یادگیری عمیق

مدرس: محمدرضا محمدی

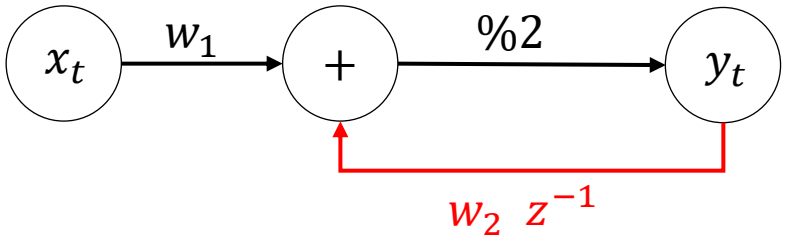
بهار ۱۴۰۲

شبکه‌های عصبی بازگشتی

Recurrent Neural Networks

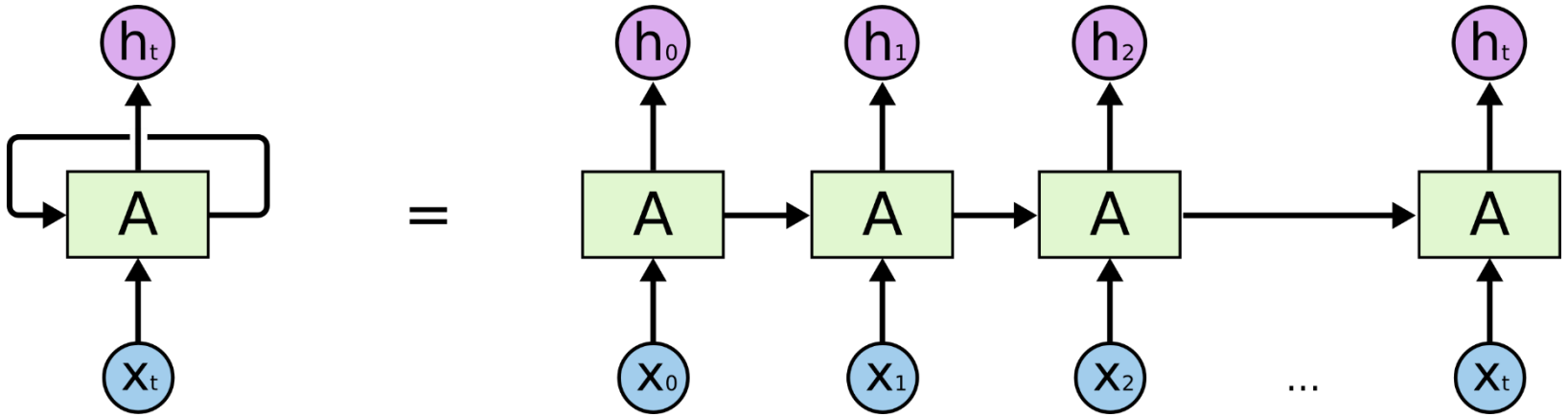
انگیزه

- در تمام مسئله‌ها نمی‌توان طول ورودی‌ها و خروجی‌ها را ثابت در نظر نگرفت
- در مسئله‌هایی مانند بازشناسی گفتار یا پیش‌بینی سری زمانی نیاز به سیستمی است که اطلاعات زمینه را ذخیره کند و از آنها به درستی استفاده نماید
- مثال ساده: اگر تعداد 1های یک دنباله فرد باشد خروجی 1 و در غیر اینصورت خروجی 0 تولید شود
  - 1000010101: 0 ، 1000110000000000000: 1 و ...
- انتخاب یک پنجره با طول ثابت سخت/غیرممکن است

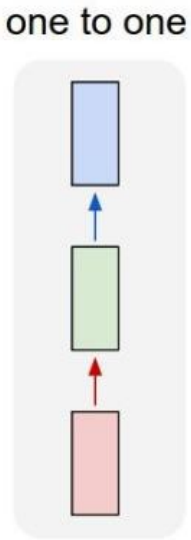


شبکه‌های عصبی بازگشتی

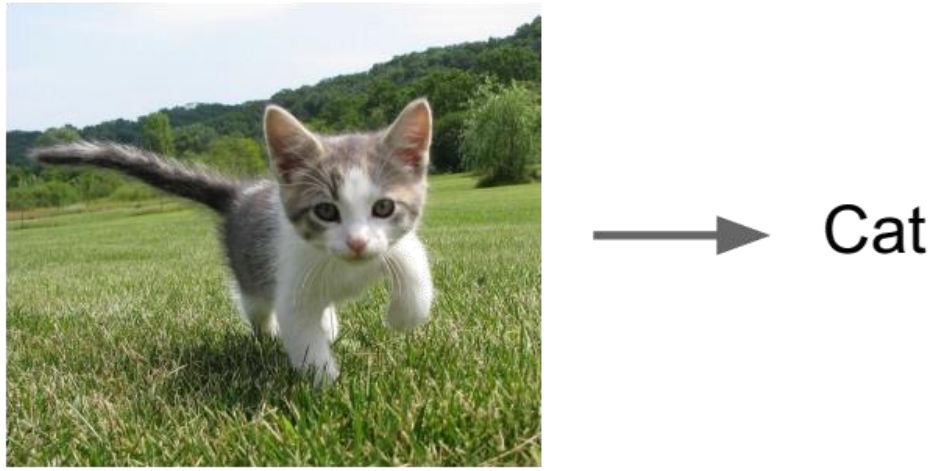
- شبکه‌های عصبی بازگشتی (RNNها) خانواده‌ای از شبکه‌های عصبی برای پردازش داده‌های دنباله‌ای هستند
  - دنباله‌ای از مقادیر $\boldsymbol{x}^{(1)}$، ...، $\boldsymbol{x}^{(\tau)}$
- اغلب شبکه‌های بازگشتی می‌توانند دنباله‌هایی با طول متغیر را نیز پردازش کنند
- یک RNN وزن‌های یکسانی را در چندین مرحله زمانی به اشتراک می‌گذارد



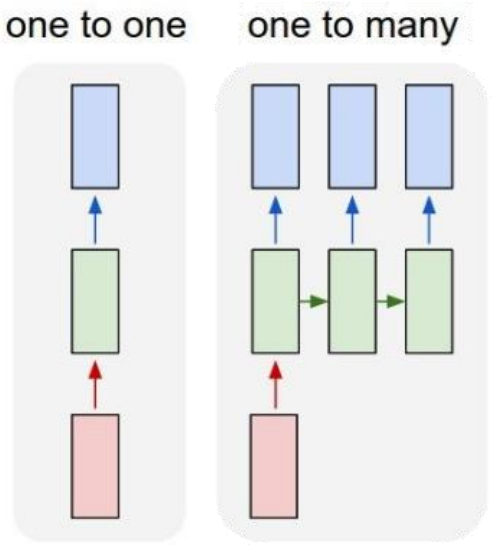
شبکه‌های عصبی



- لایه‌های FC و Conv دارای حافظه نیستند!
  - ورودی‌های خود را به صورت مستقل پردازش می‌کنند (بدون هیچ حالتی در بین آنها)
- به چنین شبکه‌هایی پیش‌خور (feedforward) می‌گویند



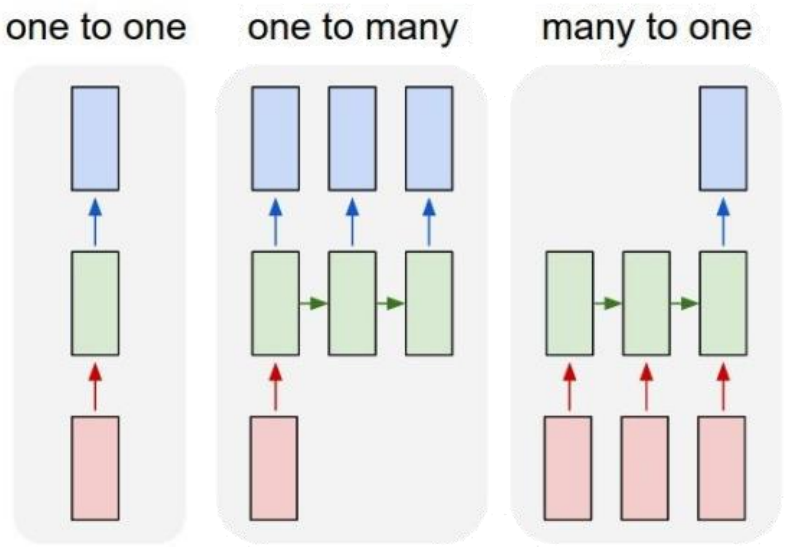
شبکه‌های عصبی



- خروجی این مثال دنباله‌ای از کلمات است که می‌تواند طول متغیر داشته باشد

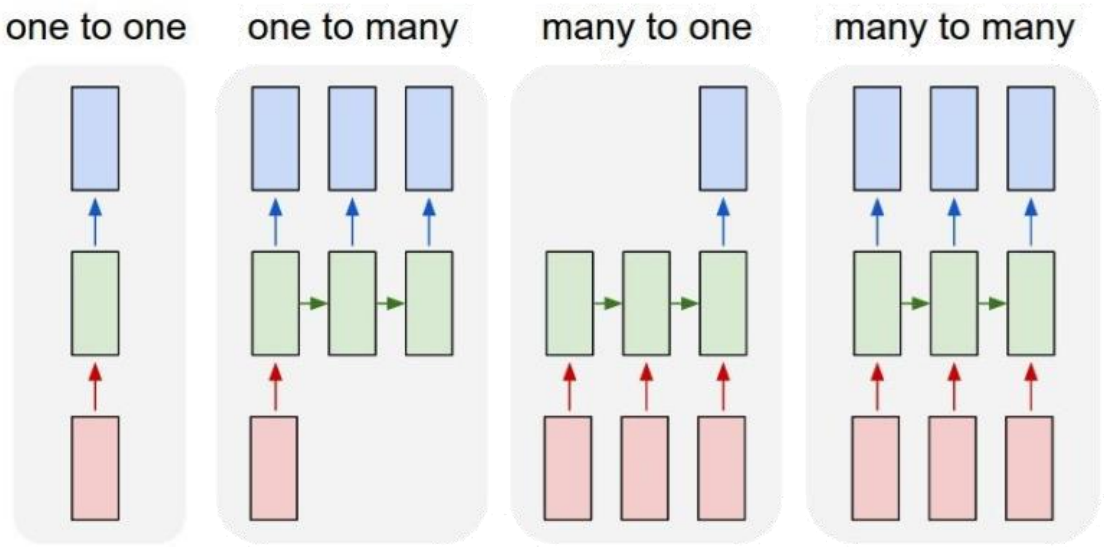
A cat is sitting on a tree branch

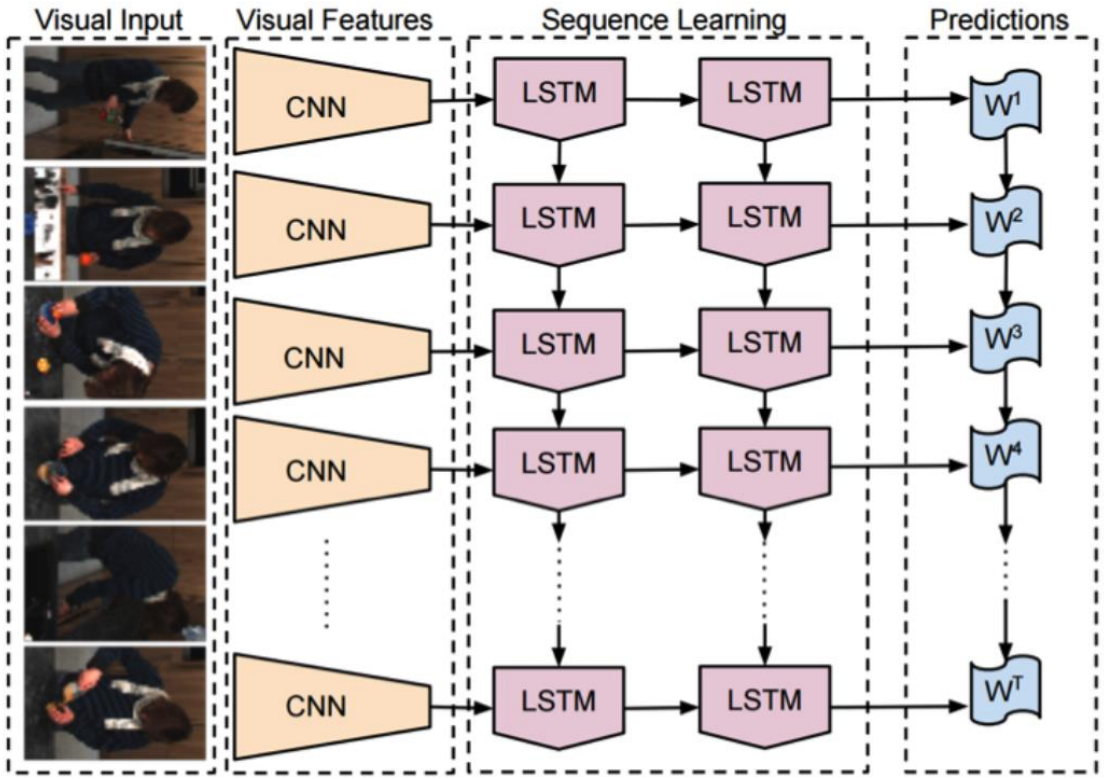
شبکه‌های عصبی



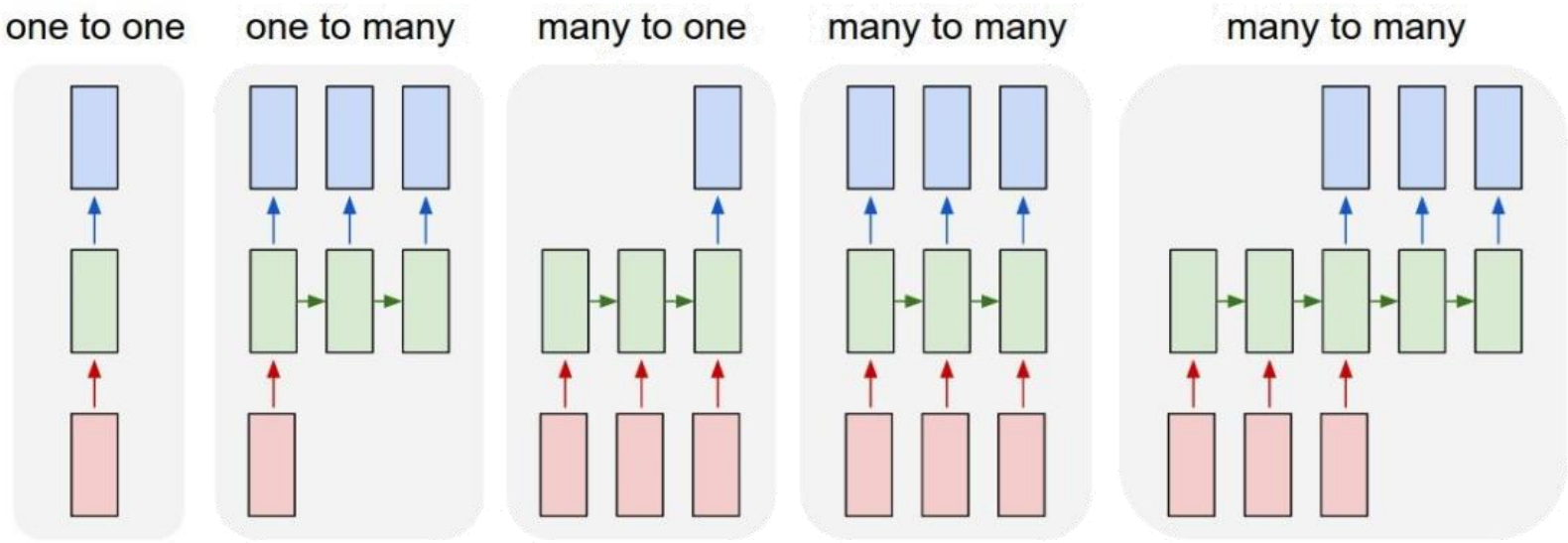
- در دسته‌بندی یک فایل صوتی، طول ورودی می‌تواند متغیر باشد

شبکه‌های عصبی





شبکه‌های عصبی

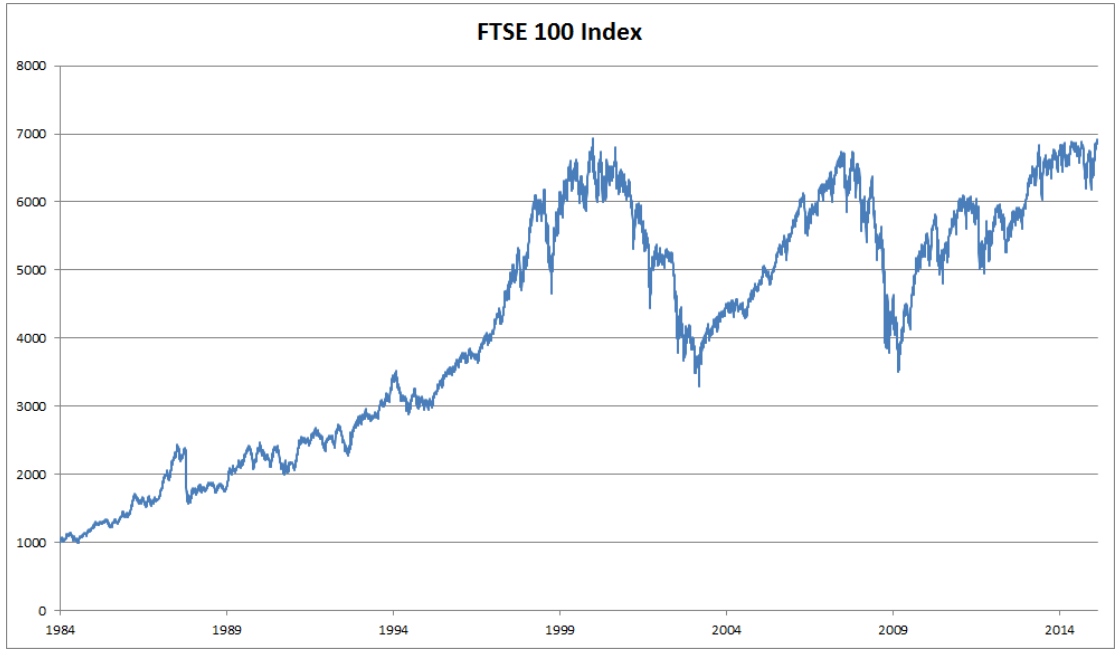


Input:	If you face a problem try to find the solution not the reason
Output:	مشکل که به وجود اومد بگرد راه حلش را پیدا کن نگرد دنبال این که چرا به وجود اومد

ابزارهای آماری

- برای کار با داده‌های دنباله‌ای به ابزارهای آماری و معماری‌های جدید نیاز داریم
- به عنوان مثال، قیمت سهام (شاخص FTSE 100) را برای یک بازه حدود ۳۰ ساله در نظر بگیرید
  - قیمت در گام زمانی t را با x_t نشان می‌دهیم
  - می‌خواهیم قیمت سهام را تخمین بزنیم

$$x_t \sim P(x_t \mid x_{t-1}, \dots, x_1)$$

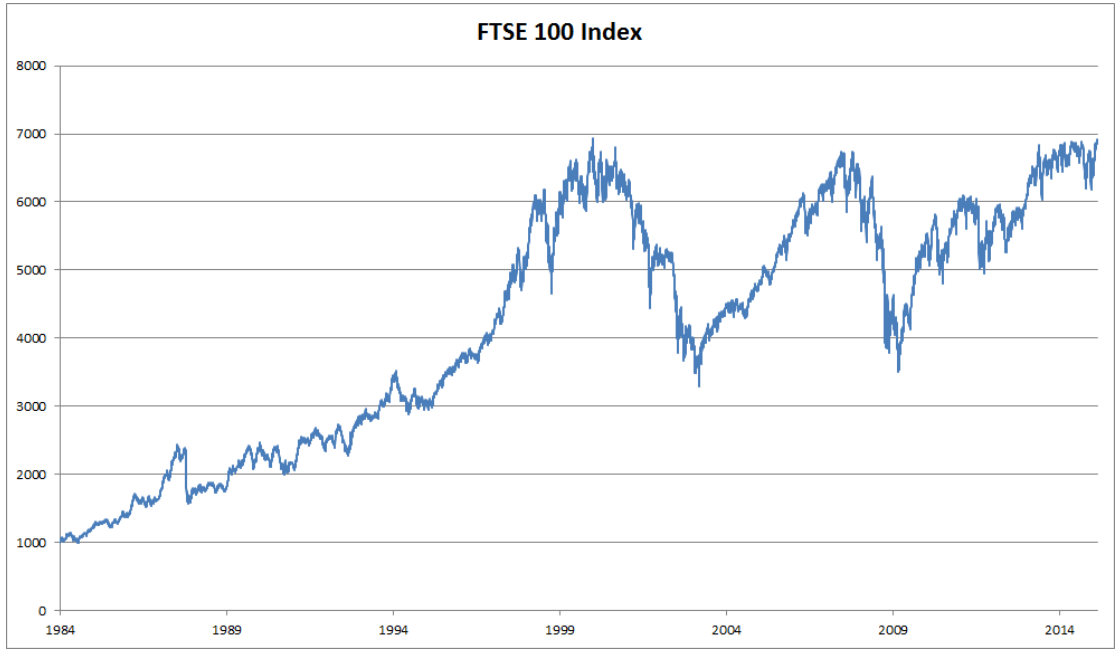


مدل‌های Autoregressive

- برای تخمین قیمت سهام در این مثال، می‌توان از مدل‌های رگرسیون استفاده کرد

$$x_t \sim P(x_t \mid x_{t-1}, \dots, x_1)$$

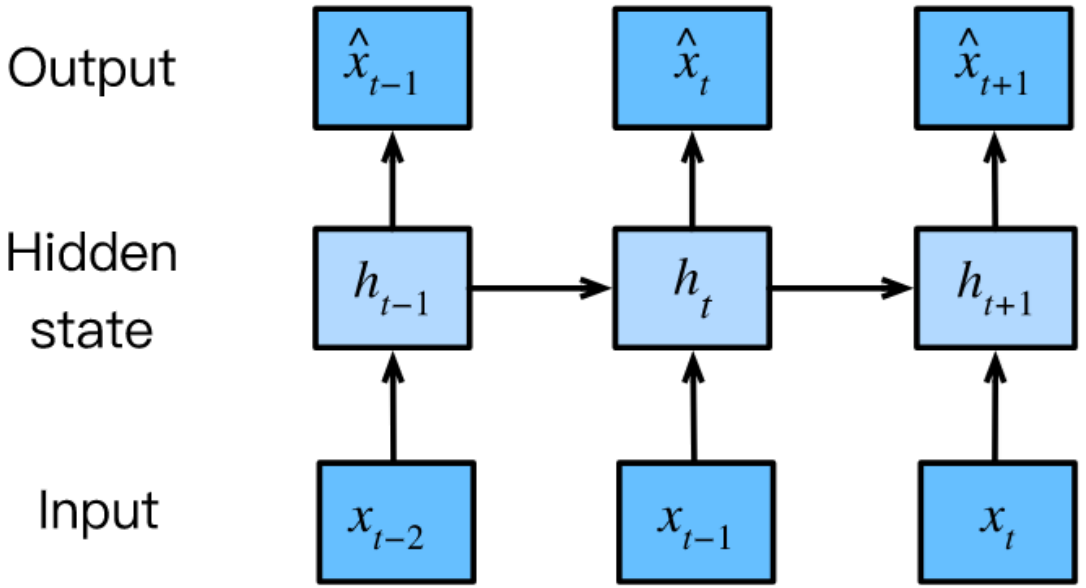
  - یک مشکل وجود دارد: تعداد ورودی‌ها $x_{t-1}, \dots, x_1$ بسته به زمان t تغییر می‌کند
  - تعداد ورودی‌ها دائم زیاد می‌شود و نیاز است تا به خوبی تقریب زده شود



- ممکن است بتوان فرض کرد که در واقع دنباله طولانی $x_{t-1}, \dots, x_1$ مورد نیاز نیست
  - تنها از $x_{t-1}, \dots, x_{t-\tau}$ با طول τ استفاده خواهد شد
  - حداقل برای $t > \tau$ طول دنباله ثابت خواهد بود
  - می‌توان از مدل‌های عمیق فعلی برای آموزش استفاده کرد
  - به این مدل‌ها Autoregressive گفته می‌شود

مدل‌های Autoregressive

- استراتژی دوم این است که خلاصه‌ای از مشاهدات گذشته h_t را نگه داریم
- در هر گام، علاوه بر پیش‌بینی $\hat{x}_t$ با استفاده از h_t، مقدار h_t نیز به روز می‌شود



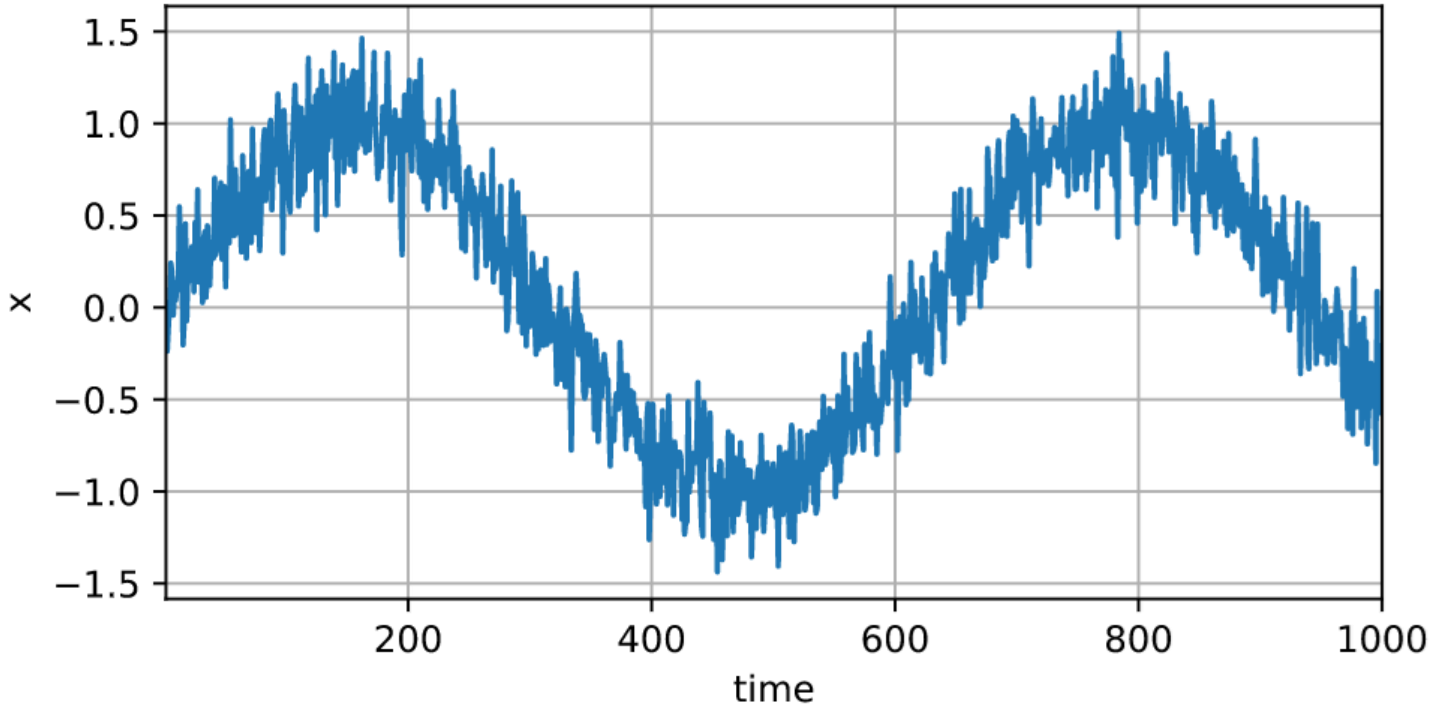
$$\hat{x}_t = P(x_t \mid h_t)$$

$$h_t = g(h_{t-1}, x_{t-1})$$

- از آنجایی که h_t هرگز مشاهده نمی‌شود، به این مدل‌ها، Autoregressive پنهان نیز می‌گویند

Fig. 8.1.2 A latent autoregressive model.

مثال شبیه‌سازی




```
%matplotlib inline
import torch
from torch import nn
from d2l import torch as d2l

T = 1000  # Generate a total of 1000 points
time = torch.arange(1, T + 1, dtype=torch.float32)
x = torch.sin(0.01 * time) + torch.normal(0, 0.2, (T,))
d2l.plot(time, [x], 'time', 'x', xlim=[1, 1000], figsize=(6, 3))
```

مثال شبیه‌سازی

- برای آموزش مدل، نیاز است تا از این دنباله نمونه‌هایی شامل ویژگی‌ها و برچسب ایجاد کنیم

$$y_t = x_t$$

$$\mathbf{x}_t = [x_{t-\tau}, \dots, x_{t-1}]$$

- برای زمان‌های $t > \tau$، می‌توان از padding استفاده کرد یا از آنها صرف نظر کرد

```
tau = 4
features = torch.zeros((T - tau, tau))
for i in range(tau):
    features[:, i] = x[i: T - tau + i]
labels = x[tau:].reshape((-1, 1))

batch_size, n_train = 16, 600
# Only the first 'n_train' examples are used for training
train_iter = d2l.load_array((features[:n_train], labels[:n_train]),
                            batch_size, is_train=True)
```

مثال شبیه‌سازی

- از یک مدل MLP دو لایه با تابع ضرر MSE استفاده می‌کنیم

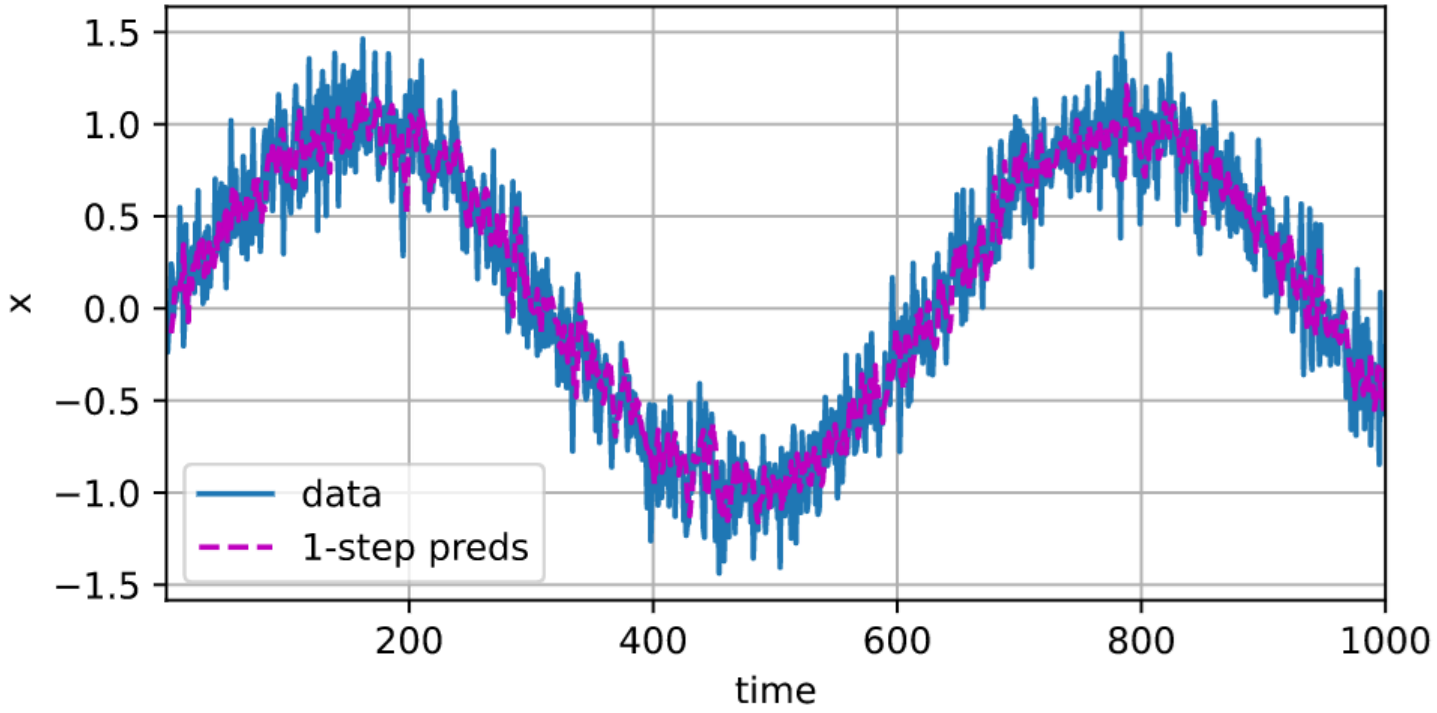
$$y_t = x_t$$

$$\mathbf{x}_t = [x_{t-\tau}, \dots, x_{t-1}]$$

```
# A simple MLP
def get_net():
    net = nn.Sequential(nn.Linear(4, 10),
                        nn.ReLU(),
                        nn.Linear(10, 1))
    net.apply(init_weights)
    return net
```

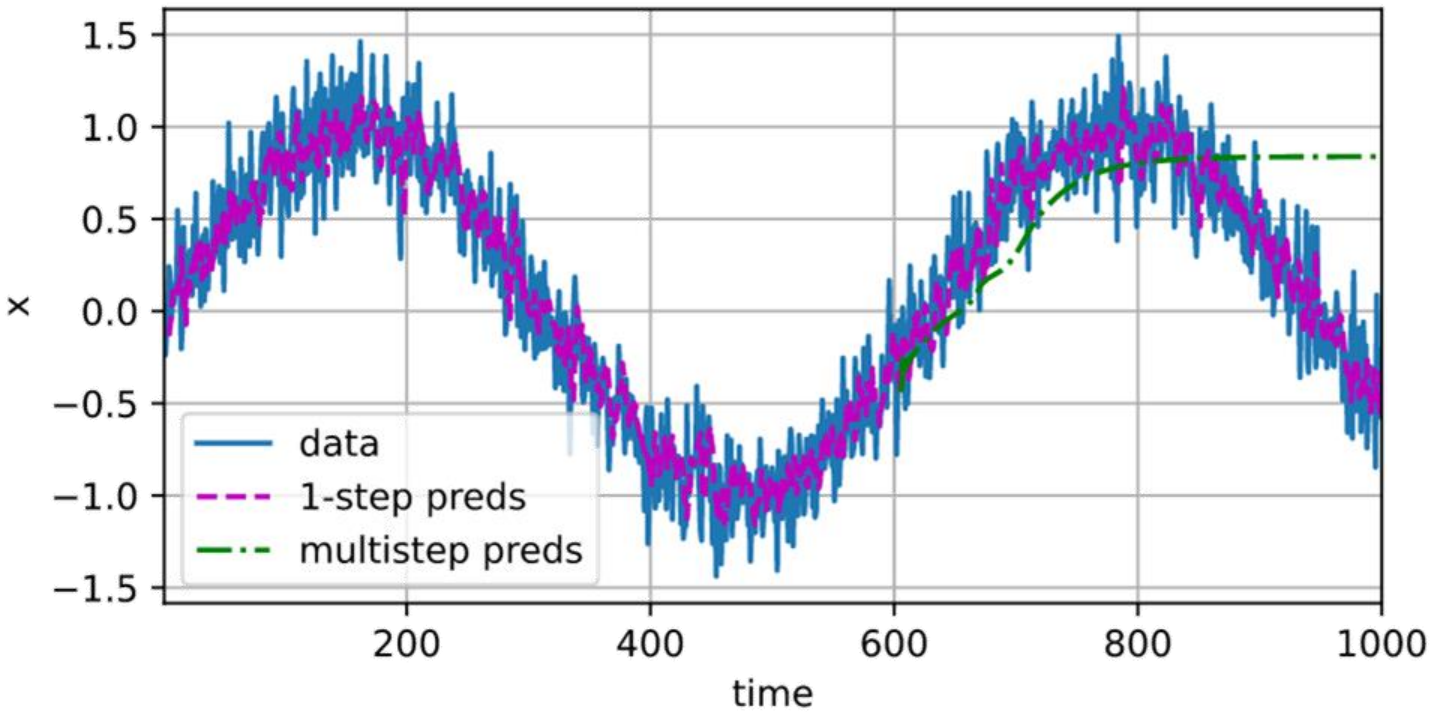
مثال شبیه‌سازی

- پس از آموزش مدل با ۶۰۰ نمونه ابتدایی، آن را بر روی کل دنباله اعمال می‌کنیم
- پیش‌بینی یک گام جلوتر برای تمام داده‌ها مناسب به نظر می‌رسد



مثال شبیه‌سازی

- پس از آموزش مدل با ۶۰۰ نمونه ابتدایی، آن را بر روی کل دنباله اعمال می‌کنیم
- پیش‌بینی یک گام جلوتر برای تمام داده‌ها مناسب به نظر می‌رسد
- چگونه می‌توانیم پیش‌بینی را برای گام‌های بیشتر انجام دهیم؟
  - می‌توان از مقادیر پیش‌بینی شده استفاده کرد
  - به آن پیش‌بینی چندگام گفته می‌شود



$$\hat{x}_{605} = f(x_{601}, x_{602}, x_{603}, x_{604})$$

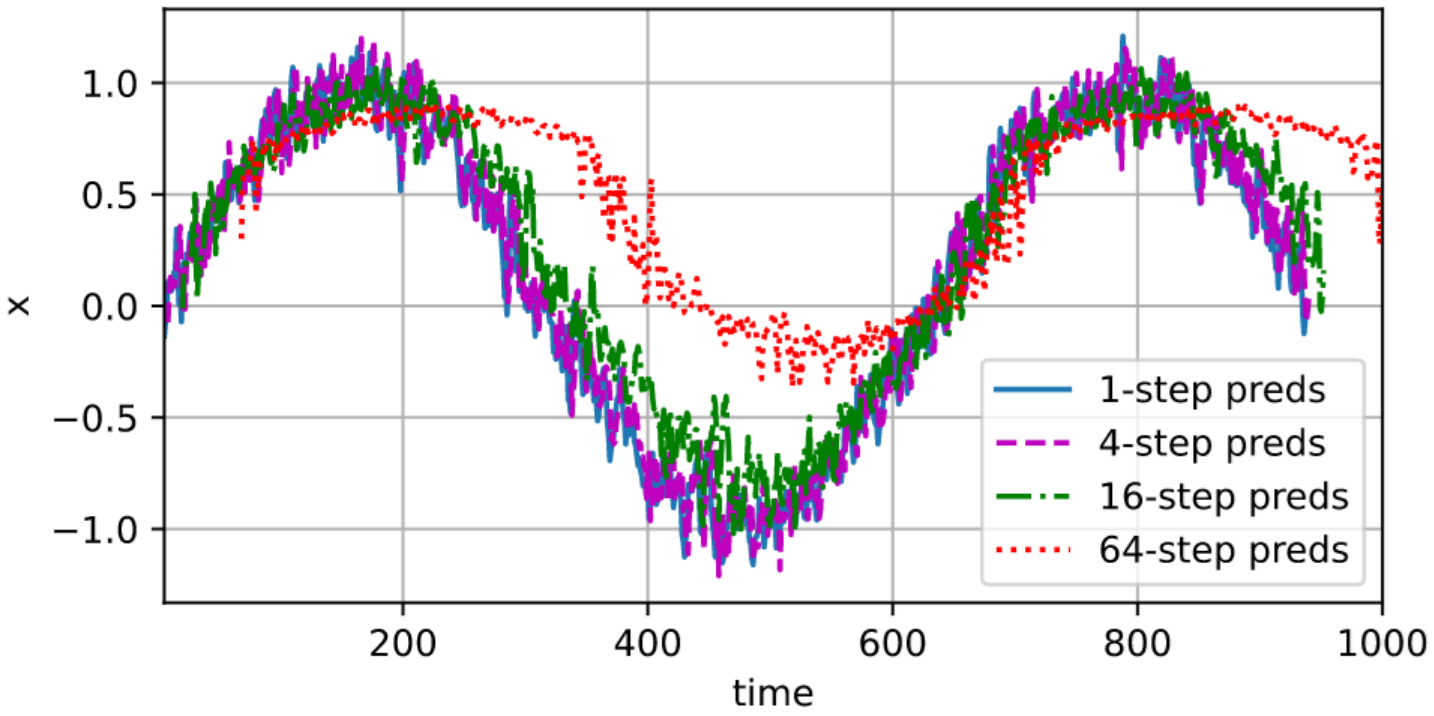
$$\hat{x}_{606} = f(x_{602}, x_{603}, x_{604}, \hat{x}_{605})$$

$$\vdots$$

$$\hat{x}_{609} = f(\hat{x}_{605}, \hat{x}_{606}, \hat{x}_{607}, \hat{x}_{608})$$

مثال شبیه‌سازی

- در این مثال پیش‌بینی‌ها بعد از چند گام به یک مقدار ثابت تبدیل می‌شوند و خطای پیش‌بینی بسیار زیاد خواهد بود
- به عنوان آخرین آزمایش، پیش‌بینی k گام جلوتر را در تمام دنباله محاسبه می‌کنیم
- همانطور که مشاهده می‌شود، هر مقدار گام دورتری را با این رویکرد پیش‌بینی می‌کنیم، خطای آن بیشتر می‌شود



داده‌های متنی

- متن را می‌توان به عنوان دنباله‌ای از کاراکترها یا کلمات در نظر گرفت
- ورودی شبکه‌های عصبی نمی‌تواند متن خام باشد: فقط با تنسورهای عددی کار می‌کنند
- بردارسازی متن:
  - تقسیم متن به کلمه‌ها، و تبدیل هر کلمه به یک بردار
  - تقسیم متن به کاراکترها، و تبدیل هر کاراکتر به یک بردار
  - استخراج n-gramها از کلمه‌ها یا کاراکترها، و تبدیل هر n-gram به یک بردار
    - n-gramها گروه‌های دارای همپوشانی متشکل از چند کلمه یا کاراکتر متوالی هستند

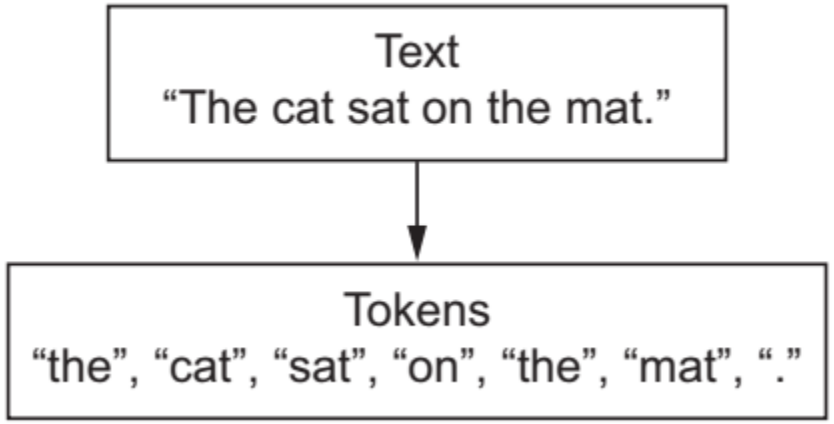
```
The cat sat on the mat
```

Bi-grams `{"The cat", "cat sat", "sat on", "on the", "the mat"}`

Tri-grams `{"The cat sat", "cat sat on", "sat on the", "on the mat"}`

Tokenization

- واحدهای مختلفی که می‌توان متن را به آنها تجزیه کرد (کلمات، کاراکترها یا n-gram) Token نامیده می‌شوند و شکستن متن به آنها Tokenization نامیده می‌شود
- سپس برای هر Token باید یک کد در نظر گرفت
  - نیاز است تا مجموعه واژگان مورد نظر مشخص شود



واژگان

- برای هر کدام از واژه‌ها یک اندیس عددی از ۰ تعریف می‌کنیم
- برای این کار، ابتدا توکن‌های منحصربه‌فرد را در تمام مجموعه آموزشی می‌شماریم و سپس به هر توکن یکتا با توجه به فراوانی آن، یک شاخص عددی اختصاص می‌دهیم
- توکن‌هایی که به ندرت ظاهر می‌شوند اغلب برای کاهش پیچیدگی حذف می‌شوند
  - برای تمام آنها از یک توکن خاص ناشناخته اختصاص داده می‌شود "<unk>"
- چند توکن خاص دیگر نیز معمولا تعریف می‌شوند مانند: "<pad>" برای padding، "<bos>" برای شروع یک دنباله، و "<eos>" برای پایان یک دنباله
- مثال:

```
[('<unk>', 0), ('the', 1), ('i', 2), ('and', 3), ('of', 4), ('a', 5), ('to', 6), ...
```

واژگان

- با داشتن واژگان می‌توان به سادگی یک متن را به یک دنباله از اعداد تبدیل کرد

```
words: ['twinkled', 'and', 'his', 'usually', 'pale', 'face', 'was', 'flushed', 'and', ...
indices: [2186, 3, 25, 1044, 362, 113, 7, 1421, 3, ...
```

- آیا این دنباله برای آموزش مدل مناسب است؟
  - فاصله هندسی میان واژه‌ها متناسب با فاصله معنایی کلمات نیست
  - با کد یک‌بعدی نمی‌توان به نحوی کدگذاری کرد که فاصله کلمات متناسب با فاصله معنایی آنها باشد
  - در ادامه نحوه اختصاص یک بردار به واژه‌ها را بررسی می‌کنیم

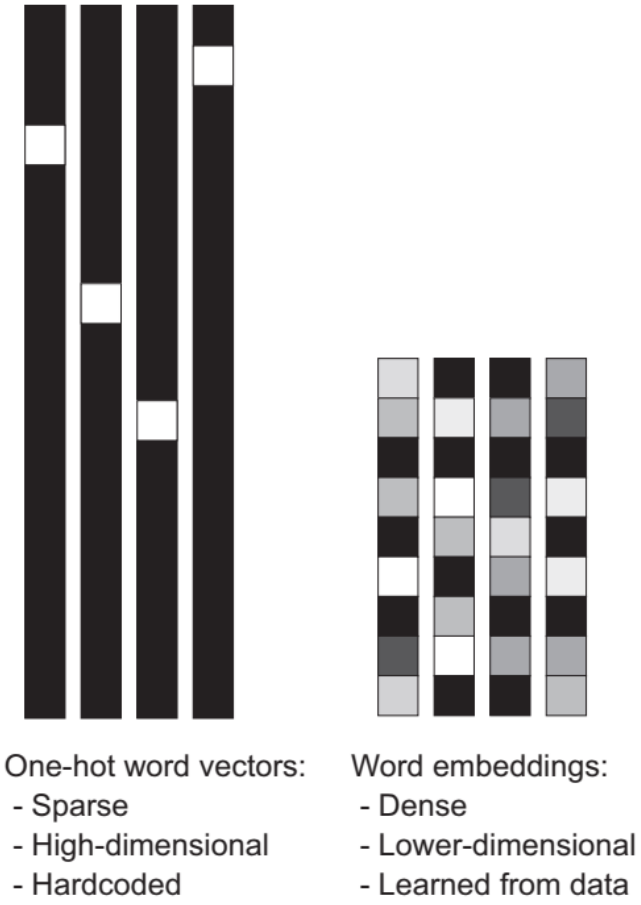
One-hot encoding

- ابتدایی‌ترین راه برای تبدیل توکن به بردار است
- یک عدد صحیح منحصر به فرد به هر کلمه اختصاص می‌یابد و سپس این عدد صحیح i به یک بردار باینری با اندازه N (اندازه واژگان) تبدیل می‌شود
- همه مقادیر این بردار صفر است به جز ورودی iام که ۱ است
- این کار در سطح کاراکتر و n-gram هم قابل انجام است

One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded

جانمایی کلمات (Word embedding)

- جانمایی کلمات اطلاعات بیشتر را در ابعاد بسیار کمتری قرار می‌دهد
- این بردارها را می‌توان با استفاده از حجم زیادی از متن پیش‌آموزش داد و در مجموعه داده‌های کوچک از آنها استفاده کرد



جانمایی کلمات (Word embedding)

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97
⋮	⋮	⋮				
size						
cost						
alive						
verb						

300

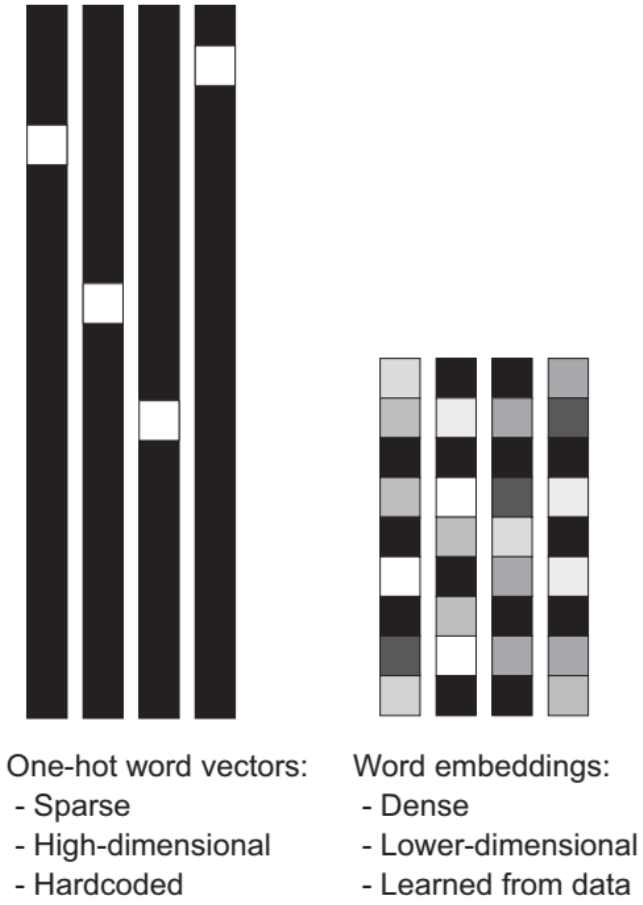
e_{5391} e_{9853}

Andrew Ng

https://www.coursera.org/lecture/nlp-sequence-models/word-representation-6Oq70

جانمایی کلمات (Word embedding)

- دو روش برای دستیابی به جانمایی کلمات وجود دارد:
  - آموزش همزمان با مسئله اصلی
    - مقداردهی اولیه به صورت تصادفی
  - بارگذاری مقادیری که از قبل بر اساس آموزش یک مسئله دیگر بدست آمده‌اند
    - جانمایی کلمات پیش‌آموخته

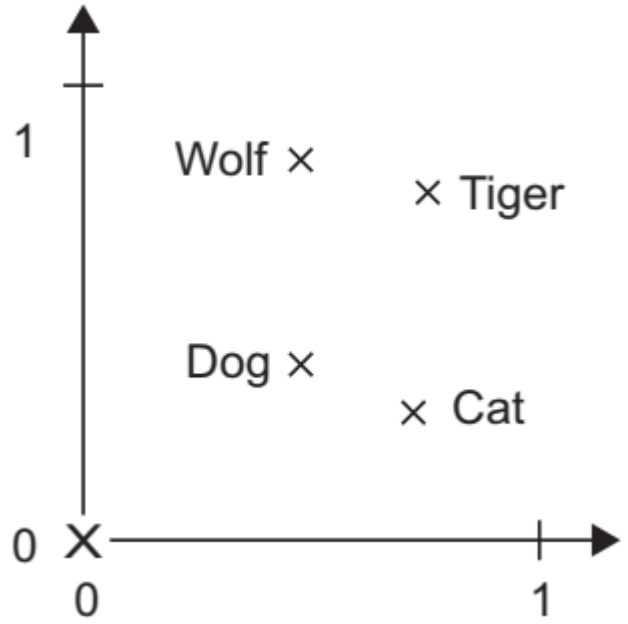


آموزش جانمایی کلمات

- اختصاص یک بردار تصادفی به هر کلمه
  - مشکل این رویکرد این است که این فضا ساختاری ندارد
    - به عنوان مثال، کلمات accurate و exact ممکن است با جانمایی‌های کاملاً متفاوتی همراه شوند، با این وجود که در اکثر جملات قابل تعویض هستند
- روابط هندسی بین بردارهای کلمات باید منعکس‌کننده روابط معنایی بین این کلمات باشد
  - جانمایی کلمات به معنای نگاشت زبان انسان به یک فضای هندسی است
  - در یک فضای جانمایی معقول، انتظار می‌رود کلمات مترادف دارای مقادیر مشابهی باشند

آموزش جانمایی کلمات

- انتظار می‌رود فاصله هندسی بین هر دو بردار کلمه با فاصله معنایی بین کلمات مرتبط باشد
- همچنین، انتظار می‌رود جهت‌های مختلف در فضای آموخته شده معنادار باشند



آموزش جانمایی کلمات

- انتظار می‌رود فاصله هندسی بین هر دو بردار کلمه با فاصله معنایی بین کلمات مرتبط باشد
- همچنین، انتظار می‌رود جهت‌های مختلف در فضای آموخته شده معنادار باشند
- در این مثال، جهت بردار از گربه به ببر و از سگ به گرگ مشابه است
  - از حیوان خانگی به حیوان وحشی
- جهت بردار از سگ به گربه و از گرگ به ببر نیز مشابه است
  - از خانواده سگ به خانواده گربه