

深度学习

PaddlePaddle基础

DAY04

PaddlePaddle概述

PaddlePaddle概述

PaddlePaddle简介

什么是PaddlePaddle

为什么要学PaddlePaddle

PaddlePaddle优点

PaddlePaddle缺点

国际竞赛获奖情况

行业应用

课程概览

学习资源

PaddlePaddle简介

什么是PaddlePaddle

- PaddlePaddle (Parallel Distributed Deep Learning , 中文名飞桨)
是百度公司推出的开源、易学习、易使用的分布式深度学习平台
- 源于产业实践，在实际中有着优异表现
- 支持多种机器学习经典模型



为什么学习PaddlePaddle

- 开源、国产
- 能更好、更快解工程决实际问题

非会员水印



PaddlePaddle优点

- 易用性。语法简洁，API的设计干净清晰
- 丰富的模型库。借助于其丰富的模型库，可以非常容易的复现一些经典方法
- 全中文说明文档。首家完整支持中文文档的深度学习平台
- 运行速度快。充分利用 GPU 集群的性能，为分布式环境的并行计算进行加速



PaddlePaddle缺点

- 教材少
- 学习难度大、曲线陡峭

非会员水印



国际竞赛获奖情况

获奖模型 / 模块		国际竞赛	
视觉领域	PyramidBox模型	WIDER FACE三项测试子集	第一
	Attention Clusters网络模型	ActivityNet Kinetics Challenge 2017	第一
	StNet模型	ActivityNet Kinetics Challenge 2018	第一
	基于Faster R-CNN的多模型	Google AI Open Images-Object Detection Track	第一
增强学习框架PARL		NIPS AI for Prosthetics Challenge	第一



行业应用



农业

智能桃子分拣机
节约 90% 人力成本



林业

病虫害监测
识别准确率达到 90%



工业

公共场所控烟



零售

商品销量预测
单店生鲜报损降低 30%



人力

AI建立匹配系统
5倍面邀成功率



制造

智能零件分拣
人工效率增加 1 倍



石油

地震波藏油预测



通讯

基站网络故障预警



地产

智能楼宇管理
制冷系统节电 20%



汽车

充电桩故障预警
准确达 90%



课程概览



学习资源

➤ 官网

- ✓ 地址：<https://www.paddlepaddle.org.cn/>
- ✓ 内容：学习指南、文档、API手册

➤ 百度云智学院

- ✓ 地址：<http://abcxueyuan.cloud.baidu.com/#/courseDetail?id=14958>
- ✓ 内容：教学视频

➤ AIStudio

- ✓ 地址：<https://aistudio.baidu.com/aistudio/projectoverview/public/1>
- ✓ 内容：项目案例



体系结构

体系结构

体系结构

总体架构

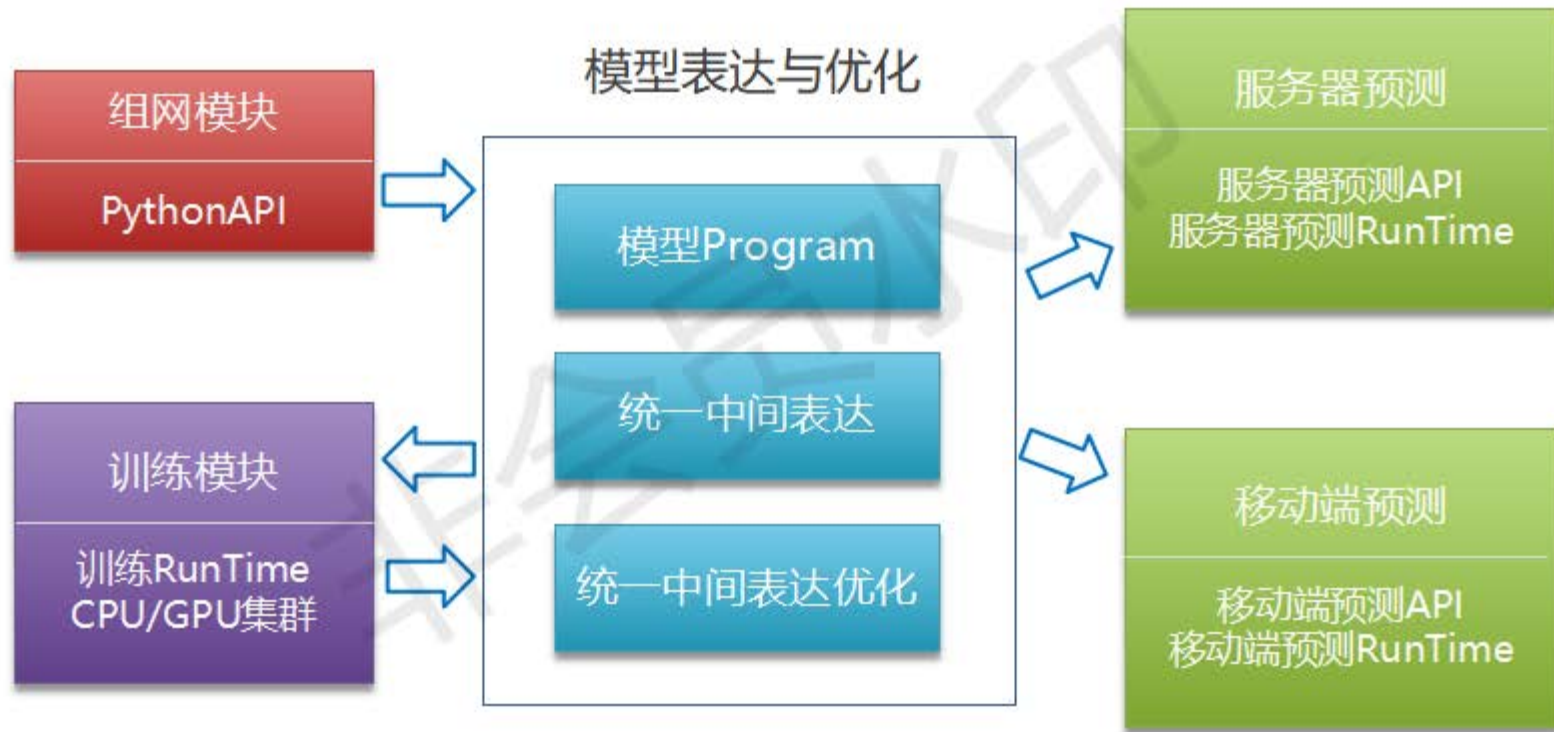
编译时与运行时

三个重要术语

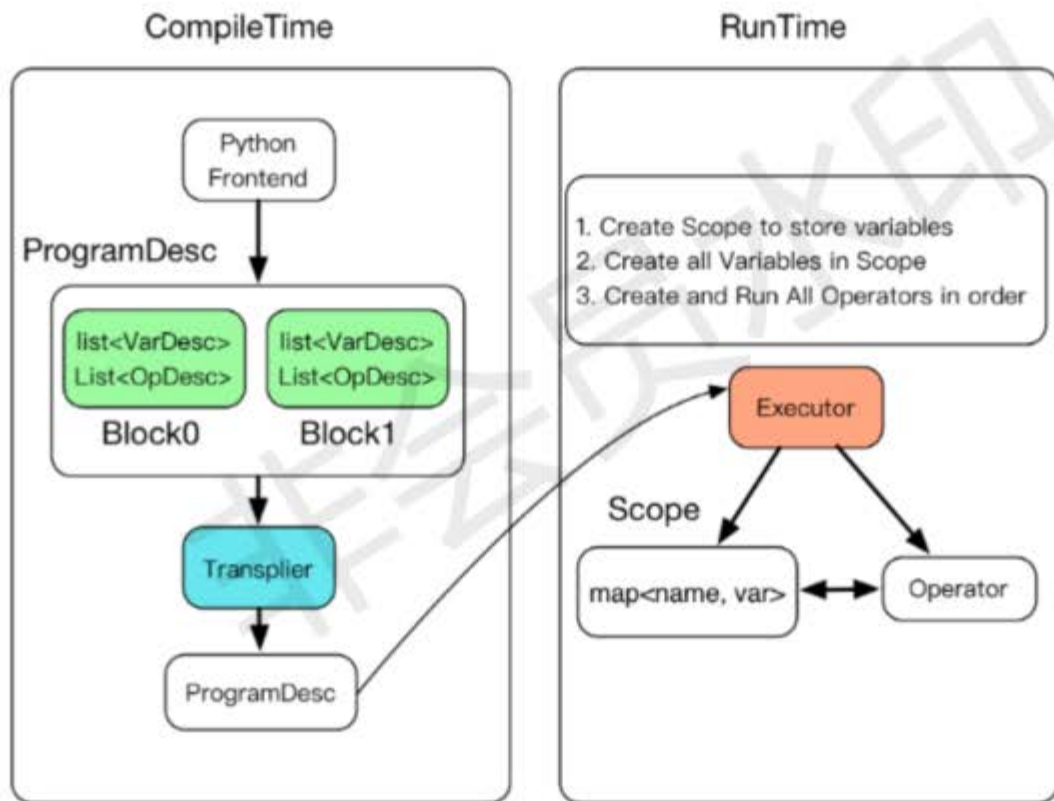
案例1：快速开始

体系结构

总体架构



编译时与执行时



三个重要术语

- Fluid : 定义程序执行流程
- Program : 对用户来说一个完整的程序
- Executor : 执行器, 执行程序

非会员水印



案例1：快速开始

```
1 import paddle.fluid as fluid
2
3 # 创建两个类型为int64，形状为1行1列的张量
4 x = fluid.layers.fill_constant(shape=[1], dtype="int64", value=5)
5 y = fluid.layers.fill_constant(shape=[1], dtype="int64", value=1)
6 z = x + y
7 # print(z) # z为对象，此时还没有值
8
9 # 创建Executor执行器
10 place = fluid.CPUPlace() # 指定在CPU上运行
11 exe = fluid.Executor(place) # 创建执行器
12
13 result = exe.run(fluid.default_main_program(), fetch_list=[z])
14 print(result[0][0]) # result为1*1的张量
```



基本概念与操作

基本概念与操作

基本概念

张量

Layer

Variable

Program

Place

Optimizer

案例2：执行两个张量计算

实现线性回归

程序执行步骤

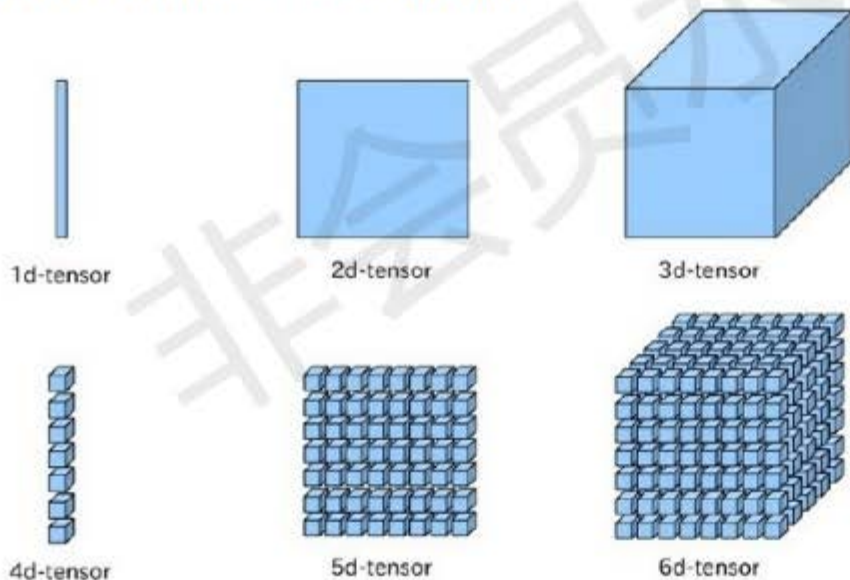
案例3：编写简单线性回归

基本概念

张量

- 什么是张量

张量 (Tensor) : 多维数组或向量，同其它主流深度学习框架一样，PaddlePaddle使用张量来承载数据



张量 (续1)

• 张量示例

灰度图像为二维张量（矩阵），彩色图像为三维张量

[illegible]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000
--	---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------

张量（续2）



一个句子是几维张量？



一篇文章是几维张量？



Layer

- 表示一个独立的计算逻辑，通常包含一个或多个operator（操作），如`layers.relu`表示ReLU计算；`layers.pool2d`表示pool操作。Layer的输入和输出为Variable。



Variable

- 表示一个变量，可以是一个张量（Tensor），也可以是其它类型。
Variable进入Layer计算，然后Layer返回Variable。创建变量方式：

```
x = fluid.layers.data(name="x", shape=[1], dtype="float32")  
y = fluid.layers.data(name="y", shape=[1], dtype="float32")
```

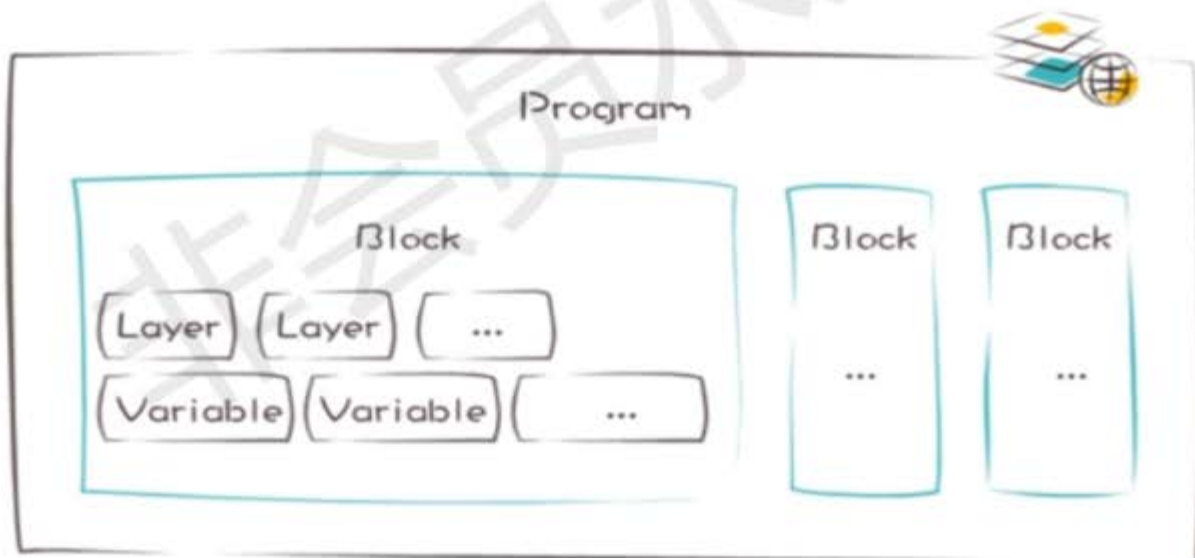
Python变量

Paddle变量



Program

- Program包含Variable定义的多个变量和Layer定义的多个计算，是一套完整的计算逻辑。从用户角度来看，Program是顺序、完整执行的。



Executor

- Executor用来接收并执行Program，会一次执行Program中定义的所有计算。通过feed来传入参数，通过fetch_list来获取执行结果。

```
outs = exe.run(fluid.default_main_program(), # 默认程序上执行  
               feed=params, # 喂入参数  
               fetch_list=[result]) # 获取结果
```



Place

- PaddlePaddle可以运行在Intel CPU , Nvidia GPU , ARM CPU和更多嵌入式设备上 , 可以通过Place用来指定执行的设备 (CPU或GPU) 。

```
1 place = fluid.CPUPlace() # 指定CPU执行  
2 place = fluid.CUDAPlace(0) # 指定GPU执行
```



Optimizer

- 优化器，用于优化网络，一般用来对损失函数做梯度下降优化，从而求得最小损失值

非会员水印

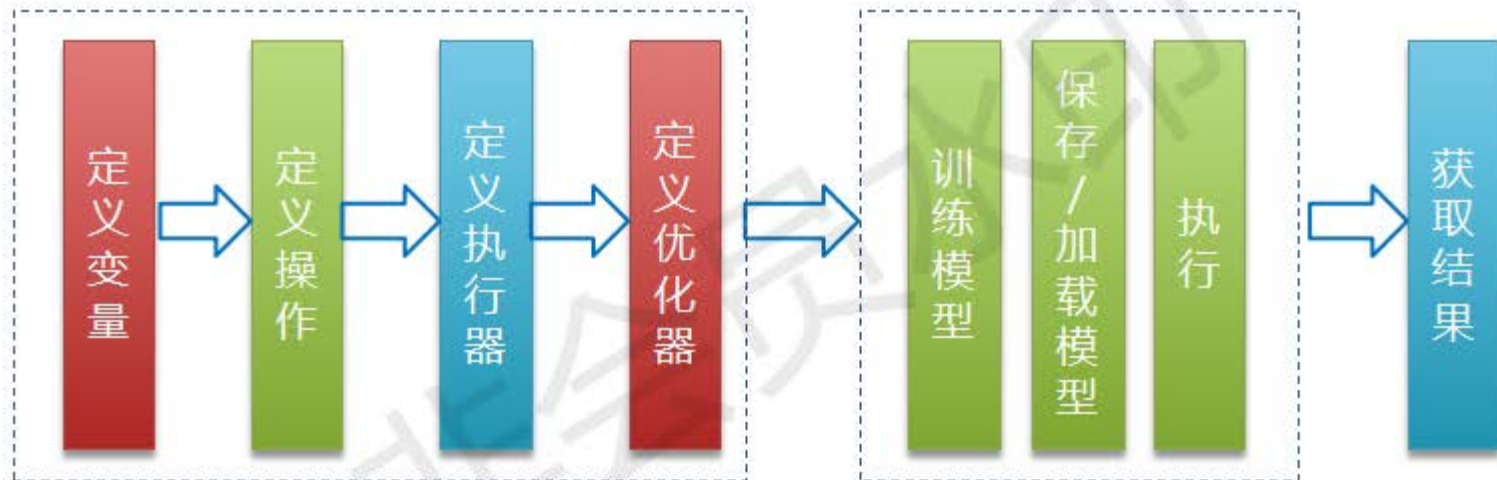


案例2：执行两个张量计算

```
1 import paddle.fluid as fluid
2 import numpy
3 # 创建x, y两个1行1列, 类型为float32的变量(张量)
4 x = fluid.layers.data(name="x", shape=[1], dtype="float32")
5 y = fluid.layers.data(name="y", shape=[1], dtype="float32")
6
7 result = fluid.layers.elementwise_add(x, y) # 两个张量按元素相加
8
9 place = fluid.CPUPlace() # 指定在CPU上执行
10 exe = fluid.Executor(place) # 创建执行器
11 exe.run(fluid.default_startup_program()) # 初始化网络
12
13 a = numpy.array([int(input("x:"))]) # 输入x, 并转换为数组
14 b = numpy.array([int(input("y:"))]) # 输入y, 并转换为数组
15
16 params = {"x": a, "y": b}
17 outs = exe.run(fluid.default_main_program(), # 默认程序上执行
18               feed=params, # 喂入参数
19               fetch_list=[result]) # 获取结果
20 print(outs[0][0])
```



程序执行步骤



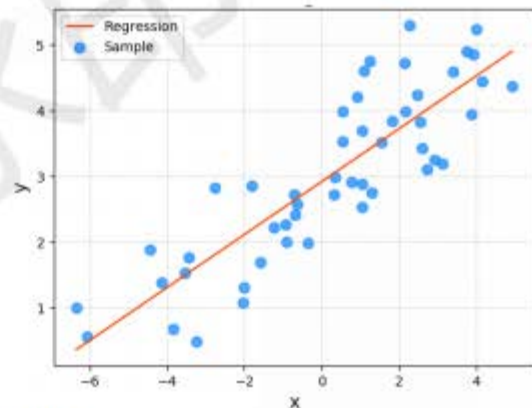
案例3：编写简单线性回归

➤ 任务：

- ✓ 给出输入样本： $[[1.0], [2.0], [3.0], [4.0]]$
- ✓ 给出实际输出样本： $[[2.0], [4.0], [6.0], [8.0]]$
- ✓ 找出 $y = wx$ 公式中的 w

➤ 思路：

- ✓ 定义输入数据、实际输出结果
- ✓ 将数据送入神经网络进行训练（全连接网络，即分类器）
- ✓ 根据实际输出、预测输出之间的损失值，进行梯度下降，直到收敛到极小值为止



案例3：编写简单线性回归（续）

➤ 技术要点：

- ✓ 神经网络，选择 `fluid.layers.fc()`，该函数在神经网络中建立一个全连接层。接收多个输入，为每个输入分配一个权重 w ，并维护一个偏置值 b ；预测时产生一个输出
- ✓ 损失函数：回归问题，选择均方差 `fluid.layers.square_error_cost` 和 `fluid.layers.mean()` 作为损失函数
- ✓ 优化器：随机梯度下降优化器 `fluid.SGD`，做梯度下降计算

代码见：simple_lr.py



关键代码

- 数据准备

```
train_data = np.array([[1.0], [2.0], [3.0], [4.0]]).astype('float32') # 输入样本  
y_true = np.array([[2.0], [4.0], [6.0], [8.0]]).astype('float32') # 实际输出样本
```



关键代码（续1）

- 搭建网络

```
# 通过全连接网络进行预测
y_preict = fluid.layers.fc(input=x, # 输入
                             size=1, # 输出结果个数
                             act=None) # 激活函数

# 添加损失函数
cost = fluid.layers.square_error_cost(input=y_preict, label=y)
avg_cost = fluid.layers.mean(cost) # 求均方差

# 定义优化方法
optimizer = fluid.optimizer.SGD(learning_rate=0.01)
optimizer.minimize(avg_cost) # 指定最小化均方差值

# 搭建网络
place = fluid.CPUPlace() # 指定在CPU执行
exe = fluid.Executor(place)
exe.run(fluid.default_startup_program()) # 初始化系统参数
```



关键代码（续2）

- 执行训练

```
# 开始训练
costs = []
iters = []
values = []
params = {"x": train_data, "y": y_true}
for i in range(200):
    outs = exe.run(feed=params, fetch_list=[y_preict.name, avg_cost.name])
    iters.append(i) # 迭代次数
    costs.append(outs[1][0]) # 损失值
    values.append(outs[0][0]) # 预测值

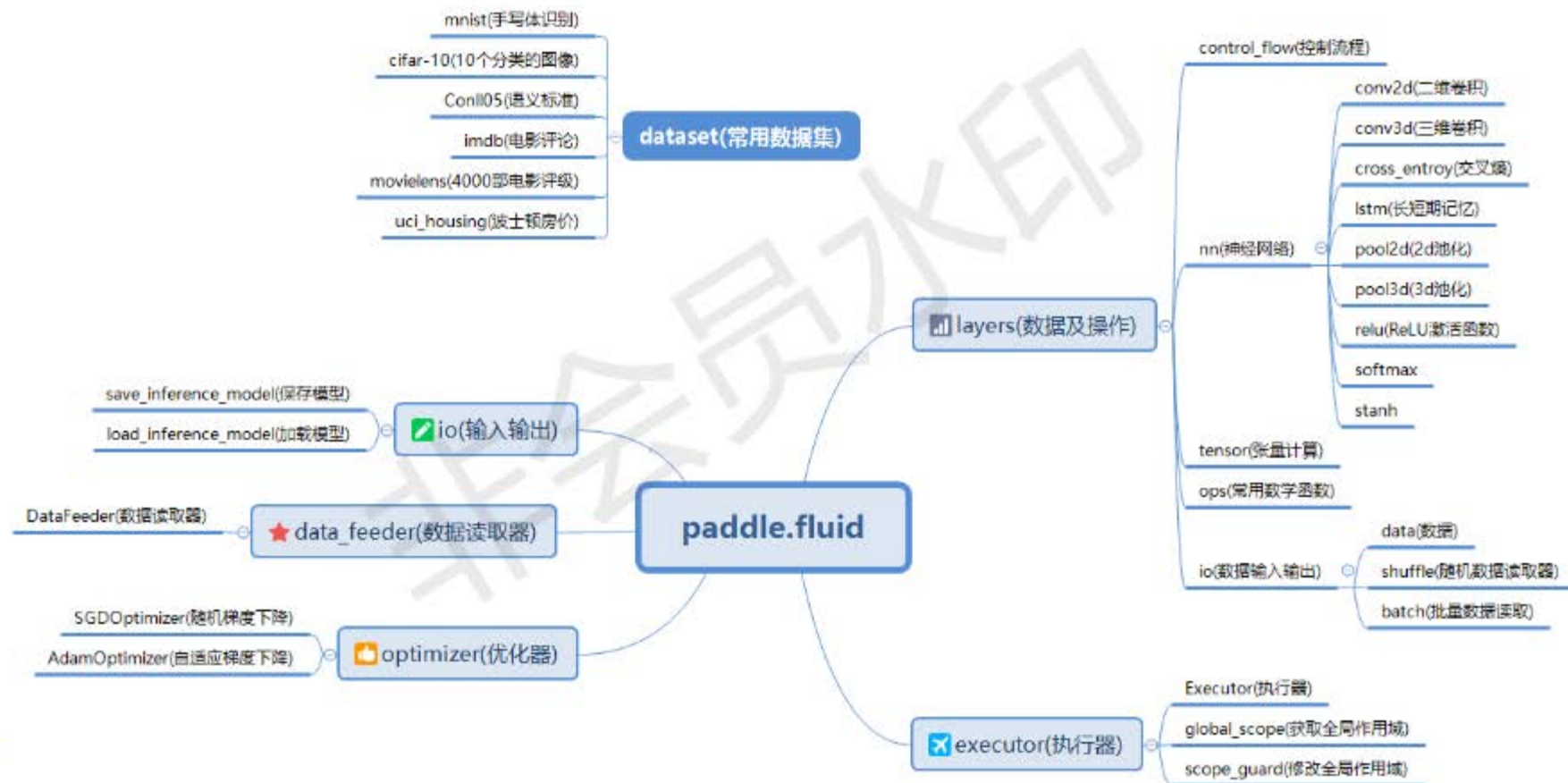
    print("avg_cost:", outs[1]) # y_preict
```



模型收敛过程



fluid API结构图



数据准备

数据准备

数据准备

案例：波士顿房价预测

什么是数据准备

为什么需要数据准备

案例4：使用reader

数据集及任务

思路

数据准备

什么是数据准备

- 数据准备是指将样本数据从外部（主要指文件）读入，并且按照一定方式（随机、批量）传递给神经网络，进行训练或测试的过程
- 数据准备包含三个步骤：
 - ✓ 第一步：自定义Reader生成训练/预测数据
 - ✓ 第二步：在网络配置中定义数据层变量
 - ✓ 第三步：将数据送入网络进行训练/预测



为什么需要数据准备

- **从文件读入数据。** 因为程序无法保存大量数据，数据一般保存到文件中，所以需要单独的数据读取操作
- **批量快速读入。** 深度学习样本数据量较大，需要快速、高效读取（批量读取模式）
- **随机读入。** 为了提高模型泛化能力，有时需要随机读取数据（随机读取模式）



案例4：使用reader

- 自定义reader creator，从文本文件test.txt中读取一行数据

```
1 import numpy
2 import paddle
3
4 def reader_creator(file_path):
5     def reader():
6         with open(file_path, "r") as f:
7             lines = f.readlines()
8             for line in lines:
9                 yield line
10    return reader
11
12 reader = reader_creator("test.txt")
13 for data in reader():
14     print(data, end="")
```



案例4：使用reader（续1）

- 从上一个reader中以随机方式读取数据

```
1 import numpy
2 import paddle
3
4 def reader_creator(file_path):
5     def reader():
6         with open(file_path, "r") as f:
7             lines = f.readlines()
8             for line in lines:
9                 yield line
10    return reader
11
12 reader = reader_creator("test.txt")
13 shuffle_reader = paddle.reader.shuffle(reader, 10)
14 for data in shuffle_reader():
15     print(data, end="")
```



案例4：使用reader（续2）

- 从上一个随机读取器中，分批次读取数据

```
1 import numpy
2 import paddle
3
4 def reader_creator(file_path):
5     def reader():
6         with open(file_path, "r") as f:
7             lines = f.readlines()
8             for line in lines:
9                 yield line
10    return reader
11
12 reader = reader_creator("test.txt")
13 shuffle_reader = paddle.reader.shuffle(reader, 10)
14 batch_reader = paddle.batch(shuffle_reader, 3)
15 for data in batch_reader():
16     print(data, end="")
```



案例：波士顿房价预测

数据集及任务

➤ 数据集介绍

- ✓ 数据量：506笔
- ✓ 特征数量：13个（见右图）
- ✓ 标签：价格中位数

➤ 任务：根据样本数据，预测房价中位数（回归问题）

属性名	解释	类型
CRIM	该镇的人均犯罪率	连续值
ZN	占地面积超过25,000平方呎的住宅用地比例	连续值
INDUS	非零售商业用地比例	连续值
CHAS	是否邻近 Charles River	离散值，1=邻近；0=不邻近
NOX	一氧化氮浓度	连续值
RM	每栋房屋的平均客房数	连续值
AGE	1940年之前建成的自用单位比例	连续值
DIS	到波士顿5个就业中心的加权距离	连续值
RAD	到径向公路的可达性指数	连续值
TAX	全值财产税率	连续值
PTRATIO	学生与教师的比例	连续值
B	$1000(BK - 0.63)^2$ ，其中BK为黑人占比	连续值
LSTAT	低收入人群占比	连续值
MEDV	同类房屋价格的中位数	连续值



思路



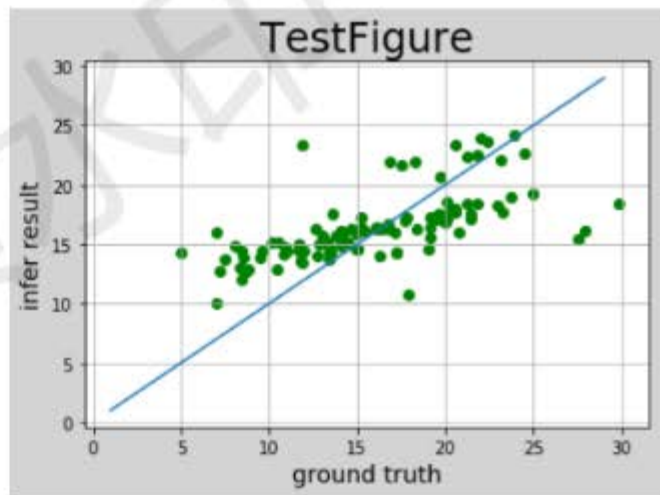
代码见：linear_regression.py



执行结果



损失函数收敛过程



预测值与实际值对比



案例5：波士顿房价预测

- 全部代码见：`uci_housing.py`

非会员水印



今日总结

- PaddlePaddle体系结构与基本概念
 - Tensor, Layer, Program, Variable, Executor, Place
 - Fluid API组织结构
- 案例：
 - 简单线性回归
 - 机器学习经典案例：波士顿房价预测