Creating Capsule Wardrobes from Fashion Images评述

许锋
3140102526
计算机学院
fxuzju@zju.edu.cn

摘要（Abstract）

本报告主要评述Creating Capsule Wardrobes from Fashion Images[1]这篇CVPR 2018 的论文。评述内容首先是胶囊衣柜生成问题的定义，然后是文章提出的生成胶囊衣柜的几个步骤，以及生成胶囊衣柜的核心算法，最后是论文里涉及的主要对比实验及其结果，以及文章的主要贡献。

1. 胶囊衣柜问题定义及意义

1.1. 胶囊衣柜的定义

通常我们把一系列衣物以及首饰的集合称为衣柜(wardrobe，fig 1 的左上角)，比如一个服装商店的服装列表就是一个衣柜，一个用户的所有衣物也是一个衣柜。从衣柜中选出一些单间可以构成一个胶囊衣柜(capsule wardrobe，fig 1 的右上角)。胶囊衣柜中的衣物可以进行组合得到套装(outfit，fig 1 的下方)。

胶囊衣柜的目的简而言之就是"do more with less"，也就是花更少的钱办更多的事。因此一个好的胶囊衣柜应该具有以下几个特征。首先，它的大小应该是相对较小的（花更少的钱）。其次，胶囊衣柜中衣物的组合套装应该具有良好的视觉兼容性(compatibility)，也就是说不会出现西装配沙滩裤的搞笑组合，保证不会浪费钱。最后，胶囊衣柜中的衣物所组成的套装应该互相之间具有多样性(versatility)，来保证办更多的事。

Figure 1：衣柜(wardrobe)、胶囊衣柜(capsule wardrobe)和套装(outfit)

1.2. 自动生成胶囊衣柜的意义

首先，胶囊衣柜本身可以帮助用户花更少的钱办更多的事。其次，胶囊衣柜也可以帮助售货商组成有吸引力的衣物组合招徕顾客。

2. 自动生成胶囊衣柜

自动生成胶囊衣柜主要面临两个挑战。首先，如何衡量胶囊衣柜兼容性和多样性？兼容性与相似性不同，相似性回答"哪个和这个看起来像？"，而兼容性则要回答"哪个和这个互补？"。文章通过借鉴主题模型很好地解决了这两个问题。其次，如何控制算法的复杂度？文章通过提出一种迭代的贪婪算法来解决算法的复杂度过高的问题。

2.1. 问题定义

胶囊衣柜的生成可以看作一个子集选择问题，如fig 2。

假设衣物分为$i = 0, \dots, (\mathrm{m}-1)$共m层（例如上衣，裤子，外套等等），每一层的衣物有N_i件，用集合$A_i = \{s_i^0, s_i^0, \dots, s_i^{N_i-1}\}$表示。那么胶囊衣柜生成就是要从每一层中分别选出$T$件衣物，每一层选出的$T$件衣物的集合用$A_{iT} = \{s_i^{j_1}, \dots, s_i^{j_T}\}$表示。这个。最终要找到这样一个胶囊衣柜，满足：

$$\mathrm{y}^* = \underset{y \subseteq \gamma}{\operatorname{argmax}}\, C(y) + V(y) \qquad (1)$$

其中，$\mathrm{y} = A_{0T} \times A_{1T} \times \dots \times A_{(m-1)T}$，是一个胶囊衣柜的所有套装的集合，而$\gamma$是所有可能的胶囊衣柜对应的y的集合。函数$C(y)$和$V(y)$分别用来衡量兼容性和多样性。

Figure 2：胶囊衣柜的子集选择过程

Figure 3：从左到右，共同购买记录、用户生成的衣物组合、穿着搭配图片

2.2. 兼容性和多样性的衡量

无论是兼容性还是多样性，根本上来说都是针对胶囊衣柜中的套装o_j而言的。

假设我们已经有了衡量一个套装o_j的兼容性的函数$c(o_j)$，那么一个胶囊衣柜的兼容性可以如下定义：

$$C(y) := \sum_{o_j \in y} c(o_j) \tag{2}$$

假设我们已经有了衡量胶囊衣柜y对某一种风格z_i的包含程度的函数$v_y(z_i)$，那么一个胶囊衣柜的多样性可以如下定义：

$$V(y) := \sum_{i=1}^{K} v_y(z_i) \tag{3}$$

其中，K是不同风格的数量。正如上面所说，多样性的定义本质上也是基于套装o_j的。如果我们可以计算出$P(z_i|o_j)$（给定套装o_j，风格z_i的概率），那么$v_y(z_i)$可以进一步如下定义：

$$v_y(z_i) := 1 - \prod_{o_j \in y} \left(1 - P(z_i|o_j)\right) \tag{4}$$

其中$1 - P(z_i|o_j)$代表套装o_j不是风格z_i的概率，$\prod_{o_j \in y} \left(1 - P(z_i|o_j)\right)$代表胶囊衣柜$y$中的每一个套装$o_j$都不是风格$z_i$的概率。

进一步地，如果用户对不同的风格有不同的偏好，我们还可以给$v_y(z_i)$加上一个权重，得到一个反应用户喜好的多样性衡量的定义：

$$V(y) := \sum_{i=1}^{K} w_i v_y(z_i) \tag{5}$$

那么如何找到计算$c(o_j)$和$P(z_i|o_j)$的方法呢？文章利用了自然语言处理中的Correlated Topic Models（CTM）模型。CTM模型和LDA模型在思想上都是类似的，区别在于CTM模型考虑了主题之间的关系，摒弃了标准LDA模型中的先验狄利克雷分布，而是使logistic Normal分布，使用正态分布中的协方差矩阵建模各个主题之间的关系。

在这个问题的场景下，语料库的每一个不同的文档就对应一个不同的套装，而文档中的每一个不同的单词就对应套装中的一个不同的属性（attribute），CTM模型的每一个不同的主题（topic）就对应一个不同的风格（style）。

如果我们已经有了一个用不同套装训练得到的CTM模型（参数μ, Σ, β），那么$c(o_j)$就可以如下定义：

$$c(o_j) := p(o_j|\mu, \Sigma, \beta) \tag{6}$$

利用这个训练好的CTM模型，对于任意一个套装o_j，我们还可以得到这个套装在不同的风格上的概率分布向量$\boldsymbol{\theta}_{o_j} = \left[\theta_{o_j 1}, \ldots, \theta_{o_j K}\right]$，因此我们可以如下定义$P(z_i|o_j)$：

$$P(z_i|o_j) := P\left(z_i \middle| \boldsymbol{\theta}_{o_j}\right) = \theta_{o_j i} \tag{7}$$

如果用户对风格有不同的偏好，我们可以选取用户喜欢的一些套装$\{o_{j_1}, \ldots, o_{j_U}\}$，然后得到$\{\boldsymbol{\theta}_{o_{j_1}}, \ldots, \boldsymbol{\theta}_{o_{j_U}}\}$，最后计算用户对风格$z_i$的偏好权重$w_i = \frac{1}{U}\sum_j \theta_{o_j i}$。

最后，我们要如何得到一个数据集来训练CTM模型呢？文章采取的方法是将每一张现成的穿着搭配图片（fig 3 的右边）中的衣物作为一个套装，从中提取出套装的属性来构成数据集。为此，文章提出了一个交叉域属性识别的方法。简单来说就是先用单件衣物图片（fig 2 中的衣物图片）的属性训练对应衣物层的深度神经网络，然后识别并分隔穿着搭配图片的上身和下身，用来训练对套装属性的识别。当然，这些属性（也就是识别或预测的label）都是预先定义好的。

到目前为止，兼容性和多样性的衡量问题已经得到了解决。简单地说，文章利用了自然语言处理中的CTM模型来计算兼容性和多样性，为了得到CTM的训练数据，又训练了一个图片属性识别的神经网络来识别套装的属性。

这种做法最大的好处是，它不需要负样本。之前的工作在训练的时候都需要负样本（不兼容的套装），但是不兼容的套装相比兼容的套装来说是更加不容易获取的。另一方面，训练CTM模型的数据来源非常广泛，任何街边拍摄的穿着搭配图片都可以作为套装，而不仅仅局限于用户的共同购买记录、用户生成的衣物组合（fig 3）等等套装的代替品。

但是我认为，文章提出的方法非常依赖图片属性识别的准确性。不论是训练CTM模型的时候还是计算兼容性和多样性的时候，套装属性识别的准确性都对结果会有非常大的影响。此外，所有的属性都是人工定义的，对于兼容性和多样性这样精妙、细微又难以言喻的特征，

用人工定义的属性去作为基本粒度，难免会有遗漏之处。

2.3. 自动生成胶囊衣柜算法

兼容性和多样性的衡量问题已经解决，接下来就要设计一个算法实现问题定义中的子集选择问题。需要注意的是，如果采取暴力的方法求解问题定义中最优解y^*，那么对于每个胶囊衣柜，需要T^m次计算（胶囊衣柜中套装的数量），而所有可能的胶囊衣柜选择共有$\binom{N}{T}^m$种。一般情况下，N可能是一个商店里所有衣物的数量，数量可能是几百或几千，因此求解最优解是不现实的。

但是，随着胶囊衣柜大小的增长，后添加的衣物相比先添加的衣物对于胶囊衣柜的风格的影响应该是更小的，也就是满足收益递减原则。也就是说，如果我们设计的目标函数是submodular的，那么我们就可以通过贪婪算法不断增加胶囊衣柜的大小来有效地迭代出一个近似解。这个贪婪算法保证得到的解的分数至少达到最优解分数的$1-\frac{1}{\epsilon}$，也就是 63%[2]。

Submodular函数的定义如下：一个函数F是submodular的，如果，$\forall D \subseteq B \subseteq V, \forall s \in V \backslash B, F(D \cup \{s\}) - F(D) \geq F(B \cup \{s\}) - F(B)$。简单地说就是收益递减。并且，submodular函数的非负线性组合也是submodular的。也就是说，只要$C(y)$和$V(y)$是submodular的，那么目标函数$C(y)+V(y)$就是submodular的。文章在设计$C(y)$和$V(y)$函数的时候已经考虑到了这一点。

显然$C(y)$是modular的，无论y的大小如何，每增加一个套装o_j，$C(y)$都会增大相同的大小$c(o_j)$。

至于$V(y)$，如果对$y_D \subseteq y_B$都添加套装s，那么

$$
\begin{aligned}
& v_{y_B \cup \{s\}}(z_i) - v_{y_B}(z_i) \\
&= \prod_{o_j \in y_B \backslash y_D} \left(1 - P(z_i|o_j)\right) \prod_{o_j \in y_D} \left(1 - P(z_i|o_j)\right) P(z_i|s) \\
&= \prod_{o_j \in y_B \backslash y_D} \left(1 - P(z_i|o_j)\right) \left(v_{y_D \cup \{s\}}(z_i) - v_{y_D}(z_i)\right) \\
&\leq v_{y_D \cup \{s\}}(z_i) - v_{y_D}(z_i)
\end{aligned}
$$

所以，$V(y)$是submodular的。同理$V'(y)$也是submodular的。

胶囊衣柜算法面临的最后一个挑战是，我们在迭代增长胶囊衣柜的大小时，每次添加的是一件衣服而不是一个套装，例如在第i层添加一件衣服$s_i^{j_t}$，会给y增加$s_i^{j_t} \times \prod_{i' \neq i} A_{i'(t-1)}$的套装。我们的目标函数在添加单个套装的时候是submodular的，但在添加一个集合的套装时却不是这样。但是可以证明，如果我们固定胶囊衣柜的其他层，一次只对某一层添加一件衣物，这时虽然仍然增加了一系列的套装，但目标函数仍然是submodular的（详见原文的Supplemental Material），因此贪婪解接近最优解仍然是成立的。

利用这一特点，文章设计了一种类似EM的迭代算法来求解胶囊衣柜。算法 1 给出了完整的步骤。

Algorithm 1 Proposed iterative greedy algorithm for submodular maximization, where $\mathbf{obj}(\mathbf{y}) := C(\mathbf{y}) + V(\mathbf{y})$.

$A_{iT} := \varnothing, \forall i$
$\Delta_{obj} := \varepsilon + 1$ ▷ ε is the tolerance degree for convergence
$\mathbf{obj}_{prev}^{m-1} := 0$
while $\Delta_{obj}^{m-1} \geq \varepsilon$ **do**
  for each layer $i = 0, 1, ...(m-1)$ **do**
    $A_{iT} = A_{i0} := \varnothing$ ▷ Reset selected pieces in layer i
    $\mathbf{obj}_{cur}^{i} := 0$
    for each time step $t = 1, 2, ...T$ **do**
      $\mathbf{y_{t-1}} = A_{i(t-1)} \times \prod_{i' \neq i} A_{i'T}$
      $s_i^{j_t} := \text{argmax}_{s \in A_i \backslash A_{i(t-1)}} \delta_s$ ▷ Max increment
      where $\delta_s = \mathbf{obj}(\mathbf{y_{t-1}} \uplus s) - \mathbf{obj}(\mathbf{y_{t-1}})$
      $A_{it} := s_i^{j_t} \cup A_{i(t-1)}$ ▷ Update layer i
      $\mathbf{obj}_{cur}^{i} := \mathbf{obj}_{cur}^{i} + \delta_{s_i^{j_t}}$
    end for
  end for
  $\Delta_{obj}^{m-1} := \mathbf{obj}_{cur}^{m-1} - \mathbf{obj}_{prev}^{m-1}$
  $\mathbf{obj}_{prev}^{m-1} := \mathbf{obj}_{cur}^{m-1}$
end while
procedure INCREMENTAL ADDITION ($\mathbf{y_t} := \mathbf{y_{t-1}} \uplus s$)
  $\mathbf{y_t^+} := s, s \in A_i \setminus A_{i(t-1)}$
  for $j \in \{1, \ldots, m\}, j \neq i$ **do**
    if $A_{jT} \neq \varnothing$ **then**
      $\mathbf{y_t^+} := \mathbf{y_t^+} \times A_{jT}$
    end if
  end for
  $\mathbf{y_t} := \mathbf{y_{t-1}} \cup \mathbf{y_t^+}$
end procedure

算法首先选取某一层并固定其他层，将这一层的大小从零迭代至目标大小T，然后依次迭代每一层得到最后的近似解。

总结来说，文章利用了submodular的目标函数可以用贪婪算法求解近似解这一特性，巧妙地设计了目标函数，从而用贪婪算法降低了算法的复杂度。

3. 实验结果

实验结果主要分为两个部分。第一部分是对主题模型计算的兼容性的评价。第二部分是对生成的胶囊衣柜的评价，也可以认为是对多样性的评价。这里要指出的是，虽然我向文章作者要了文章使用的数据集，但因为对图片属性识别部分细节的不了解，因此没有能够复现实验，下面给出的实验结果都是原文的部分实验结果。

3.1. 兼容性的评价

出于兼容性测试的需要，文章需要准备一些不兼容的套装。采取的方法是对元标签不同的套装中的衣物进行互换（fig 4）。采用的对比方法包括将 CTM 模型换成 LDA，以及 MONOMER 和 BILSTM 这两种其他文章里提出的衡量兼容性的算法。

实验结果如 fig 5 所示。从左边的图可以看到，用 CTM 取得的效果明显好于 LDA 和 PolyLDA。这是因

Figure 4 构建不兼容的负样本

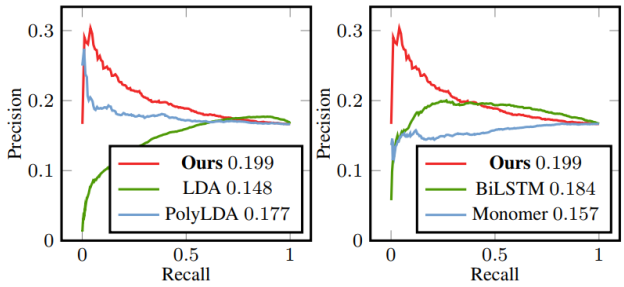

Figure 5 兼容性实验结果

Figure 7 给定套装生成胶囊衣柜，套装在左上方

为 LDA 模型不知道风格之间的相关性（前文提到）。从右边的图可以看出，文章提出的方法明显好于其他两种方法。MONOMER 只能衡量成对衣物的兼容性（计算的时候套装中的每对衣物计算兼容性，并求取平均值），而许多套装的衣物数量是超过两件的，因此难以抓住这些套装的整体兼容性。而 BiLSTM 虽然也是基于正样本训练，但由于文章提出的方法可以从穿着搭配的全身图片中学习套装的兼容性，因此效果更好。

3.2. 生成的胶囊衣柜的评价

生成的方法有两种，第一种是给定一个套装作为胶囊衣柜的初始值，然后添加衣物生成一个目标大小的胶囊衣柜。对比的方法有 naïve-greedy(贪婪算法中先对所有层迭代，再对大小迭代)、MMR 和 CLUSTER CENTERS。实验结果样例如 fig 6，个人感觉两种 greedy 算法的差距不大，但比另外两种算法更能抓住给出的套装的特点，比如第一个的单调色彩和第二个的强烈对比。

第二种是给定用户喜欢的一些套装，计算用户对不同风格的偏好，然后从零开始生成一个胶囊衣柜。结果如 fig 7 所示，感觉效果还可以。

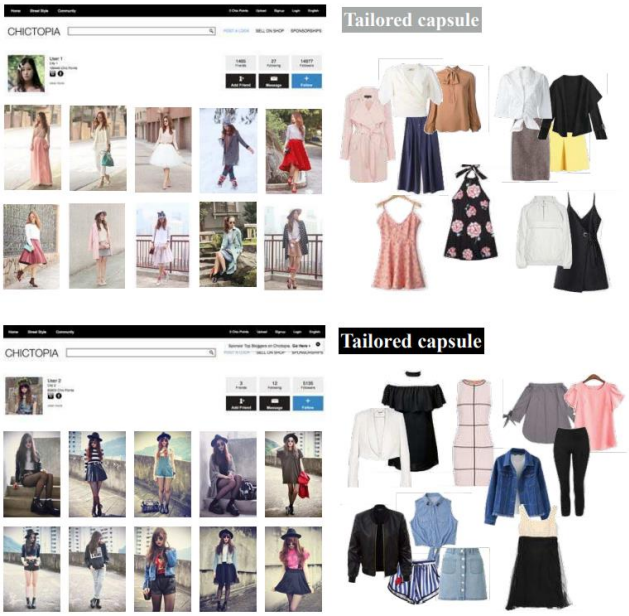

Figure 6 给定喜好生成胶囊衣柜

4. 总结

个人觉得，文章最大的亮点在于可以从街头的穿着搭配图片学习套装的兼容性，这极大地扩展了学习的范围，具有很强的实用性。此外，生成胶囊衣柜的方式也很多样，对于不会穿搭的人来说，具有较好的用户体验和可操作性。

可能的不足之处在于，对于图片中套装的属性识别很依赖于图片识别模型的准确性。另外，各种属性都是人工定义（比如蕾丝、颜色等），对图片的表征能力可能受到限制，或许可以考虑用无监督的方法来生成属性。

References

[1] Hsiao, Wei-Lin, and Kristen Grauman. Creating capsule war drobes from fashion images. Proceedings of the IEEE Confe rence on Computer Vision and Pattern Recognition. 2018.

[2] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analy sis of approximations for maximizing submodular set functi ons. Mathematical Programming, 1978.