Kontradyktoryjne uczenie maszynowe

Phillip Kuznetsov, Riley Edmunds, Ted Xiao, Humza Iqbal, Raul Puri, Noah Golmant, Shannon Shih

Wprowadzenie

Dzięki ogromnej ilości zasobów oraz znacznego zainteresowania skupionego ostatnio na SI można zaobserwować liczne formy inteligentnych agentów wyposażonych w różnorodne unikatowe i nowatorskie możliwości. Potencjał oddziaływania jest nieograniczony, aczkolwiek odnotowywano już przykłady, w których decyzje podejmowane przez sztuczną inteligencję były niezrozumiałe.

W ostatnim dziesięcioleciu uczenie maszynowe, a ściślej sieci neuronowe, przyczyniły się do eksplozji rozwoju SI. Naukowcy wykorzystali sieci neuronowe, aby poczynić bezprecedensowe postępy w rozwiązywaniu otwartych problemów w takich obszarach, jak rozpoznawanie obrazów, przetwarzanie dźwięku, modelowanie języka i wiele innych. Najnowocześniejsze sieci neuronowe często przewyższają zdolności ludzi w tych dziedzinach.

Pomimo takiego postępu niewykrywalne dla ludzi zaburzenia danych wejściowych mogą oszukać sieci neuronowe. W rzeczywistości, jeśli takie zaburzenie zostanie starannie stworzone, to sieć neuronowa będzie bardzo pewna swoich złych prognoz! Ten rodzaj zakłóceń pokazano na rysunku 1.

Tego rodzaju obrazy i dane są znane jako przykłady kontradyktoryjne, a skupiona na tym dziedzina uczenia maszynowego znana jest jako kontradyktoryjne uczenie maszynowe. W tym rozdziale przedstawiono ramy ataku polegającego na kontradyktoryjnym uczeniu maszynowym, omówiono warianty tego zagrożenia, potencjalną obronę, a na koniec kilka dziedzin, które są podatne na tego typu ataki.

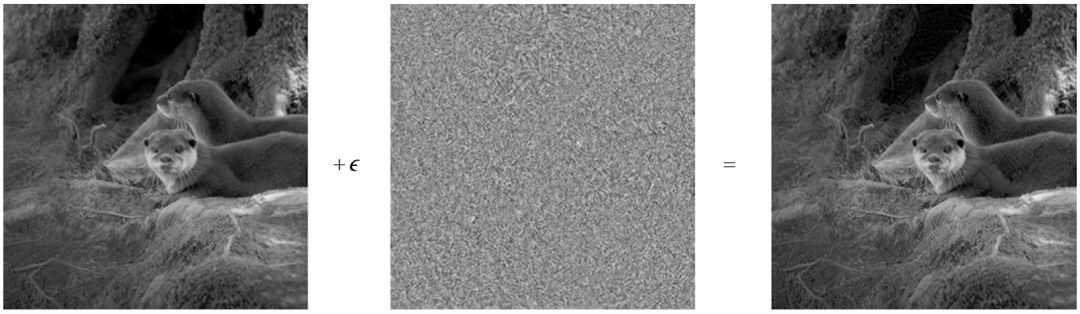
Samochody samojezdne i kontradyktoryjne uczenie maszynowe: studium przypadku

Samochody samojezdne zyskały ogromną uwagę w ciągu ostatnich kilku lat. Firmy takie, jak Uber, Google, GM i Apple, pracują nad rozwojem solidnych i bezpiecznych, w pełni autonomicznych samochodów przeznaczonych dla konsumentów.

Ostatnie zmiany w uczeniu maszynowym sprawiły, że samochody samojezdne stały się możliwe. Na co jednak dokładnie powinien zwracać uwagę dobry autonomiczny kierowca? Na inne samochody? Pogodę? Przeszkody? Uwzględnienie setek takich dynamicznych zmiennych sprawia, że opracowanie samochodów samojezdnych jest tak trudne. Jakkolwiek jednak trudny byłby to problem, to społeczna korzyść z takiego rozwiązania jest ogromna. W 2016 roku w samych Stanach Zjednoczonych odnotowano ponad 40 000 ofiar śmiertelnych wypadków drogowych [1], z których większość wynikała ze złych decyzji człowieka oraz z ograniczonej ludzkiej percepcji.

Możemy bezpiecznie założyć, że samochody samojezdne ostatecznie osiągną lepsze wyniki od ludzkich kierowców, biorąc pod uwagę ogrom danych, które komputer może przetworzyć. Dzięki zainstalowaniu kamer wokół pojazdu i wykorzystaniu zaawansowanych narzędzi, takich jak LIDAR, samochód samojezdny może postrzegać otoczenie w zakresie 360°. Samochody samojezdne wykorzystują sieci neuronowe do przekształcania tak dużej ilości danych na wejścia do układów sterowania i przyspieszania samochodu. Algorytmy te mogą skutecznie rozpoznawać przeszkody i znaki drogowe w czasie rzeczywistym. Jednak, jak można zobaczyć na rysunku 1, algorytmy te są również podatne na oszukiwanie. Podatność autonomicznych systemów prowadzenia pojazdu na ataki polegające na kontradyktoryjnym uczeniu maszynowym stanowi poważne zagrożenie dla bezpieczeństwa pasażerów i pieszych.

Co się stanie, jeśli w świecie pełnym samochodów samojezdnych złośliwy osobnik umieści naklejkę na znaku Stop, co oszuka system wizyjny samochodu, który uzna, że znak Stop to tak naprawdę zielone światło? Taki znak Stop prawdopodobnie spowodowałby liczne wypadki, a jego wadliwość byłaby bardzo



„Wydra”
98,5% zaufania

„Komputer stacjonarny”
20,4% zaufania

„Rekin”
94,8% zaufania

Rys. 1. Atakujący wykonuje zdjęcie wydry (po lewej) dodaje niewielkie zaburzenia pikseli, które wygląda jak zwykły szum i oszukuje sieć neuronową, która z dużą pewnością sklasyfikuje obraz jako rekina. Dla człowieka nowy obraz wygląda na niezmieniony

trudna do wykrycia. Nasza niezdolność do zaprojektowania sieci neuronowych odpornych na tego rodzaju ataki może prowadzić do wyjątkowo niebezpiecznych sytuacji dla pasażerów i innych samochodów na drodze.

Model funkcyjny czarnej skrzynki

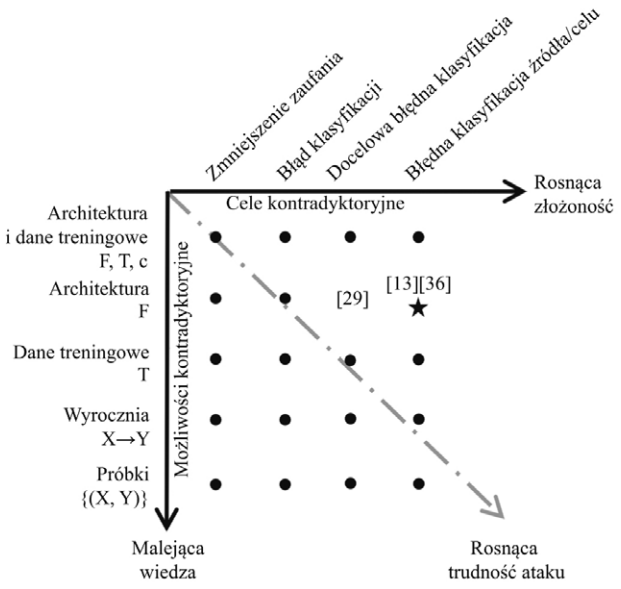
W celu zrozumienia, dlaczego istnieją przykłady kontradyktoryjne, ważne jest, aby pojąć, jak działają modele uczenia maszynowego. Na najbardziej ogólnym poziomie możemy postrzegać model uczenia maszynowego jako czarną skrzynkę. Model pobiera pewne dane wejściowe, takie jak obraz lub dźwięk, przetwarza dane zgodnie z pewną wewnętrzną logiką, a następnie wytwarza pożądany wynik. Dane wejściowe są zwykle określane jako *przykład* i oznaczone przez x, natomiast pożądane dane wyjściowe są określane jako *etykieta* oznaczona przez y[1]. Model, któremu podano przykład X, dający wynik $\hat{Y}$ równoważny etykiecie Y, jest modelem działającym zgodnie z przeznaczeniem.

Istnieją dwa główne typy problemów, do których rozwiązywania wykorzystuje się modele uczenia maszynowego. Jest to klasyfikacja, która identyfikuje kategorie, takie jak obiekty na obrazach lub nowotwory nieszkodliwe w porównaniu z nowotworami złośliwymi, oraz regresja, która pozwala przewidywać rzeczywiste etykiety, takie jak cena domu lub cena akcji za 10 dni od dzisiaj. W celu upewnienia się, że model identyfikuje odpowiednie obiekty lub dokonuje właściwych prognoz, model musi przetworzyć dużą liczbę przykładów i towarzyszących im etykiet, takich jak tysiąc zdjęć kotów i ich ras lub milion różnych domów wraz z odpowiadającymi im cenami. Ten proces jest znany jako uczenie, a wykorzystywane dane są określane jako *dane uczące*. Takie dane uczące składają się z wielu par (x, y), zwykle oznaczonych jako zbiór $\{(x_1, y_1),\ldots (x_n, y_n)\}$. Używamy metryki znanej z funkcji strat, tak aby porównać przewidywania wyjściowe z prawdziwymi etykietami. W ten sposób możemy ilościowo zmierzyć, o ile ulepszony został model. W celu nauczenia modelu obliczamy pochodną funkcji strat i używamy jej do aktualizacji parametrów modelu. Ilekroć używamy modelu do przewidywania, mówimy, że model wykonuje *wnioskowanie*.

Takie przedstawienie modelu uczenia maszynowego jest przydatne, ponieważ pozwala opisać kontradyktoryjne uczenie maszynowe przez używanie wygodnych abstrakcji. Kontradyktoryjny atakujący chciałby zamienić dane wejściowe x na kontradyktoryjne przykłady $\tilde{x}$, tak aby model zamiast poprawnego wyniku y dawał niepoprawny wynik $\tilde{y}$, a przykład $\tilde{x}$ wyglądał dla człowieka tak samo jak x. Umożliwia to atakującemu skuteczne „zhakowanie" systemu przewidywania modelu, tak by ludzie nie byli w stanie wykryć tego aż do momentu wyrządzenia szkody.

Taksonomia zagrożeń kontradyktoryjnych

Do rozważania kontradyktoryjnych zagrożeń dla modeli uczenia maszynowego konieczne jest posiadanie umiejętności kategoryzowania ataków. Została tu wykorzystana taksonomia zaproponowana przez Papernota i innych, która rozróżnia ataki na podstawie reakcji modelu pożądanej przez atakującego i ilości ujawnionych informacji o modelu [2]. Schemat tej



Rys. 2. Taksonomia ataków kontradyktoryjnych. Na osi poziomej przedstawiono cel atakującego według wzrostu złożoności. Na osi pionowej zaznaczono w kolejności rosnącej ilość informacji, którą atakujący musi znać, aby wykonać atak. Im bardziej złożony jest atak, tym trudniej jest go przeprowadzić. Podobnie, im więcej osoba atakująca wie o modelu, tym łatwiej jest wykonać atak. (Zaczerpnięte z: PAPERNOT N. I IN., 2016. *The limitations of deep learning in adversarial settings*, w: Security and Privacy (Euros&P), 2016 IEEE European Symposium, s. 372–387, IEEE)

zależności przedstawiono na rysunku 2. Kategoryzacja koncentruje się przede wszystkim na klasyfikacji, aczkolwiek zagrożenia dla problemów regresji nie różnią się zbytnio.

Kategorie zagrożeń

Zagrożenia związane z redukcją zaufania. Ten typ ataku zmniejsza zaufanie modelu do jego prognoz. W podobny sposób, w jaki szron na szybie przesłania szczegóły obiektów po drugiej stronie szyby, przeciwnik może chcieć dodać do przykładów szum w celu sprawienia, by informacja tam zawarta stała się niejasna. Są to zwykle najłatwiejsze ataki dla atakującego, ponieważ istnieje wiele możliwości dokonania zmian w obrazie, które zmniejszyłyby zaufanie modelu do jego prognozy. Na przykład system samojezdnego samochodu może zakładać, że poprawnie odczytał znak, jeśli sieć neuronowa uzyskuje 95% pewności w przewidywaniu tego znaku. Przeciwnik może chcieć zmodyfikować znak tak, aby obniżyć zaufanie sieci neuronowej poniżej takiego progu, potencjalnie powodując problemy z systemem kierowania samochodem.

Zagrożenia związane z błędną klasyfikacją źródłową. Atak tego typu jest bardziej ekstremalną wersją ataku polegającego na zmniejszeniu zaufania. Zamiast po prostu obniżyć pewność prognozy, atakujący próbuje oszukać klasyfikator, tak aby oznaczył obraz jako coś innego niż prawdziwa etykieta y. Zagrożenie spowodowane ukierunkowaną błędną klasyfikacją. Ataki tego typu mają na celu oszukanie modelu, tak aby przewidywał etykiety obrazu jako należące do określonej klasy docelowej $\tilde{y}$. Jest to wykonywane przez wygenerowanie przykładów

wyglądających dla ludzkiego oka jak szum, które jednak użyte jako dane wejściowe do modelu powodują przypisanie z dużą pewnością nieprawidłowej etykiety y.

Zagrożenia związane z błędną klasyfikacją źródło – cel. Zagrożenia te są najbardziej specyficznymi formami ataku. Ich celem jest takie oszukanie modelu, aby identyfikował przykład x (z prawdziwą etykietą y) jako należący do określonej klasy $\tilde{y}$. Są to najtrudniejsze do wykonania typy ataku, ponieważ istnieje znacznie mniej miejsca na błąd podczas przekształcania x w przykład kontradyktoryjny $\tilde{x}$, który model sklasyfikuje jako $\tilde{y}$. Błędna klasyfikacja źródło – cel jest również najbardziej destrukcyjną kategorią ataku. Wcześniej wspomniany przykład znaku stopu należy właśnie do tej kategorii.

Hierarchia informacji

Ważną rolę w stopniu trudności przeprowadzenia ataku, oprócz samego rodzaju ataku, odgrywają informacje dostępne dla atakującego. Atakujący, który zna konstrukcję modelu oraz dane użyte do jego uczenia, jest znacznie lepiej przygotowany niż atakujący, który ma dostęp tylko do prognoz modelu. Papernot i inni skategoryzowali poziomy informacji, które mogą być dostępne dla użytkownika w następujący sposób [2]:

Dane uczące i architektura. Atakujący wie wszystko o modelu i danych uczących, od dokładnego projektu architektury modelu i wyuczonych parametrów modelu, po dane użyte do uczenia modelu. Dla atakującego jest to najłatwiejszy scenariusz stworzenia kontradyktoryjnych przykładów.

Architektura modelu. Atakujący wie coś o wewnętrznej strukturze modelu, ale nie wie nic o danych używanych do uczenia modelu.

Dane uczące. Przeciwnik wie coś o danych użytych do uczenia modelu, ale nie wie nic o modelu.

Wyrocznia. Model jest dla atakującego tylko czarną skrzynką. Atakujący może przedstawić modelowi dowolne wejście i obserwować odpowiednie dane wyjściowe, jednak nie ma żadnej informacji o tym, jaka logika wewnętrzna została użyta do wygenerowania tego wyniku.

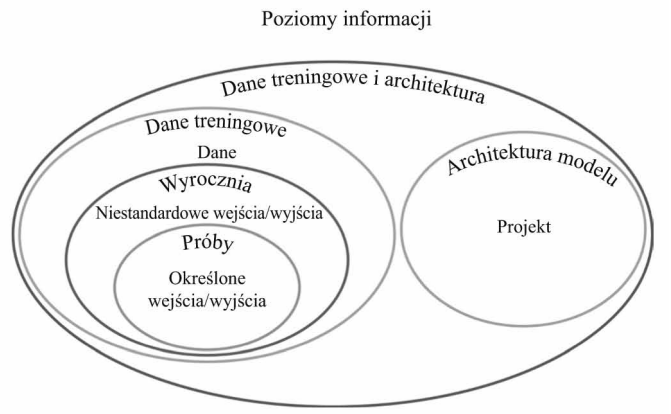
Próbki. Atakujący ma tylko listę danych wejściowych i odpowiadające im dane wyjściowe. W przeciwieństwie do poprzedniego poziomu informacji, atakujący nie może przedstawić modelowi nowych danych wejściowych i musi polegać wyłącznie na posiadanych danych wejściowych i wyjściowych.

Zachęcamy Czytelnika do odniesienia się do rysunków 2 i 3, ponieważ stanowią one trzon poniższej dyskusji na temat ataków kontradyktoryjnych i obrony. Zrozumienie tych liczb i związanych z nimi relacji stanowi duży krok w kierunku zrozumienia kompromisu pomiędzy narażeniem modelu a podatnością na zagrożenia, które leżą u podstaw kontradyktoryjnego uczenia maszynowego.

Omówiono modele z tych kategorii i poziomów hierarchii informacji, aczkolwiek skupiono się przede wszystkim na tych, które stanowią bardziej bezpośrednie zagrożenia oraz mają największy wpływ w różnych sytuacjach.

Omówienie problemu wysokiego poziomu

W tej części omówiono, w jaki sposób osoba atakująca może wygenerować przykłady kontradyktoryjne. Załóżmy, że agent,



Rys. 3. Hierarchia informacji dostępnych dla modelu

taki jak firma, rząd, hobbysta, uruchamia model uczenia maszynowego w określonym celu. Osoba atakująca może oszukać ten model z jakimś złośliwym zamiarem. Taki atakujący musiałby najpierw uzyskać pewne informacje o modelu, starając się znaleźć jak najwyżej w hierarchii z rysunku 3. Korzystając z tych informacji, atakujący tworzy następnie pseudomodel, który uczy się wykonywania tego samego zadania co model oryginalny. Pseudomodel jest przybliżoną kopią modelu docelowego, którą tworzy atakujący w celu przetestowania strategii ataku. Więcej szczegółów na temat tego procesu omówiono w części „Budowanie pseudomodelu". Atakujący może następnie znaleźć słabe punkty w pseudomodelu, przeprowadzając jeden z ataków opisanych w części „Ataki na model". Według wyników omówionych w części „Budowanie pseudomodelu" każdy skuteczny atak przeciwko pseudomodelowi ma duże prawdopodobieństwo powodzenia również w stosunku do oryginalnego modelu.

Krótko mówiąc, atakujący budują repliki modeli, które chcieliby zaatakować, znajdują wady w tych replikach, a następnie konstruują kontradyktoryjne przykłady, które oszukują repliki modelu, a w rezultacie i model oryginalny.

Dyskusję oparto na części „Samochody samojezdne i kontradyktoryjne uczenie maszynowe: studium przypadku". Załóżmy, że istnieje fikcyjna firma samochodowa o nazwie GoDriveYourself.

Opracowując samojezdny samochód, inżynierowie GoDriveYourself zgromadzili tysiące godzin nagrań wideo oraz wydali miliony dolarów swoich inwestorów na opracowanie systemu wizyjnego opartego na uczeniu maszynowym. Jak każdy inny start-up dotyczący samojezdnych samochodów, inżynierowie wykorzystali tę samą sieć neuronową, najlepiej działający model, w każdym samochodzie, który opuszcza linię produkcyjną.

GoDriveYourself sprzedaje samochód złośliwej osobie, która chce zaatakować ten system, powodując, by nie zatrzymywał się on przed znakiem Stop. W pierwszym etapie atakujący zbiera informacje o sieci neuronowej wykorzystywanej w systemie wizyjnym, dążąc do uzyskania najwyższego poziomu hierarchii informacji z rysunku 3. Następnie wykorzystuje te informacje do zbudowania pseudomodelu, co zostało omówione bardziej szczegółowo w części „Budowanie pseudomodelu". Wykorzystując metody omówione w części „Ataki na model", atakujący

może łatwo przygotować ataki na pseudomodel, które później oszukają również prawdziwy system wizyjny. Ponieważ ta sama sieć neuronowa jest używana w każdym samochodzie, atak, który może wpłynąć na jeden samochód GoDriveYourself, może wpłynąć na każdy inny samochód produkowany przez GoDriveYourself.

Budowanie pseudomodelu

Jak wspomniano w części „Omówienie problemu wysokiego poziomu", do przeprowadzenia skutecznego ataku na model atakujący musi mieć wystarczającą ilość informacji o tym modelu, tak aby utworzyć przybliżoną kopię, którą nazywamy *pseudomodelem*. Po skonstruowaniu dobrego pseudomodelu atakujący może generować kontradyktoryjne przykłady, które prawdopodobnie oszukają także pierwotny model. Pomimo że oryginalny model i pseudomodel różnią się od siebie, to – jak pokazali Papernot i inni – udany atak przeciwnika na pseudomodel ma dużą szansę na oszukanie oryginalnego modelu [3]. Co ciekawe, architektura pseudomodelu nie musi być dokładną repliką architektury modelu, by atak się powiódł. Papernot i inni pokazali, że ataki kontradyktoryjne mogą przenosić się pomiędzy modelami uczenia maszynowego, często nawet pomiędzy modelami, które działają na zupełnie innych zasadach, jak na przykład drzewa decyzyjne i sieci neuronowe [4]. Oznacza to, że atakujący może ułatwić sobie zadanie, ucząc pseudomodel, który jest znacznie mniej wymagający obliczeniowo, w celu stworzenia skutecznych kontradyktoryjnych przykładów.

Najprostsza forma ataku mieści się w najwyższej kategorii hierarchii informacji z rysunku 3, gdy atakujący ma bezpośredni dostęp do projektu modelu, danych uczących i parametrów. W przypadku rozpatrywanej firmy zajmującej się samojezdnymi samochodami osobowymi oznaczałoby to, że przeciwnik dowiedział się, jak zaprojektowaliśmy system wizyjny, uzyskał dostęp do danych dotyczących jazdy, które zostały wykorzystane do uczenia modelu, a także uzyskał dostęp do stanu wewnętrznego końcowego wyszkolonego systemu wizyjnego. Dzięki tym informacjom tworzenie pseudomodelu nie wymaga praktycznie żadnej pracy. Atakujący po prostu buduje model, składając architekturę i ucząc model, aby był zbieżny, lub w skrajnym przypadku ustawiając parametry bezpośrednio na końcowe znane wartości.

Powiedzmy, że firma GoDriveYourself sama przyznała, że tego rodzaju atak może się zdarzyć, dlatego też zaszyfrowali szczegóły sieci i ukryli dane uczące, w wyniku czego system wizyjny działa jak czarna skrzynka. System nadal jednak działa, identyfikuje znaki Stop, znaczniki drogowe i tym podobne. Jednak atakujący nie ma już dostępu do struktury modelu ani danych uczących, a zatem dokładne skopiowanie modelu lub wykorzystanie danych uczących do stworzenia pseudomodelu jest niemożliwe.

Może się wydawać, że inżynierowie zatroszczyli się o wszystkie możliwe rodzaje ataku, jakie mogą się wydarzyć. Czy jednak przeciwnik mógłby wykorzystać przewidującą zdolność modelu na swoją korzyść? Papernot i inni zademonstrowali, w jaki sposób przeciwnik może użyć modelu do zbudowania zestawu danych, stosując go do klasyfikowania lub regresji licznych danych wejściowych, które można następnie wykorzystać do uczenia pseudomodelu [3]. W takim przypadku atakujący traktuje model jako „wyrocznię", po to, by uzyskać pewne pojęcie na temat jego działania.

Idealnie byłoby, gdyby firma GoDriveYourself chciała ukryć te przewidywania przed atakującym, tak aby był on ograniczony do najniższego poziomu hierarchii informacji opisanej w części „Hierarchia informacji". Na tym poziomie jedyne informacje, do których atakujący ma dostęp, to próbki danych wejściowych i wyjściowych wybrane przez twórców modelu i wcześniej zapisane. Pomimo że model jest znacznie bezpieczniejszy niż wcześniej, to jeśli zestaw par wejściowych i wyjściowych dostępnych dla przeciwnika jest wystarczająco duży, można go skutecznie wykorzystać do uczenia pseudomodelu. Taki poziom bezpieczeństwa jest dość trudny do osiągnięcia, szczególnie w przypadku już wdrożonych ustawień. Na przykład GoDriveYourself może nie być w stanie osiągnąć tego poziomu, ponieważ wdrożony system wizyjny w samochodzie cały czas musi przetwarzać nowe zdjęcia w celu bezpiecznej autonomicznej jazdy po nieznanych drogach.

Ataki na model

Gdy atakujący zbuduje już pseudomodel, musi stworzyć kontradyktoryjne przykłady potrzebne do przeprowadzenia swojego ataku. Poniżej opisano metodę generowania przykładów kontradyktoryjnych redukcji ufności i błędnej klasyfikacji źródłowej: metodę znaku szybkiego gradientu (FGSM, ang. *Fast Gradient Sign Method*).

Metoda znaku szybkiego gradientu

W metodzie znaku szybkiego gradientu przykłady kontradyktoryjne są tworzone przez znajdowanie kierunku maksymalnej dodatniej zmiany strat przez określanie gradientu obrazu w stosunku do funkcji strat oryginalnego modelu. Następnie obraz wejściowy jest zaburzany wzdłuż tego kierunku, tworząc przykład kontradyktoryjny, który co najmniej zmniejsza zaufanie modelu do jego klasyfikacji, a w najlepszym wypadku powoduje, że model nieprawidłowo sklasyfikuje przykład.

Dla bardziej zaawansowanych matematycznie Czytelników poniżej przedstawiono formalne równanie ataku bazujące na metodzie znaku szybkiego gradientu:

$$\tilde{x} = x + \varepsilon \cdot \text{sign}\left(\nabla_x J\left(\theta, x, y\right)\right)$$

Atakujący określa kierunek największego zakłócenia wejścia, obliczając jakobian funkcji strat J w odniesieniu do parametrów modelu θ, wejścia x i prawdziwej etykiety y. Atakujący standaryzuje perturbację przez zastosowanie funkcji znaku[2], a następnie dodaje ten znormalizowany wektor przeskalowany o $\varepsilon > 0$ do przykładu wejściowego x, tworząc przykład kontradyktoryjny $\tilde{x}$. Należy zauważyć, że ta technika działa tylko na modelach, które są rozróżnialne, a zatem nie będzie w stanie atakować modeli takich, jak drzewa decyzyjne [5].

Przeciwnik zazwyczaj wybiera wartość ε na tyle małą, że zaburzenie jest niewidoczne dla ludzi, a jednocześnie jest w stanie oszukać model.

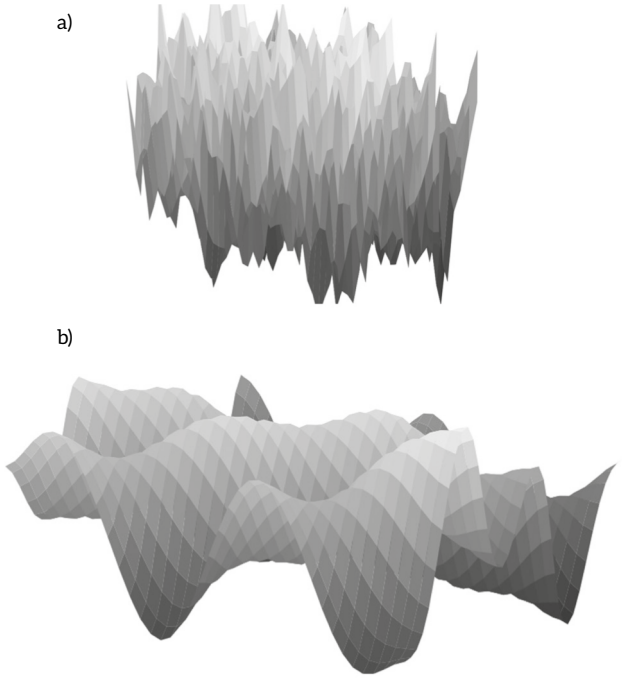
Przeciwnik może również zmodyfikować ten atak, aby dokonać błędnej klasyfikacji źródłowej:

$$\tilde{x} = x - \varepsilon \cdot \text{sign}\,(\nabla_x J\,(\theta, x, y))$$

Zamiast zwiększania gradientu funkcji straty, zwiększania wartości strat i tym samym zmniejszania zaufania do modelu, atakujący obniża gradient względem docelowego $\tilde{y}$ zamiast prawdziwej etykiety y. Zapewnia to, że model niepoprawnie klasyfikuje $\tilde{x}$ jako jako $\tilde{y}$ oraz ma duże zaufanie do prognozy $\tilde{y}$.

W obu wersjach, jeśli pojedyncza perturbacja nie jest wystarczająca do spowodowania błędnej klasyfikacji, atakujący może powtórzyć proces wprowadzania zaburzeń metodą znaku szybkiego gradientu, zastępując x w obliczeniu gradientu zaktualizowanym $\tilde{x}$ po każdej iteracji [6].

Pomimo że istnieją inne rodzaje ataków, to są one głównie rozszerzeniami algorytmu metody znaku szybkiego gradientu. Na przykład w ataku opartym na jakobianach Saliency Map Attack (JSMA) aktualizowana jest jedna funkcja naraz, zamiast zwiększać gradient dla każdej pojedynczej cechy naraz. JSMA zapewnia precyzyjniejszą kontrolę przez aktualizację cech jedna po drugiej, a zatem może generować bardziej przekonujące przykłady kontradyktoryjne ze zmianami, które są mniej widoczne dla ludzi. Jednak ta dodatkowa precyzja związana jest ze zwiększonym kosztem obliczeniowym [2].



Rys. 4. Wizualizacja powierzchni modelu błędu przed i po uczeniu kontradyktoryjnym. Osie *x* i *y* reprezentują dwie cechy wejścia, a oś z powierzchnię błędu, znaną również jako powierzchnia strat. Problemem jest klasyfikacja binarna: model z wysokim błędem jest uważany za część klasy A, a model z niskim błędem uważany za część klasy B. Na górze: powierzchnia błędu nauczonego modelu docelowego. Powierzchnia jest bardzo postrzępiona, z dużymi gradientami w wielu punktach, co ułatwia ataki oparte na gradiencie. Na dole: powierzchnia błędu modelu docelowego po uczeniu kontradyktoryjnym. Powierzchnia jest znacznie gładsza, co zmniejsza wpływ niewielkich zaburzeń na wydajność modelu. (a) Błąd powierzchni przed uczeniem kontradyktoryjnym. (b) Powierzchnia błędu w uczeniu kontradyktoryjnym

Obrona przed atakami kontradyktoryjnymi

Obrona w fazie uczenia

Najprostszą obroną przed atakami kontradyktoryjnymi jest zbudowanie silniejszego modelu podczas fazy uczenia przez rozszerzenie zestawu danych uczących albo przez zmianę schematu uczenia w celu włączenia wiedzy o atakach kontradyktoryjnych.

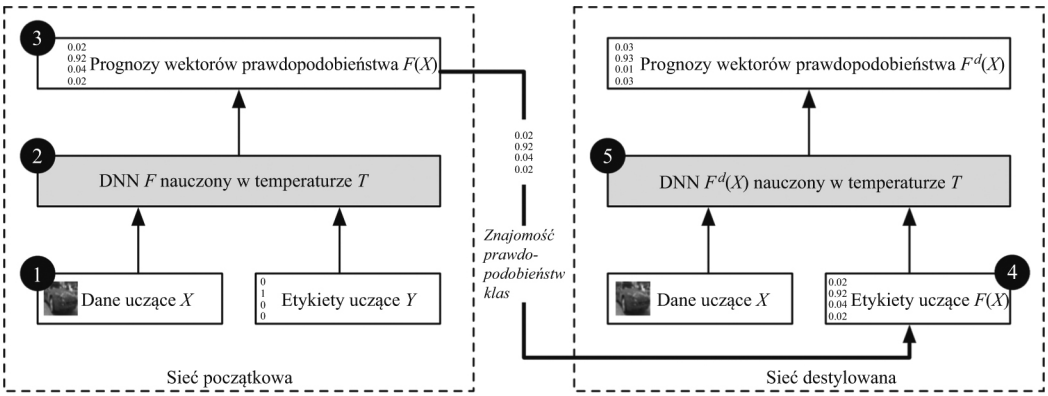
Uczenie kontradyktoryjne

Jednym ze sposobów zmniejszenia skuteczności ataków kontradyktoryjnych jest dodanie kontradyktoryjnych przykładów do zbioru danych z ich prawidłowymi etykietami. Proces ten nazywany jest *uczeniem kontradyktoryjnym*. Ucząc się poprawnie klasyfikować takie kontradyktoryjne przykłady, modele mogą lepiej zrozumieć prawdziwy rozkład danych. Po pierwsze, proces uczenia kanonicznego przebiega przez wiele iteracji. Potem za pomocą FGSM lub innej metody tworzone są przykłady kontradyktoryjne i dodawane do zestawu danych. Następnie proces zaczyna się od nowa. Przy tradycyjnych ustawieniach bezpieczeństwa uczenie kontradyktoryjne można interpretować jako rodzaj testu penetracyjnego, który pomaga ujawnić luki w zabezpieczeniach w modelu, które są następnie uzupełniane przez proces szkolenia.

Możemy lepiej zrozumieć taką obronę, wprowadzając koncepcję powierzchni błędów. Powierzchnie błędów opisują związek pomiędzy dziedziną możliwych danych wejściowych do modelu a wartością funkcji kosztu modelu. Stromy gradient na powierzchni błędu wskazuje, że niewielka zmiana na wejściu może drastycznie zwiększyć wartość funkcji kosztu modelu, ułatwiając tym samym atakującemu znalezienie kontradyktoryjnego przykładu przez nieznaczną zmianę istniejącego wejścia. Model nauczony bez narażania go na przykłady kontradyktoryjne może mieć wiele takich punktów. Powierzchnia błędu takiego modelu byłaby poszarpana, przypominając wystające z ziemi noże, jak pokazano to na rysunku 4a. W takim przypadku atakujący może łatwo znaleźć przykłady kontradyktoryjne, wykorzystując atak taki, jak FGSM.

Uczenie kontradyktoryjne wygładza niedoskonałości powierzchni błędu, jak pokazano na rysunku 4b. Atakującemu trudniej jest wtedy znaleźć zaburzenie, które jest wystarczająco małe, aby było niezauważalne przez ludzi, ale wystarczająco istotne, aby właściwie zaatakować klasyfikator.

Można łatwo sobie wyobrazić, że gdyby obrońca włączył wszystkie możliwe przykłady kontradyktoryjne do zestawu treningowego podczas szkolenia modelu, to model byłby całkowicie odporny, ponieważ nigdy nie zaskoczyłby go nieznany przykład kontradyktoryjny. Jednak przestrzeń wszystkich możliwych przykładów jest bardzo duża i generowanie każdego przykładu jest trudne obliczeniowo, tak więc obrońcy mogą mieć tylko nadzieję, że wybrane z przestrzeni próbki obrony przed *brute force* są wystarczająco dobre. Korzystanie



Rys. 5. Schemat destylacji defensywnej wykorzystującej dwa modele. Pierwsza sieć generuje bardzo gładkie wartości prawdopodobieństwa, które mają dobre właściwości obronne (jej powierzchnia strat jest naturalnie mniej poszarpana). Druga sieć wykorzystuje te prawdopodobieństwa do generowania wysoce niejednorodnych prawdopodobieństw, które są stosowane do pewnych prognoz klas

z bardzo szybkiej techniki, takiej jak FGSM, jako część reżimu uczenia kontradyktoryjnego nadal zapewnia, że obrońca może skutecznie wygenerować wystarczająco duży przykładowy zestaw kontradyktoryjny i użyć go do wygładzenia dużej części powierzchni błędu modelu.

Defensywna destylacja

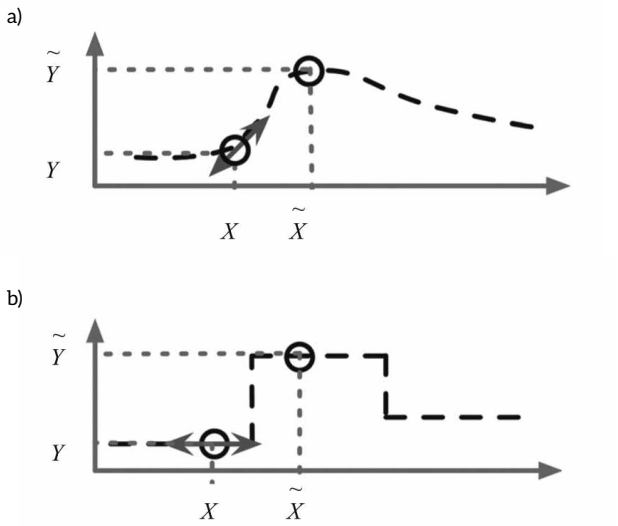
Inną metodą obrony, która została zaproponowana przez Papernota i innych [7], jest tak zwana *destylacja obronna*. Metoda ta oparta jest na technice transferu wiedzy opracowanej przez Hintona i innych [8]. Początkowo metoda wykorzystywała wiedzę na temat skomplikowanych sieci neuronowych lub zespołów sieci neuronowych do uczenia prostych sieci neuronowych, które były w stanie działać w przybliżeniu tak dobrze, jak sieci źródłowe.

Zwykle model tworzy odwzorowanie od przykładów wejściowych do wysoce pewnych prognoz. Na przykład model jest bardziej skłonny do klasyfikowania obrazu psa, który zawiera 98% rozkładu psa i 2% kota (wysoce niejednakowe prawdopodobieństwo) zamiast 54% rozkładu psa i 46% kota (bardzo jednolite prawdopodobieństwo). Niejednorodny wynik prawdopodobieństwa umożliwia dokonanie pewnych przewidywań, natomiast jednolity wynik prawdopodobieństwa jest bardziej przydatny do celów klasyfikacji, ponieważ istnieje mniejsza tendencja do wybierania pewnej klasy zamiast innej. Destylacja zachęca sieć neuronową do dokonywania bardziej jednolitych prognoz, zmniejszając potencjał wpływu przypadkowego szumu na prognozę sieci. Z perspektywy obrony kontradyktoryjnej oznacza to, że atakujący miałby większe trudności ze znalezieniem takiego szumu, który pozostaje niezauważalny dla ludzi (rysunek 5).

Zasadniczo celem destylacji defensywnej jest uwzględnienie zalet zarówno właściwości defensywnych jednolitych wyników prawdopodobieństwa, jak i korzyści zaufania wynikającego z niejednolitych wyników prawdopodobieństwa. Dokładny formalny opis tego, jak destylacja defensywna wykorzystuje ten dwustopniowy proces, został zawarty w pracy Papernota i innych na temat destylacji defensywnej [7].

Obrona polegająca na blokadzie gradientu

Obiecującym rodzajem obrony jest bezpośrednie ukrywanie sygnałów zwrotnych, na których polegają atakujący przy konstruowaniu ataków kontradyktoryjnych. Jak wyjaśniono w przypadku samochodu samojezdnego, osoby atakujące znajdują sposoby na oszukanie modeli przez nieznaczne zaburzenie danych wejściowych i obserwowanie, w jaki sposób zaburzenie to wpływa na zaufanie modelu do klasyfikacji danych wejściowych. Zaburzenie najlepiej przeprowadzić, postępując zgodnie z kierunkiem gradientu. Z tego powodu, ukrywając informacje o gradiencie przed atakującym, obrońca może skutecznie zapobiec takim atakom. Proces ten nosi nazwę maskowanie gradientu (rysunek 6).



Rys. 6. Schemat maskowania gradientu na podstawie [9].
(a) Model docelowy ma ciągłą powierzchnię z niezerowym gradientem. Przeciwnik może wykorzystać to do ataku, zakłócając X wraz ze wzrostem gradientu i tworząc przykład kontradyktoryjny $\tilde{X}$, powodując, że klasyfikator oznaczy go jako $\tilde{Y}$.
(b) Ten sam model, ale z maskowaniem gradientu. Przeciwnik nie może już używać gradientu do tworzenia przykładu kontradyktoryjnego $\tilde{X}$

Prosta forma maskowania gradientu polega na ukryciu rozkładu prawdopodobieństwa modelu klasyfikacyjnego i udostępnieniu tylko najbardziej prawdopodobnego wyniku. Rozważmy sieć neuronową, której zadaniem jest rozróżnianie psów i kotów.

Taki konkretny model zwraca prawdopodobieństwa jako dane wyjściowe. W przypadku pewnego określonego obrazu psa wyjściem może być 70% prawdopodobieństwo, że jest to pies i 30%, że jest to kot. Dzięki tym informacjom atakujący może zakłócać obrazy wejściowe i sprawdzać, czy zaufanie do przewidywania „psa" spadnie. Taka zmiana prawdopodobieństwa daje atakującemu użyteczny gradient. Obrona maskująca gradientem polegałaby na zmianie oprogramowania *front-end*, które wyświetla wyniki obliczeń tego klasyfikatora, tak aby wyświetlało wyłącznie etykietę o najwyższym prawdopodobieństwie („pies"), a nie zaufanie klasyfikatora do tej prognozy. Takie maskowanie uniemożliwia atakującemu zrozumienie gradientu optymalnej perturbacji wejściowej, co utrudnia użycie ataków opartych na gradiencie.

Okazuje się, że wyżej wspomniane techniki uczenia kontradyktoryjnego i destylacji obronnej tak naprawdę zawdzięczają swoją skuteczność niezamierzonemu maskowaniu gradientu. Uczenie kontradyktoryjne i destylacja defensywna skutkują gładszymi powierzchniami strat dotyczących przykładów uczących, szczególnie tam, gdzie atakujący może oczekiwać sygnałów gradientu z modelu docelowego.

Wydaje się to świetną perspektywą. Dlaczego zatem potrzebujemy innych środków obrony, jeśli ta prosta metoda jest dostępna? Pomimo że maskowanie gradientu znacznie utrudnia przeprowadzenie ataku, to nie eliminuje potencjalnych atakujących, po prostu ukrywa nieodłączne słabości modelu. Ukrywanie gradientu może utrudniać atakującym tworzenie próbek kontradyktoryjnych. Jednakże nadal istnieją takie przykłady, które są w stanie oszukać sieć i jest całkowicie prawdopodobne, że atakujący nadal może znaleźć takie próbki kontradyktoryjne.

Carlini i inni omówili różne sposoby, w jakie atakujący może to osiągnąć, w tym stosując odpowiednio dobrane funkcje strat, obliczanie gradientów z warstwy pre-softmax i przenoszenie próbek kontradyktoryjnych z pseudomodelu [10]. Podczas gdy wszystkie ataki koncentrują się na przypadkach, w których atakujący ma dostęp do wysokich poziomów hierarchii informacji (rysunek 3), to przykłady kontradyktoryjne łatwo przenoszą się z jednego modelu do drugiego. Oznacza to, że pseudomodel nauczony na wyjściach modelu bronionego przez maskowanie gradientu nadal może generować skuteczne ataki kontradyktoryjne na takie modele.

Obrona polegająca na wykrywaniu próbek kontradyktoryjnych

Obrona polegająca na wykrywaniu prób kontradyktoryjnych polega na aktywnym rozróżnieniu próbek normalnych i kontradyktoryjnych podczas *wnioskowania*, na którym to etapie nauczony model aktywnie klasyfikuje nowe dane wejściowe. Obronę polegającą na wykrywaniu prób kontradyktoryjnych można uznać za filtr dostrojony do wykrywania przykładów kontradyktoryjnych. Zarówno sam model, jak i osobny model mogą próbować zidentyfikować i odfiltrować szkodliwe dane wejściowe przed ich przetworzeniem.

Detektory

Jedna z metod wykorzystuje *detektor*, który identyfikuje szkodliwe dane wejściowe. Pomimo że zaproponowano wiele schematów detektorów, to w niniejszej pracy skupiono się na *ściskaniu cech* Xu i innych [11].

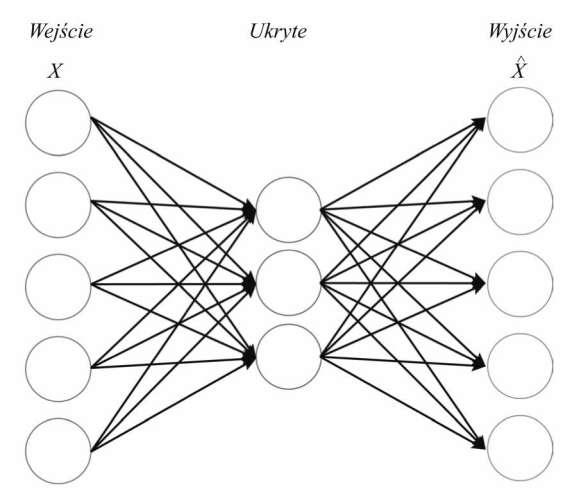
W celu oddzielenia próbek kontradyktoryjnych od normalnych ściskanie cech zwiększa odporność modelu. Model jest odporny, jeśli potrafi poprawnie sklasyfikować przykład nawet po niewielkiej transformacji. Na przykład model rozpoznawania obrazu powinien być w stanie rozpoznać obraz nawet po jego obróceniu lub skalowaniu. Detektor wykorzystuje takie informacje do porównania prognoz modelu przed i po transformacji danych wejściowych. Chodzi o to, że jeśli spreparowany kontradyktoryjnie przykład zostanie błędnie zaklasyfikowany jako kot, to nawet po przekształceniu takiego kontradyktoryjnego obrazu zostanie on z powodzeniem sklasyfikowany jako kot. W strategii ściskania cech dokonywana jest transformacja zmniejszająca głębię bitową koloru każdego piksela i wygładzająca obraz przy użyciu filtra przestrzennego. Matematyczne zmniejszenie głębi bitowej kolorów pikseli na obrazie oznacza zmniejszenie ziarnistości bitów reprezentujących intensywność każdego kanału koloru (dla obrazów zwykle RGB). Podczas gdy 8 bitów jest zwykle używanych do przechowywania intensywności zawartych w przedziale [0, 1] dla każdego kanału koloru, ściskanie cech sugeruje zmniejszenie głębokości bitowej nawet o 1 bit na kanał. Intuicyjnie, w wyniku takiej transformacji otrzymywany jest obraz, na którym mogą być zaakcentowane niewielkie zakłócenia w kanałach kolorów na wejściu, takie same jak zaburzenia wywołane przez atakującego do przeprowadzenia ataku. Inną cechą transformacji ściskania cech jest filtr wygładzania przestrzennego, zwany potocznie rozmyciem. Dla każdego piksela obrazu filtr używa ruchomego okna do uśrednienia intensywności sąsiadujących pikseli. Taka transformacja ma na celu „cofnięcie" rzadkich zaburzeń w poszczególnych pikselach.

Motywacją tych dwóch transformacji ściskania jest „cofnięcie" wszelkich kontradyktoryjnych perturbacji na poziomie pikseli, które są niewidoczne dla ludzi.

Jednak, jak pokazał to He i inni, jeśli osoba atakująca wie, jakiego rodzaju detektor jest używany, może często oszukać detektor w ramach procesu generowania próbki kontradyktoryjnej [12]. Taki rodzaj ataku, który aktywnie omija czujniki obronne, nazywa się atakiem aktywnym. Aktywni atakujący są w stanie wygenerować solidne próbki kontradyktoryjne, które konsekwentnie oszukują sieci docelowe nawet po wielu transformacjach, takich jak wydrukowanie obrazu kontradyktoryjnego, a następnie zrobienie zdjęcia wydruku.

Reformatorzy

Meng i inni w najnowszych pracach wprowadzili pojęcie *reformatora* jako metodę obrony kontradyktoryjnej [13]. Reformator znajduje na podstawie przykładu najbliższy pasujący przykład w zbiorze danych uczących, a następnie modyfikuje dane wejściowe, przybliżając je do tego innego przykładu. Specjalna sieć neuronowa o nazwie *autoencoder* okazała się szczególnie skuteczna w przeprowadzaniu takiej modyfikacji. Autoencoder próbuje nauczyć się mapowania od wejścia



Rys. 7. Prosta sieć neuronowa autoencodera z 5-wymiarowymi danymi i 3-wymiarowym wąskim gardłem. Autoencoder uczy się funkcji tożsamości z takim ograniczeniem, że informacja musi przejść przez wąskie gardło, ukrytą warstwę pośrodku schematu sieci. Nauczenie autoencodera skutkuje nauczeniem wąskiego gardła dobrej reprezentacji danych, która jest podstawą modelu reformatora autoencodera

x z powrotem do siebie, zastępując ponownie etykietę przykładem, tak aby $y = x$ (rysunek 7). Naiwnie zaprojektowany autoencoder nauczyłby się funkcji tożsamości. Jednak tworząc wąskie gardło w architekturze, model uczy się optymalnego sposobu kompresji informacji do uproszczonej reprezentacji. Uproszczona reprezentacja ułatwia reformatorowi „naprawienie" kontradyktoryjnego przykładu, ponieważ z mniejszą liczbą cech związana jest mniejsza liczba możliwych słabych punktów [14].

Reformator autoencodera jest uczony na niekontradyktoryjnych przykładach z zamiarem poznania powierzchni reprezentującej kluczowe aspekty normalnych próbek. Następnie, gdy reformatorowi przedstawi się kontradyktoryjny przykład x, wówczas reformator spróbuje znaleźć normalny przykład y na wyuczonej powierzchni wąskiego gardła, który będzie zbliżony do x. W kolejnym kroku reformator spróbuje „cofnąć" kontradyktoryjne perturbacje, których atakujący używa do przekształcenia pierwotnego wejścia. Tak poprawiony normalny przykład y zostanie sklasyfikowany przez model docelowy jako prawdziwa etykieta dla x.

Jednak reformatorzy są nadal podatni na ataki, podobnie jak detektory. Wiedząc, że obrońca korzysta z reformatora, atakujący może spróbować zaatakować go bezpośrednio, generując przykłady kontradyktoryjne, które oszukają zarówno reformatora, jak i model docelowy.

Zakres problemów

Omówione zostały kontradyktoryjne ataki uczenia maszynowego oraz zostało wyjaśnione, jak są przeprowadzane i jak się przed nimi bronić. Podczas gdy badania akademickie w zakresie kontradyktoryjnego uczenia maszynowego koncentrowały się na atakowaniu modeli klasyfikacji obrazów, przewidujemy szybki rozwój metod kontradyktoryjnych we wszystkich dziedzinach, w których modele uczenia maszynowego mogą być przydatne. Poniżej przedstawiono kilka dziedzin, które będą szczególnie podatne na ataki kontradyktoryjne w przyszłości.

- *Rozpoznawanie twarzy.* Atakujący może podszyć się pod kogoś na zasadzie ataku związanego z ukierunkowaną błędną klasyfikacją. Atakujący musiałby po prostu wygenerować kontradyktoryjny obraz, który oszuka model rozpoznawania twarzy, jednocześnie przekonując model, że obraz ten jest kimś, kto ma dostęp do informacji zabezpieczanych przez system. Systemy rozpoznawania twarzy, które opierają się wyłącznie na systemach wizyjnych opartych na uczeniu maszynowym, nie są odporne na ataki kontradyktoryjne i większość planów ataków opisanych w części „Zakres problemów" może przeniknąć do takiego systemu.
- *Antywirus.* Atakujący może wykorzystać atak związany z błędną klasyfikacją źródłową na system antywirusowy, aby stworzyć wirusa „niewidocznego", przynajmniej dla programu antywirusowego. Atakujący może zbudować wirusa, którego funkcjonalności mogą ukryć zamiary programu przed antywirusowym modelem uczenia maszynowego [15]. Haker może utworzyć trojana z określonym umieszczeniem komentarzy lub inną metodą zmiany kodu, która zamaskuje złośliwą naturę wirusa. Podczas gdy scenariusz ten jest przedmiotem współczesnych badań nad bezpieczeństwem, ataki ewoluują bardzo często, a twórcy programów antywirusowych muszą wiedzieć, jak bronić swoje modele uczenia maszynowego przed sprytnie spreparowanymi atakami kontradyktoryjnymi.
- *Google*. Algorytm wyszukiwania Google jest głównym celem dla przykładów kontradyktoryjnych. Wykorzystując zagrożenia związane z błędną klasyfikacją źródłową, atakujący może zmienić ranking i reputację jakiejkolwiek konkretnej strony internetowej, inteligentnie zaburzając jej zawartość i historię. Należy rozważyć przypadek, w którym właściciel witryny stara się poprawić reputację swojej witryny, tak aby zwiększyć liczbę wyświetleń strony, a tym samym przychody z reklam. Taka osoba może nauczyć się optymalnych „perturbacji", które zastosowane na własnej stronie internetowej sprawią, że strona internetowa stanie się przykładem kontradyktoryjnym i pomimo braku odpowiedniej treści często będzie się pojawiać w najlepszych wynikach wyszukiwania. Dałoby to stronie internetowej znacznie większy ruch i nakłoniłoby internautów do generowania przychodów z reklam dla tej osoby. Ponadto ze względu na kluczową rolę Google w społeczeństwie jako silnika wiedzy, tego typu ataki kontradyktoryjne są bardzo alarmujące ze względu na potencjalny ogromny zasięg takiego ataku. Na przykład fałszywe źródła informacji mogą przekierowywać prawidłowe wyszukiwania faktów do stronniczych lub sfabrykowanych wiadomości politycznych. W świecie, który już cierpi z powodu cenzury informacji stworzonej przez człowieka, możliwość wykorzystania sztucznej inteligencji do optymalnego „prowadzenia" dogmatu i zrozumienia mas jest bardzo przerażającą perspektywą.

Trendy, spostrzeżenia i najnowsze osiągnięcia

Niektórzy sceptycy przykładów kontradyktoryjnych twierdzą, że ataki na modele uczenia maszynowego są często

niewykonalne lub niepraktyczne. Jest to błędne założenie, ponieważ zmotywowani atakujący mogą zaakceptować żmudność ataku, jeśli będą mieć wyraźny cel. Powinno być również jasne, że atak może zostać skutecznie wdrożony po kilku dniach lub tygodniach przygotowań (zwykle podczas uczenia pseudomodelu) zamiast po tysiącleciu niezbędnym do złamania bezpieczeństwa kryptograficznego. Zabezpieczenie przed kontradyktoryjnym uczeniem maszynowym nie powinno w pełni polegać na założeniu, że osoba atakująca nie poświęci wystarczającej ilości wysiłku, aby złamać system.

Oprócz wszelkich ataków na sam model, atakujący może skupić się na danych wykorzystywanych do uczenia modelu. Jest to proces znany jako *zatrucie danych*. W takim przypadku atakujący przejmuje kontrolę nad procesem uczenia przez podstawianie starannie spreparowanych próbek do zestawu danych uczących. Próbki takie są zaprojektowane w celu sztucznego wypaczenia rozkładu danych uczących, a tym samym obniżenia wydajności modelu. Ataki zatrucia danych są skuteczne przeciwko modelom, które są często ponownie uczone przy użyciu danych rzeczywistych. Biorąc pod uwagę, że dane zebrane w celu uczenia jednego modelu będą także udostępniane wszystkim jego odpowiednikom, stosunkowo łatwo jest zatruć setki, jeśli nie tysiące, produktów, takich jak samochody i kamery bezpieczeństwa rozpoznające twarz, infekując dane zgromadzone przez tylko pojedynczy produkt.

Na przykład złośliwe kampanie e-mailowe stale ewoluują tak, aby ominąć filtry antyspamowe. Systemy antyspamowe muszą zbierać dane z ostatnich kampanii spamowych, tak aby ciągle mogły pozostawać aktualne. Ręczne etykietowanie tych danych jest niemożliwe, dlatego system polega na automatycznym gromadzeniu danych uczących. Na przykład system może przechowywać listę adresów e-mail znanych z regularnego wysyłania spamu i automatycznie oznaczać wiadomości e-mail wysyłane z adresu znajdującego się na czarnej liście jako „spam" w danych uczących. Grupa kontrolująca te adresy mailowe może mieć świadomość, że zostały one oznaczone jako spam i zacząć wysyłać wiadomości e-mail, które nie będą charakterystyczne dla wiadomości spamowych, a system nauczy się identyfikować je jako „podobne do spamu". Jeśli system automatycznego etykietowania błędnie oznaczy znaczną liczbę takich wiadomości e-mail, model wykrywania spamu nauczy się niewłaściwego rozkładu klas wiadomości e-mail, identyfikując spam jako zgodny z prawem, a następnie klasyfikując zwykłe wiadomości e-mail jako spam.

Jedną potencjalną obroną przed zatruciami danych jest dokładne odkażenie danych uczących. W praktyce sprowadza się to do wykrywania i usuwania podejrzanych danych i wartości odstających oraz korzystanie wyłącznie z zaufanych repozytoriów danych. Pomimo że duże ilości danych są często niezbędne do stworzenia dobrych modeli uczenia maszynowego, to niewiarygodne lub niezweryfikowane repozytoria danych mogą być podobnie szkodliwe dla modeli uczenia maszynowego. Inną metodą odkażania danych jest zastosowanie analizy „odrzucenie po negatywnym wpływie" do nowych danych. W takiej konfiguracji system porównuje wydajność modelu przed i po uczeniu na podejrzanych próbkach. Jeśli nowe dane znacznie zmniejszają dokładność testu na sprawdzonych przykładach, to system odrzuca nowe próbki szkoleniowe.

Wraz z rozwojem dyscypliny wydaje się, że atakujący przezwyciężają obronę kontradyktoryjną, jak tylko zostanie ona opracowana. Istnieje jednak kilka obiecujących kierunków badań. Pierwszym z nich jest zestawianie modeli: łączenie informacji z wielu modeli „podstawowych" w celu uzyskania pojedynczej klasyfikacji. Składanie ma tendencję do podkreślania mocnych stron modeli podstawowych, gdy działają one dobrze, i maskowania słabości modeli podstawowych, gdy działają słabo. Jeden z wariantów takiego podejścia wydaje się zmniejszać podatność kontradyktoryjnie uczonych modeli na ataki typu biała skrzynka [16]. Korzystając z wielu źródeł przed podjęciem decyzji, metoda ta wygładza powierzchnię błędu, czyniąc model odpornym na ataki kontradyktoryjne, tak jak omówiono to w części „Ataki na model".

Możemy również zaobserwować lepszą odporność na przykłady kontradyktoryjne, w sytuacjach, gdy duże grupy badawcze, takie jak Google i Facebook, uczą modele, wykorzystując gigantyczne zestawy danych. Obejmując większy zakres możliwych danych wejściowych, modele uczone na dużych zestawach danych mogą uczyć się bardziej wygładzonych powierzchni błędów, a tym samym większej odporności na ataki kontradyktoryjne. Jednakże rozwiązuje to tylko tę część problemów, dla których dostępne są duże ilości danych.

Wnioski

Ataki i mechanizmy obronne nieustannie ewoluują wraz ze sztuczną inteligencją. Obecnie wydaje się, że bardzo trudno obronić się przed większością ataków. Jednak niektóre mechanizmy obronne są obiecujące w przypadkach, kiedy atakujący ma dostęp tylko do ograniczonych informacji o modelu. Rozważając ten wyścig zbrojeń w nauczaniu kontradyktoryjnym, wydaje się, że obecnie przewaga leży po stronie atakującego. Aby być naprawdę odpornym na każdy atak, model musiałby nauczyć się i zachować wszelką możliwą wiedzę oraz informacje na temat problemów z procesem uczenia, niezależnie od tego, czy obejmowałoby to proaktywne i niepraktyczne uczenie na wszystkich możliwych próbach uczących, nieskończoną przestrzeń, uczenie reaktywne lub cokolwiek związane z wykrywaniem i odkażaniem wszelkich ataków kontradyktoryjnych.

Jedno jest pewne: zanim zaczniemy polegać na modelach uczenia maszynowego i integrować je z systemami bezpieczeństwa, takimi jak samochody samojezdne, konieczne jest przeprowadzenie większej liczby badań, tak aby zapewnić integralność tych systemów. Opracowanie solidnej obrony ma kluczowe znaczenie, ponieważ niebezpieczne aplikacje mogą kosztować życie ludzkie.

Przypisy

1. Warto powiedzieć, że pojęcie to pasuje tylko do nadzorowanego paradygmatu modelu uczenia maszynowego. Istnieje również uczenie nienadzorowane oraz nauczanie częściowo nadzorowane. Jednakże dyskusja na temat tych metod została pominięta, ponieważ większość najnowocześniejszych algorytmów w uczeniu maszynowym opiera się na uczeniu nadzorowanym, w wyniku czego

kontradyktoryjne uczenie maszynowe koncentruje się tylko na tym paradygmacie.

2. Funkcja sign po prostu wyodrębnia znak liczby rzeczywistej. Jeśli $x < 0$, sign(x): = −1. Jeśli $x > 0$, sign(x): = 1.

Literatura

[1] D. of Statistics 2016. *NSC motor vehicle fatality estimates.*

[2] Papernot N., McDaniel P., Jha S., Fredrikson M., Celik Z.B., Swami A.: *The limitations of deep learning in adversarial settings.* In 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 372–387. IEEE, Saarbrücken, Germany.

[3] Papernot N., McDaniel P., Goodfellow I., Jha S., Celik Z.B., Swami A.: *Practical blackbox attacks against machine learning.* In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519. ACM, Abu Dhabi, UAE.

[4] Papernot N., McDaniel P., Goodfellow I.: *Transferability in machine learning: from phenomena to black-box attacks using adversarial samples.* arXiv preprint arXiv:1605.07277, 2016.

[5] Goodfellow I.J., Shlens J., Szegedy C.: *Explaining and harnessing adversarial examples.* arXiv preprint arXiv:1412.6572, 2014.

[6] Kurakin A., Goodfellow I., Bengio S.: *Adversarial machine learning at scale.* arXiv preprint arXiv:1611.01236, 2016.

[7] Papernot N., McDaniel P., Wu X., Jha S., Swami A.: *Distillation as a defense to adversarial perturbations against deep neural networks.* In Proceedings of the 37th IEEE Symposium on Security and Privacy, pp. 582–597. IEEE, San Jose, California, 2015.

[8] Hinton G., Vinyals O., Dean J.: *Distilling the knowledge in a neural network.* arXiv preprint arXiv:1503.02531, 2015.

[9] Papernot N., McDaniel P., Sinha A., Wellman M.: *Towards the Science of Security and Privacy in Machine Learning.* ArXiv e-prints, 2016.

[10] Carlini N., Wagner D.A.: *Defensive distillation is not robust to adversarial examples.* CoRR abs/1607.04311. http://arxiv.org/abs/1607.04311, 2016.

[11] Xu W., Evans D., Qi Y.: *Feature squeezing: Detecting adversarial examples in deep neural networks.* arXiv preprint arXiv:1704.01155, 2017.

[12] He W., Wei J., Chen X., Carlini N., Song D.: *Adversarial example defenses: Ensembles of weak defenses are not strong.* arXiv preprint arXiv:1706.04701, 2017.

[13] Meng D., Chen H.: *Magnet: a two-pronged defense against adversarial examples.* arXiv preprint arXiv:1705.09064, 2017.

[14] Baldi P.: *Autoencoders, unsupervised learning, and deep architectures.* In Proceedings of ICML Workshop on Unsupervised and Transfer Learning, Bellevue, Washington, pp. 37–49, 2012.

[15] Grosse K., Papernot N., Manoharan P., Backes M., McDaniel P.: "Adversarial perturbations against deep neural networks for malware classification." arXiv preprint arXiv:1606.04435, 2016.

[16] Tramèr F., Kurakin A., Papernot N., Boneh D., McDaniel P.: *Ensemble adversarial training: Attacks and defenses.* arXiv preprint arXiv:1705.07204, 2017.

Fragment pochodzi z książki:
Sztuczna inteligencja. Bezpieczeństwo i zabezpieczenia,
Roman V. Yampolskiy (redakcja).
Wydawnictwo Naukowe PWN, Warszawa 2020