

计算机应用技术丛书

一个中文书籍的 L^AT_EX 模板

—— 这里可以放一个副标题

这个图书模板是在群主网站上的一个封面模板的基础上改写而成的，设定了一些图书出版要素，设计了封面、扉页及版权页和封底的样式，修改了 chapter 的样式，并提供了几个选项可切换色彩风格，其余则维持 book 基本文档类的设定。

图书模板部分代码的完成得到了林莲枝大神的帮助，在此表示感谢。由于作者水平有限，模板代码编写不恰当之处还请用户提出批评和指正。

感谢造字工坊提供了刻宋、郎宋和黄金时代三种非商业可免费下载使用的字体。

感谢谷歌提供自由使用的思源宋体、思源黑体。

感谢文泉驿提供的开源的文泉驿等宽微米黑字体。

张 三 著

L^AT_EXStudio 出版社

第二版
Edition II

一个中文书籍的 L^AT_EX 模板

张 三 著

图书在版编目(CIP)数据

一个中文书籍的 L^AT_EX 模板/张三著.—成都: L^AT_EXStudio 出版社, 2015/11/05

ISBN xxx-x-xxxx-xxxx-x

I.0811… II. 张三 … III. 书籍—模板—L^AT_EXStudio IV.I213

中国版本图书馆 CIP 数据核字 (2018) 第 666666 号

一个中文书籍的 L^AT_EX 模板

张 三 著

* * *

L^AT_EXStudio 出版社

<http://www.latexstudio.net>

开本: 216 mm×279 mm 字数: 有功夫您帮我数一下

印数: 001–100 册 定价: 26.5 元

本书如有缺页、倒页、脱页, 请自行处理一下

T _E X Design 目 录	第 1 章 模板使用说明	1
	第 1.1 节 颜色选择	1
	第 1.2 节 字体选择	1
	第 1.3 节 几个可有可无的 tcolorbox 设置	2
	1.3.1 Ubuntu Terminal	2
	第 2 章 代码盒子	3
	第 2.1 节 settings.sty 的一些设置和宏包的使用	9
	2.1.1 线性规划问题	9

T_EX Design

第 1 章

模板使用说明

首先我们简要说明一下目前已经开发出来的模板功能,包括字体选择和颜色选择。

第 1.1 节

颜色选择

目前颜色选择共三种, `green`, `orange` 和 `violet`, 默认为 `green`。你也可以自己设置 `cvprimary`, `cvsecondary` 以及 `cvtext`, 定制你自己喜欢的色盘。



自己来设定颜色吧！

```
% 如果想要用 xcolor 宏包已定义的颜色, 记得要在 \documentclass 之前
% 传递相应的 xcolor 选项。
\PassOptionsToPackage{dvipsnames}{xcolor}
\documentclass{lsbook}
% cvprimary 为封面、标题盒子最外围的颜色
\colorlet{cvprimary}{SkyBlue}
% cvsecondary 为封面、标题盒子主要填充的颜色
\colorlet{cvsecondary}{RoyalBlue}
% cvtext 为封面、标题盒子里文字的颜色, 也可以自己给出
% 特定的色值。
\definecolor{cvtext}{HTML}{EEEEFF}
```

第 1.2 节

字体选择

目前字体选择共两种, `customfont` 和 `systemfont`, 默认为 `systemfont`。若要使用 `customfont`, 需要安装相应的字体。若要使用 `systemfont`, 则模板直接调用系统内部字体。

第 1.3 节

几个可有可无的 tcolorbox 设置

现有 `appledark`, `applelight`, `win10dark`, `win10light` 四个选项。如果是供荧幕阅读的还好；如果是要打印出来的，除非您就是打印店的老板，不然还是不要多用 `*dark` 选项。



世界你好！

天气好吗？吃饱了吗？



世界你好！

天气好吗？吃饱了吗？

1.3.1

Ubuntu Terminal

```

TensorFlow@xubuntu:~$
----- TRAINING -----
Epoch: 004/020 cost: 1.41867 train_accaray:0.63000 test_accaray:0.62820
Epoch: 008/020 cost: 0.98922 train_accaray:0.71000 test_accaray:0.72040
Epoch: 012/020 cost: 0.81536 train_accaray:0.70000 test_accaray:0.76270
Epoch: 016/020 cost: 0.71605 train_accaray:0.83000 test_accaray:0.78570
Epoch: 020/020 cost: 0.65017 train_accaray:0.82000 test_accaray:0.80130

```

灰色主题盒子

世界你好！



天气好吗？吃饱了吗？

世界你好！



天气好吗？吃饱了吗？

T_EX Design

第 2 章

代码盒子

程序清单 1: SIFT 算法

    Python

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Jan 22 19:22:10 2014
4
5 @author: duan
6 """
7
8 import cv2
9 import numpy as np
10
11 img = cv2.imread('home.jpg')
12 gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
13
14 sift = cv2.SIFT()
15 kp = sift.detect(gray,None)
16
17 img=cv2.drawKeypoints(gray,kp)
18
19 cv2.imwrite('sift_keypoints.jpg',img)
```

程序清单 2: Matlab 求解

    Matlab

```
1 b1 = [-6;0];
2 a1 = [-3 -2;1 0];
3 d1 = [0;5/2];
4 format rat;
5 alpha1 = b1\(a1*d1) % 右乘 a1
```

Haar cascading detection in OpenCV

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Thu Jan 30 11:06:23 2014
4
5  @author: duan
6  """
7
8  import numpy as np
9  import cv2
10
11  face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
12  eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
13
14  img = cv2.imread('sachin.jpg')
15  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
16
17  # Detects objects of different sizes in the input image.
18  # The detected objects are returned as a list of rectangles.
19  # cv2.CascadeClassifier.detectMultiScale(image, scaleFactor, minNeighbors,
20  ↪ flags, minSize, maxSize)
21  # scaleFactor - Parameter specifying how much the image size is reduced at
22  ↪ each image
23  # scale.
24  # minNeighbors - Parameter specifying how many neighbors each candidate
25  ↪ rectangle should
26  ↪ have to retain it.
27  # minSize - Minimum possible object size. Objects smaller than that are
28  ↪ ignored.
29  # maxSize - Maximum possible object size. Objects larger than that are
30  ↪ ignored.
31
32  faces = face_cascade.detectMultiScale(gray, 1.3, 5)
33  for (x,y,w,h) in faces:
34      img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
```

```
29     roi_gray = gray[y:y+h, x:x+w]
30     roi_color = img[y:y+h, x:x+w]
31     eyes = eye_cascade.detectMultiScale(roi_gray)
32     for (ex,ey,ew,eh) in eyes:
33         cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
34 cv2.imshow('img',img)
35 cv2.waitKey(0)
36 cv2.destroyAllWindows()
```

Multi-GPU Basics

```
1 import numpy as np
2 import tensorflow as tf
3 import datetime
4
5 #Processing Units logs
6 log_device_placement = True
7
8 #num of multiplications to perform
9 n = 10
10
11 # Example: compute  $A^n + B^n$  on 2 GPUs
12
13 # Create random large matrix
14 A = np.random.rand(1e4, 1e4).astype('float32')
15 B = np.random.rand(1e4, 1e4).astype('float32')
16
17 # Creates a graph to store results
18 c1 = []
19 c2 = []
20
21 # Define matrix power
22 def matpow(M, n):
23     if n < 1: #Abstract cases where  $n < 1$ 
24         return M
```

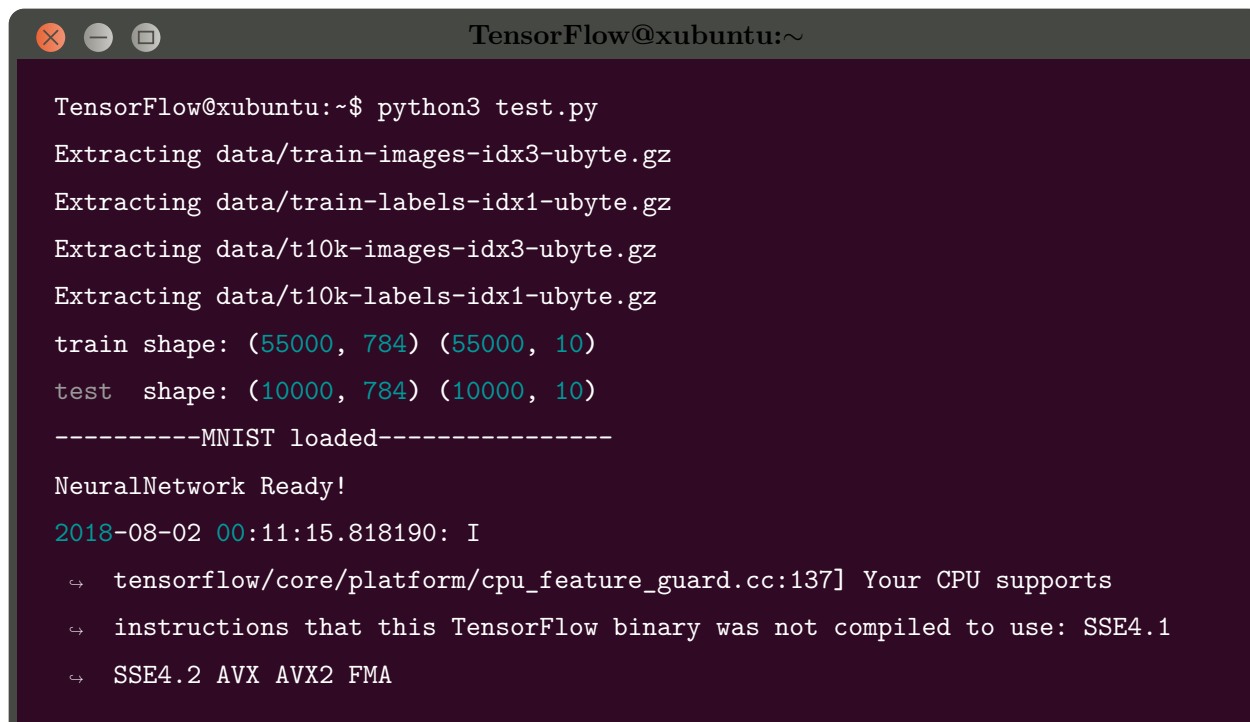
```
25     else:
26         return tf.matmul(M, matpow(M, n-1))
27
28     # Single GPU computing
29
30     with tf.device('/gpu:0'):
31         a = tf.constant(A)
32         b = tf.constant(B)
33         #compute A^n and B^n and store results in c1
34         c1.append(matpow(a, n))
35         c1.append(matpow(b, n))
36
37     with tf.device('/cpu:0'):
38         sum = tf.add_n(c1) #Addition of all elements in c1, i.e. A^n + B^n
39
40     t1_1 = datetime.datetime.now()
41     with
42         ↪ tf.Session(config=tf.ConfigProto(log_device_placement=log_device_placement))
43         ↪ as sess:
44         # Runs the op.
45         sess.run(sum)
46         t2_1 = datetime.datetime.now()
47
48     # Multi GPU computing
49     # GPU:0 computes A^n
50     with tf.device('/gpu:0'):
51         #compute A^n and store result in c2
52         a = tf.constant(A)
53         c2.append(matpow(a, n))
54
55     #GPU:1 computes B^n
56     with tf.device('/gpu:1'):
57         #compute B^n and store result in c2
58         b = tf.constant(B)
```

```

57 c2.append(matpow(b, n))
58
59 with tf.device('/cpu:0'):
60     sum = tf.add_n(c2) #Addition of all elements in c2, i.e.  $A^n + B^n$ 
61
62 t1_2 = datetime.datetime.now()
63     with
64         ↪ tf.Session(config=tf.ConfigProto(log_device_placement=log_device_placement))
65         ↪ as sess:
66     # Runs the op.
67     sess.run(sum)
68 t2_2 = datetime.datetime.now()
69
70 print "Single GPU computation time: " + str(t2_1-t1_1)
71 print "Multi GPU computation time: " + str(t2_2-t1_2)

```

Single GPU computation time: 0:00:11.833497
Multi GPU computation time: 0:00:07.085913



TensorFlow@xubuntu:~

```

TensorFlow@xubuntu:~$ python3 test.py
Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz
train shape: (55000, 784) (55000, 10)
test  shape: (10000, 784) (10000, 10)
-----MNIST loaded-----
NeuralNetwork Ready!
2018-08-02 00:11:15.818190: I
↪ tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports
↪ instructions that this TensorFlow binary was not compiled to use: SSE4.1
↪ SSE4.2 AVX AVX2 FMA

```

```
2018-08-02 00:11:16.044897: I
  ↳ tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Found device 0 with
  ↳ properties:
name: GeForce GTX 1080 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.582
pciBusID: 0000:0a:00.0
totalMemory: 10.92GiB freeMemory: 3.04GiB
2018-08-02 00:11:16.044948: I
  ↳ tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Creating TensorFlow
  ↳ device (/device:GPU:0) -> (device: 0, name: GeForce GTX 1080 Ti, pci bus
  ↳ id: 0000:0a:00.0, compute capability: 6.1)
----- TRAINING -----
Epoch: 004/020 cost: 1.41867 train_accaray:0.63000 test_accaray:0.62820
Epoch: 008/020 cost: 0.98922 train_accaray:0.71000 test_accaray:0.72040
Epoch: 012/020 cost: 0.81536 train_accaray:0.70000 test_accaray:0.76270
Epoch: 016/020 cost: 0.71605 train_accaray:0.83000 test_accaray:0.78570
Epoch: 020/020 cost: 0.65017 train_accaray:0.82000 test_accaray:0.80130

Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz
train shape: (55000, 784) (55000, 10)
test shape: (10000, 784) (10000, 10)
-----MNIST loaded-----
NeuralNetwork Ready!
2018-08-02 00:11:15.818190: I
  ↳ tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports
  ↳ instructions that this TensorFlow binary was not compiled to use: SSE4.1
  ↳ SSE4.2 AVX AVX2 FMA
2018-08-02 00:11:16.044897: I
  ↳ tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Found device 0 with
  ↳ properties:
name: GeForce GTX 1080 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.582
pciBusID: 0000:0a:00.0
totalMemory: 10.92GiB freeMemory: 3.04GiB
```

```
2018-08-02 00:11:16.044948: I
↳ tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Creating TensorFlow
↳ device (/device:GPU:0) -> (device: 0, name: GeForce GTX 1080 Ti, pci bus
↳ id: 0000:0a:00.0, compute capability: 6.1)
```

Output Data

第 2.1 节

settings.sty 的一些设置和宏包的使用

程序清单 3: Matlab 求解

    Matlab

```
1 b1 = [-6;0];
2 a1 = [-3 -2;1 0];
3 d1 = [0;5/2];
4 format rat;
5 alpha1 = b1\ (a1*d1) % 右乘 a1
```

得 $\hat{a}_k = \frac{6}{5}$
 $\succeq \succ \prec \preceq xyzxx$

$$\nabla_x L(x, \nu) = 2x + A^T \nu = 0,$$

2.1.1

线性规划问题

 考虑标准形式的线性规划问题

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b, \\ & && x \succeq 0 \end{aligned} \tag{2.1}$$

$$\begin{aligned} b^T &= [b_1 \quad b_2 \quad \cdots \quad b_n] \\ x^T &= [x_1 \quad x_2 \quad \cdots \quad x_n] \\ b^T \nu &= \nu^T b \end{aligned}$$

如果我们猜测 x 的值为 \hat{x} , 也就是隐含地猜测 v 的值为 $y - A\hat{x}$ 。假设 v (在 $\|\cdot\|$ 度量下) 越小越可信, 那么对于 x 的最有可信的猜测是

$$\hat{x} = \operatorname{argmin}_z \|Az - y\| \quad (2.2)$$

考虑子空间 $\mathcal{A} = \mathcal{R}(A) \subseteq \mathbf{R}^m$ 和一个点 $b \in \mathbf{R}^m$ 。点 b 向子空间 \mathcal{A} 的投影是 \mathcal{A} 中在 $\|\cdot\|$ 下最靠近 b 的点, 也就是说, 它是下列问题的任意最优解

$$\begin{aligned} & \text{minimize} \quad \|u - b\| \\ & \text{subject to} \quad u \in \mathcal{A}. \end{aligned} \quad (2.3)$$

将 $\mathcal{R}(A)$ 中的元素参数化为 $u = Ax$, 我们可以看出求解范数逼近问题 (6.1) 等价于计算 b 向 \mathcal{A} 的投影。

Newton 方法是求解无约束优化问题的有效算法, 干净优化问题的一个特例就是经常遇到的最小二乘问题

$$\text{minimize} \quad \|Ax - b\|_2^2 = x^T(A^T A)x - 2(A^T b)^T x + b^T b \quad (2.4)$$

其最优性条件

$$A^T A x^* = A^T b$$

被称为最小二乘问题的**正规方程** 无约束几何规划 作为第二个盒子, 我们考虑凸的无线事几何规划问题

$$\text{minimize} \quad f(x) = \log \left(\sum_{i=1}^m \exp(a_i^T x + b_i) \right) \quad (2.5)$$

其最优性条件为

$$\nabla f(x^*) = \frac{1}{\sum_{j=1}^m \exp(a_j^T x^* + b_j)} \sum_{i=1}^m \exp(a_i^T x^* + b_i) a_i = 0.$$

一般情况下该方程组没有解析解, 因此我们必须采用迭代算法。由于 $\mathbf{dom} f = \mathbf{R}^n$, 斜体点都可以用作初始点 $x^{(0)}$ 。

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{m}{2} \|y - x\|^2 \quad (2.6)$$

对任意固定的 x , 式 (2.6) 的右边是 y 的二次凸函数。令其关于 y 的层数等于零, 即

$$\frac{\partial [f(x) + \nabla f(x)^T (y - x) + \frac{m}{2} \|y - x\|_2^2]}{\partial y} = 0 \quad (2.7)$$

$$0 + \nabla f(x) + m(y - x) = 0 \quad (2.8)$$

其中 $y > x$, 可以得到该二次函数的最优解 $\tilde{y} = x - (1/m)\nabla f(x)$ 。

二次型函数极点

$$\left(-\frac{b}{2a}, \frac{4ac - b^2}{4a} \right)$$


 **unicode-math 的一些说明** 这里使用的数学宏包是 unicode-math, 所以一些命令与 amsmath 里面有些不同

Table 1: New unicode-math commands which overlap with legacy math commands. For new documents the sym versions are recommended.

Command	Synonym	Command	Synonym
<code>\symnormal</code>	<code>\mathnormal</code>		
<code>\symliteral</code>		<code>\symbfsf</code>	<code>\mathbfsf</code>
		<code>\symbfup</code>	<code>\mathbfup</code>
		<code>\symbfit</code>	<code>\mathbfit</code>
<code>\symbb</code>	<code>\mathbb</code>		
<code>\symbbit</code>	<code>\mathbbit</code>		
<code>\symcal</code>	<code>\mathcal</code>	<code>\symbfcal</code>	<code>\mathbfcal</code>
<code>\symscr</code>	<code>\mathscr</code>	<code>\symbfscr</code>	<code>\mathbfscr</code>
<code>\symfrak</code>	<code>\mathfrak</code>	<code>\symbffrak</code>	<code>\mathbffrak</code>
<code>\symsfup</code>	<code>\mathsfup</code>	<code>\symbfsfup</code>	<code>\mathbfsfup</code>
<code>\symsfit</code>	<code>\mathsfit</code>	<code>\symbfsfit</code>	<code>\mathbfsfit</code>

封面设计：张三丰
责任编辑：张三丰

ISBN 978-80-7340-097-2



9 788073 400972

定价：26.5 元