Lecture Notes for Deep Learning

1 Introduction

m	number of samples (or rows)
n	number of features (or columns)
y	expected output (i.e. actual label value(s))
$\hat{y}$	output (i.e. predicted label value(s))
w	weight(s)
b	bias
x	input
C	loss function
ς	activation function
(k)	number of neurons in layer k
$\boldsymbol{f}$	element-wise function f
$C[x]$	partial derivative of C w.r.t. x

The training set is a set of samples used for "building" or "finding" a model. The test set is a set of samples used for "evaluating" or "checking" a model. A metric is either a loss function or testing performance measure.[1]

Informally, a model is said to be overfit if its capacity is too high, and underfit if it is too low. Regularization mitigates overfitting by reducing the Vapnik–Chervonenkis (VC) dimension of the hypothesis class of the model. The ridge and lasso regularization terms "encourage" smaller weights quadratically and linearly, respectively; the latter allows weights to become zero.[2]

Conventionally, a model starts out with some randomly initialised weights and bias(es); the model then updates its weights and bias(es) iteratively, following the "negative gradient" of a chosen loss function.

[1] A "feature" in machine learning is said to correspond to an "independent variable" in statistics.

[2] In describing (the hypothesis class of) a model, "capacity" is synonymous with "complexity", "expressiveness", and "flexibility".

2 Non-Deep Machine Learning Models

Definition 1 (Linear Regression Model)**.** Let $\boldsymbol{X} \in \mathbb{R}^{m\times(n+1)}$, and $\boldsymbol{\beta} \in \mathbb{R}^{n+1}$.

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$$

Definition 2 (MSE Loss Function)**.** Let $\boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathbb{R}^m$.

$$\mathsf{MSE} = \frac{1}{m}\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2$$

Theorem 1. Let $\boldsymbol{X} \in \mathbb{R}^{m\times(n+1)}$, and $\boldsymbol{\beta} \in \mathbb{R}^{n+1}$.

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^{n+1}}{\operatorname{argmin}} \mathsf{MSE} = (\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}\boldsymbol{X}\boldsymbol{y}$$

Definition 3 (Logistic Binary Classification Model)**.** Let $\boldsymbol{X} \in \mathbb{R}^{m\times(n+1)}$, and $\boldsymbol{w} \in \mathbb{R}^{(n+1)}$.

$$\hat{\boldsymbol{y}} = \boldsymbol{\varsigma}(\boldsymbol{X}\boldsymbol{w})$$

Remark 1. Each prediction vector entry can be interpreted as the probability of the corresponding input vector entry being in the category associated with "1".

Definition 4 (Cross-Entropy Loss Function)**.** Let $\boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathbb{R}^m$.

$$\begin{aligned}\mathsf{CE} &= -\frac{1}{m}\sum_{i=1}^{m} y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i) \\ &= -\frac{1}{m}\left(\begin{bmatrix}\boldsymbol{y}\\ \mathbf{1}-\boldsymbol{y}\end{bmatrix} \odot \begin{bmatrix}\mathbf{ln}(\hat{\boldsymbol{y}})\\ \mathbf{ln}(\mathbf{1}-\hat{\boldsymbol{y}})\end{bmatrix}\right)\end{aligned}$$

3 Deep Machine Learning Models

We shall stipulate the 0th layer to be the input "layer". In deep learning, a (bracketed) superscript is commonly used to denote a layer number. In practice, RELU is often used over sigmoid as the former is unbounded and continuously differentiable. For simplicity, we shall define our multi-layer perceptron model to receive a vector as input, and fix an activation function to be used for all layers.

Definition 5 (Multi-Layer Perceptron Model)**.** Let $\boldsymbol{x} \in \mathbb{R}^m$, and $\boldsymbol{W}_l \in \mathbb{R}^{(l)\times(l-1)}$, and $\boldsymbol{b}_l \in \mathbb{R}^{(l)}$, and ς be an activation function, and C be a loss function.

$$\hat{\boldsymbol{y}}_l = \begin{cases} \boldsymbol{\varsigma}(\boldsymbol{W}_l \hat{\boldsymbol{y}}_{l-1} + \boldsymbol{b}), & \text{if } l > 0 \\ \boldsymbol{\varsigma}(\boldsymbol{x}), & \text{otherwise} \end{cases}$$

Let $\bar{\boldsymbol{y}}_l = \boldsymbol{W}_l \hat{\boldsymbol{y}}_{l-1} + \boldsymbol{b}$.

$$\boldsymbol{\varepsilon}_l = \begin{cases} (\boldsymbol{W}_{l+1}^\intercal \boldsymbol{\varepsilon}_{l+1}) \odot \boldsymbol{\varsigma}'(\bar{\boldsymbol{y}}_l), & \text{if } l < L \\ C[\hat{\boldsymbol{y}}_L] \odot \boldsymbol{\varsigma}'(\bar{\boldsymbol{y}}_L), & \text{otherwise} \end{cases}$$

$$C[\boldsymbol{b}_l] = \boldsymbol{\varepsilon}_l$$

$$C[\boldsymbol{w}_l] = \boldsymbol{\varepsilon}_l \hat{\boldsymbol{y}}_{l-1}^\intercal$$

Informally, exploding and vanishing gradients are gradients which become too large or small for computation. Fixing some number of layers, learning rate, and initial weights, exploding and vanishing gradients can be seen as results of repeated composition using weight matrices which are, "on the whole", too large or small. For concreteness, consider $\prod_{l=1}^{L} \|\boldsymbol{W}_l\|$: for some learning rate and weight initialisation, one's gradient may explode if the norm exceeds 1 for too many layers, or vanish if the norm deceeds 1 for too many layers.

4 Appendix

model	**loss function example(s)**
linear regression	MSE
logistic binary classification	cross-entropy

Proposition 1. Let $x \in \mathbb{R}$.

$$0 < \sigma(x) < 0.5 \Leftrightarrow x < 0$$

$$\sigma(x) = 0.5 \Leftrightarrow x = 0$$

$$0.5 < \sigma(x) < 1 \Leftrightarrow x > 0$$