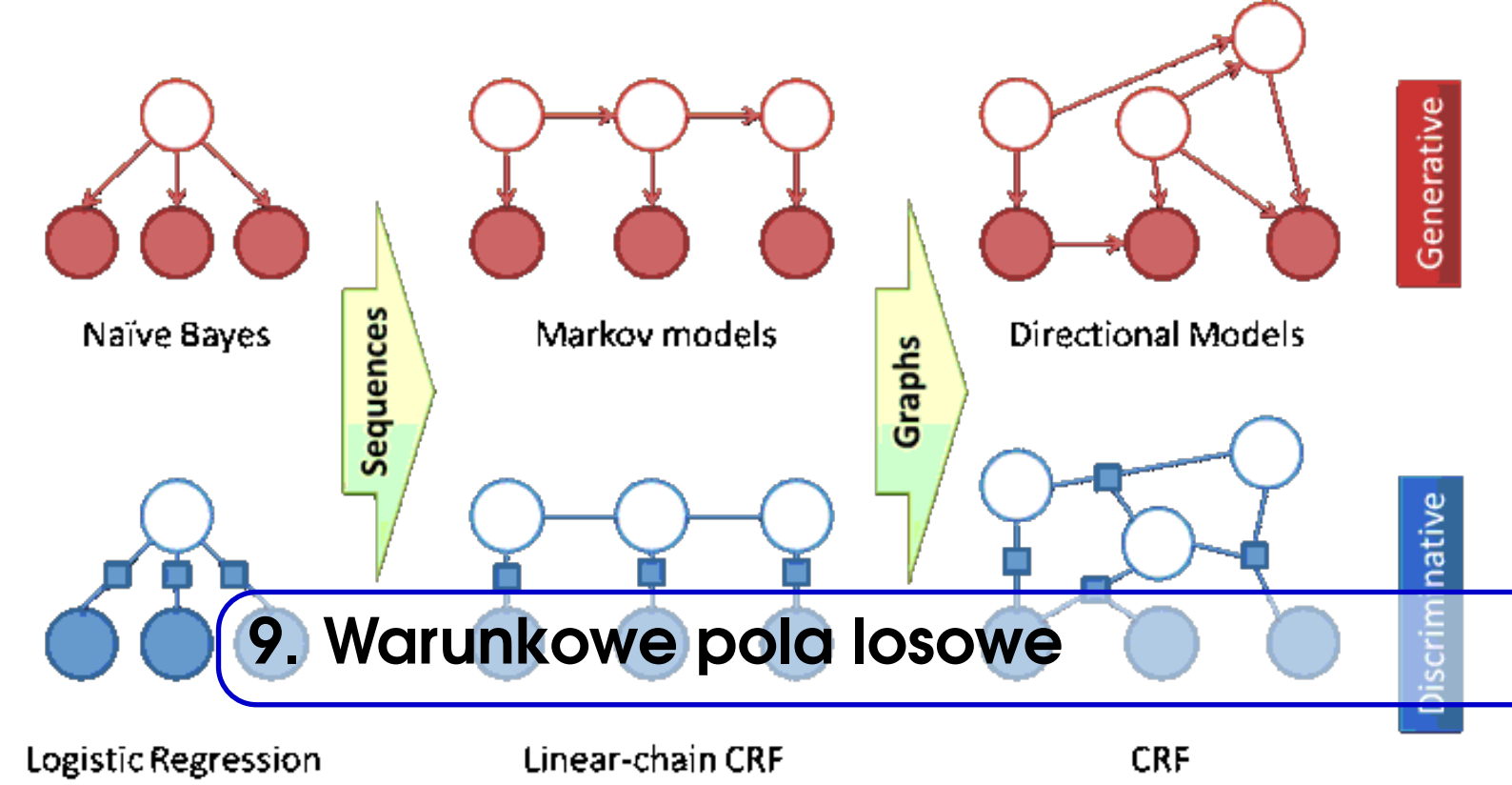


9. Warunkowe pola losowe

Od ostatnich dwóch zajęć zajmujemy się jednym z problemów predykcji struktur: predykcją sekwencji. Problem ten porównaliśmy do problemu klasyfikacji wieloklasowej, ponieważ problem predykcji struktur można do niego sprowadzić używając prostego podstawienia. Mianowicie, każdą możliwą sekwencję którą należy przewidzieć można potraktować jako jedną klasę, uzyskując problem klasyfikacji z tyloma klasami ile jest możliwych wyjściowych sekwencji. Zauważyliśmy główne trudności takie jak niezmiernie duża liczba klas (rosnąca wykładniczo z długością sekwencji) oraz potrzebę uogólniania wiedzy na nowe klasy (struktury), ponieważ z powodu ich ilości nie możemy zakładać że każda z nich pojawi się w zbiorze uczącym. Doprowadziło nas to do potrzeby dekompozycji struktury na mniejsze części, które będą podlegały ocenie przez system uczący oraz prezentacji algorytmów w trzech częściach: model (założenia), trening i wnioskowanie (dekodowanie/predykcja).

Omówiliśmy jedno podejście generatywne: ukryte modele Markowa, jednak ze względu na trudność modelowania dużej liczby cech oraz nadzieję na uzyskania predykcji o wyższej jakości – zaczęliśmy omawianie modelu dyskryminacyjnego. Konkretnie, modelu Markowa o maksymalnej entropii (MEMM), który zdekomponował problem predykcji do sekwencji indywidualnych decyzji dotyczących kolejnych etykiet:

$$P(y_1^n|x_1^n) = \prod_{i=1}^{n} P(y_i|y_{i-1}, x_1^n) = \prod_{i=1}^{n} \frac{e^{\mathbf{w}^T \phi(\mathbf{x}_i, y_{i-1}, y_i)}}{\sum_{y'} e^{\mathbf{w}^T \phi(\mathbf{x}_i, y_{i-1}, y')}}$$

Budowa tego modelu jest dość prosta i szybka, pojawia się jednak pytanie: czy czegoś nie tracimy poprzez predykcję sekwencji po kawałku?

Zauważ, że modele MEMM w czasie nauki uczą się wyłącznie podejmowania lokalnych decyzji i nie są świadome że ich wyniku używa się w sekwencji aby uzyskać jakąś całość. Chociażby, modele w czasie nauki widzą zawsze prawidłową etykietę y_{i-1} i korzystają z niej by przewidzieć następną. Naiwnym jest sądzić, że podobnie będzie

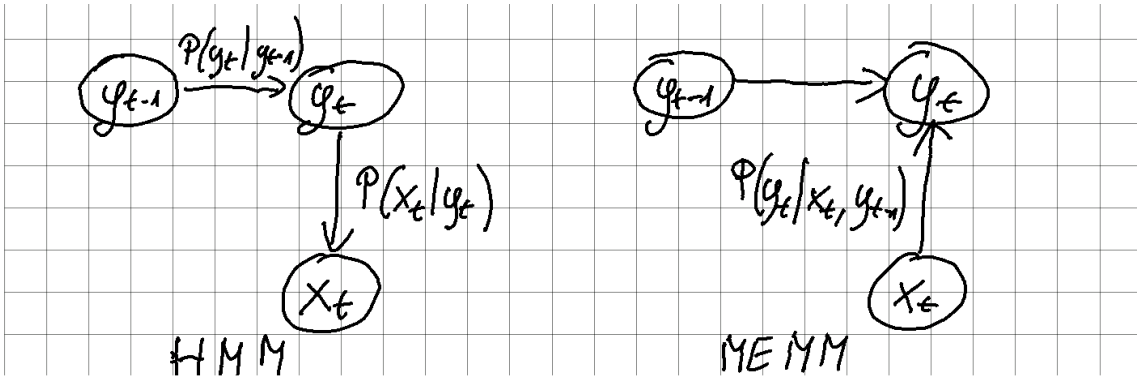
w czasie testowania. Prawie na pewno MEMM dostanie na swoje wejście jakieś y_{i-1}, które zostały błędnie przewidziane w poprzedniej iteracji, podczas gdy w czasie treningu klasyfikator nigdy nie miał styczności z taką sytuacją, przez co jego predykcje stają się mało wiarygodne. Szczególnie jeśli przypisał cechom związanym z poprzednią etykietą duże wagi (znaczenie) w modelu wiedzy.

Zauważ, że w czasie predykcji algorytmem Viterbiego, klasyfikator poddaje ocenie wszystkie możliwe ścieżki, więc w czasie jego działania *większość* klasyfikacji bazuje właśnie na błędnych etykietach y_{i-1} (rozważamy wszystkie!). To rzuca światło na kolejny problem: algorytm dekodowania (np. Viterbi) jest niezwykle ważną częścią systemu predykcji struktur gdyż to on de facto konstruuje końcowy rezultat. Jego użycie jest jednak całkowicie niewidoczne i pomijane w fazie uczenia się.

Tę sytuację można porównać do systemów klasyfikacji wieloklasowej, które korzystając z klasyfikatorów binarnych starają się rozwiązywać problemy wieloklasowe działając w schemacie jeden kontra wszyscy (ang. *one-vs-all, OVA*). Przypomnijmy, korzysta się wtedy z tylu klasyfikatorów binarnych ile jest klas, a każdy klasyfikator binarny stara się wykrywać jedną wybraną klasę i pomijać wszystkie inne. Jednakże porównując binarną regresję logistyczną w schemacie OVA z prawdziwie wieloklasową, wielomianową regresją logistyczną (softmax), ta druga zwykle działa dużo lepiej. Jednym z powodów jest uczenie klasyfikatorów binarnych podejmowania decyzji osobno tj. bez wzięcia pod uwagę decyzji innych klasyfikatorów zespołu. Wyobraźmy sobie sytuację w której klasyfikator binarny wykrywający klasę przykładu zwrócił prawdopodobieństwo równe 80%. Rozpatrując ten klasyfikator binarnie – jest to bardzo dobry wynik. Z kolei gdy dla drugiego przykładu ze swojej klasy zwrócił tylko 20% – popełnił on dość poważny błąd. Jednak patrząc na cały zespół klasyfikatorów może być dokładnie na odwrót. Dla pierwszego przykładu inny klasyfikator w zespole może zwrócić 82%, co sprawi że 80% odpowiedź nie wystarczy by przegłosować złą decyzję, a dla drugiego przykładu, wszystkie inne klasyfikatory mogły zwrócić wyniki poniżej 5%, co sprawia że 20% nie tylko wystarcza do podjęcia prawidłowej decyzji, ale także zostawia spory margines pewności. W przypadku uczenia tych klasyfikatorów razem – czyli przy połączeniu tych klasyfikatorów liniowych we wspólnego wieloklasowego softmaxa – takie sytuacje są jasne w czasie nauki, a algorytm uczący może ukarać funkcją straty odpowiednie przykłady.

Analogiczne sytuacje mogą pojawić się przy dokonywaniu predykcji systemami MEMM. Jeśli wszystkie kolejne decyzje na prawidłowej ścieżce są podejmowane z wysoką pewnością, lokalne wskazanie jej z nie za dużym prawdopodbieństwem może być wystarczające do podjęcia prawidłowej decyzji.

W końcu, nie można zapominać o poznanym na zeszłych zajęciach problemie obciążenia etykietą (ang. *label bias*), który pojawia się dla klasyfikatorów MEMM z powodu podejmowania przez nie lokalnych decyzji. Głównym źródłem tego problemu jest normalizacja prawdopodobieństw do wartości 1 dla każdego znacznika – normalizacja lokalna. W tym rozdziale poznamy modele globalne czyli podejmującą globalną decyzję o sekwencji zamiast kilku lokalnych. Normalizacja prawdopodobieństwa do 1 również odbędzie się po całej sekwencji a nie po poszczególnych decyzjach, a algorytm predykcji zostanie włączony w proces uczenia.



Rysunek 9.1: Grafy Markowa dla ukrytego modelu Markowa i modelu Markowa o maksymalnej entropii

9.1 Przypomnienie: probabilistyczne modele grafowe

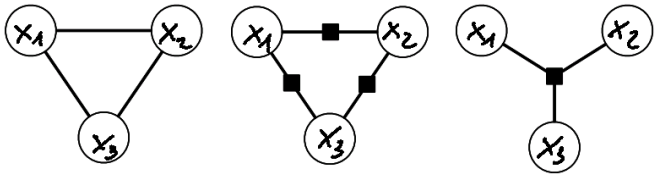
W modelowaniu wielowymiarowych rozkładów prawdopodobieństwa pomocny bywa ich zapis w postaci skierowanych grafów Markowa (⇒ Metody Inteligencji Sztucznej i Obliczeniowej). Każdy wierzchołek takiego grafu symbolizuje zmienną losową, a skierowane krawędzie pomiędzy nimi wskazują na to która zmienna wpływa na którą. Przykładowe grafy dla modelu HMM i MEMM możesz znaleźć na rysunku 9.1. Poprzez analizę grafu można w łatwy sposób odczytać informacje o warunkowych (nie)zależnościach różnych zmiennych np. w modelu HMM przy znajomości etykiety y_{t-1} zmienna y_t jest (warunkowo) niezależna od wszystkich innych zmiennych. Analogicznie, w narysowanym modelu MEMM zmienna y_t jest warunkowo niezależna od innych zmiennych przy znajomości y_{t-1} i x_t.

Innym sposobem reprezentowania graficznie rozkładu prawdopodobieństwa jest nieskierowany graf Markowa, który nie identyfikuje rozkładów warunkowych z krawędziami. Zamiast tego w grafie *nie* ma krawędzi pomiędzy parą zmiennych jeżeli są one warunkowo niezależne pod warunkiem wszystkich innych zmiennych. Również w tym zapisie można określić warunkową niezależność: para zmiennych jest warunkowo niezależna jeśli każda ścieżka w grafie przechodzi przez zmienne warunkujące. Inaczej: zmienne są warunkowo niezależne jeśli można narysować graf w ten sposób, że ma on dwie części zawierające te zmienne z łączącym je przesmykiem w którym wszystkie zmienne są znane (warunkujące). W szczególności: warunkując po wszystkich sąsiadach zmiennej jest ona warunkowo niezależna od wszystkich innych zmiennych (tzw. koc Markowa).

Modele skierowane można przepisać na prawdopodobieństwo po prostu wymnażając odpowiednie prawdopodobieństwa warunkowe, w przypadku modeli nieskierowanych jest to trochę bardziej skomplikowane. Prawdopodobieństwo modeluje się tam w następujący sposób:

$$P(\mathsf{x}) = \frac{1}{Z}\prod_{c\in C}\Psi_c(\mathsf{x}_c)$$

gdzie C to zbiór wszystkich klik w grafie, a x_c to zmienne znajdujące się w klice $c \in C$. Stała $Z = \sum_{\mathsf{x}}\prod_{c\in C}\Psi_c(\mathsf{x}_c)$ to suma statystyczna, normalizująca prawdopodobieństwo do 1. Najważniejszym elementem wzoru jest jednak funkcja oceny (ang. *score function*) $\Psi_c(\mathsf{x}_c)$ danej kliki, która czasami nazywana jest też funkcją potencjału lub funkcją kompatybilności. Funkcja oceny zawsze zwraca wartość nieujemną i zwraca ocenę tym wyższą im



Rysunek 9.2: Graf Markowa (po lewej) i dwie możliwe jego reprezentacje jako grafy rozłożenia na czynniki [2].

bardziej wartości przypisane do zmiennych w klice pasują do siebie. Mówiąc nieformalnie, funkcja ocenia jak bardzo lubimy takie ustawienie zmiennych. Oprócz nieujemności nie ma żadnych innych ograniczeń na funkcję oceny: może przyjmować wartości dowolnie duże, w szczególności nie musi sumować się do 1 po wszystkich możliwych ustawieniach.

Mając zestaw funkcji potencjału, którymi chcemy modelować rozkład prawdopodobieństwa, możemy bez problemu stworzyć odpowiadający im graf Markowa. Wystarczy narysować węzeł dla każdej zmiennej, a następnie połączyć krawędziom te zmienne, które występują wspólnie w argumencie przynajmniej jednej funkcji potencjału.

Widok samego grafu Markowa pozostawia jednak pewne nieścisłości dotyczące tego jakie funkcje oceny są rzeczywiście używane. Rozważmy prosty graf przedstawiony na rysunku 9.2. Możemy stwierdzić, że graf ten ma jedną trzy-elementową klikę i zapisać rozkład w postaci

$$P(\mathsf{x}) = \frac{\Psi(x_1, x_2, x_3)}{Z}$$

Równie dobrze graf może przedstawiać też rozkład z trzema funkcjami oceny:

$$P(\mathsf{x}) = \frac{\Psi_1(x_1, x_2)\Psi_2(x_1, x_3)\Psi_3(x_2, x_3)}{Z}$$

gdyż mamy na grafie trzy dwu-elementowe kliki. Aby uniknąć niejednoznaczności wprowadzono grafy rozłożenia na czynniki (ang. *factor graphs*) w których funkcje oceny są przedstawione jako kwadraciki połączone krawędziami z argumentami danej funkcji. Rysunek 9.2 przedstawia grafy rozkładu na czynniki dla pierwszej parametryzacji rozkładu (prawa) oraz dla drugiej (środek).

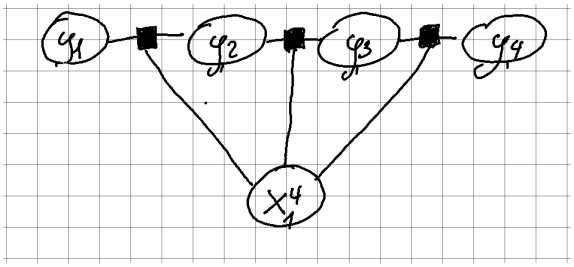
9.2 Warunkowe pola losowe: model

Warunkowe pola losowe (ang. *conditional random fields, CRF*) modelują warunkowy rozkład prawdopodobieństwa po strukturze używając funkcji potencjału i są często przedstawiane w postaci grafów rozkładów na czynniki. Modele CRF służą do predykcji struktur różnych typów i można je definiować dla grafów o różnych kształtach.

Tutaj skupimy się na tzw. warunkowym polu losowym na łańcuchu (ang. *linear-chain CRF*) który jest najpopularniejszym modelem dla predykcji sekwencji. Model ten jest wyrażony wzorem:

$$P(\mathsf{y}|\mathsf{x}) = \frac{1}{Z(\mathsf{x})} \prod_{i=1}^{n} \Psi_i(y_i, y_{i-1}, \mathsf{x}) \tag{9.1}$$

a jego wizualizacje znajdziesz na rysunku 9.3. Problem modelowania skomplikowanego



Rysunek 9.3: Warunkowe pola losowe na łańcuchu. Wierzchołek x_1^4 zawiera w sobie 4 zmienne losowe.

rozkładu warunkowego po sekwencji został sprowadzony do oceny jak bardzo nam się podoba dana sekwencja, a dla ułatwienia ocenę całej sekwencji rozdzieliliśmy na kilka ocen cząstkowych poszczególnych jej kawałków. Taką ocenę cząstkową dostarcza nam funkcja potencjału, której jedynym ograniczeniem jest to, że nie może ona przyznawać punktów ujemnych. Łączna ocena kompatybilności całej sekwencji łączona jest poprzez przemnożenie ocen cząstkowych (jeden całkowicie niepasujący kawałek wyklucza całą sekwencję – mnożenie przez 0). Następnie taka łączna ocena jest w prosty sposób przekształcana na prawdopodobieństwo poprzez podzielenie jej przez sumę ocen wszystkich możliwych sekwencji znaczników. Zwróć uwagę, że wartość sumy statystycznej $Z(\mathsf{x}) = \sum_{\mathsf{y}'} \prod_{i=1}^{n} \Psi_i(y'_i, y'_{i-1}, \mathsf{x})$ zależy od znakowanego zdania, bo modelujemy rozkład warunkowy (czyli rozkład sumuje się do 1 dla konkretnego, stałego warunku).

Zadaniem funkcji $\Psi_i(y_i, y_{i-1}, \mathsf{x})$ jest sprawdzenie czy fragment grafu z wierzchołkami y_i, y_{i-1} i x wygląda w porządku czyli czy etykiety na pozycjach i i $i-1$ pasują do sekwencji wejściowej. Każda etykieta, oprócz tych znajdujących się na końcach łańcucha, jest obecna w dwóch takich funkcjach oceniających. Funkcje potencjału są zdefiniowane w następujący sposób:

$$\Psi_i(y_i, y_{i-1}, \mathsf{x}) = e^{\mathsf{w}^T \phi(y_i, y_{i-1}, \mathsf{x})}$$

gdzie wektor wag w będzie uczony przez model, tak aby dobrze oceniać fragmenty sekwencji przy wykorzystaniu wektora cech $\phi(y_i, y_{i-1}, \mathsf{x})$ opisującego ten fragment. Funkcja eksponencjalna zabezpiecza nas przed ujemnymi wynikami funkcji liniowej i oferuje inne pożądane własności teoretyczne i praktyczne.

Ostatecznie warunkowy model losowy zdefiniowany na łańcuchu ma postać:

$$P(\mathsf{y}|\mathsf{x}) = \frac{1}{Z(\mathsf{x})} \prod_{i=1}^{n} \Psi_i(y_i, y_{i-1}, \mathsf{x}) = \frac{1}{Z(\mathsf{x})} \prod_{i=1}^{n} e^{\mathsf{w}^T \phi(y_i, y_{i-1}, \mathsf{x})} = \frac{1}{Z(\mathsf{x})} e^{\sum_{i=1}^{n} \mathsf{w}^T \phi(y_i, y_{i-1}, \mathsf{x})} \quad (9.2)$$

gdzie $Z(\mathsf{x}) = \sum_{\mathsf{y}'} e^{\sum_{i=1}^{n} \mathsf{w}^T \phi(y'_i, y'_{i-1}, \mathsf{x})}$. Model opisany jest więc wzorem matematycznym

$$P(\mathsf{y}|\mathsf{x}) = \frac{e^{\sum_{i=1}^{n} \mathsf{w}^T \phi(y_i, y_{i-1}, \mathsf{x})}}{\sum_{\mathsf{y}'} e^{\sum_{i=1}^{n} \mathsf{w}^T \phi(y'_i, y'_{i-1}, \mathsf{x})}}$$

który wygląda podobnie do wzoru opisującego MEMM. Zwróć jednak uwagę, że jest on normalizowany po wszystkich możliwych sekwencjach znaczników zamiast po każdej lokalnej decyzji. Znika więc problem obciążenia etykietą, ponieważ model nie wymusza normalizacji po każdej indywidualnej decyzji. Klasyfikator który jest ukryty w funkcji

oceny może zwrócić wysoką wartość oceny jeśli jest pewny, że dana część sekwencji jest prawidłowa. Jeśli jednak jakiś fragment nie był widziany w zbiorze i model zwyczajnie nie wiem, może zwrócić niską wartość oceny.

9.2.1 Porównanie do innych modeli uczenia maszynowego

■ **Przykład 9.1** Nic nie stoi na przeszkodzie by klasyfikator CRF użyć do zwykłego problemu klasyfikacji (w końcu możemy go również zamodelować w postaci grafu w którym jest tylko jeden węzeł y). Jaką postać przyjmuje model CRF dla klasyfikacji binarnej? Załóż, że funkcja potencjału zależy tylko od y_i i x, a funkcja ekstrakcji cech ma postać

$$\phi(y,\mathsf{x}) = \begin{cases} 0 & \text{dla } y = 0 \\ \mathsf{x} & \text{dla } y = 1 \end{cases}$$

Podstawmy do wzoru, zauważając że długość „sekwencji" wynosi 1.

$$P(\mathsf{y}|\mathsf{x}) = \frac{e^{\sum_{i=1}^{n} \mathsf{w}^T \phi(y_i,\mathsf{x})}}{\sum_{y'} e^{\sum_{i=1}^{n} \mathsf{w}^T \phi(y'_i,\mathsf{x})}} = \frac{e^{\mathsf{w}^T \phi(y_i,\mathsf{x})}}{\sum_{y' \in \{0,1\}} e^{\mathsf{w}^T \phi(y'_i,\mathsf{x})}}$$

Aby rozwiązać problem klasyfikacji binarnej wystarczy określić wartość prawdopodobieństwa warunkowego dla jednej z klas.

$$P(y = 1|\mathsf{x}) = \frac{e^{\mathsf{w}^T \phi(1,\mathsf{x})}}{e^{\mathsf{w}^T \phi(0,\mathsf{x})} + e^{\mathsf{w}^T \phi(1,\mathsf{x})}} = \frac{e^{\mathsf{w}^T \mathsf{x}}}{e^0 + e^{\mathsf{w}^T \mathsf{x}}} = \frac{e^{\mathsf{w}^T \mathsf{x}}}{1 + e^{\mathsf{w}^T \mathsf{x}}}$$

Jest to postać funkcji logistycznej – CRF jest uogólnieniem regresji logicznej dla sekwencji. ■

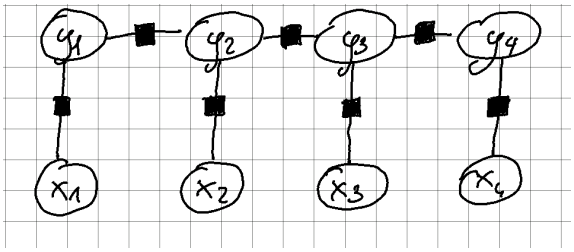
Czasami stosuje się uproszczoną wersję modelu 9.1, w którym funkcję oceny $\Psi_i(y_i, y_{i-1}, \mathsf{x})$ rozbija się na dwie mniejsze funkcje oceniające: $\Psi_{i'}(y_i, y_{i-1})$ oraz $\Psi_{i''}(y_i, \mathsf{x})$. Wtedy taki model w postaci graficznej wygląda tak jak na rysunku 9.4. Ten graf nieprzypadkowo ma taką samą strukturę jak graf dla ukrytych modeli Markowa: CRF na liniowym łańcuch jest dyskryminacyjną parą do generatywnego klasyfikatora HMM. Odnoszą się więc do niego analogiczne zależności i twierdzenia, które dotyczą poznanej na uczeniu maszynowym pary: regresja logistyczna i naiwny Bayes.

Dalej model warunkowych pól losowych na łańcuchu będziemy nazywać po prostu CRFem lub warunkowym polem losowym.

Problem 9.1 Narysuj graf Markowa dla naiwnego Bayesa oraz graf rozkładu na czynniki dla regresji logistycznej.

9.2.2 Inżynieria cech

Cechy dla klasyfikatora CRF $\phi(y_i, y_{i-1}, \mathsf{x})$ mają taką samą postać jak dla klasyfikatora MEMM, więc pomysły na ich tworzenie są analogiczne: włączając w to korzystanie z tokenów START i STOP, budowę cech patrzących na słowa w przód itd. Poruszymy jednak temat obsługi cech, które nigdy nie były aktywne w zbiorze uczącym.



Rysunek 9.4: Warunkowe pola losowe na łańcuchu w wersji uproszczonej

Cechy bez wsparcia w zbiorze uczącym

Cechy w inżynierii lingwistycznej często tworzone są masowo, używając pewnych szablonów. Na przykład, dla każdego możliwego słowa $w \in V$ i każdej możliwej etykiety y stwórz cechę binarną, przyjmującą jeden gdy $x_i = w \wedge y_i = y$ oraz zero w innym przypadku. Analogiczne cechy tworzy się często dla bigramów, kształtów słów itd. W rezultacie duża liczba (większość?) takich cech nigdy nie zwraca wartości innej niż 0 dla przykładów w zbiorze uczącym np. słowo „ale" raczej nie wystąpi w nazwie ulicy czy osoby.

Co interesujące, cechy bez pokrycia w zbiorze uczącym mogą być użyte przez klasyfikator CRF do poprawy wyników a wartość skojarzonych z nimi wag może zostać zmodyfikowana w procesie uczenia. Dzieje się tak, bo klasyfikator normalizuje swoją odpowiedź po wszystkich możliwych sekwencjach znaczników, niezależnie od tego czy występują one w zbiorze uczącym czy nie. Obniżenie oceny fragmentów sekwencji, które nie występują w zbiorze uczącym prowadzi do podwyższenia prawdopodobieństw sekwencji które w nim występują. Metody uczące będą więc zwykle przypisywać ujemne wagi cechom które nie występują w zbiorze uczącym.

Choć praca na pełnym zbiorze uczącym może skutkować wyższą trafnością modelu, jednak znacznie zwiększając liczbę jego parametrów. Sutton i McCallum [2] opisują tzw. „unsupported features trick" polegający na włączaniu tylko tych cech bez pokrycia w zbiorze uczącym, które prawdopodobnie pozwolą klasyfikatorowi na uniknięcie pewnych błędów. Na początku trenuje się wstępnie model CRF (bez uzyskania zbieżności, tylko kilka iteracji) używający tylko cechy z pokryciem w zbiorze danych. Następnie wstawia się do klasyfikatora cechy bez wsparcia w zbiorze, które występują we fragmentach sekwencji, które nie są prawidłowo klasyfikowane z dużą pewnością. Np. cecha aktywująca się przy użyciu słowa „ale" w nazwie ulicy zostałaby dodana do klasyfikatora tylko wtedy gdy po wstępnym treningu znaleźlibyśmy sekwencję ze słowem „ale" na i-tej pozycji w której $P(y_i = \text{STREET}|\mathsf{x})$ jest większe niż jakaś zdefiniowana wcześniej mała wartość.

Globalna interpretacja cech

Warto też zauważyć, że wzór na klasyfikator warunkowych pól losowych, można przekształcić dalej do postaci

$$P(\mathsf{y}|\mathsf{x}) = \frac{e^{\sum_{i=1}^{n} \mathsf{w}^T \phi(y_i, y_{i-1}, \mathsf{x})}}{\sum_{\mathsf{y}'} e^{\sum_{i=1}^{n} \mathsf{w}^T \phi(y'_i, y'_{i-1}, \mathsf{x})}} = \frac{e^{\mathsf{w}^T \sum_{i=1}^{n} \phi(y_i, y_{i-1}, \mathsf{x})}}{\sum_{\mathsf{y}'} e^{\mathsf{w}^T \sum_{i=1}^{n} \phi(y'_i, y'_{i-1}, \mathsf{x})}}$$

czyli do postaci klasyfikatora liniowego, którego predykcją jest cała sekwencja a cała sekwencja jest opisane poprzez wektor cech będący sumą cech poszczególnych fragmentów sekwencji $\sum_{i=1}^{n} \phi(y_i, y_{i-1}, \mathsf{x})$.

Cechy z innych modeli uczenia maszynowego

Na koniec, zauważmy że jedną z cech modelu może być też prawdopodobieństwo znacznika uzyskane przez inny model np. ukryty model Markowa. Jeśli użyjemy wynik modelu nauczonego na tym samym zbiorze danych taka cecha przyniesie znikome korzyści (jeśli nawet nie pogorszy trafności), jednak mając dostępny gotowy system HMM wytrenowany na innym zbiorze danych – taka cecha może być przydatna.

9.3 Algorytm propagacji przekonania dla łańcuchowych CRF

Trening powyżej przedstawionego modelu CRF nie wydaje się trudny: mając zdefiniowany i wyrażony probabilistycznie model wystarczy zastosować estymację maksymalnej wiarygodności i przystąpić do optymalizacji. Stosując nowoczesne środowiska programistyczne z funkcją automatycznego liczenia pochodnej, nawet wyprowadzenie wzorów na gradient nie powinno nam zaprzątać głowy. Niestety, naiwna implementacja i zastosowanie automatycznej pochodnej uniemożliwi ci wykonanie nawet jednego kroku treningu algorytmu CRF przed emeryturą.

Powodem zamieszania jest globalna normalizacja modelu CRF, która jest jego głównym wyróżnikiem i zaletą, jednak także główną trudnością w czasie treningu i optymalizacji. Przyglądając się bowiem równaniu 9.2 dostrzeżesz złowrogą sumę statystyczną $Z(\mathsf{x})$ która iteruje po wszystkich możliwych sekwencjach. Jest ich oczywiście c^n gdzie n to długość sekwencji, a c to liczba możliwych znaczników. Już dla średniej długości sekwencji i małych, z praktycznego punktu widzenia, zbiorów znaczników uzyskujemy liczbę iteracji bilony razy większą niż rozmiar słownika $|V|$, a przecież już przy tej liczbie spotykanej w modelach językach stosowaliśmy zaawansowane techniki tak jak softmax hierarchiczny czy próbkowanie ważnościowe aby przyśpieszyć trening i skończyć go w rozsądnym czasie.

W tył...

Z pomocą znów przyjdzie nam paradygmat programowania dynamicznego, tym razem pod postacią algorytmu propagacji przekonania[1] dla łańcuchowych CRF czyli algorytm w przód i w tył (ang. *forward–backward algorithm*). Rozpocznijmy przekształcanie wzoru na sumę statystyczną dla danego zdania, tak aby wyróżnić potencjalne wyniki cząstkowe które moglibyśmy buforować w pamięci.

$$
\begin{aligned}
Z(\mathsf{x}) &= \sum_{\mathsf{y}'} \prod_{i=1}^{n} e^{\mathsf{w}^T \phi(y'_i, y'_{i-1}, \mathsf{x})} = \sum_{y'_1} \sum_{y'_2} \cdots \sum_{y'_n} \prod_{i=1}^{n} e^{\mathsf{w}^T \phi(y'_i, y'_{i-1}, \mathsf{x})} \\
&= \sum_{y'_1} \sum_{y'_2} \cdots \sum_{y'_n} \left(e^{\mathsf{w}^T \phi(y'_1, y'_0, \mathsf{x})} e^{\mathsf{w}^T \phi(y'_2, y'_1, \mathsf{x})} \cdots e^{\mathsf{w}^T \phi(y'_n, y'_{n-1}, \mathsf{x})} \right) \\
&= \sum_{y'_1} \left(e^{\mathsf{w}^T \phi(y'_1, y'_0, \mathsf{x})} \sum_{y'_2} \left(e^{\mathsf{w}^T \phi(y'_2, y'_1, \mathsf{x})} \cdots \sum_{y'_{n-1}} \left(e^{\mathsf{w}^T \phi(y'_{n-1}, y'_{n-2}, \mathsf{x})} \sum_{y'_n} e^{\mathsf{w}^T \phi(y'_n, y'_{n-1}, \mathsf{x})} \right) \right) \right)
\end{aligned}
$$

[1] Algorytm propagacji przekonania (ang. *belief propagation*) jest algorytmem dla warunkowych pół losowych zdefiniowanych na dowolnych grafach. Algorytm ten polega na przekazywaniu specyficznych wiadomości pomiędzy wierzchołkami w grafie rozdziału na czynniki. Tu prezentujemy jego uproszczoną wersję dla łańcuchowych CRF: algorytm w przód i w tył.

Zauważ, że wartość najbardziej zagnieżdżonej sumy tak naprawdę zależy tylko od y'_{n-1} i można by ją policzyć tylko c razy, a potem już odczytywać gotową wartość z bufora. Zdefiniujmy więc bufor:

$$\beta_n(y) = \sum_{y'_n} e^{\mathsf{w}^T \phi(y'_n, y, \mathsf{x})}$$

z którego użyciem wzór na sumę statystyczną przekształca się w:

$$Z(\mathsf{x}) = \sum_{y'_1} \left(e^{\mathsf{w}^T \phi(y'_1, y'_0, \mathsf{x})} \sum_{y'_2} \left(e^{\mathsf{w}^T \phi(y'_2, y'_1, \mathsf{x})} \cdots \sum_{y'_{n-1}} \left(e^{\mathsf{w}^T \phi(y'_{n-1}, y'_{n-2}, \mathsf{x})} \beta_n(y'_{n-1}) \right) \right) \right)$$

Nietrudno się zorientować, że obecnie najbardziej zagnieżdżona suma również może zostać zbuforowana i możemy uzyskać ogólny wzór:

$$\beta_i(y) = \sum_{y'_i} e^{\mathsf{w}^T \phi(y'_i, y, \mathsf{x})} \beta_{i+1}(y'_i)$$

a po podstawieniu

$$\begin{aligned} Z(\mathsf{x}) &= \sum_{y'_1} \left(e^{\mathsf{w}^T \phi(y'_1, y'_0, \mathsf{x})} \sum_{y'_2} \left(e^{\mathsf{w}^T \phi(y'_2, y'_1, \mathsf{x})} \cdots \sum_{y'_{n-2}} \left(e^{\mathsf{w}^T \phi(y'_{n-2}, y'_{n-3}, \mathsf{x})} \beta_{n-1}(y'_{n-2}) \right) \right) \right) \\ &= \cdots = \sum_{y'_1} e^{\mathsf{w}^T \phi(y'_1, y'_0, \mathsf{x})} \beta_2(y'_1) = \beta_1(y'_0) = \beta_1(\boxed{\text{START}}) \end{aligned}$$

Przy czym dla ostatniej sumy wzór na bufor przeradza się w $\beta_1(y) = \sum_{y'_1} e^{\mathsf{w}^T \phi(y'_1, y, \mathsf{x})} \beta_2(y'_1)$, ale tylko jeden znacznik jest możliwy na pozycji y_0 i jest to $\boxed{\text{START}}$. Wystarczy więc odczytać $\beta_1(\boxed{\text{START}})$ aby uzyskać wynik całej sumy statystycznej. Bufory β_i nazywa się czasami buforami w tył lub – ponieważ mają one wartość dla każdego znacznika – wektorami w tył. Nazwa bierze się z tego, że musimy rozpocząć ich uzupełnianie od końca sekwencji tj. bufora β_n, a następnie przesuwać się w tył sekwencji aż do β_1.

... i w przód

Analogiczne przekształcenie można też zrobić aby uzyskać wektory/bufory w przód, po prostu zapisując sumy w odwrotnym kierunku niż poprzednio:

$$\begin{aligned} Z(\mathsf{x}) &= \sum_{\mathsf{y}'} \prod_{i=1}^{n} e^{\mathsf{w}^T \phi(y'_i, y'_{i-1}, \mathsf{x})} = \sum_{y'_n} \sum_{y'_{n-1}} \cdots \sum_{y'_1} \prod_{i=1}^{n} e^{\mathsf{w}^T \phi(y'_i, y'_{i-1}, \mathsf{x})} \\ &= \sum_{y'_n} \sum_{y'_{n-1}} \cdots \sum_{y'_1} \left(e^{\mathsf{w}^T \phi(y'_n, y'_{n-1}, \mathsf{x})} e^{\mathsf{w}^T \phi(y'_{n-1}, y'_{n-2}, \mathsf{x})} \cdots e^{\mathsf{w}^T \phi(y'_1, y'_0, \mathsf{x})} \right) \\ &= \sum_{y'_n} \sum_{y'_{n-1}} \left(e^{\mathsf{w}^T \phi(y'_n, y'_{n-1}, \mathsf{x})} \sum_{y'_{n-2}} \left(e^{\mathsf{w}^T \phi(y'_{n-1}, y'_{n-2}, \mathsf{x})} \cdots \sum_{y'_2} \left(e^{\mathsf{w}^T \phi(y'_2, y'_1, \mathsf{x})} \sum_{y'_1} e^{\mathsf{w}^T \phi(y'_1, y'_0, \mathsf{x})} \right) \right) \right) \end{aligned}$$

Wartości buforów w przód mogą zostać obliczone wzorem:

$$\alpha_1(y) = 1$$

$$\alpha_i(y) = \sum_{y'_{i-1}} e^{\mathsf{w}^T \phi(y, y'_{i-1}, \mathsf{x})} \alpha_{i-1}(y'_{i-1})$$

co pozwala nam na zwijające przekształcenia o ile przemnożymy najbardziej zagnieżdżoną sumę przez $\alpha_1(y)$ czyli z definicji nie zmieniając wyniku (mnożenie przez 1).

$$\begin{aligned}
Z(\mathsf{x}) &= \sum_{y'_n} \sum_{y'_{n-1}} \left(e^{\mathsf{w}^T \phi(y'_n, y'_{n-1}, \mathsf{x})} \sum_{y'_{n-2}} \left(e^{\mathsf{w}^T \phi(y'_{n-1}, y'_{n-2}, \mathsf{x})} \cdots \sum_{y'_2} \left(e^{\mathsf{w}^T \phi(y'_2, y'_1, \mathsf{x})} \sum_{y'_1} e^{\mathsf{w}^T \phi(y'_1, y'_0, \mathsf{x})} \alpha_1(y'_1) \right) \right) \right) \\
&= \sum_{y'_n} \sum_{y'_{n-1}} \left(e^{\mathsf{w}^T \phi(y'_n, y'_{n-1}, \mathsf{x})} \sum_{y'_{n-2}} \left(e^{\mathsf{w}^T \phi(y'_{n-1}, y'_{n-2}, \mathsf{x})} \cdots \sum_{y'_2} \left(e^{\mathsf{w}^T \phi(y'_2, y'_1, \mathsf{x})} \alpha_2(y'_2) \right) \right) \right) \\
&= \cdots = \sum_{y'_n} \sum_{y'_{n-1}} \left(e^{\mathsf{w}^T \phi(y'_n, y'_{n-1}, \mathsf{x})} \alpha_{n-1}(y'_{n-1}) \right) \\
&= \sum_{y'_n} \alpha_n(y'_n)
\end{aligned}$$

Wektory α należy wypełniać w przód tj. zaczynając od wektora α_1 i przesuwać się wzdłuż sekwencji aż do α_n. Aby uzyskać wartość sumy statystycznej należy zsumować wartości ostatniego wektora po wszystkich możliwych etykietach. Jeśli z definicji y_n może być jedynie $\boxed{\text{STOP}}$ to wystarczy odczytać wartość $Z(\mathsf{x}) = \alpha_n(\boxed{\text{STOP}})$.

Z punktu widzenia obliczenia prawdopodobieństwa i uruchomienia mechanizmu automatycznego liczenia gradientu poznaliśmy dwa równie dobre sposoby na policzenie sumy statystycznej: albo iterując w przód uzupełniając wektory α albo iterując w tył konstruując wektory β. Nie ma sensu liczyć tę samą rzecz dwukrotnie – chyba że dla sprawdzenia poprawności implementacji! Jak się jednak okaże wartości α i β pełnią niebagatelną rolę w predykcji i obliczeniu gradientów drogą analityczną – stąd w praktyce konieczne jest obliczenie obydwu zestawów wektorów.

Uwaga implementacyjna

Liczenie sumy statystycznej rozkładu zdefiniowanej przez warunkowe pola losowe wymaga od nas sumowania wyników funkcji eksponencjalnej. Wartość funkcji eksponencjalnej już dla wartości 20 jest liczba 9 cyfrową ($e^{20} = 485165195$), więc należy spodziewać się sumowania dużych liczb. Powoduje to oczywiście niestabilności numeryczne, których możemy uniknąć pracując w przestrzeni logarytmicznej tj. przechowując w pamięci wartości logarytmów wyników. Na przykład dla wektorów w przód otrzymujemy:

$$\log \alpha_1(y) = \log 1 = 0$$

$$\log \alpha_i(y) = \log \sum_{y'_{i-1}} e^{\mathsf{w}^T \phi(y, y'_{i-1}, \mathsf{x})} \alpha_{i-1}(y'_{i-1}) = \log \sum_{y'_{i-1}} e^{\mathsf{w}^T \phi(y, y'_{i-1}, \mathsf{x}) + \log \alpha_{i-1}(y'_{i-1})}$$

Niestety nie możemy wejść z logarytmem do środka sumy i za każdym razem, aby obliczyć nową wartość bufora, i tak musimy sumować potencjalnie duże wartości potęg i dopiero potem je logarytmować.

Z pomocą przychodzi nam następująca własność logarytmu sumy funkcji eksponencjalnych dla dowolnej stałej c.

$$\log \sum_i e^{z_i} = \log \sum_i e^{z_i - c + c} = \log \left(e^c \sum_i e^{z_i - c} \right) = c + \log \sum_i e^{z_i - c} \tag{9.3}$$

W szczególności, aby uniknąć sumowania potęg o dużych wykładnikach (co może nie być stabilne numerycznie) możemy podstawić $c = \max z_i$ i uzyskać wzór:

$$\log \sum_i e^{z_i} = \max_i(z_i) + \log \sum_i e^{z_i - \max_i(z_i)}$$

co pozwala nam na uzyskanie znaczenie stabilniejszych wyników. Pomimo tego, że nadal mamy sumowanie wyników funkcji eksponencjalnej wykładniki potęg są w nowym wzorze znacznie mniejsze (są mniejsze równe 0).

9.4 Predykcja modelem CRF

Podobnie jak w przypadku innych modeli sekwencyjnych, mamy możliwość zdekodowania sekwencji na dwa sposoby: algorytmem zachłannym oraz algorytmem Vietrbiego.

W przypadku algorytmu zachłannego iterujemy po sekwencji np. od prawej do lewej za każdym razem wybierając najbardziej prawdopodobną etykietę na pozycji i-tej. Rozkład prawdopodobieństwa modelowany przez CRF jest jednak zdefiniowany na całej sekwencji znaczników. Aby otrzymać rozkład prawdopodobieństwa znaczników na konkretnej pozycji należy przekształcić wzór 9.2 poprzez marginalizację prawdopodobieństwa. W czasie obliczeń można zauważyć, że prawdopodobieństwo to można wyrazić przy pomocy wektorów w przód i w tył.

$$P(y_i|\mathsf{x}) = \frac{\alpha_i(y_i)\beta_{i+1}(y_i)}{Z(\mathsf{x})}$$

Problem 9.2 Wyprowadź powyższy wzór poprzez marginalizację rozkładu prawdopodobieństwa.

Inną metodą jest wyznaczenie najbardziej prawdopodobnej sekwencji algorytmem Viterbiego. Ponieważ interesuje nas znalezienie maksymalnej wartości prawdopodobieństwa - możemy szukać maksimum po nieznormalizowanych wartościach

$$\begin{aligned}
\max_{\mathsf{y}} P(\mathsf{y}|\mathsf{x}) &= \max_{\mathsf{y}} e^{\sum_{i=1}^n \mathsf{w}^T \phi(y_i, y_{i-1}, \mathsf{x})} = \max_{\mathsf{y}} \sum_{i=1}^n \mathsf{w}^T \phi(y_i, y_{i-1}, \mathsf{x}) \\
&= \max_{y_1, y_2, \ldots, y_n} \sum_{i=1}^n \mathsf{w}^T \phi(y_i, y_{i-1}, \mathsf{x}) = \max_{y_n} \max_{y_1, y_2, \ldots, y_{n-1}} \sum_{i=1}^n \mathsf{w}^T \phi(y_i, y_{i-1}, \mathsf{x}) \\
&= \max_{y_n} \max_{y_{n-1}} \mathsf{w}^T \phi(y_n, y_{n-1}, \mathsf{x}) + \max_{y_1, y_2, \ldots, y_{n-2}} \sum_{i=1}^{n-1} \mathsf{w}^T \phi(y_i, y_{i-1}, \mathsf{x})
\end{aligned}$$

Rozwijając dalej to równanie otrzymamy w końcu skrajnie po prawej term $\mathsf{w}^T \phi(y_1, y_0, \mathsf{x})$ po którym będzie wyciągane maksimum po wszystkich możliwych wartościach y_1. Wartość tego termu dla każdego możliwego podstawienia y_1 może zostać zapisana w pamięci podręcznej:

$$B_1(y) = \mathsf{w}^T \phi(y, y_0, \mathsf{x}) = \mathsf{w}^T \phi(y, \boxed{\text{START}}, \mathsf{x})$$

Kolejny składnik wzoru także można obliczyć i zapisać w pamięci podręcznej:

$$B_i(y) = \max_{y'} \mathsf{w}^T \phi(y, y', \mathsf{x}) + B_{i-1}(y')$$

Ostatecznie, aby odczytać ostatni znacznik najbardziej prawdopodobnej sekwencji należy wykonać $\hat{y}_n = \arg\max_{y_n} B_n(y_n)$, a następnie prześledzić ścieżkę jaką przeszedł algorytm Viterbiego w przestrzeni znaczników, aby skonstruować ten wynik. Algorytm, jak poprzednio, ma złożoność $O(c^2 n)$.

9.5 Trening modeli CRF

Trening warunkowych łańcuchowych pól losowych najczęściej jest wykonywany poprzez optymalizację funkcji wiarygodności. Funkcja wiarygodności modelu CRF jest funkcją wklęsłą, co oznacza że używając odpowiednich metod optymalizacji ciągłej mamy gwarancję osiągnięcia globalnego optimum (lub przynajmniej znalezienia się w jego ścisłym sąsiedztwie).

Większość metod optymalizacji wymaga od nas obliczenia gradientu optymalizowanej funkcji. Przystąpmy więc do obliczeń, które przeprowadzimy dla jednej konkretnej obserwacji uczącej x, y zamiast dla całego zbioru. Będzie to więc gradient dla aktualizacji w stylu stochastycznego spadku wzdłuż gradientu (SGD).

$$
\begin{aligned}
\nabla_{\mathsf{w}} \log P(\mathsf{y}|\mathsf{x}) &= \nabla_{\mathsf{w}} \log \frac{e^{\mathsf{w}^T \sum_{i=1}^{n} \phi(y_i, y_{i-1}, \mathsf{x})}}{\sum_{\mathsf{y}'} e^{\mathsf{w}^T \sum_{i=1}^{n} \phi(y'_i, y'_{i-1}, \mathsf{x})}} \\
&= \nabla_{\mathsf{w}} \left[\mathsf{w}^T \sum_{i=1}^{n} \phi(y_i, y_{i-1}, \mathsf{x}) - \log \sum_{\mathsf{y}'} e^{\mathsf{w}^T \sum_{i=1}^{n} \phi(y'_i, y'_{i-1}, \mathsf{x})} \right] \\
&= \sum_{i=1}^{n} \phi(y_i, y_{i-1}, \mathsf{x}) - \frac{1}{\sum_{\mathsf{y}''} e^{\mathsf{w}^T \sum_{i=1}^{n} \phi(y''_i, y''_{i-1}, \mathsf{x})}} \nabla_{\mathsf{w}} \sum_{\mathsf{y}'} e^{\mathsf{w}^T \sum_{i=1}^{n} \phi(y'_i, y'_{i-1}, \mathsf{x})} \\
&= \sum_{i=1}^{n} \phi(y_i, y_{i-1}, \mathsf{x}) - \sum_{\mathsf{y}'} \underbrace{\frac{e^{\mathsf{w}^T \sum_{i=1}^{n} \phi(y'_i, y'_{i-1}, \mathsf{x})}}{\sum_{\mathsf{y}''} e^{\mathsf{w}^T \sum_{i=1}^{n} \phi(y''_i, y''_{i-1}, \mathsf{x})}}}_{CRF} \sum_{i=1}^{n} \phi(y'_i, y'_{i-1}, \mathsf{x}) \\
&= \sum_{i=1}^{n} \phi(y_i, y_{i-1}, \mathsf{x}) - \mathbb{E}_{\mathsf{y}' \sim P(\mathsf{y}|\mathsf{x})} \left[\sum_{i=1}^{n} \phi(y'_i, y'_{i-1}, \mathsf{x}) \right] \\
&= \sum_{i=1}^{n} \phi(y_i, y_{i-1}, \mathsf{x}) - \sum_{i=1}^{n} \mathbb{E}_{\mathsf{y}' \sim P(\mathsf{y}|\mathsf{x})} \left[\phi(y'_i, y'_{i-1}, \mathsf{x}) \right] \\
&= \sum_{i=1}^{n} \phi(y_i, y_{i-1}, \mathsf{x}) - \sum_{i=1}^{n} \mathbb{E}_{(y'_i, y''_{i-1}) \sim P(y_i, y_{i-1}|\mathsf{x})} \left[\phi(y'_i, y'_{i-1}, \mathsf{x}) \right]
\end{aligned}
$$

Do policzenia powyższego wzoru potrzebne jest wyznaczenie $P(y_i, y_{i-1}|\mathsf{x})$. Na szczęście z pomocą w jego wyznaczeniu przychodzą nam wektory w przód i w tył:

$$
P(y_i, y_{i-1}|\mathsf{x}) = \frac{\alpha_{i-1}(y_{i-1}) e^{\mathsf{w}^T \phi(y_i, y'_{i-1}, \mathsf{x})} \beta_{i+1}(y_i)}{Z(\mathsf{x})}
$$

które sprawiają że możemy je obliczyć w czasie stałym (mając gotowy wynik działania algorytmu propagacji przekonania).

Zakończmy kilkoma uwagami praktycznymi. Algorytm spadku wzdłuż gradientu okazują się w praktyce bardzo wolny w optymalizacji modeli CRF. Z tego powodu korzystamy z metod optymalizacyjnych cieszącymi się lepszymi gwarancjami zbieżności (⇒ Optymalizacja ciągła). Popularne jest stosowanie metod quasi-Newtonowskich, które estymują odwrotność Hesjanu np. poprzez zaledwie kilka wartości i wektorów własnych. Jedną z najczęściej spotykanych metod w pakietach do uczenia modeli CRF jest algorytm L-BFGS (ang. *limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm*), który –

podobnie jak metoda Newtona – nie wymaga od nas strojenia żadnego parametru kontrolującego szybkość optymalizacji η, a ponieważ Hesjan jest tam przybliżany nie wymaga od nas implementacji obliczania Hesjanu. Innym algorytmem często stosowanym w praktyce jest metoda gradientów sprzężonych.

Duże znaczenie praktyczne w uzyskaniu wysokiej wydajności działania modeli CRF jest wykorzystanie faktu rzadkości cech. Operacje mnożenia wektorowego warto implementować na wektorach rzadkich, ponieważ liczba cech w modelach warunkowych pól losowych często przekracza kilka milionów.

Do funkcji wiarygodności, podobnie jak dla innych modeli uczących, zwykle dodajemy jakąś formę regularyzacji. Na przykład, poprzez jednoczesne minimalizowanie normy L_2 wag. Stałą kontrolującą wielkość regularyzacji ustala się na stałą wartość, ale w podejściach stochastycznych czasami zmienia się ją w zależności od długości analizowanej sekwencji.

Trening CRF dla innych struktur niż łańcuch (lub drzewo) rodzi dodatkowe komplikacje. Funkcja wiarygodności przestaje być wklęsła, algorytm propagacji przekonania trzeba zastosować w pełnej wersji i może nie osiągać zbieżności. Z tego powodu stosowane są różne podejścia przybliżone m.in. oparte na próbkowaniu Monte Carlo lub zamianie optymalizowanej funkcji na pseudo-wiarygodność. Pomimo tego modele te są często używane np. w rozpoznawaniu obrazów oraz innych dziedzinach wymagających predykcji struktur.

Dodatki

Materiały powtórkowe

Dobry opis warunkowych modeli losowych można znaleźć w materiałach Charlesa Elkana `http://cseweb.ucsd.edu/~elkan/250B/CRFs.pdf`. Bardziej wyczerpujący opis, wraz z uogólnieniem na predykcję dowolnych struktur można znaleźć w tutorialu [2]

Materiały dla chętnych

Dla chętnych polecam opis innych klasycznych metod do predykcji struktur: strukturalnego perceptronu i strukturalnych maszyn wektorów podpierających w dostępnej online książce [1].

Bibliografia

[1] Hal Daumé III. A course in machine learning, 2017. URL `http://ciml.info/dl/v0_99/ciml-v0_99-ch17.pdf`.

[2] Charles Sutton, Andrew McCallum, i in. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373, 2012.