Logistic Regression

Let's take target function f to be a 0/1 probability so y is a deterministic function of $\mathbb{x}$ and create a data set $\mathcal{D}$ to try out logistic regression.

We'll take d = 2 and let $\mathcal{X}$ = [-1,1]×[-1,1] with uniform probability of picking each $\mathbb{x} \in \mathcal{X}$, then choose a line in the plane as the boundary between $f(\mathbb{x})$ = 1 (where y has to be +1) and $f(\mathbb{x})$ = 0 (where y has to be -1) by taking two random, uniformly distributed points from X and taking the line passing through them as the boundary between y = ±1. Then we'll pick N = 100 training points at random from $\mathcal{X}$, and evaluate the outputs y_n for each of these points $\mathbb{x}_n$.

Then let's run logistic regression with stochastic gradient descent to find g, and estimate E_{out} (the **cross entropy** error) by generating a sufficiently large, separate set of points to evaluate the error. We'll then repeat the experiment for 100 runs with different targets and take the average, initializing the weight vector of Logistic Regression to all zeros in each run. We'll stop the algorithm when $|| \mathbb{w}^{(t-1)} - \mathbb{w}^{(t)} ||$ < 0.01, where $\mathbb{w}^{(t)}$ denotes the weight vector at the end of epoch t. We will use a learning rate of 0.01.

```
(* Clear globals *)
Clear[GenerateX, DoLogisticRegressionExperiment]
```

```
(* Generate our X=[-1,1]×[-1,1] space with d uniformly distributed
   points and a random X-partitioning solution function f. *)

GenerateX[OptionsPattern[]] :=
 Module[{
   D, f, d = 2,
   f1, f2,
   m, a, b, c,
   t1, t2,
   dotTest1, dotTest2, DotTest
  },
  f1 = RandomReal[{-1, 1}, {1, d}][[1]];
  f2 = RandomReal[{-1, 1}, {1, d}][[1]];

  m = (f1[[2]] - f2[[2]]) / (f1[[1]] - f2[[1]]);
  a = -m;
  b = 1;
  c = m f1[[1]] - f1[[2]];
  f = {{c}, {a}, {b}};

  (* f should not dot to zero for our two original points! *)
  t1 = {1, f1[[1]], f1[[2]]};
  t2 = {1, f2[[1]], f2[[2]]};

  DotTest[v_, s_] := If[Abs[v] > 0.00000000001, Throw[s], 0];
  DotTest[t1.f, "t1 dot test failed"];
  DotTest[t2.f, "t2 dot test failed"];
```

```
  𝒟 = RandomReal[{-1, 1}, {OptionValue[𝒟Size], d}];

  {𝒟, f}
 ]
Options[Generate𝒳] = {𝒟Size → 100};

(*
LinearTarget[f_,𝕏_]:=Sign[𝕏.f];
NoFeature[𝕏_]:=𝕏;
RiskNoiseFlip[percent_]:=If[RandomReal[]≤percent/100,-1,1];
*)
DoLogisticRegressionExperiment[𝒳_, OptionsPattern[]] :=
 Module[{
   𝒟, f, 𝕪,
   𝕏, (*𝕏f,𝕏fdag,*)
   w, wPrev, 𝕩, y, t, η,
   sgdSamples,
   Ee, gradE,
   epoch, maxEpoch, tDist,
   tst
  },
  {𝒟, f} = 𝒳;
  (*
  𝕏= Function[x,Prepend[x,1]]/@𝒟;
  𝕏f=OptionValue[DataFeature][𝕏];
  𝕪=RiskNoiseFlip[OptionValue[Noise]]*OptionValue[TargetFunction][f,𝕏];
  *)
  gradE[w_, 𝕩_, y_] := -y 𝕩 / (1 + e^(y (w^T.𝕩))[[1,1]]);
  η = OptionValue[LearningRate];
  maxEpoch = OptionValue[MaxEpoch];
  tDist = OptionValue[TDist];
  w = OptionValue[w0];
  wPrev = w + 2 tDist;

  (* 𝕏 is the 1-prepended version of 𝒟 to be used for regression *)
  (* 𝕪 is the vector of labels to be used for regression *)
  𝕏 = Function[x, Prepend[x, 1]] /@ 𝒟;
  (*𝕪=e^Sign[𝕏.f]/(1+e^Sign[𝕏.f]);*)
  𝕪 = Sign[𝕏.f];
  (*𝕪=Map[(If[#<0,0,#])&,𝕪,{2}];*) (* turn 1/-1 elems to 1/0 *)

  For[epoch = 1, epoch ≤ maxEpoch ⋀ Norm[wPrev - w] ≥ tDist, epoch++, (
    (* generate Stochastic Gradient Descent sample order
     for this entire epoch and keep track of the previous
     w so that we can check progress at the end of the epoch *)
    wPrev = w;
    sgdSamples = Permute[Range[1, Length[𝒟]], RandomPermutation[Length[𝒟]]];
    tst = Length[sgdSamples];
    For[t = 1, t ≤ Length[𝒟], t++, (
      𝕩 = {𝕏[[sgdSamples[[t]]]]}^T;
```

```
      y = 𝕪[[sgdSamples[[t]], 1]];
      w = w - η gradE[w, 𝕩, y];
     )];
   )];
  {w, 𝕏, 𝕪, 𝒟, f, epoch, Norm[wPrev - w]}
 ]
Options[DoLogisticRegressionExperiment] =
  {MaxEpoch → 10 000, TDist → 0.01, LearningRate → 0.01, w0 → {{0}, {0}, {0}}};
```

E_{out} for N = 100

```
(* Clear globals *)
Clear[DoLRs]
Clear[lrsfs, lrsgs, lrs𝒟, (*lrsAvEin,*)lrsRuns, lrsN, lrsEpochs, lrsTDists]
```

```
DoLRs[] :=
 Module[{
   runs = 100, N = 100,
   g, gs,
   y, ys,
   X, Xs,
   f, fs,
   D,
   avEin,
   i, n,
   epoch, epochs,
   tDist, tDists
  },
  fs = {};
  gs = {};
  ys = {};
  Xs = {};
  epochs = {};
  tDists = {};
  For[i = 1, i ≤ runs, i++, (
    {g, X, y, D, f, epoch, tDist} =
     DoLogisticRegressionExperiment[GenerateX[DSize → N]];
    AppendTo[fs, f];
    AppendTo[gs, g];
    AppendTo[ys, y];
    AppendTo[Xs, X];
    AppendTo[epochs, epoch];
    AppendTo[tDists, tDist];
   )];
  (*avEin=
    Mean[MapThread[Function[{x,y,z},ClassificationEin[x,y,z]],{Xs,gs,ys}]];*)
  n = Length[D];
  {fs, gs, D, (*avEin,*)runs, n, epochs, tDists}
 ]

{lrsfs, lrsgs, lrsD, (*lrsAvEin,*)lrsRuns, lrsN, lrsEpochs, lrsTDists} = DoLRs[];
StringForm["Logistic regression runs complete
    (runs=``, N=``, ave. epochs=``, ave. t-distance=``).",
 lrsRuns, lrsN, Mean[lrsEpochs] × 1., Mean[lrsTDists]]
```

```
Logistic regression runs complete (runs=100, N=100,
  ave. epochs=344.35`, ave. t-distance=0.009987253724137581`).
```

```
(* clear globals *)
Clear[Experiment1]
Clear[e1fs, e1gs, e1runs, n, lastEpoch, tDist]
```

```
RegressionEin[𝕏_, w_, 𝕪_, OptionsPattern[]] :=
 Module[{
   N, n
  },
  N = Length[𝕪];
  1/N (∑_{n=1}^{N} Log[1 + ℯ^(-𝕪[[n]] (w^T . {𝕏[[n]]}^T)[[1,1]])])
 ]

DoEoutTest[fs_, 𝕘s_] :=
 Module[{
   runs = 100, N = 100,
   𝕘,
   𝕪, 𝕪s,
   𝕏, 𝕏s,
   f,
   𝒟,
   avEout,
   i,
   epoch, tDist
  },
  𝕪s = {};
  𝕏s = {};
  For[i = 1, i ≤ runs, i++, (
    {𝕘, 𝕏, 𝕪, 𝒟, f, epoch, tDist} =
     DoLogisticRegressionExperiment[{Generate𝒳[𝒟Size → N][[1]], fs[[i]]}];
    AppendTo[𝕪s, 𝕪];
    AppendTo[𝕏s, 𝕏];
   )];
  avEout =
   Mean[MapThread[Function[{x, y, z}, RegressionEin[x, y, z]], {𝕏s, 𝕘s, 𝕪s}]];
  n = Length[𝒟];
  {fs, 𝕘s, avEout, runs, n}
 ]

Experiment1[] := DoEoutTest[lrsfs, lrs𝕘s];
{e1fs, e1𝕘s, e1avEout, e1runs, e1n} = Experiment1[];
StringForm[
 "Logistic regression results\n(runs=``, N=``):\naverage (class.) E_out=``",
 e1runs, e1n, e1avEout × 1.]
```

```
Logistic regression results
(runs=100, N=100):
average (class.) E_out={0.102754}
```