Course Outline

- Regression Techniques
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
- Deep Reinforcement Learning
- Probabilistic Graphical Models
  - Bayesian Networks
  - Markov Random Fields
  - Inference (exact & approximate)
  - Latent Variable Models
- Deep Generative Models
  - Generative Adversarial Networks
  - Variational Autoencoders

Topics of This Lecture

- Recap: Bayesian Mixture Models
  - Full Bayesian Treatment
    - Given a dataset, we are interested in the cluster assignments
    \[ p(Z|X) = \frac{p(X|Z)p(Z)}{\sum_Z p(X|Z)p(Z)} \]
    - The likelihood is obtained by marginalizing over the parameters \( \theta \)
    \[ p(X|\theta) = \int p(X|Z, \theta)p(Z)\theta)dz \theta \]
    - The posterior over assignments is intractable!
- Recap: Generative Adversarial Networks (GANs)
  - Applications & Extensions
    - GANs for image generation
    - GANs for superresolution
    - Conditional GANs
- Recap: Problems of GANs
  - Problems during training
  - Conceptual problems
  - Extension: Wasserstein GANs

Recap: Bayesian Mixture Models

- Let’s be Bayesian about mixture models
  - Place priors over our parameters
  - Again, introduce variable \( z_n \) as indicator which component data point \( x_n \) belongs to.
    \[ z_n \pi \sim \text{Multinomial}(\pi) \]
    \[ x_n|z_n = k, \mu_k, \Sigma_k \sim \mathcal{N}(\mu_k, \Sigma_k) \]
  - Introduce conjugate priors over parameters
    \[ \pi \sim \text{Dirichlet}(\frac{\alpha}{K}, \ldots, \frac{\alpha}{K}) \]
  - Denominator requires summing over all possible partitions of the data into \( K \) groups!

Recap: Mixture Models with Dirichlet Priors

- Integrating out the mixing proportions \( \pi \)
  \[ p(\pi|\alpha) = \frac{\Gamma(\alpha)}{\prod_{k=1}^{K} \Gamma(N_k + \alpha/K)} \]

- Conditional probabilities
  - Examine the conditional of \( z_n \), given all other variables \( z_{\bar{n}} \)
    \[ p(z_n|z_{\bar{n}}, \pi) = \frac{p(z_n, z_{\bar{n}}|\pi)\pi}{p(z_{\bar{n}}|\pi)} \]
    \[ = \frac{\Gamma(\alpha)}{\prod_{k=1}^{K} \Gamma(N_k + \alpha/K)} \]

  \Rightarrow The more populous a class is, the more likely it is to be joined!
Recap: Infinite Dirichlet Mixture Models

- Conditional probabilities: Finite $K$
  \[ p(z_{nk} = 1|z_{n-1}, \alpha) = \frac{N_{nk} + \alpha / K}{N - 1 + \alpha} \]
  \[ N_{nk} \triangleq \sum_{i=1}^{N} z_{ik} \]

- Conditional probabilities: Infinite $K$
  - Taking the limit as $K \to \infty$ yields the conditionals
  \[ p(z_{nk} = 1|z_{n-1}, \alpha) = \begin{cases} \frac{N_{nk} + \alpha / K}{N - 1 + \alpha} & \text{if } k \text{ represented} \\ \frac{\alpha}{N - 1 + \alpha} & \text{if all } k \text{ not represented} \end{cases} \]

  - Left-over mass $\alpha$ : countably infinite number of indicator settings

Recap: Gibbs Sampling for Finite Mixtures

- We need approximate inference here
  - Gibbs Sampling: Conditionals are simple to compute
    \[ p(z_n = k|\text{others}) \propto \sum_{k=1}^{K} z_n N(x_n; \mu_k, \Sigma_k) \]
    \[ \pi(\mu_k, \Sigma_k) \propto N - IW(\nu', s', d', \nu'') \]

  - However, this will be rather inefficient...
    - In each iteration, algorithm can only change the assignment for individual data points.
    - There are often groups of data points that are associated with high probability to the same component. \( \Rightarrow \) Unlike it that group is moved.
    - Better performance by collapsed Gibbs sampling which integrates out the parameters $\mu_k, \Sigma_k$.

Recap: Collapsed Finite Bayesian Mixture

- More efficient algorithm
  - Conjugate priors allow analytic integration of some parameters
  - Resulting sampler operates on reduced space of cluster assignments (implicitly considers all possible cluster shapes)

- Procedure
  - The model implies the factorization
    \[ p(x_n|z_{n-1}, x, H) \propto p(z_n|z_{n-1}, \alpha)p(x_n|x, x_{n-1}, H) \]
  - Derive
    \[ p(z|\alpha) = \int p(z|x)p(x|\alpha)d\alpha \]
    \[ p(x_n|z_{n-1}, H) = \sum_{k=1}^{K} z_{nk} p(x_n|\theta_k)p(\theta_k|H)d\theta \]
  - Conjugate prior, Normal - Inverse Wishart

Topics of This Lecture

- Recap: Bayesian Mixture Models
- Generative Adversarial Networks (GANs)
  - Generative networks
  - GAN loss and training procedure
- Applications & Extensions
  - GANs for image generation
  - GANs for superresolution
  - Conditional GANs
- Problems of GANs
  - Problems during training
  - Conceptual problems
  - Extension: Wasserstein GANs

Generative Networks

- Using a network to generate images
  - Sampling from noise distribution
  - Sequence of upsampling layers to generate an output image
  - How can we train such a model to produce the desired output?

Generative Adversarial Networks (GANs)

- Conceptual view
  - Simultaneously train an image generator $G$ and a discriminator $D$.
  - Interpreted as a two-player game
Two-Player Game

- **Generator**
  - Tries to draw samples from $p(x)$.
  - Analogy: counterfeiter

- **Discriminator**
  - Tries to determine whether the sample came from the generator or the data distribution.
  - Analogy: police investigator

- Both generator and discriminator are deep networks
  - We can train them with backprop.

Training the Discriminator

- Procedure
  - Fix generator weights
  - Train discriminator to distinguish between real and generated images

Formalizing This Procedure

- This corresponds to a two-player minimax game:
  $$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}} \left[ \log D(x) \right] + \mathbb{E}_{z \sim p_z} \left[ \log \left(1 - D(G(z))\right) \right]$$

- Explanation
  - Train $D$ to maximize the probability of assigning the correct label to both training examples and samples from $G$.
  - Simultaneously train $G$ to minimize $\log \left(1 - D(G(z))\right)$.

- The Nash equilibrium of this game is achieved at
  - $p_{G}(x) = p_{data}(x)$  $\forall x$
  - $D(x) = \frac{1}{2}$  $\forall x$

GAN Algorithm

```
for n = 1 : steps do
    for m = 1 : updates do
        Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_z(z)$.
        Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
        Update the discriminator by assigning its stochastic gradient:
        $$\nabla_{\theta_{D}} \mathbb{E}_{x \sim p_{data}} \left[ \log D(x) \right] + \mathbb{E}_{z \sim p_z} \left[ \log \left(1 - D(G(z))\right)\right].$$
        Discriminator updates
    end for
    Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_z(z)$.
    Update the generator by assigning its stochastic gradient:
    $$\nabla_{\theta_{G}} \mathbb{E}_{z \sim p_z} \left[ \log \left(1 - D(G(z))\right)\right].$$
    Generator updates
end for
```

Intuition

- Behavior near convergence
  - In the inner loop, $D$ is trained to discriminate samples from data.
  - Gradient of $D$ guides $G$ to flow to regions that are more likely to be classified as data.
  - After several steps of training, $G$ and $D$ will reach a point at which they cannot further improve, because $p_{G} = p_{data}$.
  - Now, the discriminator is unable to differentiate between the two distributions, i.e., $D(x) = 0.5$. 
Topics of This Lecture

- Recap: Bayesian Mixture Models
- Generative Adversarial Networks (GANs)
  - Generative networks
  - GAN loss and training procedure
- Applications & Extensions
  - GANs for image generation
  - GANs for superresolution
  - Conditional GANs
- Problems of GANs
  - Problems during training
  - Conceptual problems
  - Extension: Wasserstein GANs

Example: Deep Convolutional GAN (DCGAN)

- Generator for images
  - Remove fully-connected layers
  - Upsampling with fractional strided convolutions
  - Batch normalization after each layer (important!)
  - Use ReLu in generator for hidden layers, tanh for output layer
  - Use Leaky ReLu in the discriminator for all layers

Example Application: Image Generation

- Generating bedroom images
  - Each sample is generated from a sampled random number

Example Application: Image Generation

- Interpolating between the random points in latent space…

Interpolation between Face Images

Karras et al., “Progressive growing of GANs for improved quality, stability, and variation”, ICLR’18

Interpolation between Arbitrary Images

Karras et al., “Progressive growing of GANs for improved quality, stability, and variation”, ICLR’18
Example Application: Super-Resolution (SRGAN)

bicubic | SRRResNet | SRGAN | original

Extension: Conditional GANs

• Idea
  – Condition the latent space representation on an input image
  – Used to create the pix2pix network


Extension: Conditional GANs


Artist Project: edges2cats [Christopher Hesse]

Topics of This Lecture

• Recap: Bayesian Mixture Models
• Generative Adversarial Networks (GANs)
  – Generative networks
  – GAN loss and training procedure
• Applications & Extensions
  – GANs for image generation
  – GANs for superresolution
  – Conditional GANs
• Problems of GANs
  – Problems during training
  – Conceptual problems
  – Extension: Wasserstein GANs

What Can Possibly Go Wrong? Problems with GANs

• Problem 1: Vanishing gradients
  – When the discriminator is perfect, the loss function falls to zero.
  – No gradient to update the loss during learning iterations.
  – Dilemma: Walking a fine line...
    • Discriminator behaves badly ⇒ generator does not have accurate feedback
    • Discriminator does a great job ⇒ gradient of the loss drops close to zero
What Can Possibly Go Wrong? Problems with GANs

• Problem 2: Mode collapse
  - Even though the generator might be able to trick the discriminator, it may fail to represent the complex real-world data distribution.
  - Training gets stuck in a small space with little variety

• Problem 3: Non-convergence
  - GANs involve two players
  - Each model updates its cost independently
  - This means we are performing simultaneous gradient descent
  - Problem: might not converge to Nash equilibrium

• Problem 4: Low-dimensional support
  - Both $p_{data}$ and $p_g$ lie on low-dimensional manifolds.
  - Those manifolds most likely do not intersect
  - The Jensen divergence implicitly optimized in GANs cannot deal with this well.
  - Wasserstein GANs fix this by introducing a different loss function.

Summary

• Advantages
  - GANs can be trained with backpropagation
  - Generated images are sharper than from VAEs
  - Robust to overfitting, since generator never sees the training data
  - Fast process: single forward pass generates a single sample

• Disadvantages
  - GANs are well known for being delicate and unstable
  - Problems with non-convergence
  - Problems with mode collapse

• Extensions
  - Wasserstein GANs fix several major problems in the GAN formulation
  - Energy-based GANs allow general loss functions

2017: Explosion of GANs

• "The GAN Zoo"
  - https://github.com/hindupuravinash/the-gan-zoo

Interest in GANs Is Still Growing...
References

- Original GAN

- GAN Extensions

- Evaluations