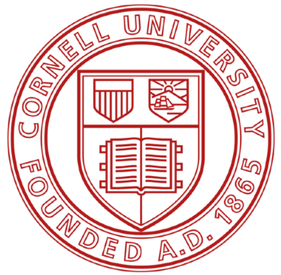
Lecture 2: Maximum likelihood estimation

CS 5788: Introduction to Generative Models



Reminders

- PS1 out soon
- Due on Feb. 10
- Covers material from Lectures 2 and 3.
- Links on webpage to linear algebra and probability reviews from Kevin Murphy's "Probabilistic Machine Learning" book.
- New (still tentative!) midterm date: April 17, 2-5pm.
- Lecture recordings should be available.
- Ask questions on Ed Discussion.

Training a discriminative model

Training data

$\{x_1, y_1\}$
$\{x_2, y_2\}$
$\{x_3, y_3\}$
$\ldots$

$\rightarrow$ Learner $\rightarrow$ $p_\theta(y \mid \mathbf{X})$

Model parameters

Adapted from Isola, Torralba, Freeman

Training a discriminative model

Training data

X Y


"bowling alley"


"desert"

$\vdots$

$\rightarrow$ Learner $\rightarrow$ $p_\theta(y \mid \mathbf{X})$

Adapted from Isola, Torralba, Freeman

Training a generative model

Training data

X

$\rightarrow$ Learner $\rightarrow$ $p_\theta(\mathbf{X})$

Adapted from Isola, Torralba, Freeman

Training a *conditional* generative model

Training data

Y X

{ "bowling alley" ,  }

{ "desert" ,  }

⋮

$\rightarrow$ Learner $\rightarrow$ $p_\theta(\mathbf{X} \mid y)$

Adapted from Isola, Torralba, Freeman

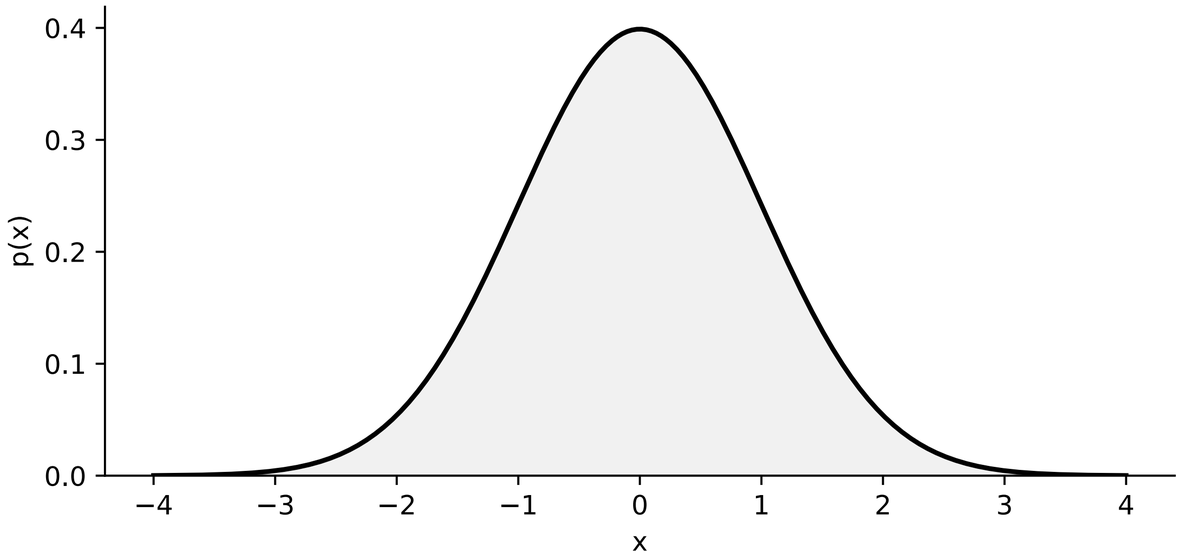
Learning simple generative models

Gaussian distribution

Probability density function (pdf):

$$p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Parameters θ: mean μ, standard deviation σ.



Example: $\mu = 0, \sigma = 1$

Gaussian distribution

Probability density function (pdf):

$$p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Parameters θ: mean μ, standard deviation σ.

Integrates to 1:

$$\int_x p_\theta(x)dx = 1$$

Expected value:

$$\mathbb{E}[X] = \int_x x p_\theta(x)dx = \mu$$

Variance:

$$\text{Var}[X] = \mathbb{E}[(X-\mu)^2] = \int_x (x-\mu)^2 p_\theta(x)dx = \sigma^2$$

Maximum likelihood estimation (MLE)

Goal: Find best parameters θ.

How do we quantify this? One option is **maximum likelihood estimation**.

Suppose we have a sample of points $x_1, x_2, \ldots, x_N$ from the distribution. We can find θ by maximizing their likelihood under the model:

$$\text{argmax}_\theta \prod_{i=1}^{N} p_\theta(x_i)$$

Assuming data is **iid** (independent and identically distributed).

$$= \text{argmax}_\theta \sum_{i=1}^{N} \log(p_\theta(x_i))$$

Equivalent to *minimizing* the **negative log likelihood:** $\text{NLL}(\theta) = -\sum_{i=1}^{N} \log(p_\theta(x_i))$

MLE for a Gaussian

$$\text{argmax}_\theta \sum_{i=1}^{N} \log(p_\theta(x_i))$$

Gaussian pdf

$$= \text{argmax}_{\mu,\sigma} \sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\right)$$

$$= \text{argmax}_{\mu,\sigma} \sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + -\frac{(x_i-\mu)^2}{2\sigma^2}$$

How do we solve for μ and σ? Take derivatives and set to 0!

MLE for a Gaussian

Let's set $\frac{\partial}{\partial \mu} \sum_{i=1}^{N} \log(p_\theta(x_i)) = 0$ and solve for μ

$$0 = \frac{\partial}{\partial \mu} \sum_{i=1}^{N} \log(p_\theta(x_i)) = \sum_{i=1}^{N} \frac{\partial}{\partial \mu} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + -\frac{(x_i - \mu)^2}{2\sigma^2}$$

$$= \sum_{i=1}^{N} \frac{(x_i - \mu)}{\sigma^2} = \frac{1}{\sigma^2}\left[\sum_{i=1}^{N} x_i - N\mu\right] \quad \Longrightarrow \quad \mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Similarly, can show $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$

Multivariate Gaussian

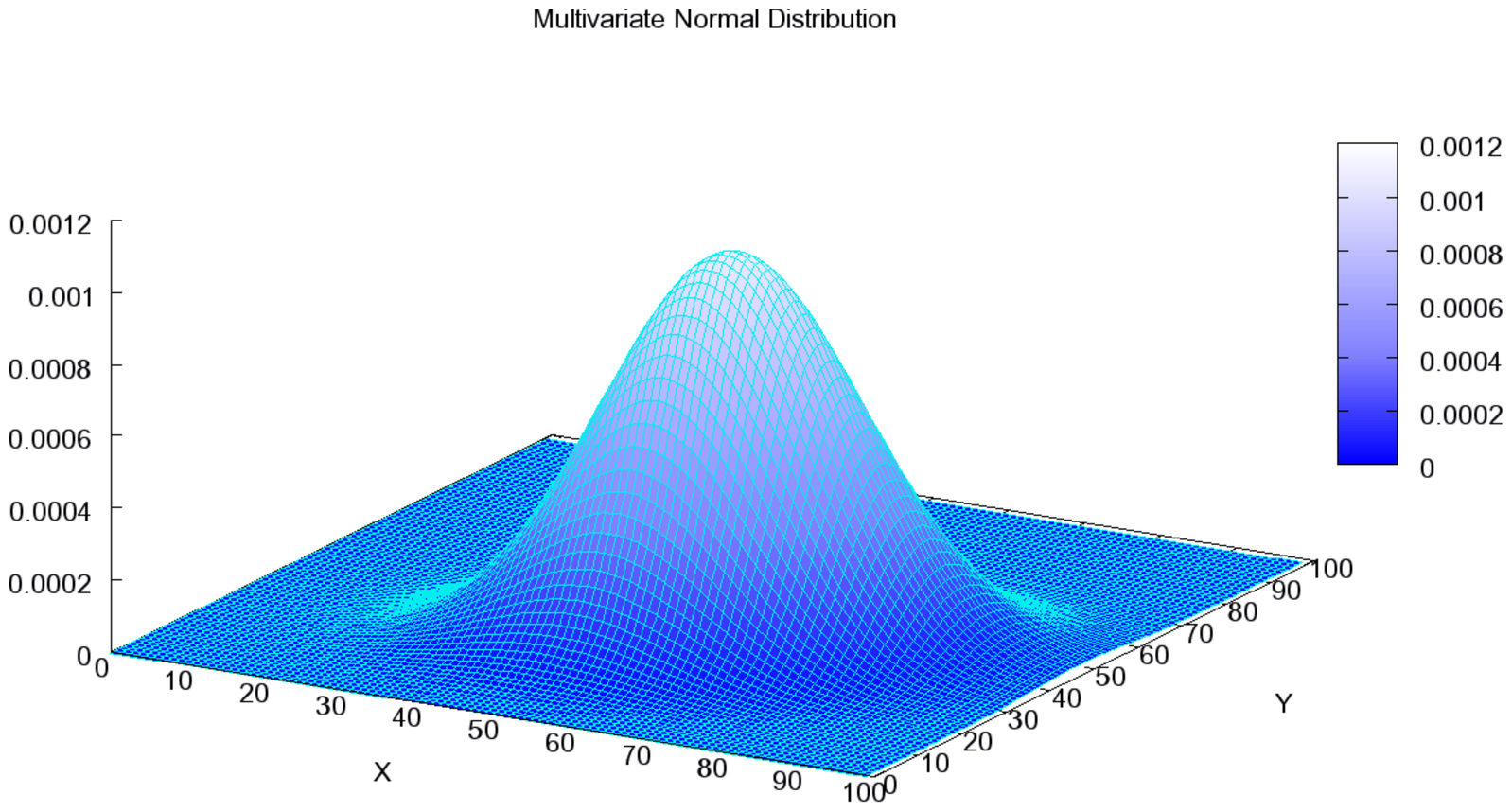


Figure source: Wikipedia

Multivariate Gaussian

Probability density function (pdf):

Mahalanobis distance

Normalizing constant

$$p_\theta(x) = \frac{1}{\sqrt{(2\pi)^k \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Parameters θ: mean $\boldsymbol{\mu}$, covariance matrix $\boldsymbol{\Sigma}$.

Fitting a multivariate Gaussian

$$\frac{\partial}{\partial \boldsymbol{\mu}} \log p_\theta(\mathbf{x}) = \frac{\partial}{\partial \boldsymbol{\mu}} \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

$$= -\frac{1}{2} \left[\frac{\partial}{\partial \mathbf{u}} \mathbf{u}^\top \boldsymbol{\Sigma}^{-1} \mathbf{u} \frac{\partial \mathbf{u}}{\partial \boldsymbol{\mu}} \right] \quad \text{for } \mathbf{u} = \mathbf{x} - \boldsymbol{\mu}$$

$$= -\frac{1}{2} \left[\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-\top} \right] \mathbf{u} \cdot (-1) = \frac{1}{2} \left[\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-\top} \right] (\mathbf{x} - \boldsymbol{\mu}) = \frac{1}{2} \left[2\boldsymbol{\Sigma}^{-1} \right] (\mathbf{x} - \boldsymbol{\mu})$$

$$= \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$$

$\boldsymbol{\Sigma}$ is symmetric

Identity:

$$\frac{\partial \left(\mathbf{u}^\top A \mathbf{u} \right)}{\partial \mathbf{u}} = \left(A + A^\top \right) \mathbf{u}$$

Fitting a multivariate Gaussian

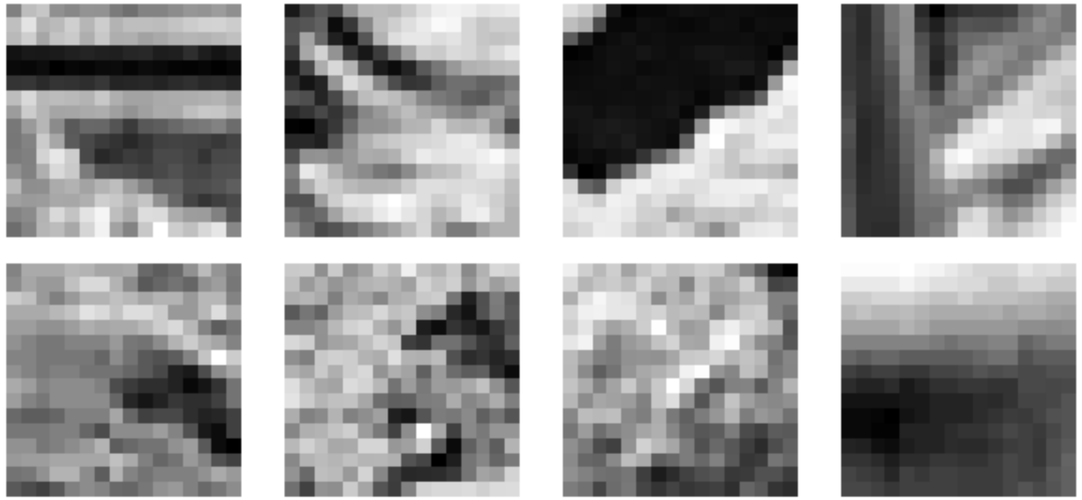
$$\frac{\partial}{\partial \boldsymbol{\mu}} \log p_\theta(\mathbf{x}) = \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})$$

$$\frac{\partial}{\partial \boldsymbol{\mu}} \sum_{i=1}^{N} \log(p_\theta(\mathbf{x}_i)) = 0 \implies \Sigma^{-1} \left[\sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu}) \right] = 0$$

$$\implies \mu = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$ i.e., average the training examples!

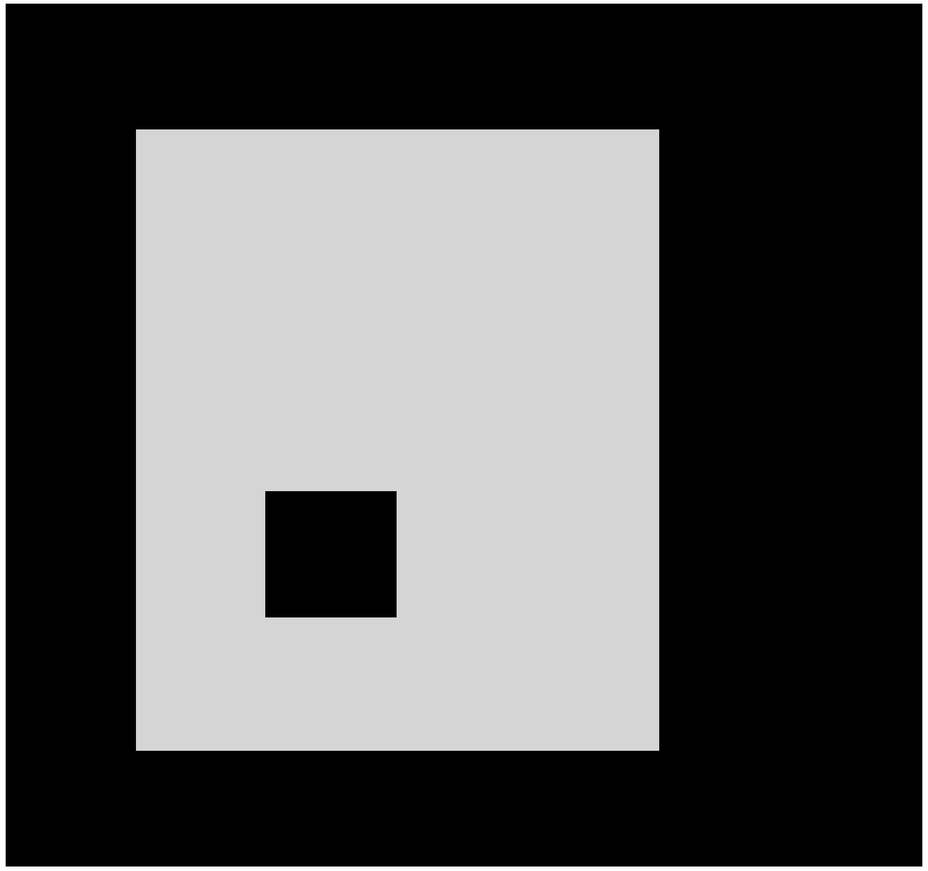
Example: fitting a multivariable Gaussian to image patches

Image patches (16×16)



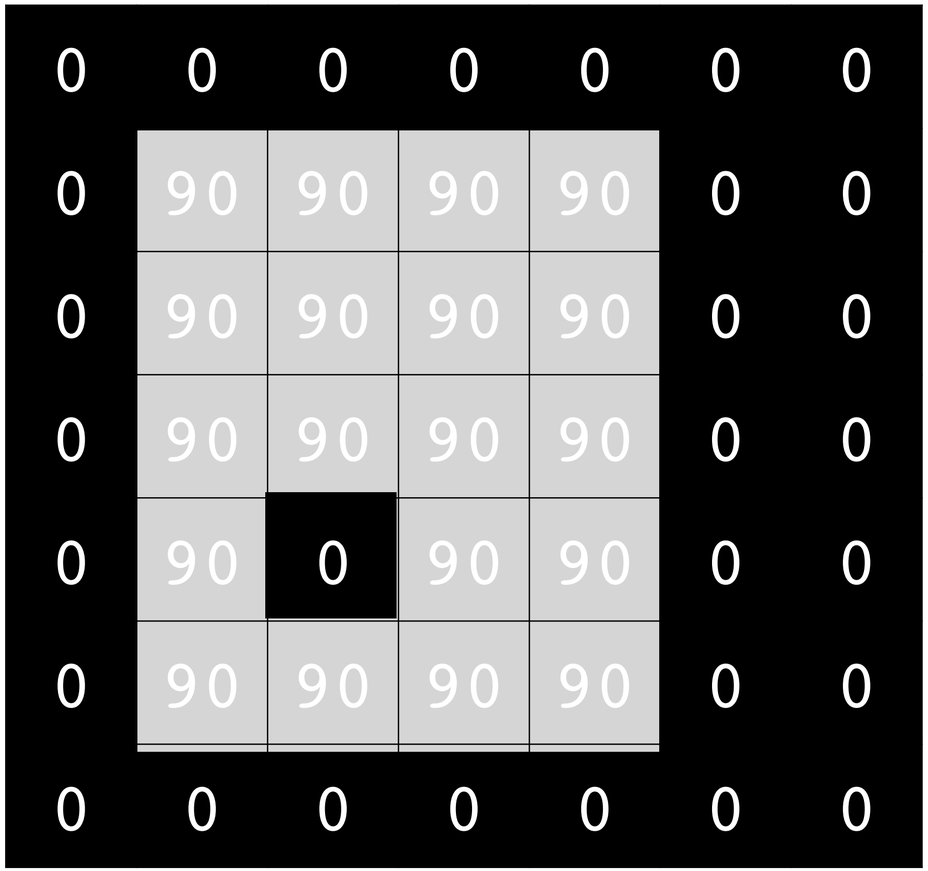
Images as arrays

An image



Images as arrays

How it's represented on the computer:



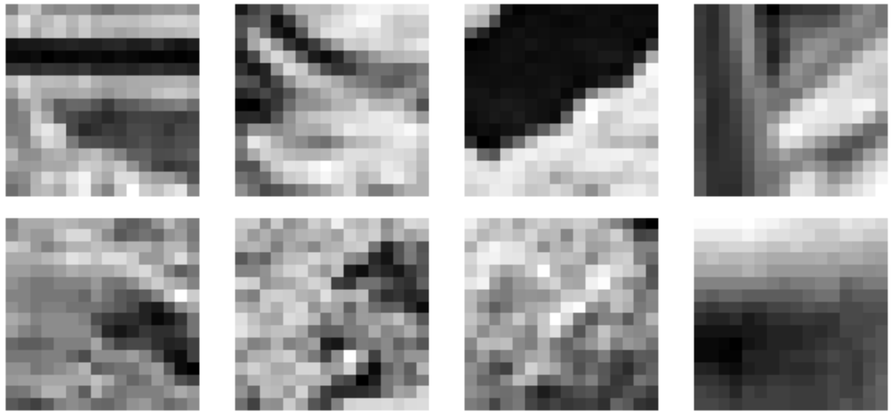
Images as arrays

How it's represented on the computer:

0	0	0	0	0	0	0
0	90	90	90	90	0	0
0	90	90	90	90	0	0
0	90	90	90	90	0	0
0	90	0	90	90	0	0
0	90	90	90	90	0	0
0	0	0	0	0	0	0

Fitting a multivariable Gaussian to image patches

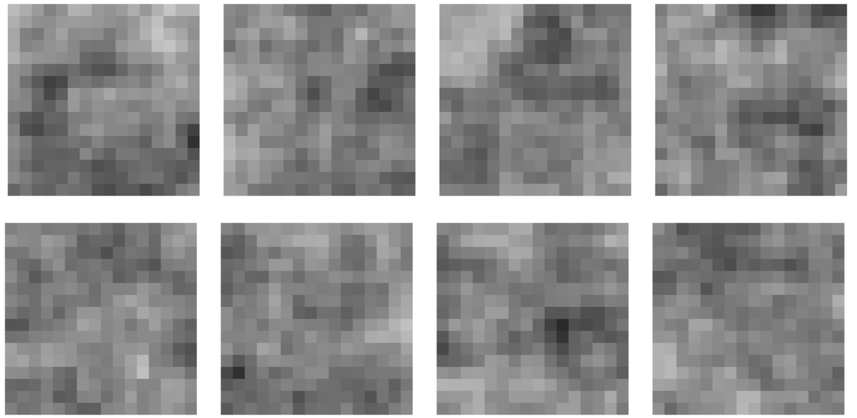
Training image patches (16×16)



Flatten each patch to get a 256-dimensional vector.

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Samples from Gaussian



(not so great!)

You'll do this in PS1!

KL divergence

Why use maximum likelihood estimation?

Consider the **KL divergence**, which measures how much a distribution q differs from a distribution p:

$$\mathrm{D}_{KL}(p \parallel q) = \int_x p(x) \log \left(\frac{p(x)}{q(x)} \right) dx$$

- **What this quantifies:** how many more bits would I need to code examples from $p(x)$ if I used a code designed to minimize the length of codes drawn from $q(x)$?
- Not symmetric! $D_{KL}(p \| q) \neq D_{KL}(q \| p)$
- Ideally, we assume that p is the true distribution and q is our approximation of p.

MLE $\iff$ minimizing KL divergence to true distribution

$$\text{argmax}_\theta \sum_{i=1}^{N} \log(p_\theta(x_i))$$

This new term doesn't depend on θ, and scaling by $\frac{1}{N}$ doesn't change max!

$$= \text{argmax}_\theta \frac{1}{N} \left[\sum_{i=1}^{N} \log(p_\theta(x_i)) - \sum_{i=1}^{N} \log(p^*(x_i)) \right]$$ where p^* is the true distribution

$$= \text{argmin}_\theta \frac{1}{N} \left[\sum_{i=1}^{N} \log(p^*(x_i)) - \sum_{i=1}^{N} \log(p_\theta(x_i)) \right]$$ multiply by -1

$$= \text{argmin}_\theta \frac{1}{N} \left[\sum_{i=1}^{N} \log \left(\frac{p^*(x_i)}{p_\theta(x_i)} \right) \right]$$ Monte Carlo estimate of expected value

$$\approx \text{argmin}_\theta \int_x p^*(x) \log \left(\frac{p^*(x)}{p_\theta(x)} \right) = \text{D}_{KL}(p^* \,\|\, p_\theta)$$ for large N

Discrete distributions

Discrete distributions

We'd also like to model discrete distributions, like sequences of words:

$p_\theta(\text{“it was the best of times, it was the worst of times”})$

Discrete distributions

Consider a sequence of N words:

$$p(x_1, x_2, \ldots, x_N)$$

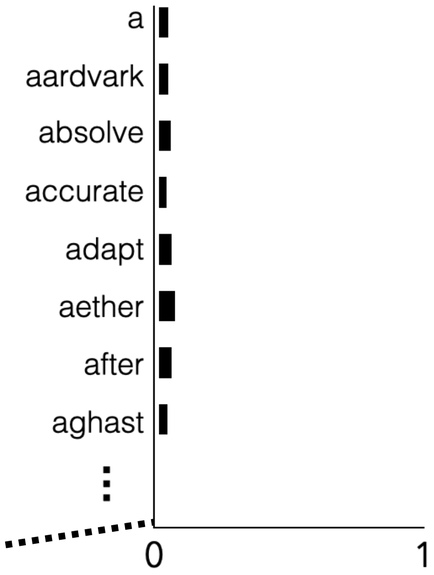
Can we store this as an N-dimensional table?

Problem: it's huge! It is D^N entires where D is the number of words in the dictionary.

Can't fill it, since each given sentence is rare.

Bag of words model

Discard word order and count the number of times each word appears. Sample a sequence by sampling each word independently.



$$p_\theta(x) = p(x^{(1)}, x^{(2)}, \ldots, x^{(T)}) = \prod_{i=1}^{T} p_\theta(x^{(i)})$$ where $x^{(i)} \in \mathbb{Z}$ is index of word i.

What's the problem with this model?

Figure source: P. Isola

Learning a bag of words model

$$p_\theta(x) = p(x^{(1)}, x^{(2)}, \ldots, x^{(T)}) = \prod_{t=1}^{T} p_\theta(x^{(t)})$$

$$\sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i) = \sum_{i=1}^{N} \sum_{t=1}^{T} \log p_\theta(\mathbf{x}_i^{(t)})$$

$$= \sum_{w=1}^{D} \sum_{i=1}^{N} \sum_{t=1}^{T} \delta(\mathbf{x}_i^{(t)} = w) \log(p_\theta(\mathbf{x}_i^{(t)} = w))$$

$$= \sum_{w=1}^{D} C_w \log(p_\theta(\mathbf{x}_i^{(t)} = w)) \qquad C_w = \text{word count in dataset}$$

$$= \sum_{w=1}^{D} C_w \log(\theta_w) \qquad \theta_w = \text{probability we assign word } w \text{ in BOW model}$$

Lagrange multiplier

constraint

Constrained optimization:

$$J(\theta, \lambda) = \sum_{w=1}^{D} C_w \log(\theta_w) + \lambda \left(\sum_{w=1}^{D} \theta_w - 1 \right)$$

Take derivative w.r.t. both θ and λ!

$$0 = \frac{\partial}{\partial \lambda} J(\theta, \lambda) = \sum_{w=1}^{D} \theta_w - 1 \implies \sum_{w=1}^{D} \theta_w = 1$$

$$0 = \frac{\partial}{\partial \theta_w} J(\theta, \lambda) = \frac{C_w}{\theta_w} + \lambda \implies \theta_w = -\frac{C_w}{\lambda}$$

Now, let's put these equations together

total words in dataset

$$\sum_{w=1}^{D} -\frac{C_w}{\lambda} = 1 \implies \lambda = -\sum_{w=1}^{D} C_w = -NT \implies \boxed{\theta_w = \frac{C_w}{NT}}$$

Just the word probability!

Bigram models

Problem with bag-of-words (unigram) models: no order to the sequence!

Consider **pairs** of words.

$$p_\theta(\mathbf{x}) = p\left(x^{(1)}, x^{(2)}, \ldots, x^{(T)}\right) = p_\theta\left(x^{(1)}\right) p_\theta\left(x^{(2)} \mid x^{(1)}\right) p_\theta\left(x^{(3)} \mid x^{(2)}\right) \ldots p_\theta\left(x^{(T)} \mid x^{(T-1)}\right)$$

Bigram models

Implement $p(x_{i+1} \mid x_i)$ as a table:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

(in practice, "smooth" to assign non-zero probability to empty cells).

Is this a valid probability distribution?

Source: Jurafsky & Martin SLP book

Probability chain rule

Last class, we used the rule: $p(a, b) = p(a \mid b)p(b)$

Generalization:

$$p(\mathbf{x}) = \prod_{i=1}^{T} p(x^{(i)} \mid x^{(1)}, x^{(2)}, \ldots, x^{(i-1)})$$

Multiplying all conditionals evaluates the probability of a full joint configuration of words.

Source: Torralba, Freeman, Isola

N-gram models

Generalization of bigrams. Consider n-tuples of words for more context.

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{T} p(x^{(i)} \mid x^{(1)}, x^{(2)}, \ldots, x^{(i-1)})$$

$$\approx \prod_{i=1}^{T} p\left(x^{(i)} \mid x^{(i-1)}, \ldots, x^{(i-2)}, x^{(i-N+1)}\right)$$

Large Language Models in Machine Translation

Thorsten Brants **Ashok C. Popat** **Peng Xu** **Franz J. Och** **Jeffrey Dean**

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94303, USA
{brants,popat,xp,och,jeff}@google.com

Abstract

This paper reports on the benefits of large-scale statistical language modeling in machine translation. A distributed infrastructure is proposed which we use to train on up to 2 trillion tokens, resulting in language models having up to 300 billion n-grams. It is capable of providing smoothed probabilities for fast, single-pass decoding. We introduce a new smoothing method, dubbed *Stupid Backoff*, that is inexpensive to train on large data sets and approaches the quality of Kneser-Ney Smoothing as the amount of training data increases.

1 Introduction

Given a source-language (e.g., French) sentence $\mathbf{f}$, the problem of *machine translation* is to automatically produce a target-language (e.g., English) trans-

How might one build a language model that allows scaling to very large amounts of training data? (2) How much does translation performance improve as the size of the language model increases? (3) Is there a point of diminishing returns in performance as a function of language model size?

This paper proposes one possible answer to the first question, explores the second by providing learning curves in the context of a particular statistical machine translation system, and hints that the third may yet be some time in answering. In particular, it proposes a *distributed* language model training and deployment infrastructure, which allows direct and efficient integration into the hypothesis-search algorithm rather than a follow-on re-scoring phase. While it is generally recognized that two-pass decoding can be very effective in practice, single-pass decoding remains conceptually attractive because it eliminates a source of potential information loss.

Sampling from N-gram models

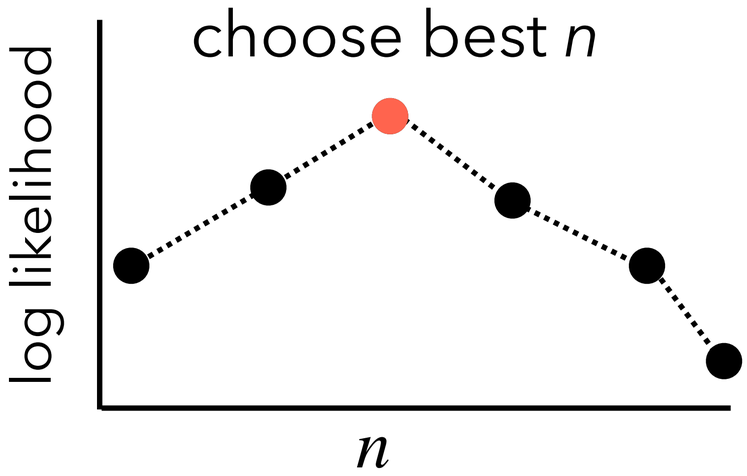
N-gram models trained on Wall Street Journal articles

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Source: Jurafsky & Martin SLP book

How do we choose hyperparameters?

How do we choose hyperparameters?



Training Set | Validation Set | Test Set

Measures *generalization*

Choose **hyperparameters** like *n*, number of mixture components, etc.

Next lecture: Gaussian mixture models