theta_j} L(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + 2\lambda \cdot \theta_j$$

Na poniższym wykresie przedstawiam zależność funkcji straty od liczby iteracji.



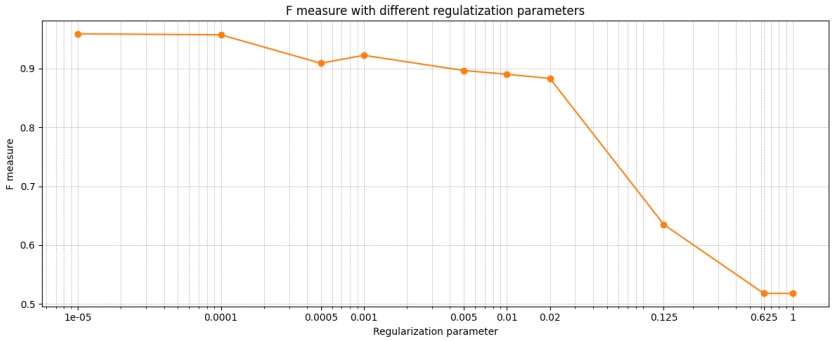
Rysunek 1: $n = 3000$ iteracji



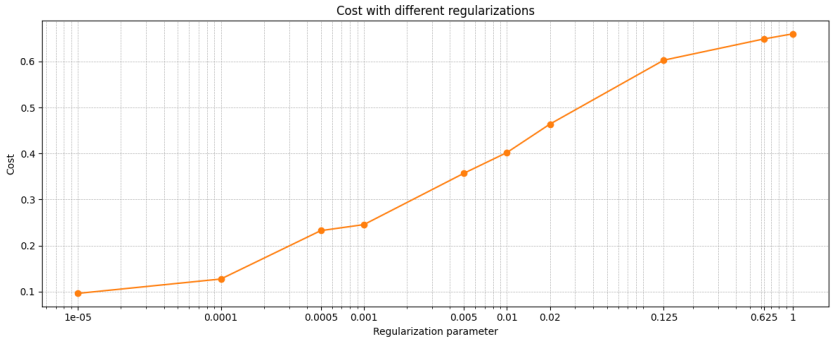
Rysunek 2: $n = 30000$ iteracji

Tempo malenia funkcji straty jest nieporównywalnie mniejsze gdy rozważamy duże iteracje, więc stwierdziłem że wystarczy trenować model do $n = 15000$ iteracji. Warto nadmienić, że 3000 iteracji zajęło mi ok. $1.66s$, a 30000 iteracji ok. $16.1s$.

Kolejnym aspektem, nad którym się chciałbym pochylić, to jak regularyzacja wpływa na wynik naszego modelu na testowym zbiorze danych. W tym projekcie używam regularyzacji grzbietowej. Nie korzystam ze zbioru walidacyjnego, zatem to w jaki sposób zmierzę współczynnik λ to przez uśrednianie wyniku mojego algorytmu na kilku podziałach danych na zbiory treningowy i testowy. Porównanie różnych współczynników regularyzacji λ przedstawia się następująco.



Rysunek 3: Miara F_1 dla różnych wyborów współczynnika λ



Rysunek 4: Błąd L na zbiorze testowym obliczony dla różnych wyborów współczynnika λ

Jak widać, model nie overfituje się na zbiorze treningowym i regularyzacja nie poprawia wyników naszego modelu. Wraz ze wzrostem parametru błąd też znacząco rośnie, zatem algorytm zamiast "wygładzać"funkcję, koncentruje się na minimalizacji współczynników przez co algorytm nie osiąga lepszych efektów. Przez to że regularyzacja nie poprawia wyników naszego modelu, w końcowym porównaniu nie będę jej stosował.

Biorąc wcześniejsze wnioski pod uwagę, zmierzmy teraz kilka parametrów trenując nasz model na pełnym zbiorze treningowym. Wyniki które otrzymałem są następujące.

Parametr	**Wartość**
Dokładność	0.987
Precyzja	0.975
Czułość	0.987
Swoistość	0.986
Miara F	0.981

Tabela 1: Wyniki najlepszego modelu

Naiwny klasyfikator Bayesowski

Przejdźmy teraz do generatywnego algorytmu klasyfikacji. W tym projekcie zaimplementowałem `naiwny klasyfikator Bayesowski` z wykorzystaniem `wygładzenia Laplace'a`. W tym celu dla danych treningowych T liczę następujące parametry:

$$\Phi_y = \frac{1+\sum_{i=1}^{m} \mathbf{1}[y^{(i)}=1]}{2+m}$$

$$\Phi_{c,d}^{j} = \frac{1+\sum_{i=1}^{m} \mathbf{1}[y^{(i)}=c \wedge x_j^{(i)}=d]}{10+\sum_{i=1}^{m} \mathbf{1}[y^{(i)}=c]}$$

Pierwszą rzeczą, na którą zwróciłem uwagę, była szybkość tego algorytmu. Wytrenowanie modelu na pełnym zbiorze treningowym zajmowało mi ok. 10 ms, co było nieporównywalnie szybsze względem regresji logistycznej.

Ciekawą rzeczą, którą zauważyłem analizując proces trenowania naiwnego klasyfikatora bayesowskiego, było to że gdy używałem więcej niż 12.5% zbioru treningowego to im więcej miał danych, tym bardziej pewne decyzje na zbiorze testowym podejmował. Zatem zdecydowaną część wyników szacował albo na 0.00%, albo na 99.99% (decyzje były *troszeczkę* bardziej zrównoważone gdy użyłem `log-prawdopodobieństwa`, ale odpowiedzi nadal były, delikatnie rzecz ujmując, **stanowcze**).

Poniżej przedstawiam uśrednione wyniki, jakie udało mi się osiągnąć używając **naiwnego klasyfikatora Bayesowskiego**.

Parametr	**Wartość**
Dokładność	0.978
Precyzja	0.963
Czułość	0.975
Swoistość	0.980
Miara F	0.969

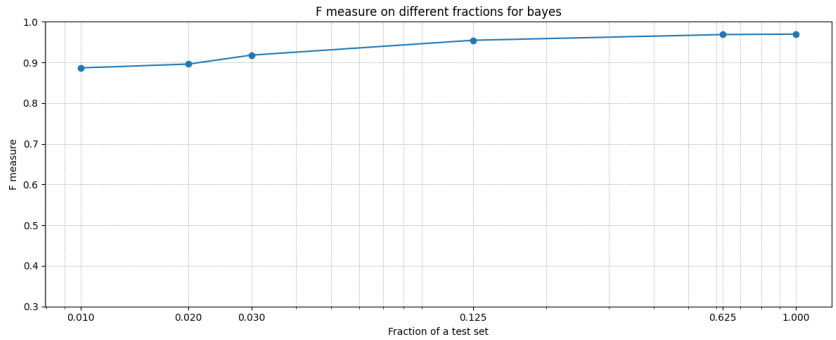
Tabela 2: Wyniki najlepszego modelu

Zauważmy, że są one równie wysokie co dla algorytmu **regresji logistycznej**, mimo naszego silnego założenia że cechy są warunkowo niezależne.

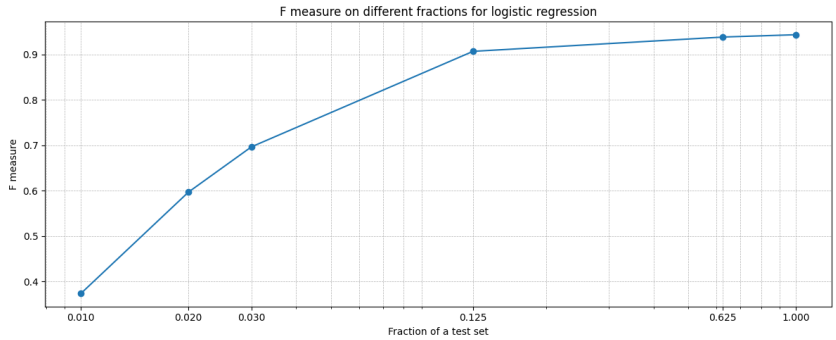
Trenowanie na frakcjach zbioru treningowego

Pytanie na które chciałbym odpowiedzieć, to ile danych wystarczy żeby satysfakcjonująco wytrenować klasyfikator. W tym celu wytrenowałem obydwa modele na różnych frakcjach zbioru treningowego, a wyniki znajdują się poniżej.

Od początku widzimy bardzo dużą różnicę. Otóż, gdy popatrzymy na skalę miary F_1, naiwny klasyfikator bayesowski od początku radzi sobie dużo lepiej, przy czym już 1% zbioru treningowego pozwala mu na nauczenie się danych na tyle dobrze, żeby otrzymywać średnio miarę 0.89. Jest to wynik do którego regresja logistyczna zbliża się dopiero przy 12.5% zbioru treningowego, a nasz klasyfikator Bayesowski otrzymuje już miary rzędu 0.94



Rysunek 5: Miara F_1 dla frakcji przy uczeniu naiwnego klasyfikatora Bayesowskiego



Rysunek 6: Miara F_1 dla frakcji przy uczeniu algorytmem regresji logistycznej

Wnioski

Według autora artykułu powszechnie uważa się modele dyskryminatywne za dokładniejsze od modeli generatywnych. W przypadku naiwnej klasyfikacji Bayesowskiej bardzo silnym założeniem jest warunkowa niezależność cech, taki model nie bierze tych zależności pod uwagę. Jednak w naszym przypadku obydwa modele osiągnęły bardzo satysfakcjonujące wyniki. Teza płynąca z artykułu to fakt, że `naiwny klasyfikator Bayesowski` potrafi się uczyć już na bardzo małej ilości danych (mowa jest o rozmiarze danych rzędu $O(\log n)$) i rzeczywiście, naiwny klasyfikator Bayesowski osiąga nienajgorszy wynik biorąc jedynie 1% danych treningowych.