第三章　机器学习概述

机器学习是对能通过经验自动改进的计算机算法的研究。

— Mitchell [1997]

在介绍人工神经网络之前，我们先来了解下机器学习的基本概念。然后再介绍下最简单的神经网络：感知器。

机器学习主要是研究如何使计算机从给定的数据中学习规律，即从观测数据（样本）中寻找规律，并利用学习到的规律（模型）对未知或无法观测的数据进行预测。目前，主流的机器学习算法是基于统计的方法，也叫统计机器学习。

机器学习系统的示例见图3.1。

DRAFT
编译时间：2016-09-27 23:14

3.1　机器学习概述

狭义地讲，机器学习是给定一些训练样本 $(x_i, y_i), 1 \leq i \leq N$ （其中 x_i 是输入，y_i 是需要预测的目标），让计算机自动寻找一个**决策函数** $f(\cdot)$ 来建立 x 和

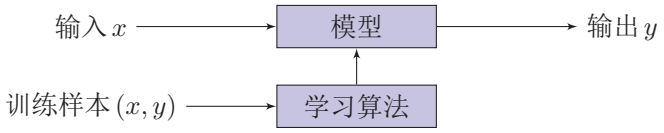

图 3.1: 机器学习系统示例

y之间的关系。

$$\hat{y} = f(\boldsymbol{\phi}(x), \theta), \tag{3.1}$$

这里，$\hat{y}$是模型输出，θ为决策函数的参数，$\boldsymbol{\phi}(x)$表示样本x对应的特征表示。因为x不一定都是数值型的输入，因此需要通过$\boldsymbol{\phi}(x)$将x转换为数值型的输入。如果我们假设x是已经处理好的标量或向量，公式3.1也可以直接写为

$$\hat{y} = f(x, \theta). \tag{3.2}$$

此外，我们还要建立一些准则来衡量决策函数的好坏。在很多机器学习算法中，一般是定义一个**损失函数** $L(y, f(x, \theta))$，然后在所有的训练样本上来评价决策函数的风险。

$$R(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)}, \theta)). \tag{3.3}$$

这里，风险函数$R(\theta)$是在已知的训练样本（经验数据）上计算得来的，因此被称之为**经验风险**。用对参数求经验风险来逐渐逼近理想的期望风险的最小值，就是我们常说的**经验风险最小化原则**（Empirical Risk Minimization）。这样，我们的目标就是变成了找到一个参数θ^*使得经验风险最小。

$$\theta^* = \arg\min_{\theta} R(\theta). \tag{3.4}$$

因为用来训练的样本往往是真实数据的一个很小的子集或者包含一定的噪声数据，不能很好地反映全部数据的真实分布。经验风险最小化原则很容易导致模型在训练集上错误率很低，但是在未知数据上错误率很高。这就是所谓的**过拟合** index过拟合。过拟合问题往往是由于训练数据少和噪声等原因造成的。过拟合的标准定义为：给定一个假设空间H，一个假设h属于H，如果存在其他的假设h属于H, 使得在训练样例上h的损失比h小，但在整个实例分布上h比h的损失小，那么就说假设h过度拟合训练数据 [Mitchell, 1997]。

和过拟合相对应的一个概念是**泛化错误**。泛化错误是衡量一个机器学习模型是否可以很好地泛化到未知数据。泛化错误一般表现为一个模型在训练集和测试集上错误率的差距。

为了解决过拟合问题，一般在经验风险最小化的原则上上加参数的**正则化**（Regularization），也叫**结构风险最小化原则**（Structure Risk Minimization）。

$$\theta^* = \arg\min_{\theta} R(\theta) + \lambda \|\theta\|_2^2 \tag{3.5}$$

DRAFT 草稿时间：2016-09-27 23:14

$$= \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f(x^{(i)}, \theta)) + \lambda \|\theta\|^2. \tag{3.6}$$

这里，$\|\theta\|_2$ 是 L_2 范数的**正则化项**，用来减少参数空间，避免**过拟合**。λ 用来控制正则化的强度。

正则化项也可以使用其它函数，比如 L_1 范数。L_1 范数的引入通常会使得参数有一定稀疏性，因此在很多算法中也经常使用。在 Bayes 估计的角度来讲，正则化是假设了参数的先验分布，不完全依赖训练数据。

3.1.1 损失函数

给定一个实例 (x, y)，真实目标是 y，机器学习模型的预测为 $f(x, \theta)$。如果预测错误时（$f(x, \theta) \neq y$），我们需要定义一个度量函数来定量地计算错误的程度。常见的损失函数有如下几类：

0-1 损失函数 0-1 损失函数（0-1 loss function）是

$$L(y, f(x, \theta)) = \begin{cases} 0 & \textbf{if } y = f(x, \theta) \\ 1 & \textbf{if } y \neq f(x, \theta) \end{cases} \tag{3.7}$$

$$= I(y \neq f(x, \theta)), \tag{3.8}$$

这里 I 是指示函数。

平方损失函数 平方损失函数（quadratic loss function）是

$$L(y, \hat{y}) = (y - f(x, \theta))^2 \tag{3.9}$$

交叉熵损失函数 对于分类问题，预测目标 y 为离散的类别，模型输出 $f(x, \theta)$ 为每个类的条件概率。

假设 $y \in \{1, \cdots C\}$，模型预测的第 i 个类的条件概率 $P(y = i|x) = f_i(x, \theta)$，则 $f(x, \theta)$ 满足

$$f_i(x, \theta) \in [0, 1], \qquad \sum_{i=1}^{C} f_i(x, \theta) = 1 \tag{3.10}$$

DRAFT 编译时间：2016-09-27 23:14

$f_y(x,\theta)$可以看作真实类别y的似然函数。参数可以直接用最大似然估计来优化。考虑到计算问题，我们经常使用最小化负对数似然，也就是**负对数似然损失函数**（Negative Log Likelihood function）。

$$L(y, f(x,\theta)) = -\log f_y(x,\theta). \tag{3.11}$$

如果我们用one-hot向量$\mathbf{y}$来表示目标类别c，其中只有$y_c = 1$，其余的向量元素都为0。

负对数似然函数也可以写为：

$$L(y, f(x,\theta)) = -\sum_{i=1}^{C} y_i \log f_i(x,\theta). \tag{3.12}$$

y_i也也可以看成是真实类别的分布，这样公式3.12恰好是交叉熵的形式。因此，负对数似然损失函数也常叫做**交叉熵损失函数**（Cross Entropy Loss function）是负对数似然函数的一种改进。

Hinge损失函数 对于两类分类问题，假设y和$f(x,\theta)$的取值为$\{-1,+1\}$。Hinge损失函数（Hinge Loss Function）的定义如下：

$$L(y, f(x,\theta)) = \max(0, 1 - yf(x,\theta)) \tag{3.13}$$

$$= |1 - yf(x,\theta)|_+. \tag{3.14}$$

3.1.2 机器学习算法的类型

根据训练数据提供的信息以及反馈方式的不同，机器学习算法一般可以分为以下几类：

有监督学习（Supervised Learning） 有监督学习是利用一组已知输入x和输出y的数据来学习模型的参数，使得模型预测的输出标记和真实标记尽可能的一致。有监督学习根据输出类型又可以分为**回归**和**分类**两类。

回归（Regression） 如果输出y是连续值（实数或连续整数），$f(x)$的输出也是连续值。这种类型的问题就是回归问题。对于所有已知或未知的(x,y)，使得$f(x,\theta)$和y尽可能地一致。损失函数通常定义为平方误差。

$$L(y, f(x,\theta)) = \|y - f(x,\theta)\|^2$$

DRAFT 编译时间：2016-09-27 23:14

分类（Classification） 如果输出y是离散的类别标记（符号），就是分类问题。损失函数有很多种定义方式。一种常用的方式0-1损失函数。

$$L(\hat{y}, y) = I(f(x, \theta) \neq y),$$

这里$f(x, \theta)$的输出也是离散值，$I(\cdot)$是指示函数，若条件为真，$I(\cdot) = 1$；否则$I(\cdot) = 0$。另一种常用的方式是让$f_i(x, \theta)$去估计给定x的情况下第i个类别的条件概率$P(y = i|x)$。损失函数定义为负对数似然函数。

$$L(y, f(x, \theta)) = -\log f_y(x, \theta).$$

在分类问题中，通过学习得到的决策函数$f(x, \theta)$也叫**分类器**。

无监督学习（Unsupervised Learning） 无监督学习是用来学习的数据不包含输出目标，需要学习算法自动学习到一些有价值的信息。一个典型的无监督学习问题就是**聚类**（Clustering）。

增强学习（Reinforcement Learning） 增强学习也叫强化学习，强调如何基于环境做出一系列的动作，以取得最大化的累积收益。每做出一个动作，并不一定立刻得到收益。增强学习和有监督学习的不同在于增强学习不需要显式地以输入/输出对的方式给出训练样本，是一种在线的学习机制。

有监督的学习方法需要每个数据记录都有类标号，而无监督的学习方法则不考虑任何指导性信息。一般而言，一个监督学习模型需要大量的有标记数据集，而这些数据集是需要人工标注的。因此，也出现了很多**弱监督学习**和**半监督学习**的方法，希望从大规模的未标记数据中充分挖掘有用的信息，降低对标记数据数量的要求。

3.1.3 机器学习中的一些概念

上述的关于机器学习的介绍中，提及了一些基本概念，比如"数据"，"样本"，"特征"，"数据集"等。我们首先来解释下这些概念。

数据

在计算机科学中，**数据**是指所有能计算机程序处理的对象的总称，可以是数字、字母和符号等。在不同的任务中，表现形式不一样，比如图像、声音、文字、传感器数据等。

DRAFT 编译时间：2016-09-27 23:14

特征

机器学习中很多算法的输入要求是数学上可计算的。而在现实世界中，原始数据通常是并不都以连续变量或离散变量的形式存在的。我们首先需要将抽取出一些可以表征这些数据的数值型特征。这些数值型特征一般可以表示为向量形式，也称为**特征向量**。

特征学习

数据的原始表示转换为。原始数据的特征有很多，但是并不是所有的特征都是有用的。并且，很多特征通常是冗余并且易变的。我们需要抽取有效的、稳定的特征。传统的特征提取是通过人工方式进行的，这需要大量的人工和专家知识。即使这样，人工总结的特征在很多任务上也不能满足需要。因此，如何自动地学习有效的特征也成为机器学习中一个重要的研究内容，也就是**特征学习**，也叫**表示学习**。特征学习分成两种，一种是**特征选择**，是在很多特征集合选取有效的子集；另一种是**特征提取**，是构造一个新的特征空间，并将原始特征投影在新的空间中。

样本

样本是按照一定的抽样规则从全部数据中取出的一部分数据，是实际观测得到的数据。在有监督学习中，需要提供一组有输出目标的样本用来学习模型以及检验模型的好坏。

训练集和测试集

一组样本集合就称为**数据集**。在很多领域，数据集也经常称为**语料库**。为了检验机器学习算法的好坏，一般将数据集分为两部分：训练集和测试集。训练集用来进行模型学习，测试集用来进行模型验证。通过学习算法，在训练集得到一个模型，这个模型可以对测试集上样本x预测一个类别标签$\hat{y}$。假设测试集为T,模型的正确率为：

$$Acc = \frac{1}{|T|} \sum_{(x_i, y_i) \in T} |\hat{y}_i = y_i|, \tag{3.15}$$

其中$|T|$为测试集的大小。第**??**节中会介绍更多的评价方法。

DRAFT 编译时间：2016-09-27 23:14

正例和负例

对于两类分类问题，类别可以表示为$\{+1,-1\}$，或者直接用正负号表示。因此，常用**正例**和**负例**来分别表示属于不同类别的样本。

判别函数

经过特征抽取后，一个样本可以表示为k维特征空间中的一个点。为了对这个特征空间中的点进行区分，就需要寻找一些超平面来将这个特征空间分为一些互不重叠的子区域，使得不同类别的点分布在不同的子区域中，这些超平面就成为判别界面。

为了定义这些用来进行空间分割的超平面，就需要引入判别函数的概念。假设变量$\mathbf{z} \in \mathbb{R}^m$为特征空间中的点，这个超平面由所有满足函数$f(\mathbf{z}) = 0$的点组成。这里的$f(\mathbf{z})$就称为**判别函数**。

有了判别函数，分类就变得很简单，就是看一个样本在特征空间中位于哪个区域，从而确定这个样本的类别。

判别函数的形式多种多样，在自然语言处理中，最为常用的判别函数为线性函数。

3.1.4 参数学习算法

学习算法就是如何从训练集的样本中，自动学习决策函数的参数。不同机器学习算法的区别在于决策函数和学习算法的差异。相同的决策函数可以有不同的学习算法。比如线性分类器，其参数的学习算法可以是感知器、支持向量机以及梯度下降法等。通过一个学习算法进行自动学习参数的过程也叫作**训练过程**。

这里我们介绍一种常用的参数学习算法：**梯度下降法**（Gradient Descent Method）。

梯度下降法也叫最速下降法（Steepest Descend Method）。如果一个实值函数$f(\mathbf{x})$在点$\mathbf{a}$处可微且有定义，那么函数$f(\mathbf{x})$在$\mathbf{a}$点沿着梯度相反的方向$-\nabla f(\mathbf{a})$下降最快。梯度下降法经常用来求解无约束优化的极值问题。梯度下

DRAFT 编译时间：2016-09-27 23:14

降法的迭代公式为：

$$\mathbf{a}_{t+1} = \mathbf{a}_t - \lambda \nabla f(\mathbf{a_t}), \tag{3.16}$$

其中$\lambda > 0$是梯度方向上的搜索步长。

对于λ为一个够小数值时，那么 $f(\mathbf{a}_{k+1}) \leq f(\mathbf{a}_1)$。因此，我们可以从一个初始值$\mathbf{x}_0$开始，并通过迭代公式得到$\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_n$，并满足

$$f(\mathbf{x}_0) \geq f(\mathbf{x}_1) \geq f(\mathbf{x}_2) \geq \cdots \geq f(\mathbf{x}_n),$$

最终$\mathbf{x}_n$收敛到期望的极值。

搜索步长的取值必须合适，如果过大就不会收敛，如果过小则收敛速度太慢。一般步长可以由线性搜索算法来确定。

在机器学习问题中，我们需要学习到参数θ，使得风险函数最小化。

$$\theta^* = \arg\min_{\theta} \mathcal{R}(\theta_t) \tag{3.17}$$

$$= \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(y^{(i)}, f(x^{(i)}, \theta)\big). \tag{3.18}$$

如果用梯度下降法进行参数学习，

$$\mathbf{a}_{t+1} = \mathbf{a}_t - \lambda \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t} \tag{3.19}$$

$$= \mathbf{a}_t - \lambda \sum_{i=1}^{N} \frac{\partial \mathcal{R}\big(\theta_t; x^{(i)}, y^{(i)}\big)}{\partial \theta}, \tag{3.20}$$

搜索步长λ在机器学习中也叫作**学习率**（Learning Rate）。

这里，梯度下降是求得所有样本上的风险函数最小值，叫做**批量梯度下降法**。若样本个数N很大，输入$\mathbf{x}$的维数也很大时，那么批量梯度下降法每次迭代要处理所有的样本，效率会较低。为此，有一种改进的方法即**随机梯度下降法**。

随机梯度下降法（Stochastic Gradient Descent，SGD）也叫**增量梯度下降**，每个样本都进行更新

$$\mathbf{a}_{t+1} = \mathbf{a}_t - \lambda \frac{\partial \mathcal{R}\big(\theta_t; x^{(t)}, y^{(t)}\big)}{\partial \theta}, \tag{3.21}$$

DRAFT 编辑时间：2016-09-27 23:14

$x^{(t)}, y^{(t)}$ 是第 t 次迭代选取的样本。

批量梯度下降和随机梯度下降之间的区别在于每次迭代的风险是对所有样本汇总的风险还是单个样本的风险。随机梯度下降因为实现简单，收敛速度也非常快，因此使用非常广泛。

还有一种折中的方法就是**mini-batch随机梯度下降**，每次迭代时，只采用一小部分的训练样本，兼顾了批量梯度下降和随机梯度下降的优点。

Early-Stop

在梯度下降训练的过程中，由于过拟合的原因，在训练样本上收敛的参数，并不一定在测试集上最优。因此，我们使用一个**验证集**（Validation Dataset）（也叫**开发集**（Development Dataset））来测试每一次迭代的参数在验证集上是否最优。如果在验证集上的错误率不再下降，就停止迭代。这种策略叫Early-Stop。如果没有验证集，可以在训练集上进行**交叉验证**。

学习率设置

在梯度下降中，学习率的取值非常关键，如果过大就不会收敛，如果过小则收敛速度太慢。一般步长可以由线性搜索算法来确定。在机器学习中，经常使用自适应调整学习率的方法。

动量法 动量法（Momentum Method）[Rumelhart et al., 1988]对当前迭代的更新中加入上一次迭代的更新。我们记 $\nabla\theta_t = \theta_t - \theta_{t-1}$。在第 t 迭代时，

$$\theta_t = \theta_{t-1} + (\rho\nabla\theta_t - \lambda g_t), \tag{3.22}$$

其中，ρ 为动量因子，通常设为0.9。这样，在迭代初期，使用前一次的梯度进行加速。在迭代后期的收敛值附近，因为两次更新方向基本相反，增加稳定性。

AdaGrad AdaGrad（Adaptive Gradient）算法[Duchi et al., 2011]是借鉴L2正则化的思想。在第 t 迭代时，

$$\theta_t = \theta_{t-1} - \frac{\rho}{\sqrt{\sum_{\tau=1}^{t} g_\tau^2}} g_t, \tag{3.23}$$

DRAFT 编译时间：2016-09-27 23:14

其中，ρ是初始的学习率，$g_\tau \in \mathbb{R}^{|\theta|}$是第$\tau$次迭代时的梯度。

随着迭代次数的增加，梯度逐渐缩小。

AdaDelta AdaDelta算法 [Zeiler, 2012] 用指数衰减的移动平均来累积历史的梯度信息。第t次迭代的梯度的期望$E(g^2)_t$为：

$$E(g^2)_t = \rho E(g^2)_{t-1} + (1-\rho) g_t^2, \tag{3.24}$$

其中，ρ是衰减常数。

本次迭代的更新为

$$\nabla\theta_t = -\frac{\sqrt{E(\nabla\theta^2)_{t-1} + \epsilon}}{\sqrt{E(g^2)_t + \epsilon}} g_t \tag{3.25}$$

其中，$E(\nabla\theta^2)_t$为前一次迭代时$\nabla\theta^2$的移动平均，ϵ为常数。

累计更新本次迭代$\nabla\theta^2$的移动平均

$$E(\nabla\theta^2)_t = \rho E(\nabla\theta^2)_{t-1} + (1-\rho)\nabla\theta_t^2. \tag{3.26}$$

最后更新参数

$$\theta_t = \theta_{t-1} + \nabla\theta_t. \tag{3.27}$$

3.2 线性回归

如果输入$\mathbf{x}$是列向量，目标y是连续值（实数或连续整数），预测函数$f(\mathbf{x})$的输出也是连续值。这种机器学习问题是回归问题。

如果我们定义$f(\mathbf{x})$是线性函数，

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b, \tag{3.28}$$

这就是线性回归问题（Linear Regression）。

为了简单起见，我们将公式3.28写为

$$f(\mathbf{x}) = \hat{\mathbf{w}}^{\mathrm{T}}\hat{\mathbf{x}}, \tag{3.29}$$

其中$\hat{\mathbf{w}}$和$\hat{\mathbf{x}}$分别称为**增广权重向量**和**增广特征向量**。

DRAFT 编译时间：2016-09-27 23:14

$$\hat{\mathbf{x}} = \mathbf{x} \oplus 1 \triangleq \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \tag{3.30}$$

$$\hat{\mathbf{w}} = \mathbf{w} \oplus b \triangleq \begin{bmatrix} w_1 \\ \vdots \\ w_k \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}, \tag{3.31}$$

这里$\oplus$只两个向量的拼接。

在后面的描述中，我们一般采用简化的表示方法，直接用$\mathbf{w}$和$\mathbf{x}$来表示增广权重向量和增广特征向量。

平方损失参见第3.1.1节

线性回归的损失函数通常定义为平方损失函数。

$$L(y, f(\mathbf{x}, \mathbf{w})) = \|y - f(\mathbf{x}, \mathbf{w})\|^2. \tag{3.32}$$

给定N给样本$(x^{(i)}, y^{(i)}), 1 \leq i \leq N$，模型的经验风险为

$$R(Y, f(X, \mathbf{w})) = \sum_{i=1}^{N} L(y^{(i)}, f(\mathbf{x}^{(i)}, \mathbf{w})) \tag{3.33}$$

$$= \sum_{i=1}^{N} \|\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} - y^{(i)}\|^2 \tag{3.34}$$

$$= \|X^{\mathrm{T}}\mathbf{w} - \mathbf{y}\|^2, \tag{3.35}$$

其中，$\mathbf{y}$是一个目标值$y^{(i)}, \cdots, y^{(N)}$的列向量，X是所有输入组成的矩阵：

$$X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(N)} \\ \vdots & \vdots & \ddots & \vdots \\ x_k^{(1)} & x_k^{(2)} & \cdots & x_k^{(N)} \\ 1 & 1 & \cdots & 1 \end{pmatrix} \tag{3.36}$$

要最小化$R(Y, f(X, \mathbf{w}))$，我们要计算$R(Y, f(X, \mathbf{w}))$对$\mathbf{w}$的导数

$$\frac{\partial \mathcal{R}\big(Y, f(X, \mathbf{w})\big)}{\partial \mathbf{w}} = \frac{\partial \|X^{\mathrm{T}}\mathbf{w} - \mathbf{y}\|^2}{\partial \mathbf{w}} \tag{3.37}$$

DRAFT 编译时间：2016-09-27 23:14

$$= X(X^{\mathrm{T}}\mathbf{w} - \mathbf{y}). \tag{3.38}$$

让 $\frac{\partial \mathcal{R}(Y, f(X,\mathbf{w}))}{\partial \mathbf{w}} = 0$，则可以得到

$$\mathbf{w} = (XX^{\mathrm{T}})^{-1}X\mathbf{y} \tag{3.39}$$

$$= \big(\sum_{i=1}^{N}\mathbf{x}^{(i)}(\mathbf{x}^{(i)})^{\mathrm{T}}\big)^{-1}\big(\sum_{i=1}^{N}\mathbf{x}^{(i)}y^{(i)}\big). \tag{3.40}$$

这里要求 XX^T 是满秩的，存在逆矩阵，也就是要求 $\mathbf{x}$ 的每一维之间是非线性相关的。这样的参数求解方法也叫**最小二乘法估计**。

否则，就需要用梯度下降法来求解。初始化 $\mathbf{w}_0 = 0$，迭代公式为：

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda\frac{\partial \mathcal{R}\big(Y, f(X,\mathbf{w})\big)}{\partial \mathbf{w}}, \tag{3.41}$$

其中 λ 是学习率。

3.3 线性分类

线性分类是机器学习中最常见并且应用最广泛的一种分类器。

3.3.1 两类分类

首先对于两类分类问题，假设类别 $y \in \{0, 1\}$，线性分类函数为

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x} > 0 \\ 0 & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x} \leq 0 \end{cases} \tag{3.42}$$

$$= I(\mathbf{w}^{\mathrm{T}}\mathbf{x} > 0), \tag{3.43}$$

其中，$I()$ 是指示函数。

公式3.59定义了一个两类分类问题的线性判别函数。在高维的特征空间中，所有满足 $\mathbf{f}(z) = 0$ 的点组成用一个超平面，这个超平面将特征空间一分为二，划分成两个区域。这两个区域分别对应两个类别。

图3.2中给了一个两维数据的判别函数以及对应的判别界面。在二维空间中，分类界面为一个直线。在三维空间中，分类界面为一个平面。在高维空间中，分类界面为一个超平面。对于线性函数来说，权重向量在线性空间中垂直于分类界面的向量。

DRAFT 编译时间：2016-09-27 23:14

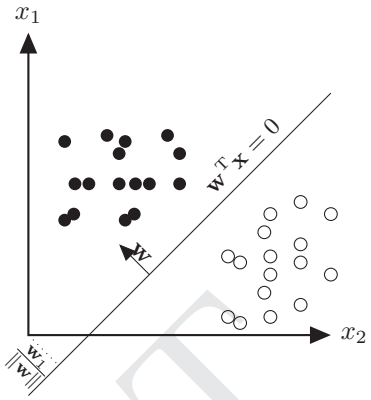

图 3.2: 两类分类线性判别函数

Logistic 回归

线性分类函数的参数 $\mathbf{w}$ 有很多种学习方式，比如第四章中介绍的感知器。这里使用另一种常用的学习算法：**Logistic 回归**。

我们定义目标类别 $y=1$ 的后验概率为：

$$P(y=1|\mathbf{x})=\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})=\frac{1}{1+\exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x})}, \tag{3.44}$$

其中，$\sigma(\cdot)$ 为logistic 函数，$\mathbf{x}$ 和 $\mathbf{w}$ 为增广的输入向量和权重向量。

$y=0$ 的后验概率为 $P(y=0|\mathbf{x})=1-P(y=1|\mathbf{x})$。

给定 N 给样本 $(x^{(i)},y^{(i)}),1\leq i\leq N$，我们使用交叉熵损失函数，模型在训练集的风险函数为：

$$\mathcal{J}(\mathbf{w})=-\sum_{i=1}^{N}\left(y^{(i)}\log\left(\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})\right)+(1-y^{(i)})\log\left(1-\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})\right)\right) \tag{3.45}$$

$$=-\sum_{i=1}^{N}\left(y^{(i)}\log\left(\frac{1}{1+\exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})}\right)+(1-y^{(i)})\log\left(1-\frac{1}{1+\exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})}\right)\right) \tag{3.46}$$

$$=-\sum_{i=1}^{N}\left(y^{(i)}\log\left(\frac{\exp(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})}{1+\exp(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})}\right)\right.$$

DRAFT 编译时间：2016-09-27 23:14

$$+(1-y^{(i)})\log\left(\frac{1}{1+\exp(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})}\right)\Big)\tag{3.47}$$

$$=-\sum_{i=1}^{N}\Big(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)}y^{(i)}-\log\Big(1+\exp(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})\Big)\Big)\tag{3.48}$$

$$\tag{3.49}$$

采样梯度下降法，$\mathcal{J}(\mathbf{w})$ 关于 $\mathbf{w}$ 的梯度为：

$$\frac{\partial\mathcal{J}(\mathbf{w})}{\partial\mathbf{w}}=-\sum_{i=1}^{N}\left(\mathbf{x}^{(i)}y^{(i)}-\mathbf{x}^{(i)}\cdot\exp(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})\cdot\frac{1}{1+\exp(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})}\right)\tag{3.50}$$

$$=-\sum_{i=1}^{N}\left(\mathbf{x}^{(i)}y^{(i)}-\mathbf{x}^{(i)}\cdot\frac{1}{1+\exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})}\right)\tag{3.51}$$

$$=-\sum_{i=1}^{N}\Big(\mathbf{x}^{(i)}\cdot\Big(y^{(i)}-\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})\Big)\Big)\tag{3.52}$$

$$=\sum_{i=1}^{N}\Big(\mathbf{x}^{(i)}\cdot\Big(\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)})-y^{(i)}\Big)\Big)\tag{3.53}$$

我们可以初始化 $\mathbf{w}_0=0$，然后用梯度下降法进行更新

$$\mathbf{w}_{t+1}=\mathbf{w}_t+\lambda\frac{\partial\mathcal{R}(\mathbf{w})}{\partial\mathbf{w}},\tag{3.54}$$

其中 λ 是学习率。

3.3.2 多类线性分类

对于多类分类问题（假设类别数为 $C(C>2)$），有很多种利用判别函数的方法。比较常用的是把多类分类问题转换为两类分类问题，在得到两类分类结果后，还需要通过投票方法进一步确定多类的分类结果。一般有两种多类转两类的转换方式：

1. 把多类分类问题转换为 C 个两类分类问题，构建 C 个一对多的分类器。每个两类分类问题都是把某一类和其他类用一个超平面分开。
2. 把多类分类问题转换为 $C(C-1)/2$ 个两类分类问题，构建 $C(C-1)/2$ 个两两分类器。每个两类分类问题都是把 C 类中某两类用一个超平面分开。

DRAFT 编译时间：2016-09-27 23:14

当对一个样本进行分类时，首先用多个两个分类器进行分类，然后进行投票，选择一个得分最高的类别。

但是上面两种转换方法都存在一个缺陷：空间中的存在一些区域，这些区域中点的类别是不能区分确定的。因为如果用多个两个分类器对这些点进行分类，然后进行投票时，会发现有两个或更多个类别的得分是一样的。

为了避免上述缺陷，可以使用一个更加有效的决策规则，直接建立多类线性分类器。假设 $y = \{1, \cdots, C\}$ 共 C 个类别，首先定义 C 个判别函数：

$$f_c(\mathbf{x}) = \mathbf{w}_c^{\mathrm{T}}\mathbf{x}, \qquad c = 1, \cdots, C, \tag{3.55}$$

这里 $\mathbf{w}_c$ 为类 c 的权重向量。

这样，对于空间中的一个点 $\mathbf{x}$，如果存在类别 c，对于所有的其他类别 $\tilde{c}(\mathbf{w}_c^T\mathbf{x} \neq c)$ 都满足 $\mathbf{f}_c(\mathbf{x}) > \mathbf{f}_{\tilde{c}}(\mathbf{x})$，那么 $\mathbf{x}$ 属于类别 c。相应的分类函数可以表示为：

$$\hat{y} = \arg\max_{c=1}^{C} \mathbf{w}_c^{\mathrm{T}}\mathbf{x} \tag{3.56}$$

当 $C = 2$ 时，

$$\begin{aligned} \hat{y} &= \arg\max_{y\in\{0,1\}} f_y(\mathbf{x}) \\ &= I(f_1(\mathbf{x}) - f_0(\mathbf{x}) > 0) \\ &= I(\mathbf{w}_1^{\mathrm{T}}\mathbf{x} - \mathbf{w}_0^{\mathrm{T}}\mathbf{x} > 0) \\ &= I((\mathbf{w}_1 - \mathbf{w}_0)^{\mathrm{T}}\mathbf{x} > 0) \end{aligned} \tag{3.57}$$

这里，$I()$ 是指示函数。对比公式3.59中的两类分类判别函数，可以发现 $\mathbf{w} = \mathbf{w}_1 - \mathbf{w}_0$。

SoftMax 回归

SoftMax 回归是 Logistic 回归的多类推广。

多类线性分类函数的参数 $\mathbf{w}$ 也有很多种学习方式。这里我们介绍一种常用的学习算法：**SoftMax 回归**。在 SoftMax 回归中，机器学习模型预测目标为每一个类别的后验概率。这就需要用到 **softmax** 函数。

利用 **softmax** 函数，我们定义目标类别 $y = c$ 的后验概率为：

$$P(y = c|\mathbf{x}) = \mathbf{softmax}(\mathbf{w}_c^{\mathrm{T}}\mathbf{x}) = \frac{\exp(\mathbf{w}_c^{\top}\mathbf{x})}{\sum_{i=1}^{C}\exp(\mathbf{w}_i^{\top}\mathbf{x})}. \tag{3.58}$$

DRAFT 编译时间：2016/9/27 23:14

对于样本$(\mathbf{x}, y)$，输出目标$y = \{1, \cdots, C\}$，我们用C维的one-hot向量$\mathbf{y}$来表示输出目标。对于类别c，

$$\mathbf{y} = [I(1 = c), I(2 = c), \cdots, I(C = c)]^{\mathrm{T}}, \tag{3.59}$$

这里，$I()$是指示函数。

同时，我们将公式3.58重新定义一下，直接输出k维向量。

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{softmax}(W^{\mathrm{T}}\mathbf{x}) \\ &= \frac{\exp(W^{\mathrm{T}}\mathbf{x})}{\underline{1}^{\mathrm{T}} \exp\left((W^{\mathrm{T}}\mathbf{x})\right)} \\ &= \frac{\exp(\mathbf{z})}{\underline{1}^{\mathrm{T}} \exp\left(\mathbf{z}\right)},\end{aligned} \tag{3.60}$$

其中，$W = [\mathbf{w}_1, \cdots, \mathbf{w}_C]$是$C$个类对应权重向量组成的矩阵。$\hat{\mathbf{y}}$的第$c$维的值是第$c$类的预测后验概率。其中，$\hat{\mathbf{z}} = W^{\mathrm{T}}\mathbf{x}$，为**softmax**函数的输入向量。

给定N给样本$(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}), 1 \leq i \leq N$，我们使用交叉熵损失函数，模型在训练集的风险函数为：

$$\begin{aligned}\mathcal{J}(W) &= -\sum_{i=1}^{N}\sum_{c=1}^{C} \mathbf{y}_c^{(i)} \log \hat{\mathbf{y}}_c^{(i)} \\ &= -\sum_{i=1}^{N} (\mathbf{y}^{(i)})^{\mathrm{T}} \log \hat{\mathbf{y}}^{(i)} \\ &= -\sum_{i=1}^{N} (\mathbf{y}^{(i)})^{\mathrm{T}} \log \left(\mathbf{softmax}(\mathbf{z}^{(i)})\right) \\ &= -\sum_{i=1}^{N} (\mathbf{y}^{(i)})^{\mathrm{T}} \log \left(\mathbf{softmax}(W^{\mathrm{T}}\mathbf{x}^{(i)})\right).\end{aligned} \tag{3.61}$$

采样梯度下降法，我们要计算$\mathcal{J}(W)$关于W的梯度。首先，我们列下要用到的公式。

（1）softmax函数的导数为

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial\, \mathbf{softmax}(\mathbf{x})}{\partial \mathbf{x}} \tag{3.62}$$

$$= \mathbf{diag}\left(\mathbf{softmax}(\mathbf{x})\right) - \mathbf{softmax}(\mathbf{x})\, \mathbf{softmax}(\mathbf{x})^{\top} \tag{3.63}$$

$$= \mathbf{diag}\left(\mathbf{y}\right) - \mathbf{y}\mathbf{y}^{\top}. \tag{3.64}$$

DRAFT 编译时间：2016-09-27 23:14

（2）$\mathbf{z} = W^{\mathrm{T}}\mathbf{x}$，则

$$\frac{\partial \mathbf{z}}{\partial \mathbf{w}_c} = M(\mathbf{x}, c), \tag{3.65}$$

$M(\mathbf{x}, c)$ 为第 c 列为 $\mathbf{x}$，其余为 0 的矩阵。

（3）$\mathbf{x}^{\top}\,\mathbf{diag}(\mathbf{x})^{-1} = \underset{\sim}{1}_K^{\top}$

（4）因为 $\mathbf{y}$ 为 onehot 向量，所以 $\underset{\sim}{1}_K^{\top}\mathbf{y} = 1$

采样梯度下降法，$\mathcal{J}(W)$ 关于 $\mathbf{w}_c$ 的梯度为：

$$\begin{aligned}
\frac{\partial \mathcal{J}(W)}{\partial \mathbf{w}_c} &= -\sum_{i=1}^{N} \frac{\left((\mathbf{y}^{(i)})^{\mathrm{T}} \log\left(\hat{\mathbf{y}}^{(i)}\right)\right)}{\partial \mathbf{w}_c} \\
&= -\sum_{i=1}^{N} \frac{\partial \mathbf{z}^{(i)}}{\partial \mathbf{w}_c} \frac{\partial \hat{\mathbf{y}}^{(i)}}{\partial \mathbf{z}^{(i)}} \frac{\partial \log \hat{\mathbf{y}}^{(i)}}{\partial \hat{\mathbf{y}}^{(i)}} \mathbf{y}^{(i)} \\
&= -\sum_{i=1}^{N} \frac{\partial \mathbf{z}^{(i)}}{\partial \mathbf{w}_c} \frac{\partial \mathbf{softmax}(\mathbf{z}^{(i)})}{\partial \mathbf{z}^{(i)}} \frac{\partial \log \hat{\mathbf{y}}^{(i)}}{\partial \hat{\mathbf{y}}^{(i)}} \mathbf{y}^{(i)} \\
&= -\sum_{i=1}^{N} M(\mathbf{x}^{(i)}, c) \left(\mathbf{diag}\left(\hat{\mathbf{y}}^{(i)}\right) - \hat{\mathbf{y}}^{(i)}(\hat{\mathbf{y}}^{(i)})^{\top}\right) \left(\mathbf{diag}(\hat{\mathbf{y}}^{(i)})\right)^{-1} \mathbf{y}^{(i)} \\
&= -\sum_{i=1}^{N} M(\mathbf{x}^{(i)}, c) \left(I - \hat{\mathbf{y}}^{(i)} \underset{\sim}{1}_K^{\top}\right) \mathbf{y}^{(i)} \\
&= -\sum_{i=1}^{N} M(\mathbf{x}^{(i)}, c) \left((\mathbf{y}^{(i)}) - \hat{\mathbf{y}}^{(i)} \underset{\sim}{1}_K^{\top} \mathbf{y}^{(i)}\right) \\
&= -\sum_{i=1}^{N} M(\mathbf{x}^{(i)}, c) \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\right) \\
&= -\sum_{i=1}^{N} \mathbf{x}^{(i)} \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\right)_c
\end{aligned} \tag{3.66}$$

如果采用 $y \in \{1, \cdots, C\}$ 的离散表示形式，梯度公式也可以写为：

$$\frac{\partial \mathcal{J}(W)}{\partial \mathbf{w}_c} = -\frac{1}{N} \sum_{i=1}^{C} \mathbf{x}^{(i)} \left(I(y^{(i)} = c) - p(y^{(i)} = c|\mathbf{x}^{(i)}; W)\right), \tag{3.67}$$

其中，$I()$ 为指示函数。

DRAFT 公开时间：2016-09-27 23:14

根据公式3.3.2，我们可以得到

$$\frac{\partial \mathcal{J}(W)}{\partial W} = -\sum_{i=1}^{N} \mathbf{x}^{(i)} \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\right)^{\top} \tag{3.68}$$

我们可以初始化$W=0$，然后用梯度下降法进行更新

$$W_{t+1} = W_t + \lambda - \sum_{i=1}^{N} \mathbf{x}^{(i)} \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)}\right)^{\top}, \tag{3.69}$$

其中λ是学习率。

3.4 评价方法

为了衡量一个分类算法好坏，需要给定一个测试集，用分类器对测试集中的每一个样本进行分类，并根据分类结果计算评价分数。常见的评价标准有正确率、准确率、召回率和F 值等。

给定测试集$T=(x_1,y_1),\cdots,(x_N,y_N)$，对于所有的$y_i \in \{\omega_1,\cdots,\omega_C\}$。假设分类结果为$Y=\hat{y_1},\cdots,\hat{y_N}$。

则**正确率**（Accuracy，Correct Rate）为：

$$Acc = \frac{\sum_{i=1}^{N} |y_i = \hat{y}_i|}{N} \tag{3.70}$$

其中，$|\cdot|$为指示函数

和正确率相对应的就是**错误率**（Error Rate）。

$$Err = \frac{\sum_{i=1}^{N} |y_i \neq \hat{y}_i|}{N} \tag{3.71}$$

正确率是平均的整体性能。

在很多情况下，我们需要对每个类都进行性能估计，这就需要计算准确率和召回率。正确率和召回率是广泛用于信息检索和统计学分类领域的两个度量值，在机器学习的评价中也被大量使用。

DRAFT 编译时间：2016-09-27 23:14

准确率（Precision，P），也叫查准率，精确率或精度，是识别出的个体总数中正确识别的个体总数的比例。对于类c来说，

$$P_c = \frac{\sum_{\substack{i=1 \\ \hat{y}_i=c}}^{N} |y_i = \hat{y}_i|}{\sum_{\substack{i=1 \\ \hat{y}_i=c}}^{N} 1} \tag{3.72}$$

召回率（Recall，R），也叫查全率，是测试集中存在的个体总数中正确识别的个体总数的比例。

$$R_c = \frac{\sum_{\substack{i=1 \\ y_i=c}}^{N} |y_i = \hat{y}_i|}{\sum_{\substack{i=1 \\ y_i=c}}^{N} 1} \tag{3.73}$$

F1值是根据正确率和召回率二者给出的一个综合的评价指标，具体定义如下：

$$F1_c = \frac{P_c * R_c * 2}{(P_c + R_c)} \tag{3.74}$$

为了计算分类算法在整个数据集上的总体准确率、召回率和F1值，经常使用两种平均方法，分别称为**宏平均**（macro average）和**微平均**（micro average）。

宏平均是每一个类的性能指标的算术平均值，

$$R_{macro} = \sum_{i=1}^{C} R_c/C, \tag{3.75}$$

$$P_{macro} = \sum_{i=1}^{C} P_c/C, \tag{3.76}$$

$$F1_{macro} = \frac{P_{macro} * R_{macro} * 2}{(P_{macro} + R_{macro})}. \tag{3.77}$$

而微平均是每一个样本的性能指标的算术平均。对于单个样本而言，它的准确率和召回率是相同的（要么都是1，要么都是0）因此准确率和召回率的微平均是相同的，根据F1 值公式，对于同一个数据集它的准确率、召回率和F1的微平均指标是相同的。

DRAFT 编译时间：2016/9/27 23:14

3.5 定理

在机器学习中，有一些非常有名的定理。这些定理对理解机器学习的内在特性非常有帮助。

3.5.1 没有免费午餐定理

没有免费午餐定理(No Free Lunch Theorem，NFL) 是由 Wolpert 和 Macerday 在最优化理论中提出的。没有免费午餐定理证明：对于基于迭代的最优化算法，不存在某种算法对所有问题（有限的搜索空间内）都有效。如果一个算法对某些问题有效，那么它一定在另外一些问题上比纯随机搜索算法更差。也就是说，不能脱离具体问题来谈论算法的优劣，任何算法都有局限性。必须要"具体问题具体分析"。

没有免费午餐定理对于机器学习算法也同样适用。不存在一种机器学习算法适合于任何领域或任务。如果有人宣称自己的模型在所有问题上都好于其他模型，那么他肯定是在吹牛。

3.5.2 丑小鸭定理

丑小鸭定理（Ugly Duckling）1960 年代，，美籍日本学者渡边慧提出了丑小鸭定理："丑小鸭与白天鹅之间的区别和两只白天鹅之间的区别一样大"。这个定理初看好像不符合常识，但是仔细思考后是非常有道理的。因为世界上不存在相似性的客观标准，一切相似性的标准都是主观的。如果以体型大小的角度来看，丑小鸭和白天鹅的区别大于两只白天鹅的区别；但是如果以基因的角度来看，丑小鸭与它父母的差别要小于他父母和其他白天鹅之间的差别。

渡边慧，模式识别研究的鼻祖之一。

这里的"丑小鸭"是指白天鹅的幼雏，而不是"丑陋的小鸭子"。

3.6 总结和深入阅读

本章简单地介绍了机器学习的理论知识，主要为后面讲解人工神经网络铺垫一些基础知识。如果需要快速全面地了解机器学习的基本概念可以阅读《Pattern Classification》[Duda et al., 2001] 和《Pattern Recognition and Machine

DRAFT 版本时间：2016-09-27 23:14

Learning》[Bishop, 2006]，进一步深入了解可以阅读《The Elements of Statistical Learning》[Hastie et al., 2001] 以及《Learning in Graphical Models》[Jordan, 1998]。

DRAFT
编译时间：2016-09-27 23:14

参考文献

James A Anderson and Edward Rosenfeld. *Talking nets: An oral history of neural networks*. MiT Press, 2000.

Yoshua Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

Yoshua Bengio, Jean-Sébastien Senécal, et al. Quick training of probabilistic neural nets by importance sampling. In *AISTATS Conference*, 2003.

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8624–8628. IEEE, 2013.

C.M. Bishop. *Pattern recognition and machine learning*. Springer New York., 2006.

P.F. Brown, P.V. Desouza, R.L. Mercer, V.J.D. Pietra, and J.C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4): 467–479, 1992.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

DRAFT 编译时间：2016/09/27 23:14

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, 2002.

Hal Daumé III. A course in machine learning. http://ciml.info/. [Online].

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, New York, 2nd edition, 2001. ISBN 0471056693.

Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. *Advances in Neural Information Processing Systems*, pages 472–478, 2001.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

Ian Goodfellow, Aaron Courville, and Yoshua Bengio. Deep learning. Book in preparation for MIT Press, 2015. URL http://goodfeli.github.io/dlbook/.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

DRAFT 编译时间：2016/9/27 23:14

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

M.I. Jordan. *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464. Association for Computational Linguistics, 2010.

Marvin Minsky and Papert Seymour. Perceptrons. 1969.

Marvin L Minsky and Seymour A Papert. *Perceptrons - Expanded Edition: An Introduction to Computational Geometry*. MIT press Boston, MA:, 1987.

T.M. Mitchell. *Machine learning*. Burr Ridge, IL: McGraw Hill, 1997.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

Albert BJ Novikoff. On convergence proofs for perceptrons. Technical report, DTIC Document, 1963.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958.

DRAFT 编译时间：2016-09-27 23:14

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(3):328–339, 1989.

Paul Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. 1974.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

DE Rumelhart GE Hinton RJ Williams and GE Hinton. Learning representations by back-propagating errors. *Nature*, pages 323–533, 1986.

Matthew D Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

DRAFT 编译时间：2016/9/27 23:14