למידה עמוקה 2022ב

הערות לפתרון ממ"ן 11

שאלה 1

היו הרבה מימושים מוצלחים, וגם כאלו שמימשו את שיטת ה-alias table תחת המגבלה של לולאה יחידה.

שימו לב שהשימוש במתודה `apply` הולך כנגד הרעיון של שימוש בפונקציות מובנות של PyTorch לצורך ייעול החישובים: `apply` עוטפת פונקציה פייתונית, אך לא מתרגמת ומקמפלת אותה לשפה יעילה יותר, כך שעלות השימוש בפייתון עודנה קיימת.

אני מציע את הפתרון הבא, נסו להבין איך הוא עובד.

```
def my_sampler(size,dist,requires_grad=False):
  import torch
  A=torch.rand(size)
  out=torch.zeros(size)
  dist=torch.tensor(dist)
  cdf=torch.cumsum(dist,dim=0)
  for i in range(dist.numel()-1):
    B=A>cdf[i]
    out=out+B
  out.requires_grad=requires_grad
  return out
```

שאלה 2

הרוב פתרו תוך שימוש בלולאה שרצה על המימדים ושימוש חוזר בפונקציה `cat`, וזו באמת אחת הדרכים הטבעיות.

אני מציע פתרון יותר low-level, שעובר על כל איברי הטנזור החדש (בהנחה שבדקנו כבר שניתן לשדר את B למימדי A, וששמרנו את מימדי הטנזור B כולל מימדים מנוונים שיש להוסיף במשתנה `Bsize`).

```
for memory_idx in range(A.numel()):
  C_idx=np.unravel_index(memory_idx, A.size())
  tensor_idx=torch.tensor(C_idx)
  B_idx=torch.min(tensor_idx,Bsize-1)
  B_idx=tuple(B_idx.squeeze().tolist())
  C[C_idx]=B[B_idx]
```

הפונקציה `unravel_index` ממירה אידנקס חד מימדי לאינדקס רב מימדי לפי הגודל של `A`, כך שבלולאה אחת אנו עוברים על כל איברי C מה"ראשון" (בזיכרון) עד לאחרון.

שאלה 3

ראו כאן מימוש של מערכת דומה:

https://sidsite.com/posts/autodiff/

שימו לב שבהמשך העמוד עוברים לשמור את הפונקציה לפיה יש לחשב את הנגזרת, במקום את הערך עצמו - בדומה למערכת ה-Autograd של PyTorch.