Machine Learning (CS 181):
8. Neural Networks 1

David Parkes and Sasha Rush

Contents

1 Fitting Logistic Regression

2 Basis Functions Revisited

3 Neural Networks

4 Fitting Neural Networks

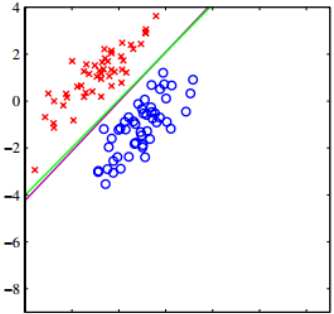
Practical 1

#	Δrank	Team Name	Score	Entries	Last Submission UTC (Best – Last Submission)
1	—	B3-D1 • BrabeebaWang • DemiGuo	0.02029	14	Fri, 10 Feb 2017 05:29:46 (-27.5h)
2	—	flower_power • dgrishin • C. N. Weisser • tktk	0.04161	15	Fri, 10 Feb 2017 01:07:51 (-1h)
3	—	. • Alex Wei • Angie Rao • Susan Xu	0.04237	11	Fri, 10 Feb 2017 16:58:25 (-10.5h)
4	—	Mediterranean Sea • Nebras Jemel • artidoro	0.04495	7	Fri, 10 Feb 2017 16:35:34 (-14.2h)
5	—	glr • boreas • LydiaGoldberg • GabbiMerz	0.04872	8	Fri, 10 Feb 2017 05:47:29
6	—	BioDeep • Joe Sedlak • Jason Qian • nathannakatsuka	0.05711	9	Fri, 10 Feb 2017 16:47:27 (-11.4h)
7	—	big play central • AndrewChen • adit	0.05719	5	Fri, 10 Feb 2017 16:34:09

Overview: Linear Classification

- Perceptron - SGD, Perceptron algorithm
- Naive Bayes - Closed-form, counts, generative probabilistic
- Logistic Regression - SGD (today), discriminative probabilistic

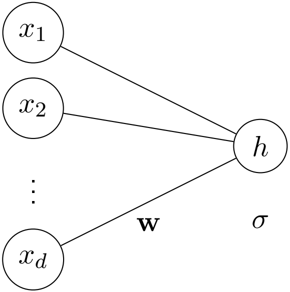
$$h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x} + w_0$$



Linear Model with Logistic

$$h(\mathbf{x}; \mathbf{w}) = \mathbf{x}^\top \mathbf{w} + w_0$$

$$p(y = 1|\mathbf{x}) = \sigma(h(\mathbf{x}; \mathbf{w}))$$



Contents

1 Fitting Logistic Regression

2 Basis Functions Revisited

3 Neural Networks

4 Fitting Neural Networks

Linear Discriminative Model

Set log prob to be proportional to some linear model, shorthand h

$$\ln p(y = 1|\mathbf{x}; \mathbf{w}) \propto \mathbf{w}^\top \mathbf{x} + w_0 = h$$

As before threshold at $h > 0$,

$$\ln p(y = 0|\mathbf{x}; \mathbf{w}) \propto 0$$

Now remove log and normalize,

$$\begin{aligned} p(y = 1|\mathbf{x}; \mathbf{w}) &= \frac{\exp h}{\exp h + \exp 0} = (1 + \exp -h)^{-1} \\ p(y = 0|\mathbf{x}; \mathbf{w}) &= \frac{\exp 0}{\exp h + \exp 0} = (1 + \exp h)^{-1} \end{aligned}$$

Call this function the logistic sigmoid activation.

$$\sigma(h) = (1 + \exp -h)^{-1}$$

Linear Discriminative Model

Set log prob to be proportional to some linear model, shorthand h

$$\ln p(y = 1|\mathbf{x}; \mathbf{w}) \propto \mathbf{w}^\top \mathbf{x} + w_0 = h$$

As before threshold at $h > 0$,

$$\ln p(y = 0|\mathbf{x}; \mathbf{w}) \propto 0$$

Now remove log and normalize,

$$\begin{aligned} p(y = 1|\mathbf{x}; \mathbf{w}) &= \frac{\exp h}{\exp h + \exp 0} = (1 + \exp -h)^{-1} \\ p(y = 0|\mathbf{x}; \mathbf{w}) &= \frac{\exp 0}{\exp h + \exp 0} = (1 + \exp h)^{-1} \end{aligned}$$

Call this function the logistic sigmoid activation.

$$\sigma(h) = (1 + \exp -h)^{-1}$$

Linear Discriminative Model

Set log prob to be proportional to some linear model, shorthand h

$$\ln p(y = 1|\mathbf{x}; \mathbf{w}) \propto \mathbf{w}^\top \mathbf{x} + w_0 = h$$

As before threshold at $h > 0$,

$$\ln p(y = 0|\mathbf{x}; \mathbf{w}) \propto 0$$

Now remove log and normalize,

$$\begin{aligned} p(y = 1|\mathbf{x}; \mathbf{w}) &= \frac{\exp h}{\exp h + \exp 0} = (1 + \exp -h)^{-1} \\ p(y = 0|\mathbf{x}; \mathbf{w}) &= \frac{\exp 0}{\exp h + \exp 0} = (1 + \exp h)^{-1} \end{aligned}$$

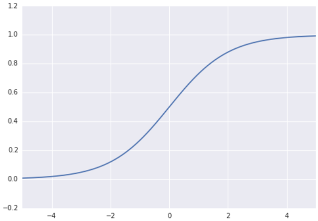
Call this function the logistic sigmoid activation.

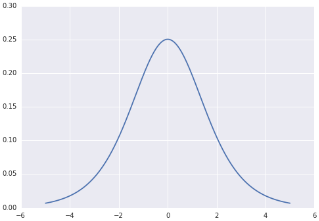
$$\sigma(h) = (1 + \exp -h)^{-1}$$

(Logistic) Sigmoid Activation

$$\sigma(h) = (1 + \exp(-h))^{-1}$$

$$\sigma'(h) = \sigma(h)^2 \exp(-h)$$





Sigmoid Function and Derivative

- “Squashes” $\mathbb{R}$ to a probabilities.

Logistic Regression

Linear model converted to probability estimated by sigmoid

$$p(y=1|\mathbf{x};\mathbf{w}) = \sigma(\mathbf{w}^\top\mathbf{x} + w_0) = (1+\exp(-h))^{-1}$$

$$p(y=0|\mathbf{x};\mathbf{w}) = 1 - \sigma(\mathbf{w}^\top\mathbf{x} + w_0) = (1+\exp h)^{-1}$$

- Linear "Regression" transformed to probability estimate.
- Name is confusing, mostly used for *classification*.

Fitting Model

Reminder:

$$p(y=1|\mathbf{x};\mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x} + w_0) = (1+\exp(-h))^{-1}$$

$$p(y=0|\mathbf{x};\mathbf{w}) = 1 - \sigma(\mathbf{w}^\top \mathbf{x} + w_0) = (1+\exp h)^{-1}$$

As this is now a probabilistic model, can fit with MLE.

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= -\sum_{i=1}^{n} \ln p(y_i|\mathbf{x}_i;\mathbf{w}) = -\sum_{i=1}^{n} \ln \sigma(h_i)^{y_i}(1-\sigma(h_i))^{1-y_i} \\ &= \sum_{i=1}^{n} y_i \ln(1+\exp(-h_i)) + (1-y_i)\ln(1+\exp h_i)\end{aligned}$$

Likelihood and Estimation (1)

Reminder:

$$\begin{aligned} h(\mathbf{x}_i;\mathbf{w}) = h_i &= \mathbf{w}^\top \mathbf{x}_i + w_0 \\ \mathcal{L}(\mathbf{w}) &= \sum_{i=1}^{n} y_i \ln(1+\exp(-h_i)) + (1-y_i)\ln(1+\exp h_i) \end{aligned}$$

Take gradients wrt h_i:

$$\frac{\partial}{\partial h_i}\ln(1+\exp h_i) = \frac{\exp h_i}{1+\exp h_i} = p(y_i = 1|\mathbf{x}_i)$$

$$\frac{\partial}{\partial h_i}\ln(1+\exp(-h_i)) = -\frac{\exp -h_i}{1+\exp(-h_i)} = -p(y_i = 0|\mathbf{x}_i)$$

Chain rule (with h_i scalar function of $\mathbf{w}$):

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial h_i}{\partial \mathbf{w}}\frac{\partial \mathcal{L}(\mathbf{w})}{\partial h_i}$$

Likelihood and Estimation (1)

Reminder:

$$\begin{aligned} h(\mathbf{x}_i; \mathbf{w}) = h_i &= \mathbf{w}^\top \mathbf{x}_i + w_0 \\ \mathcal{L}(\mathbf{w}) &= \sum_{i=1}^{n} y_i \ln(1 + \exp(-h_i)) + (1 - y_i) \ln(1 + \exp h_i) \end{aligned}$$

Take gradients wrt h_i:

$$\frac{\partial}{\partial h_i} \ln(1 + \exp h_i) = \frac{\exp h_i}{1 + \exp h_i} = p(y_i = 1 | \mathbf{x}_i)$$

$$\frac{\partial}{\partial h_i} \ln(1 + \exp(-h_i)) = -\frac{\exp -h_i}{1 + \exp(-h_i)} = -p(y_i = 0 | \mathbf{x}_i)$$

Chain rule (with h_i scalar function of $\mathbf{w}$):

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial h_i}{\partial \mathbf{w}} \frac{\partial \mathcal{L}(\mathbf{w})}{\partial h_i}$$

Likelihood and Estimation (1)

Reminder:

$$\begin{aligned} h(\mathbf{x}_i; \mathbf{w}) = h_i &= \mathbf{w}^\top \mathbf{x}_i + w_0 \\ \mathcal{L}(\mathbf{w}) &= \sum_{i=1}^{n} y_i \ln(1 + \exp(-h_i)) + (1 - y_i) \ln(1 + \exp h_i) \end{aligned}$$

Take gradients wrt h_i:

$$\frac{\partial}{\partial h_i} \ln(1 + \exp h_i) = \frac{\exp h_i}{1 + \exp h_i} = p(y_i = 1 | \mathbf{x}_i)$$

$$\frac{\partial}{\partial h_i} \ln(1 + \exp(-h_i)) = -\frac{\exp -h_i}{1 + \exp(-h_i)} = -p(y_i = 0 | \mathbf{x}_i)$$

Chain rule (with h_i scalar function of $\mathbf{w}$):

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial h_i}{\partial \mathbf{w}} \frac{\partial \mathcal{L}(\mathbf{w})}{\partial h_i}$$

Likelihood and Estimation (2)

$$h_i = \mathbf{w}^\top \mathbf{x}_i + w_0$$

Take gradients wrt $\mathbf{w}$:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} \ln(1 + \exp h_i) &= \frac{\partial h_i}{\partial \mathbf{w}} \times p(y_i = 1|\mathbf{x}) = \mathbf{x}_i p(y_i = 1|\mathbf{x}) \\
\frac{\partial}{\partial \mathbf{w}} \ln(1 + \exp(-h_i)) &= \frac{\partial h_i}{\partial \mathbf{w}} \times -p(y_i = 0|\mathbf{x}) = -\mathbf{x}_i p(y_i = 0|\mathbf{x})
\end{aligned}$$

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n} -y_i \mathbf{x}_i p(y_i = 0|\mathbf{x}_i) + (1 - y_i)\mathbf{x}_i p(y_i = 1|\mathbf{x}_i)$$

- If $y_i = 1$, gradient is $-\mathbf{x}_i p(y_i = 0|\mathbf{x}_i)$
- if $y_i = 0$, gradient is $\mathbf{x}_i p(y_i = 1|\mathbf{x}_i)$

Likelihood and Estimation (2)

$$h_i = \mathbf{w}^\top \mathbf{x}_i + w_0$$

Take gradients wrt $\mathbf{w}$:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} \ln(1 + \exp h_i) = &\quad \frac{\partial h_i}{\partial \mathbf{w}} \times p(y_i = 1|\mathbf{x}) = \mathbf{x}_i p(y_i = 1|\mathbf{x}) \\
\frac{\partial}{\partial \mathbf{w}} \ln(1 + \exp(-h_i)) = &\quad \frac{\partial h_i}{\partial \mathbf{w}} \times -p(y_i = 0|\mathbf{x}) = -\mathbf{x}_i p(y_i = 0|\mathbf{x})
\end{aligned}$$

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n} -y_i \mathbf{x}_i p(y_i = 0|\mathbf{x}_i) + (1 - y_i)\mathbf{x}_i p(y_i = 1|\mathbf{x}_i)$$

- If $y_i = 1$, gradient is $-\mathbf{x}_i p(y_i = 0|\mathbf{x}_i)$
- if $y_i = 0$, gradient is $\mathbf{x}_i p(y_i = 1|\mathbf{x}_i)$

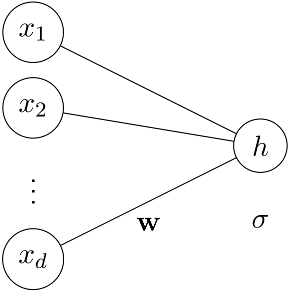
High-Level Gradient Computation

Compute the loss function:

$$\mathbf{w} \rightarrow h(\mathbf{x}; \mathbf{w}) \rightarrow \mathcal{L}$$

Compute the gradient of loss wrt the weight, use chain rule:

$$\frac{\partial h}{\partial \mathbf{w}} \leftarrow \frac{\partial \mathcal{L}}{\partial h} \leftarrow \mathcal{L}$$



Reminder: Stochastic Gradient Descent

Instead of computing gradient of entire loss, compute stochastic gradients.

1. Sample data point i, corresponding to $(\mathbf{x}_i, y_i)$
2. Compute loss and gradient for just that data point $\mathcal{L}^{(i)}(\mathbf{w})$
3. Update $\mathbf{w}$ based on this *stochastic gradient*

Similar to standard gradient descent, often more efficient in practice.

SGD on Logistic Regression

(Exercise: derive)

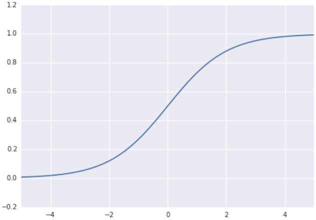
1. Iterate over the data:
   - Compute $p(y_i = 1|\mathbf{x}_i)$.
   - If $(y_i = 1)$, add $\eta \times \mathbf{x}_i p(y_i = 0|\mathbf{x}_i)$ to $\mathbf{w}$
   - If $(y_i = 0)$, add $\eta \times -\mathbf{x}_i p(y_i = 1|\mathbf{x}_i)$ to $\mathbf{w}$
2. Repeat until convergence.

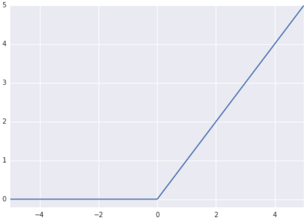
Guaranteed to maximize conditional likelihood of data.

Recall: Perceptron Algorithm

1. Iterate over the data:
   - If correct $(y_i = \hat{y}_i)$, do nothing.
   - If incorrect, add/subtract $\eta \times \mathbf{x}_i$ to weights
2. If errors, repeat process.
3. Otherwise separator is found.

- What is the difference between Perceptron and Logistic Regression?





Sigmoid Function versus Hinge

Contents

1 Fitting Logistic Regression

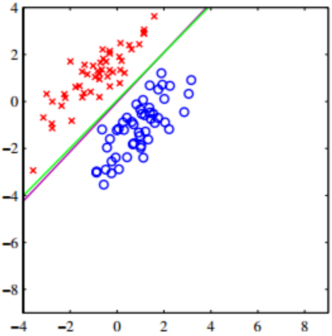
2 Basis Functions Revisited

3 Neural Networks

4 Fitting Neural Networks

Linear Models

- Linear models aren't very good for most problems.



Binary Classification

Basis Functions

- What should we do?
- Our answer so far: use basis functions.

$$h(\mathbf{x}; \mathbf{w}, w_0) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + w_0.$$

- Basis function provides another representation that makes it linear-separable (in $\mathbb{R}^d$)

Downside of Basis Functions

Choosing basis functions is hard. Either need:

- Specific domain knowledge of the problem
  - Frequency domain in speech, visual detection for images, etc.
- Very large transformation with aggressive regularization.
  - Radial basis functions, high-degree polynomials

Adaptive Basis Functions

Alternative approach: learn the basis functions from data.

$$\boldsymbol{\phi}(\mathbf{x}; \mathbf{w}^1) : \mathbb{R}^m \mapsto \mathbb{R}^d$$

- Combines basis selection and learning.
- Several different approaches, we will focus on neural networks.
- Downside: Complicates the optimization problem.

Contents

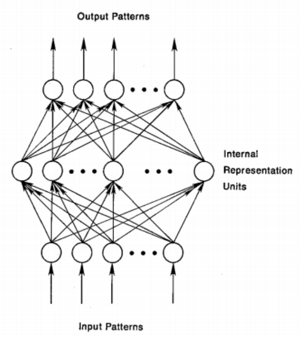
1 Fitting Logistic Regression

2 Basis Functions Revisited

3 Neural Networks

4 Fitting Neural Networks

- AI Winter (post-perceptron)
- Paul Werbos (1974,1982)
- Popularized by work of Rumelhart et al (1986)



Learning Internal Representations by Error Propagation

D. E. RUMELHART, G. E. HINTON, and R. J. WILLIAMS

Neural Networks: Formally

Define basis function as logistic regression:

For all $j \in \{1, \ldots, d\}$, define basis function as,

$$\phi_j(\mathbf{x}; \mathbf{w}_j^1, w_{j0}^1) = \sigma(\mathbf{w}_j^{1\top} \mathbf{x} + w_{j0}^1)$$

Where $\mathbf{w}_j^1 \in \mathbb{R}^{m \times 1}$

Interpretation:

- Predicting feature basis value.
- Different *linear* weight vector $\mathbf{w}_j$ for each basis dimension

Neural Networks: Formally

Define basis function as logistic regression:

For all $j \in \{1, \ldots, d\}$, define basis function as,

$$\phi_j(\mathbf{x}; \mathbf{w}_j^1, w_{j0}^1) = \sigma(\mathbf{w}_j^{1\top} \mathbf{x} + w_{j0}^1)$$

Where $\mathbf{w}_j^1 \in \mathbb{R}^{m \times 1}$

Interpretation:

- Predicting feature basis value.
- Different *linear* weight vector $\mathbf{w}_j$ for each basis dimension

Neural Networks: Matrix-view

Define entire adaptive basis layer as d problems,

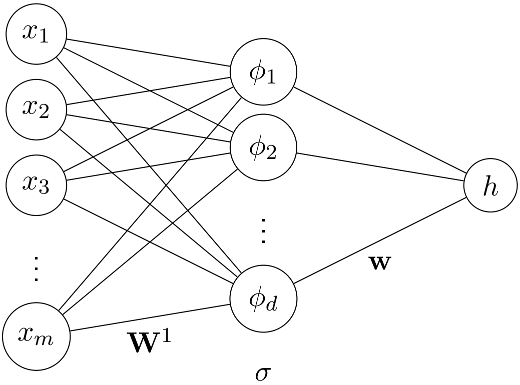
$$\boldsymbol{\phi}(\mathbf{x}; \mathbf{W}^1, \mathbf{w}_0^1) = \boldsymbol{\sigma}(\mathbf{W}^1 \mathbf{x} + \mathbf{w}_0^1)$$

Where $\boldsymbol{\sigma}$ is pointwise sigmoid and $\mathbf{W}^1 \in \mathbb{R}^{d \times m}$

Full classification problem:

$$\begin{aligned} h(\mathbf{x}; \mathbf{w}, w_0, \mathbf{W}^1, \mathbf{w}_0^1) &= \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}; \mathbf{W}^1, \mathbf{w}_0^1) + w_0 \\ &= \mathbf{w}^\top \boldsymbol{\sigma}(\mathbf{W}^1 \mathbf{x} + \mathbf{w}_0^1) + w_0 \end{aligned}$$

Neural Network Diagram

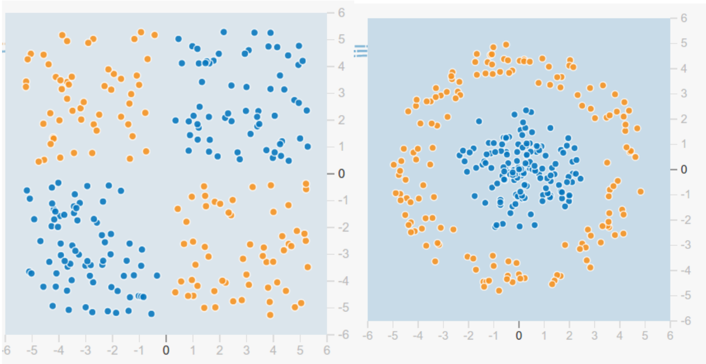


$$\boldsymbol{\phi}(\mathbf{x}) = \boldsymbol{\sigma}(\mathbf{W}^1\mathbf{x} + \mathbf{w}_0^1)$$

$$h(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + w_0$$

Non-Linear Problems

- Clearly no linear separator.



Two class classification: (1) xor classification, (2) ring

Exercise: XOR with Neural Network

- $\mathbf{x} = [0; 1]$ $\mathbf{x} = [1; 0]$ in class $y = 1$.
- $\mathbf{x} = [0; 0]$ $\mathbf{x} = [1; 1]$ in class $y = 0$.

$$
\begin{aligned}
\phi_1(x_1, x_2) &= \sigma(-x_1 - x_2) + 0.5 \\
\phi_2(x_1, x_2) &= \sigma(x_1 + x_2) - 0.5 \\
h(\mathbf{x}) &= -\phi_1 - \phi_2 + 0.5
\end{aligned}
$$

- $\phi_1 > 0$, only $[0; 0]$
- $\phi_2 > 0$, only $[1; 1]$
- h, only $\phi_1 = 0$ and $\phi_2 = 0$

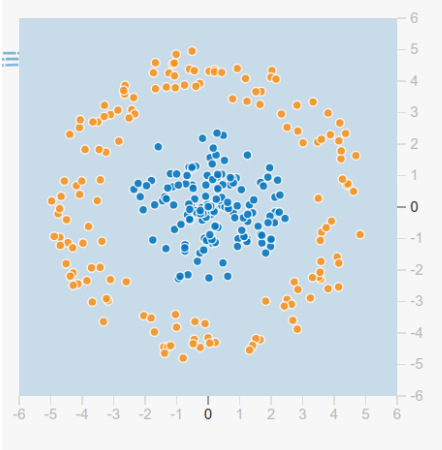
Exercise: XOR with Neural Network

- $\mathbf{x} = [0; 1]$ $\mathbf{x} = [1; 0]$ in class $y = 1$.
- $\mathbf{x} = [0; 0]$ $\mathbf{x} = [1; 1]$ in class $y = 0$.

$$\begin{aligned} \phi_1(x_1, x_2) &= \sigma(-x_1 - x_2) + 0.5 \\ \phi_2(x_1, x_2) &= \sigma(x_1 + x_2) - 0.5 \\ h(\mathbf{x}) &= -\phi_1 - \phi_2 + 0.5 \end{aligned}$$

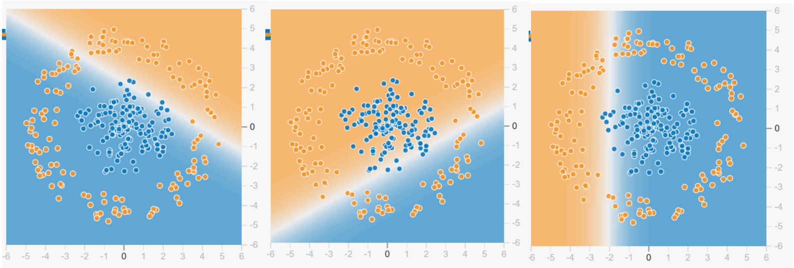
- $\phi_1 > 0$, only $[0; 0]$
- $\phi_2 > 0$, only $[1; 1]$
- h, only $\phi_1 = 0$ and $\phi_2 = 0$

Ring



Visual Description

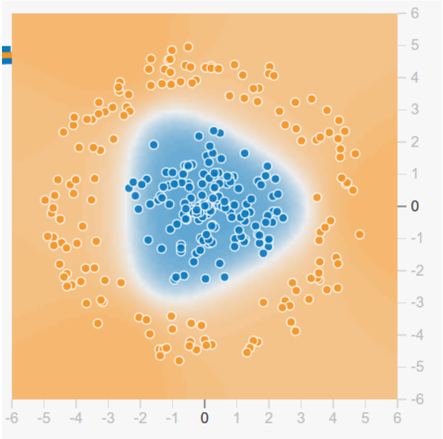
- ϕ is constructed by learned linear models.
- Logistic regression to select ϕ values.



Separators for learned features ϕ_1, ϕ_2, ϕ_3.

Final Non-Linear Output

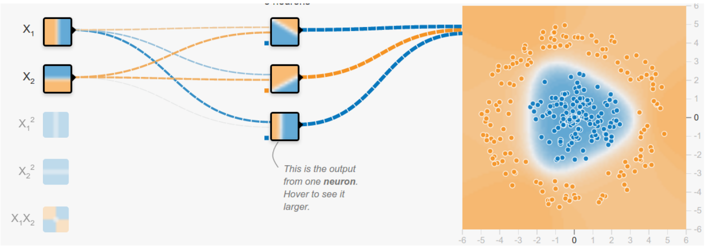
- Combining multiple layers give non-linear decision boundary.



Final classification h using ϕ basis.

Neural Network

- Multi-layer linear decisions for classification.



A 2-layer neural network

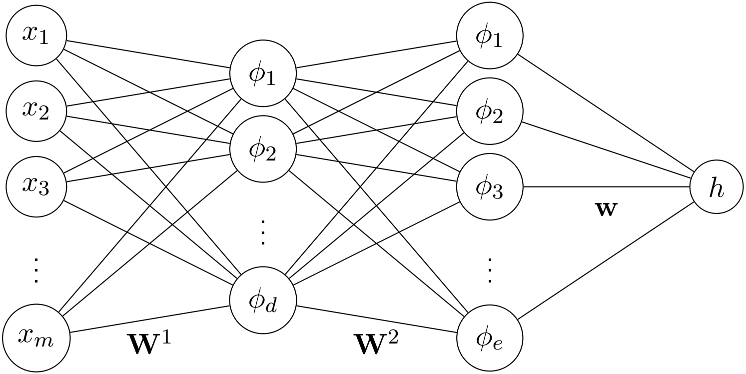
Multilayer “Deep” Neural Networks

Can stack arbitrary layers of basis functions.

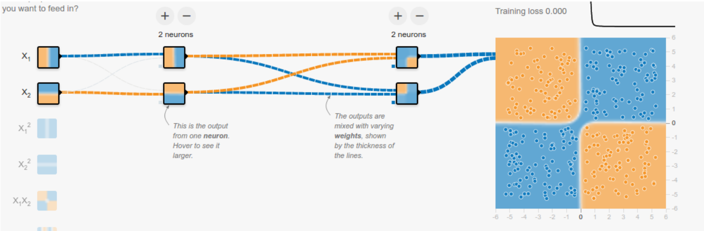
$$\boldsymbol{\phi}^l(\mathbf{x}; \mathbf{W}^l, \mathbf{w}_0^l) = \boldsymbol{\sigma}(\mathbf{W}^l \boldsymbol{\phi}^{(l-1)} + \mathbf{w}_0^l)$$

- Each layer learns adaptive basis based on previous layers.

Multilayer NN



3-Layer Xor Network



Demos

Contents

1 Fitting Logistic Regression

2 Basis Functions Revisited

3 Neural Networks

4 Fitting Neural Networks

Loss Function

Typical to train neural networks with maximum likelihood estimation.

$$\mathcal{L}(\mathbf{w}, \mathbf{W}) = -\sum_{i=1}^{n} \ln p(y_i | \mathbf{x}_i; \mathbf{w}, \mathbf{W})$$

- Can also use other losses.
- Or even train for regression, other tasks.

$$\mathcal{L}(\mathbf{w}, \mathbf{W}) = \frac{1}{2} \sum_{i=1}^{n} (y_i - h(\mathbf{x}_i; \mathbf{w}, \mathbf{W}))^2$$

Recall: Estimation (1)

$$\mathcal{L}(\mathbf{w}, \mathbf{W}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - h(\mathbf{x}_i; \mathbf{w}, \mathbf{W}))^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - h_i)^2$$

Take gradients wrt h_i:

$$\frac{\partial}{\partial h_i}\mathcal{L} = -(y_i - h_i)$$

(New:) Need loss gradient to train $\mathbf{W}$ inside basis.

$$h(\mathbf{x}; \mathbf{w}, \mathbf{W}) = h_i = \mathbf{w}^\top \phi(\mathbf{x}_i; \mathbf{W}) + w_0$$

Recall: Estimation (1)

$$\mathcal{L}(\mathbf{w}, \mathbf{W}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - h(\mathbf{x}_i; \mathbf{w}, \mathbf{W}))^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - h_i)^2$$

Take gradients wrt h_i:

$$\frac{\partial}{\partial h_i}\mathcal{L} = -(y_i - h_i)$$

(New:) Need loss gradient to train $\mathbf{W}$ inside basis.

$$h(\mathbf{x}; \mathbf{w}, \mathbf{W}) = h_i = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i; \mathbf{W}) + w_0$$

Estimation (2)

$$\mathbf{w} \in \mathbb{R}^m \rightarrow \mathbf{g}(\mathbf{w}) \in \mathbb{R}^d \rightarrow f(\mathbf{g}(\mathbf{w})) \in \mathbb{R}$$

$$\frac{\partial f(\mathbf{g}(\mathbf{w}))}{\partial \mathbf{w}} \in \mathbb{R}^m \leftarrow \frac{\partial f(\mathbf{g}(\mathbf{w}))}{\partial \mathbf{g}(\mathbf{w})} \in \mathbb{R}^d \leftarrow f(\mathbf{g}(\mathbf{w})) \in \mathbb{R}$$

Chain rule, vector-valued functions:

$$\frac{\partial f(\mathbf{g}(\mathbf{w}))}{\partial \mathbf{w}} = \sum_{j=1}^{m} \frac{\partial g(\mathbf{w})_j}{\partial \mathbf{w}} \frac{\partial f(g(\mathbf{w}))}{\partial g(\mathbf{w})_j}$$

Estimation (2)

$$\mathbf{w} \in \mathbb{R}^m \to \mathbf{g}(\mathbf{w}) \in \mathbb{R}^d \to f(\mathbf{g}(\mathbf{w})) \in \mathbb{R}$$

$$\frac{\partial f(\mathbf{g}(\mathbf{w}))}{\partial \mathbf{w}} \in \mathbb{R}^m \leftarrow \frac{\partial f(\mathbf{g}(\mathbf{w}))}{\partial \mathbf{g}(\mathbf{w})} \in \mathbb{R}^d \leftarrow f(\mathbf{g}(\mathbf{w})) \in \mathbb{R}$$

Chain rule, vector-valued functions:

$$\frac{\partial f(\mathbf{g}(\mathbf{w}))}{\partial \mathbf{w}} = \sum_{j=1}^{m} \frac{\partial g(\mathbf{w})_j}{\partial \mathbf{w}} \frac{\partial f(g(\mathbf{w}))}{\partial g(\mathbf{w})_j}$$

Estimation (2)

$$\mathbf{w} \in \mathbb{R}^m \to \mathbf{g}(\mathbf{w}) \in \mathbb{R}^d \to f(\mathbf{g}(\mathbf{w})) \in \mathbb{R}$$

$$\frac{\partial f(\mathbf{g}(\mathbf{w}))}{\partial \mathbf{w}} \in \mathbb{R}^m \leftarrow \frac{\partial f(\mathbf{g}(\mathbf{w}))}{\partial \mathbf{g}(\mathbf{w})} \in \mathbb{R}^d \leftarrow f(\mathbf{g}(\mathbf{w})) \in \mathbb{R}$$

Chain rule, vector-valued functions:

$$\frac{\partial f(\mathbf{g}(\mathbf{w}))}{\partial \mathbf{w}} = \sum_{j=1}^{m} \frac{\partial g(\mathbf{w})_j}{\partial \mathbf{w}} \frac{\partial f(g(\mathbf{w}))}{\partial g(\mathbf{w})_j}$$

Estimation (3)

$$\frac{\partial}{\partial h_i}\mathcal{L} = -(y_i - h_i)$$

$$\frac{\partial h_i}{\partial \phi_j} = w_j$$

$$\frac{\partial \phi_j}{\partial \mathbf{w}_j^1} = \mathbf{x}_i \sigma'(\mathbf{w}_j^{1\top}\mathbf{x}_i + w_{j0})$$

Apply chain rule:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_j^1} = \mathbf{x}_i \sigma'(\mathbf{w}_j^{1\top}\mathbf{x}_i + w_{j0}) \times \sum_{i=1}^{n} w_j \times -(y_i - h_i)$$

Chain Rule

$$\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{W}} \leftarrow \frac{\partial h_i}{\partial \boldsymbol{\phi}} \leftarrow \frac{\partial \mathcal{L}}{\partial h_i} \leftarrow \mathcal{L}$$

- The chain rule is applied to compute gradient of loss wrt $\mathbf{W}$.
- For efficiency, the chain rule can be applied step-wise without instantiating full Jacobian.
- This methodology trick in general is known as backpropagation.