的综合和交互与目标分类类别之间的关系更紧密。有时多个特征之间是相关的，使用特征的交互往往能建立更有效的模型。例如，对年龄、职业和种族这三个特征，我们可以使用crossed_column()建立交叉特征列：

```
combined = layers.crossed_column([age_range, race, occupation],
                                 hash_bucket_size=int(1e7))
```

建立好各式各样的特征列之后，我们可以直接将它们传入不同的 TF.Learn Estimator。以下面这个简单的逻辑回归分类模型为例，我们可以使用之前介绍过的 fit()、predict()等方法训练和评估模型。

```
classifier = tf.contrib.learn.LinearClassifier(feature_columns=[
    gender, education, occupation, combined, age_range, race, income,
    spending], model_dir=model_dir)
```

这里我们只是简单地介绍了一些比较常用的函数，在实际应用中有各种各样的需求，例如有时想取一部分特征的加权求和作为一个新的特征列，可以使用weighted_sum_from_feature_columns()来很快地实现。读者可以在官方文档里找到更多需要的函数。

11.2.4 Embeddings

在许多深度模型应用中，包含许多稀疏的、高维的类别特征向量，我们通常先把它们转换成低维的、稠密的实数值的向量，也通常将它们和连续特征向量联合起来，一起输入进神经网络模型中进行训练和优化损失函数，这些被统一称为嵌入向量（Embedding Vectors）。大部分文本识别都是先将文本转换成嵌入向量，然后对它们进行分析并用在模型训练中。

contrib.layers 模块里的 embedding_column()能迅速将高维稀疏的类别特征向量转换为读者想要的维数的嵌入向量，以下是一个例子。

```
embedding_columns = [
    tf.contrib.layers.embedding_column(title, dimension=8),
    tf.contrib.layers.embedding_column(education, dimension=8),
    tf.contrib.layers.embedding_column(gender, dimension=8),
    tf.contrib.layers.embedding_column(race, dimension=8),
    tf.contrib.layers.embedding_column(country, dimension=8)]
```

这里的 title、education、gender、race，以及 country 都是比较稀疏的类别特征向量，我们通过使用 embedding_column()，把它们转换为低维数的稠密向量，从而更好地归纳数据中的特性，特别是当一组特征中的交互矩阵比较稀疏，级别比较高时，这种方法会使模型更具有概括性且更有效。

接下来，可以直接将它们传入 TF.Learn 的 Estimator 里进行模型的建立、训练，以及评估。例如，可以将 embedding_columns 传入 DNNLinearCombinedClassifier 里的深度神经网络特征列里。

```
est = tf.contrib.learn.DNNLinearCombinedClassifier(
          model_dir=model_dir,
          linear_feature_columns=wide_columns,
          dnn_feature_columns=embedding_columns,
          dnn_hidden_units=[100, 50])
```

embedding_columns()是 contrib.layers 模块里最简单易用的一个，当涉及实际数据时，许多稀疏高维的数据里通常有空的特征及无效的 ID，这时可以使用 safe_embedding_lookup_sparse()安全地建立嵌入向量。这里先用 tf.SparseTensor 建立好稀疏的 ID 及稀疏的权重。

```
indices = [[0, 0], [0, 1], [0, 2], [1, 0], [3, 0], [4, 0], [4, 1]]
ids = [0, 1, -1, -1, 2, 0, 1]
weights = [1.0, 2.0, 1.0, 1.0, 3.0, 0.0, -0.5]
shape = [5, 4]

sparse_ids = tf.SparseTensor(
    tf.constant(indices, tf.int64), tf.constant(ids, tf.int64),
    tf.constant(shape, tf.int64))

sparse_weights = tf.SparseTensor(
    tf.constant(indices, tf.int64), tf.constant(weights, tf.float32),
    tf.constant(shape, tf.int64))
```

接下来，建立嵌入向量的权重 embedding_weights，这取决于词汇量大小、嵌入向量维数，以及 shard 数量。然后，使用 initializer.run()和 eval()初始化嵌入向量的权重，具体

细节和参数的说明请参考最新的官方文档。

```
vocab_size=4
embed_dim=4
num_shards=1
embedding_weights = tf.create_partitioned_variables(
     shape=[vocab_size, embed_dim],
     slicing=[num_shards, 1],
     initializer=tf.truncated_normal_initializer(mean=0.0,
                     stddev=1.0 / math.sqrt(vocab_size), dtype=tf.float32))
for w in embedding_weights:
    w.initializer.run()
embedding_weights = [w.eval() for w in embedding_weights]
```

最后，可以使用 safe_embedding_lookup_sparse()将原来的特征向量安全地转换为低维和稠密的特征向量，这里使用 eval()，然后将它们收集到一个 tuple 里。

```
embedding_lookup_result = (tf.contrib.layers.safe_embedding_lookup_sparse(
    embedding_weights, sparse_ids, sparse_weights).eval())
```

11.3 性能分析器 tfprof

TensorFlow 也在 Contrib 模块里提供了自己的性能分析器 tfprof，可以通过它帮助分析模型的架构及衡量系统的性能。它涵盖了许多实用的功能，例如衡量模型的参数、浮点运算、op 执行时间、要求的存储大小、探索模型的结构等。本节将简单地介绍一些功能。

首先，通过以下命令安装 tfprof 命令行的工具。

```
bazel build -c opt tensorflow/contrib/tfprof/...
```

可以通过以下命令查询帮助文件。

```
bazel-bin/tensorflow/contrib/tfprof/tools/tfprof/tfprof help
```

可以执行互动模式，然后指定 graph_path 来分析模型的 shape 和参数。

```
bazel-bin/tensorflow/contrib/tfprof/tools/tfprof/tfprof \
    --graph_path=/graph.pbtxt
```

类似地，我们用 graph_path 和 checkpoint_path 查看 checkpoint 里 Tensor 的数据和相对应的值。

```
bazel-bin/tensorflow/contrib/tfprof/tools/tfprof/tfprof \
    --graph_path=graph.pbtxt \
    --checkpoint_path=model.ckpt
```

与此同时，我们可以多提供一个 run_meta_path 来查看不同 op 请求的存储和计时。

```
bazel-bin/tensorflow/contrib/tfprof/tools/tfprof/tfprof \
    --graph_path=graph.pbtxt \
    --run_meta_path=run_meta \
    --checkpoint_path=model.ckpt
```

值得注意的是，上面用到了 run_meta_path、graph_path 和 checkpoint_path 几个路径，我们是怎么得到这几种类型的文件的呢？

graph_path 的文件是 GraphDef 文本文件，用来在内存里建立模型的代表，例如用 tf.Supervisor 写出来的 graph.pbtxt 就是一个 GraphDef 文本文件的例子。如果不使用 tf.Supervisor，那么可以使用 tf.Graph.as_graph_def()或者其他类似的 API 存储模型的定义到一个 GraphDef 文件里。

run_meta_path 所需的文件是 tensorflow::RunMetadata 的结果，这个方程是用来得到模型中每个 op 所需的存储和时间消耗的，以下简单的几行代码可以写出一个 RunMetadata 文件。

```
run_options = config_pb2.RunOptions(
    trace_level=config_pb2.RunOptions.FULL_TRACE)
run_metadata = config_pb2.RunMetadata()
_ = self._sess.run(..., options=run_options, run_metadata=run_metadata)
with gfile.Open(os.path.join(output_dir, "run_meta"), "w") as f:
    f.write(run_metadata.SerializeToString())
```

checkpoint_path 是模型的 checkpoint，它包含了所有 checkpoint 的变量的 op 类型、shape 和它们的值。

读者也可以提供其他的路径，例如 op_log_path 路径，它是 tensorflow::tfprof::OpLog

的结果，包含了额外的 op 的信息，由于它包含了 op 的组的类别名字，用户可以很简单地综合 op 的一些数据，而不会不小心错过其中一部分 op。以下用一个暴露出来的 API 来很快地写出了一个 OpLog 文件。

```
tf.contrib.tfprof.tfprof_logger.write_op_log(graph, log_dir, op_log=None)
```

由于 tfprof 是一个 CLI 命令行的工具，当输入之前的 tfprof 命令按下回车键时，会进入互动模式，再按一下回车键会看到一些类似以下的命令行参数的默认值。

```
tfprof>
-max_depth                  4
-min_bytes                  0
-min_micros                 0
-min_params                 0
-min_float_ops              0
-device_regexes             .*
-order_by                   name
-account_type_regexes       Variable
-start_name_regexes         .*
-trim_name_regexes
-show_name_regexes          .*
-hide_name_regexes          VariableInitialized_[0-9]+,save\/.*,^zeros[0-9_]*
-account_displayed_op_only false
-select                     params
-viz                        false
-dump_to_file
```

然后就可以调节里面的参数，例如用 show_name_regexes 查找 scope 名字正则式为 unit_1_0.*gamma，max_depth 为 5 的变量，查看这些 tensor 的值：

```
tfprof> scope -show_name_regexes unit_1_0.*gamma -select tensor_value \
    -max_depth 5
```

读者会得到类似以下符合条件的 tensor 的值：

```
unit_1_0/shared_activation/init_bn/gamma ()
[1.80 2.10 2.06 1.91 2.26 1.86 1.81 1.37 1.78 1.85 1.96 1.54 2.04 2.34 2.22
```

```
1.99 ],
unit_1_0/sub2/bn2/gamma ()
[1.57 1.83 1.30 1.25 1.59 1.14 1.26 0.82 1.19 1.10 1.48 1.01 0.82 1.23 1.21
1.14 ]
```

tfprof 提供了两种类型的分析：scope 和 graph。当读者想查看一些变量和 scope 的值的时候，你可以使用 scope，也就是我们上面阐述过的例子。当读者想查看 op 在 graph 里所花的内存和时间时，可以使用 graph 查看。

类似地，可以改动一些命令行参数，例如使用 start_name_regexes 选择想要查看的 op 的名字，这里我们假设需要查看命名为 cost 的损失 op 的内存和时间花费的情况：

```
tfprof> graph -start_name_regexes cost.* -max_depth 100 -min_micros 10000 \
    -select micros -account_type_regexes .*

init/init_conv/Conv2D (11.75ms/3.10sec)
random_shuffle_queue_DequeueMany (3.09sec/3.09sec)
unit_1_0/sub2/conv2/Conv2D (74.14ms/3.19sec)
unit_1_3/sub2/conv2/Conv2D (60.75ms/3.34sec)
unit_2_4/sub2/conv2/Conv2D (73.58ms/3.54sec)
unit_3_3/sub2/conv2/Conv2D (10.26ms/3.60sec)
```

我们就先简单地介绍以上这些模块，这一部分变化很大，更多的功能还需要读者自己摸索并查看官方文档。

参考文献

1. **Large Scale Distributed Deep Networks**, *NIPS 2012*, Authors: Jeffrey Dean and Greg S. Corrado and Rajat Monga and Kai Chen and Matthieu Devin and Quoc V. Le and Mark Z. Mao and Marc'Aurelio Ranzato and Andrew Senior and Paul Tucker and Ke Yang and Andrew Y. Ng

2. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**, *http://tensorflow.org/*, Authors: Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng

3. http://eigen.tuxfamily.org/

4. http://www.netlib.org/blas/

5. https://developer.nvidia.com/cublas

6. https://github.com/akrizhevsky/cuda-convnet2

7. https://developer.nvidia.com/cudnn

8. **Caffe: Convolutional Architecture for Fast Feature Embedding**, *https://arxiv.org/abs/1408.5093*, Authors: Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell

9. https://keras.io/

10. **An Introduction to Computational Networks and the Computational Network Toolkit**, *Microsoft Research*, Authors: Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Oleksii Kuchaiev, Yu Zhang, Frank Seide, Zhiheng Huang, Brian Guenter, Huaming Wang, Jasha Droppo, Geoffrey Zweig, Chris Rossbach, Jie Gao, Andreas Stolcke, Jon Currey, Malcolm Slaney, Guoguo Chen, Amit Agarwal, Chris Basoglu, Marko Padmilac, Alexey Kamenev, Vladimir Ivanov, Scott Cypher, Hari Parthasarathi, Bhaskar Mitra, Baolin Peng, Xuedong Huang

11. https://github.com/torch/torch7

12. **MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems**, *NIPS 2015*, Authors: Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang

13. https://github.com/autumnai/leaf

14. http://deeplearning.net/software/theano/

15. https://deeplearning4j.org/

16. https://github.com/Lasagne/Lasagne

17. https://github.com/NervanaSystems/neon

18. https://developer.nvidia.com/cuda-zone

19. https://tensorflow.github.io/serving/

20. https://www.continuum.io/downloads

21. http://yann.lecun.com/exdb/mnist/

22. https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/mnist/mnist_softmax.py

23. **Reducing the dimensionality of data with neural networks**, *Science*, Authors: G. E. Hinton and R. R. Salakhutdinov

24. **Deep belief networks**, *Scholarpedia*, Geoffrey E. Hinton

25. https://github.com/tensorflow/models/blob/master/autoencoder/AdditiveGaussianNoiseAutoencoderRunner.py

26. **Understanding the difficulty of training deep feedforward neural networks**, *AISTATS 2010*, Authors: Xavier Glorot and Yoshua Bengio

27. **Dropout: A Simple Way to Prevent Neural Networks from Overfitting**, *Journal of Machine Learning Research*, Authors: Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov

28. **Adaptive Subgradient Methods for Online Learning and Stochastic Optimization**, *Journal of Machine Learning Research*, Authors: John Duchi, Elad Hazan, Yoram Singer

29. **Digital selection and analogue amplification coesist in a cortex-inspired silicon circuit**, *Nature*, Authors: Hahnloser, R. Sarpeshkar, M A Mahowald, R. J. Douglas, H.S. Seung

30. **ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION**, *ICLR 2015*, Authors: Diederik P. Kingma and Jimmy Lei Ba

31. **ADADELTA: AN ADAPTIVE LEARNING RATE METHOD**, *https://arxiv.org/abs/1212.5701*, Authors: Matthew D. Zeiler

32. **On the momentum term in gradient descent learning algorithms**, *Neural Networks : The Official Journal of the International Neural Network*, Authors: Ning Qian

33. **A method for unconstrained convex minimization problem with the rate of**

convergence o(1/k2), *Doklady ANSSSR*, Authors: Nesterov, Y.

34. **Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)**, *ICLR 2016*, Authors: Djork-Arné Clevert, Thomas Unterthiner, Sepp Hochreiter

35. **Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification**, *https://arxiv.org/abs/1502.01852*, Authors: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

36. **Empirical Evaluation of Rectified Activations in Convolutional Network**, *https://arxiv.org/abs/1505.00853*, Authors: Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li

37. **Observations on the scratch-reflex in the spinal dog**, *Journal of Physiology*, Authors: Sherrington, C. S.

38. **Neocognitron: a neural network model for a mechanism of visual pattern recognition**, *IEEE Transactions on Systems, Man, and Cybernetics*, Authors: Fukushima, K.; Miyake, S.; Ito, T.

39. **Gradient-based learning applied to document recognition**, *Proceedings of the IEEE*, Authors: Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner

40. **ImageNet Classification with Deep Convolutional Neural Networks**, *NIPS 2012*, Authors: Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

41. https://github.com/tensorflow/tensorflow/blob/master/tensorflow/g3doc/tutorials/mnist/pros/index.md

42. **Learning Multiple Layers of Features from Tiny Images**, *https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf*, Authors: Alex Krizhevsky

43. http://groups.csail.mit.edu/vision/TinyImages/

44. https://github.com/tensorflow/models/tree/master/tutorials/image/cifar10

45. **ImageNet Classification with Deep Convolutional Neural Networks**, *NIPS 2012*, Authors: Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

46. **VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION**, *ICLR 2015*, Authros: Karen Simonyan and Andrew Zisserman

47. **Going Deeper with Convolutions**, *CVPR 2015*, Authors: Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

48. **Deep Residual Learning for Image Recognition**, *https://arxiv.org/abs/1512.03385*, Authors: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

49. **ImageNet Large Scale Visual Recognition Challenge**, *IJCV 2015*, Authors: Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei

50. **ImageNet: A Large-Scale Hierarchical Image Database**, *CVPR 2009*, Authors: J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei

51. https://github.com/tensorflow/models/blob/master/tutorials/image/alexnet/alexnet_benchmark.py

52. https://github.com/machrisaa/tensorflow-vgg

53. https://github.com/tensorflow/models/blob/master/slim/nets/inception_v3.py

54. https://github.com/tensorflow/models/blob/master/slim/nets/resnet_v2.py

55. **Efficient Estimation of Word Representations in Vector Space**, *https://arxiv.org/abs/1301.3781*, Authors: Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean

56. https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/word2vec/word2vec_basic.py

57. **Long short-term memory**, *https://dx.doi.org/10.1162%2Fneco.1997.9.8.1735*, Authors: Sepp Hochreiter, Jürgen Schmidhuber

58. https://github.com/tensorflow/models/blob/master/tutorials/rnn/ptb/ptb_word_lm.py

59. **Bidirectional recurrent neural networks**, *IEEE Transactions on Signal Processing*,

Authors: Mike Schuster and Kuldip K. Paliwal

60. https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/3_NeuralNetworks/bidirectional_rnn.py

61. **Human-level control through Deep Reinforcement Learning**, *Nature*, Authors: Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, Demis Hassabis

62. **Mastering the game of Go with deep neural networks and tree search**, *Nature*, Authors: David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner,Ilya Sutskever,Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, Demis Hassabis

63. **OpenAI Gym**, *http://arxiv.org/abs/1606.01540*, Greg Brockman and Vicki Cheung and Ludwig Pettersson and Jonas Schneider and John Schulman and Jie Tang and Wojciech Zaremba

64. https://github.com/awjuliani/DeepRL-Agents/blob/master/Policy-Network.ipynb

65. **Learning from Delayed Rewards**, *Ph.D. thesis, Cambridge University*, Authors: Watkins, C.J.C.H.

66. https://github.com/awjuliani/DeepRL-Agents/blob/master/Double-Dueling-DQN.ipynb

67. https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/mnist/mnist_with_summaries.py

68. https://github.com/tensorflow/models/blob/master/tutorials/image/cifar10/cifar10_multi_gpu_train.py

69. https://github.com/tensorflow/tensorflow/blob/master/tensorflow/tools/dist_test/python/mnist_replica.py

作者简介

·黄文坚·

PPmoney大数据算法总监，负责集团的风控、理财、互联网证券等业务的数据挖掘工作。Google TensorFlow Contributor。前明略数据技术合伙人，领导了对诸多大型银行、保险公司、基金的数据挖掘项目，包括建立金融风控模型、新闻舆情分析、保险复购预测等。曾就职于阿里巴巴搜索引擎算法团队，负责天猫个性化搜索系统。曾参加阿里巴巴大数据推荐算法大赛，于7000多支队伍中获得前10名。本科、研究生就读于香港科技大学，曾在顶级会议和期刊*SIGMOBILE MobiCom*、*IEEE Transactions on Image Processing*发表论文，研究成果获美国计算机协会移动计算大会（MobiCom）最佳移动应用技术冠军，并获得两项美国专利和一项中国专利。

·唐　源·

目前在芝加哥的Uptake公司带领团队建立用于多个物联网领域的数据科学引擎进行条件和健康监控，也建立了公司的预测模型引擎，现在被用于航空、能源等大型机械领域。一直活跃在开源软件社区，是TensorFlow和DMLC的成员，是TensorFlow、XGBoost、MXNet等软件的committer，TF.Learn、ggfortify等软件的作者，以及caret、pandas等软件的贡献者。曾获得谷歌Open Source Peer Bonus，以及多项高校和企业编程竞赛的奖项。在美国宾州州立大学获得荣誉数学学位，曾在本科学习期间成为创业公司DataNovo的核心创始成员，研究专利数据挖掘、无关键字现有技术搜索、策略推荐等。

博文视点Broadview

新浪微博 weibo.com
@博文视点Broadview

策划编辑：郑柳洁　欢迎投稿
责任编辑：郑柳洁　邮箱：zhenglj@phei.com.cn
封面设计：侯士卿　微信号：Alinamercy

上架建议：人工智能＞深度学习

ISBN 978-7-121-30912-0

9 787121 309120 >

定价：79.00元