Neural Networks
and
Genetic Algorithms

ddebarr@uw.edu

2017-03-16

Course Outline

1. Introduction to Statistical Learning
2. Linear Regression
3. Classification
4. Resampling Methods
5. Linear Model Selection and Regularization
6. Moving Beyond Linearity
7. Tree-Based Methods
8. Support Vector Machines
9. Unsupervised Learning
10. Neural Networks and Genetic Algorithms

Agenda

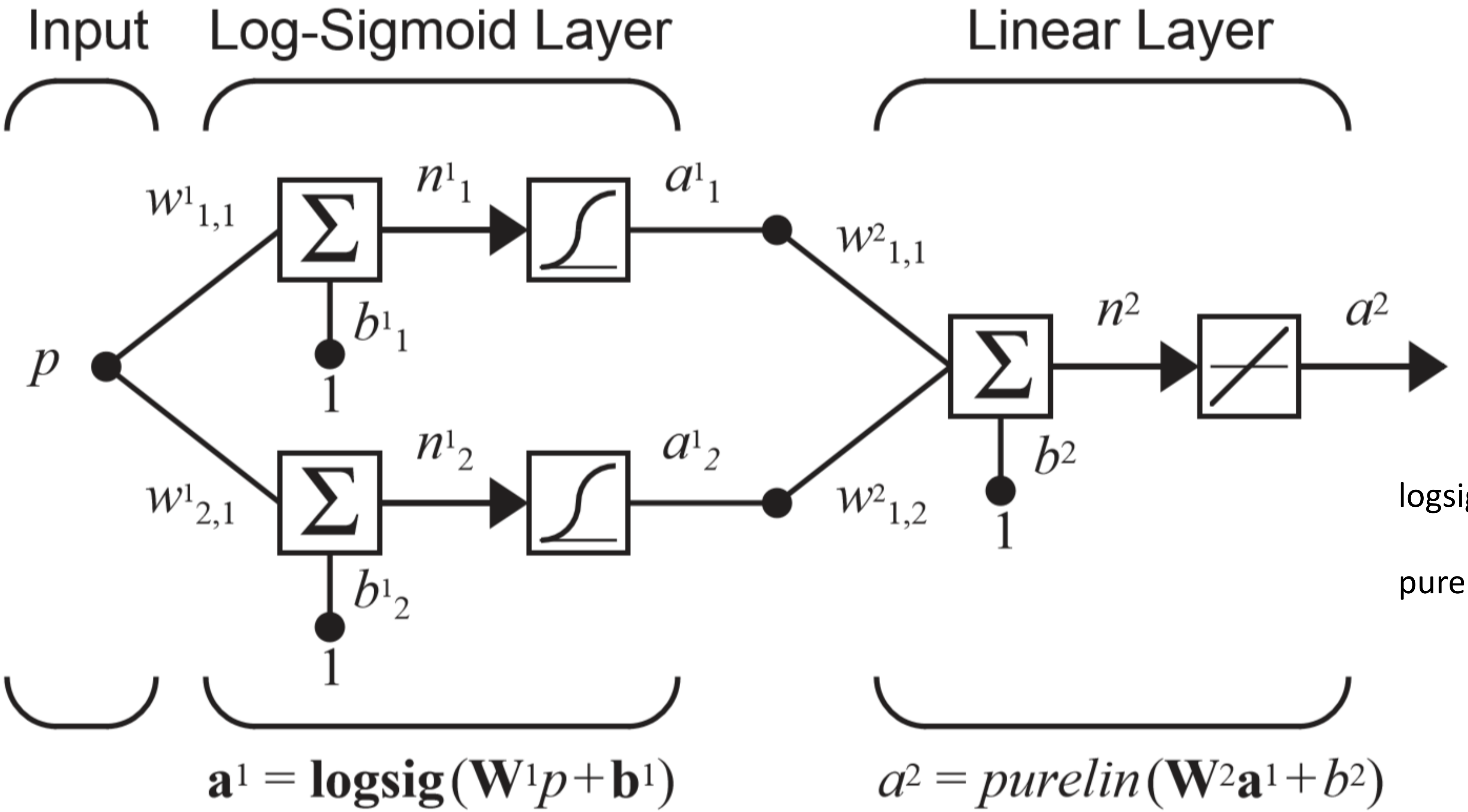
- Machine Learning Tribes
- Neural Networks
- Genetic Algorithms
- Naïve Bayes
- Wrap Up

The Five Tribes of Machine Learning

Tribe	Origins	Master Algorithm
Symbolists	Logic, philosophy	Inverse deduction
Connectionists	Neuroscience	Backpropagation
Evolutionaries	Evolutionary biology	Genetic programming
Bayesians	Statistics	Probabilistic inference
Analogizers	Psychology	Kernel machines

From Pedro Domingos' The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World

Example Function Approximation Network



logsig(x) = 1 / (1 + exp(-x))

purelin(x) = x

Example: Initial Weights and Training Example

Suppose …

- we're trying to learn $f(p) = 1 + \sin(0.25 * \pi * p)$ for $p \in \{-2,2\}$
- we've initialized our weights as follows

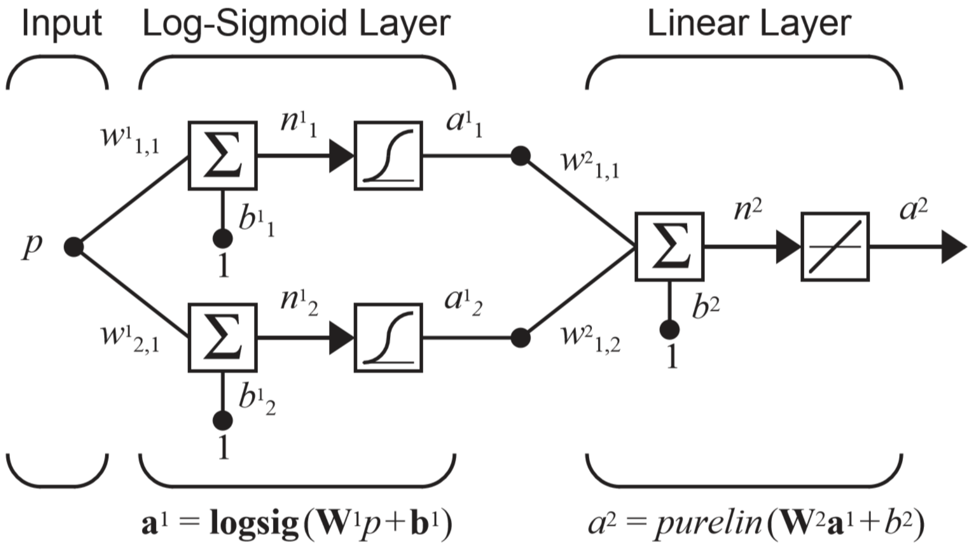
$$\mathbf{W}^1(0) = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix},\ \mathbf{b}^1(0) = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix},\ \mathbf{W}^2(0) = \begin{bmatrix} 0.09 & -0.17 \end{bmatrix},\ \mathbf{b}^2(0) = \begin{bmatrix} 0.48 \end{bmatrix}$$

- we're given the following input

$$a^0 = p = 1$$

- and the following output

f(1) = 1.707

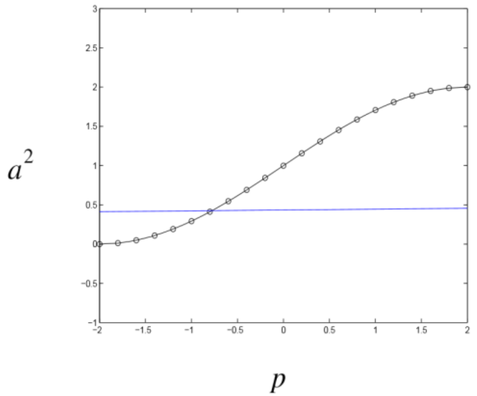


Example: Forward Propagation of Activations

$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{a}^0 + \mathbf{b}^1) = \mathbf{logsig}\left(\begin{bmatrix}-0.27\\-0.41\end{bmatrix}\begin{bmatrix}1\end{bmatrix} + \begin{bmatrix}-0.48\\-0.13\end{bmatrix}\right) = \mathbf{logsig}\left(\begin{bmatrix}-0.75\\-0.54\end{bmatrix}\right)$$

$$= \begin{bmatrix}\dfrac{1}{1+e^{0.75}}\\[2ex]\dfrac{1}{1+e^{0.54}}\end{bmatrix} = \begin{bmatrix}0.321\\0.368\end{bmatrix}$$

$$a^2 = f^2(\mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2) = purelin\left(\begin{bmatrix}0.09 & -0.17\end{bmatrix}\begin{bmatrix}0.321\\0.368\end{bmatrix} + \begin{bmatrix}0.48\end{bmatrix}\right) = \begin{bmatrix}0.446\end{bmatrix}$$



Example: Back Propagation of Error

$$e = t - a = \left\{1 + \sin\left(\frac{\pi}{4}p\right)\right\} - a^2 = \left\{1 + \sin\left(\frac{\pi}{4}1\right)\right\} - 0.446 = 1.261$$

$$\dot{f}^2(n) = \frac{d}{dn}(n) = 1$$

$$\mathbf{s}^2 = -2\dot{\mathbf{F}}^2(\mathbf{n}^2)(\mathbf{t} - \mathbf{a}) = -2\left[\dot{f}^2(n^2)\right](1.261) = -2\left[1\right](1.261) = -2.522$$

$$\mathbf{W}^2(1) = \mathbf{W}^2(0) - \alpha\mathbf{s}^2(\mathbf{a}^1)^T = \begin{bmatrix}0.09 & -0.17\end{bmatrix} - 0.1\begin{bmatrix}-2.522\end{bmatrix}\begin{bmatrix}0.321 & 0.368\end{bmatrix}$$

$$= \begin{bmatrix}0.171 & -0.0772\end{bmatrix},$$

$$\mathbf{b}^2(1) = \mathbf{b}^2(0) - \alpha\mathbf{s}^2 = \begin{bmatrix}0.48\end{bmatrix} - 0.1\begin{bmatrix}-2.522\end{bmatrix} = \begin{bmatrix}0.732\end{bmatrix}$$

Example: Back Propagation of Error

$$\dot{f}^1(n) = \frac{d}{dn}\left(\frac{1}{1+e^{-n}}\right) = \frac{e^{-n}}{(1+e^{-n})^2} = \left(1 - \frac{1}{1+e^{-n}}\right)\left(\frac{1}{1+e^{-n}}\right) = (1-a^1)(a^1)$$

$$\mathbf{s}^1 = \dot{\mathbf{F}}^1(\mathbf{n}^1)(\mathbf{W}^2)^T\mathbf{s}^2 = \begin{bmatrix} (1-a_1^1)(a_1^1) & 0 \\ 0 & (1-a_2^1)(a_2^1) \end{bmatrix}\begin{bmatrix} 0.09 \\ -0.17 \end{bmatrix}\begin{bmatrix} -2.522 \end{bmatrix}$$

$$= \begin{bmatrix} (1-0.321)(0.321) & 0 \\ 0 & (1-0.368)(0.368) \end{bmatrix}\begin{bmatrix} 0.09 \\ -0.17 \end{bmatrix}\begin{bmatrix} -2.522 \end{bmatrix}$$

$$= \begin{bmatrix} 0.218 & 0 \\ 0 & 0.233 \end{bmatrix}\begin{bmatrix} -0.227 \\ 0.429 \end{bmatrix} = \begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix}$$

$$\mathbf{W}^1(1) = \mathbf{W}^1(0) - \alpha\mathbf{s}^1(\mathbf{a}^0)^T = \begin{bmatrix} -0.27 \\ -0.41 \end{bmatrix} - 0.1\begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix}\begin{bmatrix} 1 \end{bmatrix} = \begin{bmatrix} -0.265 \\ -0.420 \end{bmatrix}$$

$$\mathbf{b}^1(1) = \mathbf{b}^1(0) - \alpha\mathbf{s}^1 = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix} - 0.1\begin{bmatrix} -0.0495 \\ 0.0997 \end{bmatrix} = \begin{bmatrix} -0.475 \\ -0.140 \end{bmatrix}.$$

The Gradient of the Logistic Function

$$\frac{d}{dx}\left[\frac{1}{1+\exp(-x)}\right] = -\frac{\frac{d}{dx}\left[1+\exp(-x)\right]}{\left(1+\exp(-x)\right)^2} = -\frac{\frac{d}{dx}[1]+\frac{d}{dx}\left[\exp(-x)\right]}{\left(1+\exp(-x)\right)^2}$$

$$= -\frac{0+\exp(-x)\frac{d}{dx}[-x]}{\left(1+\exp(-x)\right)^2} = -\frac{-\exp(-x)\frac{d}{dx}[x]}{\left(1+\exp(-x)\right)^2} = \frac{\exp(-x)\frac{d}{dx}[x]}{\left(1+\exp(-x)\right)^2}$$

$$= \frac{\exp(-x)}{\left(1+\exp(-x)\right)^2} = \left(\frac{1}{1+\exp(-x)}\right)\left(\frac{\exp(-x)}{1+\exp(-x)}\right)$$

$$= \left(\frac{1}{1+\exp(-x)}\right)\left(\frac{\exp(-x)\ /\ \exp(-x)}{1\ /\ \exp(-x)+\exp(-x)\ /\ \exp(-x)}\right)$$

$$= \left(\frac{1}{1+\exp(-x)}\right)\left(\frac{1}{\exp(x)+1}\right) = \left(\frac{1}{1+\exp(-x)}\right)\left(1-\frac{1}{1+\exp(-x)}\right)$$

Did We Learn Anything?

Yes!

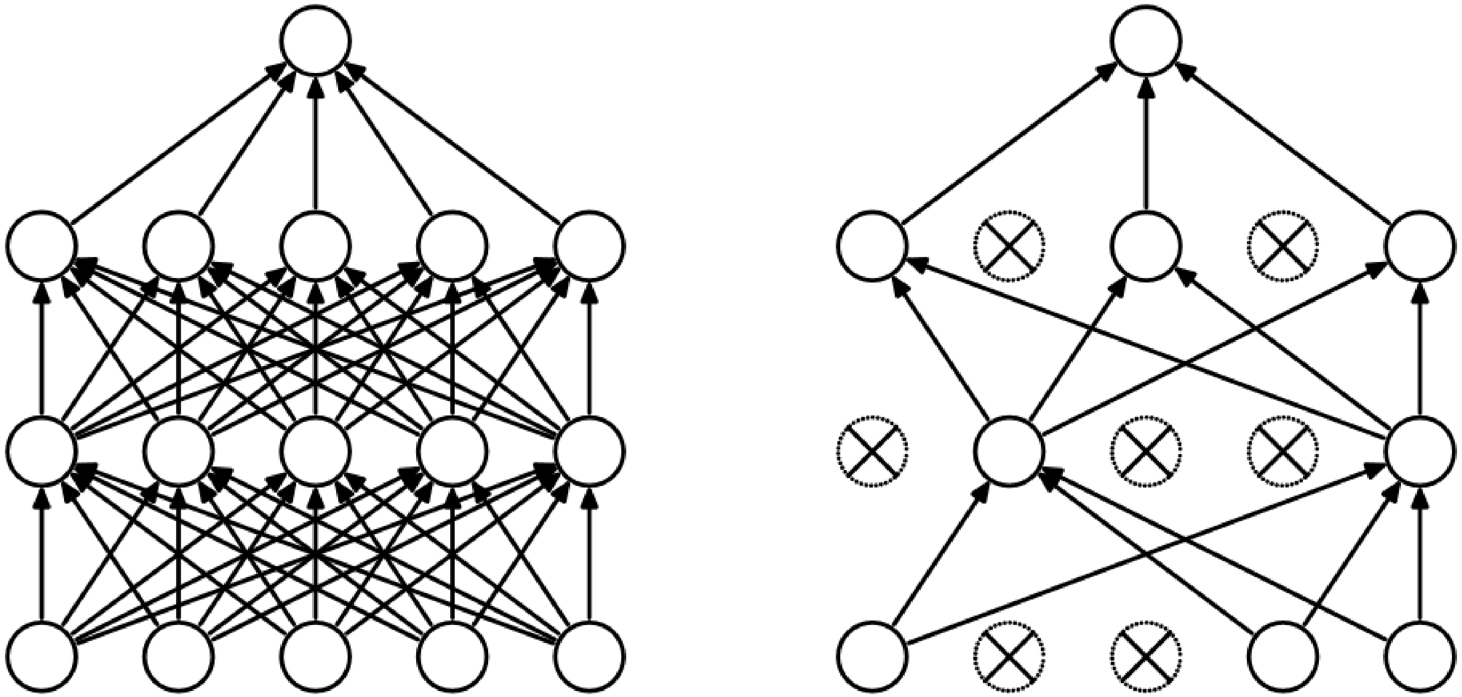
Our new output (0.759) is closer to 1.707 than our old output (0.446)

```
> w2 = c(0.171, -0.0772)
> b2 = c(0.732)
> w1 = c(-0.265, -0.420)
> b1 = c(-0.475, -0.140)
> logsig = function(x) { return(1 / (1 + exp(-x))) }
> purelin = function(x) { return(x) }
> a0 = 1
> a1 = logsig(w1 * a0 + b1)
> a2 = purelin(t(w2) %*% a1 + b2)
> a2
 0.759
```

Choices for Neural Networks

- How many layers to use?
- How many neurons (activation functions) per layer?
- Which activation functions to use?
- How to connect neurons of one layer to the next?

Using Dropout to Prevent Overfitting



(a) Standard Neural Net (b) After applying dropout.

Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Example Genetic Algorithm for Feature Selection

Randomly generate an initial population of chromosomes

repeat:

  for each chromosome do

    Tune and train a model and compute each chromosome's fitness

  end

  for each reproduction 1 … P/2 do

    Select 2 chromosomes based on fitness

    Crossover: randomly select a locus and exchange genes on either side of locus

      (head of one chromosome applied to tail of the other and vice versa)

      to produce 2 child chromosomes with mixed genes

    Mutate the child chromosomes with probability p_m

  end

until stopping criterion are met

The Naïve Bayes Model

Posterior = Prior * Likelihood / Evidence

$$prob\left(class = c \mid \bar{x}\right) = \frac{prob\left(class = c\right) * \prod_{j=1}^{p} prob\left(x_j \mid class = c\right)}{prob\left(class = c\right) * \prod_{j=1}^{p} prob\left(x_j \mid class = c\right) + prob\left(class \neq c\right) * \prod_{j=1}^{p} prob\left(x_j \mid class \neq c\right)}$$

This model is called "naïve" because it assumes conditional independence to derive the likelihood estimates:

$$prob\left(feature1 = value1, feature2 = value2 \mid class = c\right) = prob\left(feature1 = value1 \mid class = c\right) * prob\left(feature2 = value2 \mid class = c\right)$$

Add a small weight to the observed frequency counts for all possible values: this amounts to incorporating a Bayesian prior to avoid the certainty of zero or one (use 1 for Laplace smoothing)

```
> table(HouseVotes84$Class)

  democrat republican
       267        168
> table(HouseVotes84$Class, HouseVotes84$V4)

               n   y
  democrat   245  14
  republican   2 163
```

Libraries

- library(akima)
- library(boot)
- library(car)
- library(class)
- library(e1071)
- library(gam)
- library(gbm)
- library(glmnet)
- library(ISLR)
- library(leaps)
- library(MASS)
- library(pls)
- library(randomForest)
- library(ROCR)
- library(splines)
- library(tree)
- library(caret)
- library(mxnet)
- library(xgboost)

Model Construction Commands (from book)

- lm()
- glm()
- knn()
- lda()
- qda()
- cv.glm()
- regsubsets()
- glmnet()
- cv.glmnet()
- pcr()
- plsr()
- smooth.spline()
- loess()
- gam(): poly(), bs(), ns(), s(), lo()
- tree()
- cv.tree()
- randomForest()
- gbm()
- svm()
- prcomp()
- kmeans()
- hclust()

http://www-bcf.usc.edu/~gareth/ISL/All%20Labs.txt

Final Notes

- For model selection: never evaluate a model on the data used for training the model
- The train() method from library(caret) is convenient for model selection
- Remember that small data means large uncertainty: use repeated cross validation for smaller data sets
- Consider evaluating stacking/blending to boost your performance
- Review: A Few Useful Things to Know About Machine Learning

  http://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf
- Keep an open mind: new methods/tools are always being developed

Survey

- Please help support my boss learning about me learning about you learning about machine learning ☺ [please fill out the survey]

- Best Wishes for Your New Adventures!