Breaking NormalHedge — Voorstel voor een Bachelor Project

Peter D. Grünwald
Mathematisch Instituut, Universiteit Leden en CWI, Amsterdam
pdg@cwi.nl

January 17, 2014

Sequentieel Voorspellen

Dit project gaat over sequentieel voorspellen, bijvoorbeeld beurskoersen of een reeks enen en nullen . We hebben een aantal (mogelijk oneindig veel) *experts* die op elk moment t een voorspelling doen voor de volgende uitkomst. De kwaliteit van die voorspelling wordt gemeten aan de hand van een bepaalde *gain* of *loss* functie. In de beurs-applicatie kan deze functie bijvoorbeeld gedefinieerd worden in termen van de winst gemaakt tussen tijdstip t en $t-1$. Maar voor nu houd ik het eenvoudig en kijk naar het simpele geval waarin de te voorspellen data een reeks enen en nullen is, en de loss functie de standaard 0/1-loss: op elk tijdstip voorspel je een 1 of een 0, en je loss is 1 als je voorspelling onjuist is, en anders 0. De totale loss van een expert op tijdstip T is dan de som van de losses $\ell_1, \ldots, \ell_T$ die hij tot dan toe heeft gemaakt, met andere woorden, het aantal fouten dat hij heeft gemaakt. Onze experts kunnen mensen zijn, of statistische modellen, of computer- modellen, die eventueel toegang hebben tot allerlei extra data die wij niet zien. Wij krijgen alleen de voorspellingen van de experts op elk tijdstip te zien, en moeten daaruit zelf een voorspelling destilleren. Ons doel is de voorspellingen op zo'n manier te combineren dat *wat voor data er ook komt, wij op z'n minst (vrijwel) zo goed voorspellen als de expert die het achteraf gezien het beste heeft gedaan voor die data*. Met andere woorden, we willen leren uit het verleden welke expert (of combinatie van experts) het het beste doet, en deze expert/combinatie van experts volgen voor de volgende voorspelling.

Follow the Leader vs. Hedge

Als het aantal experts eindig is, dan is het meest voor de hand liggende algoritme *Follow the Leader* (FTL): kopieer, op elk tijdstip t, de voorspelling van de expert die tot nu toe het minst aantal fouten heeft gemaakt. Dit werkt in de praktijk vaak heel goed, maar in bepaalde 'worst-case' situaties gaat het flink de mist in. Een standaard voorbeeld is de volgende simpele datareeks:

01010101010101...

Er zijn maar twee experts: Expert E_0 voorspelt steeds 0, en Expert E_1 voorspelt steeds 1. Als deze reeks T rondes doorgaat maken beide experts ongeveer $T/2$ fouten, maar FTL maakt ongeveer $(3/4)T$ fouten, dus véél (linear in T) slechter dan beide experts (als je ziet waarom dat zo is, is dit project misschien wel iets voor jou!). We zeggen dat de *regret* van FTL gelijk is aan $T/4$.

Dit slechte gedrag van FTL is een van de redenen dat er de laatste 20 jaar een hele reeks algoritmes ontworpen zijn die het voor elkaar krijgen om *wat voor data je ook krijgt*, een regret van $O(\sqrt{T})$ te halen; dus ze zijn nooit meer dan $O(\sqrt{T})$ slechter dan de expert die het minste fouten maakte tot aan tijdstip T. Dat is redelijk goed, omdat de regret-per outcome (gemiddelde fout) dan naar 0 gaat voor grote T. Het bekendste van deze algoritmes is *Hedge* van Freund en Shapire, dat met een slim gewogen gemiddelde voorspelt, in plaats van al zijn kaarten op de beste tot nu toe te zetten.

Nu is het probleem echter dat Hedge en de vele aanverwante algoritmes ook een regret van orde $\sqrt{T}$ halen op zogenoemde 'makkelijke' data waarbij FTL of andere 'naieve' algoritmes het juist erg goed doen. Een voorbeeld (bij lange na niet het enige) van 'makkelijke' data is een reeksq waarbij een expert E op een gegeven moment t_0 het beste wordt en dat dan voor altijd blijft. Hedge haalt $\sqrt{T}$ op zulke data, maar FTL haalt een constante regret — vanaf t_0 volgt het E en blijft dat voor altijd doen. FTL is hier dus veel beter dan Hedge. De vraag is nu of er algoritmes bestaan die (net als Hedge) nooit regret halen groter dan $\sqrt{T}$, maar tevens (net als FTL) het in specifieke gevallen veel beter doen.

NormalHedge vs. FlipFlop

Het is verbazingwekkend lastig dat soort algoritmes te bouwen, maar in een recent artikel hebben we laten zien dat het door onszelf bedachte *FlipFlop*-voorspelalgoritme dit inderdaad voorelkaar krijgt. We kunnen wiskundig bewijzen dat de regret van FlipFlop altijd kleiner is dan 5 maal de regret van Hedge, en altijd kleiner dan 5 maal de regret van FTL. In dat artikel testen we FlipFlop, Hedge en een hele hoop andere algoritmes op allerlei soorten data, en we vinden dat alle Hedge-achtige algoritmes het in de praktijk ook echt veel slechter doen dan FlipFlop...op één uitzondering na: het zgn. *NormalHedge* algoritme van Freund, Hsu en Chaudhuri. Maar in tegenstelling tot FlipFlop is NormalHedge niet per se gemaakt om goed te werken in 'makkelijke' gevallen, en wij hebben het sterke vermoeden dat er een reeks 1en en 0en te verzinnen moet zijn waarbij FlipFlip het goed doet en NormalHedge slecht — we hebben die alleen nog niet gevonden. De opdracht zou in eerste instantie zijn om zo'n reeks te zoeken, zowel door te kijken naar de wiskunde achter FlipFlop en NormalHedge als door met kandidaat-data te experimenteren — het project vereist dus enig (niet veel, de algoritmes passen op 1/2 A4) programmeer werk. De precieze invulling is flexibel; we kunnen in een later stadium ook naar andere (continue) data, andere vormen van experts en andere algoritmes kijken.

Wellicht is het nog belangrijk te vermelden dat varianten van algoritmes zoals Hedge e.d. ook toepasbaar zijn op zeer grote verzamelingen experts en veel in de praktijk gebruikt worden, bijvoorbeeld voor het voorspellen van electriciteitsconsumptie, vermogen geleverd door windmolenparken e.d.