Neural Networks II: Deep Learning

Mengye Ren

NYU

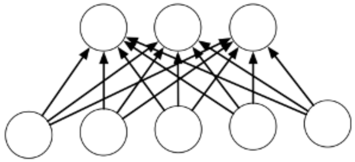
Nov 28, 2023

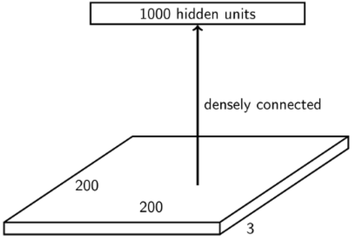
Fully connected vs. locally connected

- So far we apply a layer where all output neurons are connected to all input neurons.
- In matrix form, $\mathbf{z} = W\mathbf{x}$.
- This is also called a fully connected layer or a dense layer or a linear layer.

Fully connected vs. locally connected

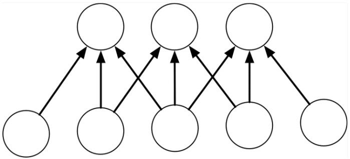
- So far we apply a layer where all output neurons are connected to all input neurons.
- In matrix form, $\mathbf{z} = W\mathbf{x}$.
- This is also called a fully connected layer or a dense layer or a linear layer.
- For 200×200 image and 1000 hidden units, the matrix of a single layer will have 40M parameters!

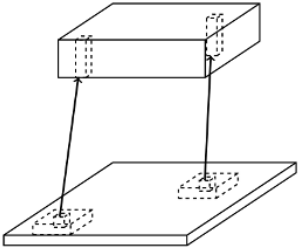




Fully connected vs. locally connected

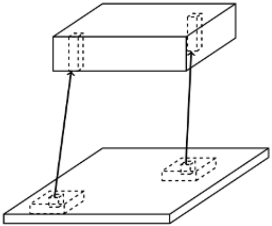
- An alternative strategy is to use local connection.
- For neuron i, only connects to its neighborhood (e.g. [i+k, i-k])
- For images, we index neurons with three dimensions i, j, and c.
- i = vertical index, j = horizontal index, c = channel index.





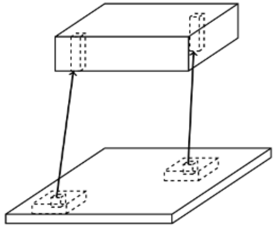
Local connection patterns

- The typical image input layer has 3 channels R G B for color or 1 channel for grayscale.
- The hidden layers may have C channels, at each spatial location (i, j).



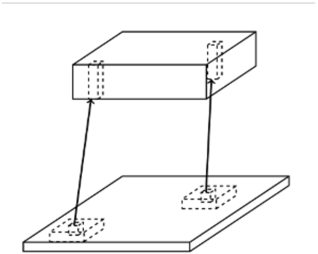
Local connection patterns

- The typical image input layer has 3 channels R G B for color or 1 channel for grayscale.
- The hidden layers may have C channels, at each spatial location (i, j).
- Now each hidden neuron $z_{i,j,c}$ receives inputs from $x_{i\pm k, j\pm k, \cdot}$
- k is the "kernel" size - do not confuse with the other kernel we learned.
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i'j'c'} w_{i,j,i'-i,j'-j,c',c}$



Local connection patterns

- The typical image input layer has 3 channels R G B for color or 1 channel for grayscale.
- The hidden layers may have C channels, at each spatial location (i, j).
- Now each hidden neuron $z_{i,j,c}$ receives inputs from $x_{i \pm k, j \pm k, \cdot}$
- k is the "kernel" size - do not confuse with the other kernel we learned.
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i' j' c'} w_{i,j,i'-i,j'-j,c',c}$
- The spatial awareness (receptive field) of the neighborhood grows bigger as we go deeper.



Weight sharing

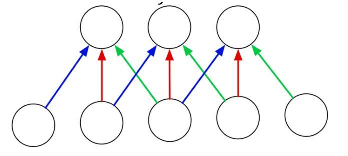
- Still a lot of weights: If we have 100 channels in the second layer, then $200 \times 200 \times 3 \times 100 = 12M$

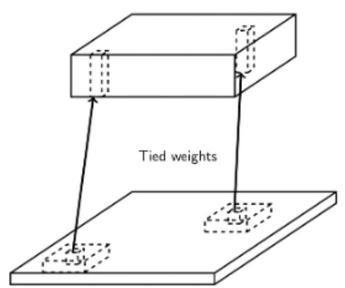
Weight sharing

- Still a lot of weights: If we have 100 channels in the second layer, then $200 \times 200 \times 3 \times 100 = 12M$
- Local information is the same regardless of the position of an element.

Weight sharing

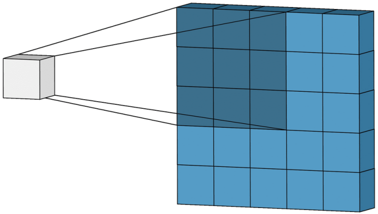
- Still a lot of weights: If we have 100 channels in the second layer, then $200 \times 200 \times 3 \times 100 = 12M$
- Local information is the same regardless of the position of an element.
- Solution: We can tie the weights at different locations.





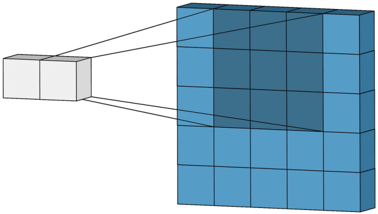
2D convolution

- Using the same weight connections for each activation spatial location works like the "filtering operation" or "convolution"
- The neighborhood window is the filter window.
- The weight connection is called "convolution filter"
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i'j'c'} w_{i-i',j-j',c',c}$



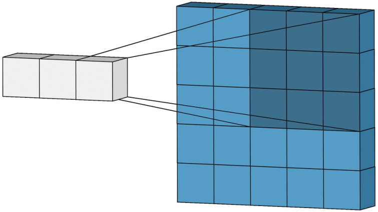
2D convolution

- Using the same weight connections for each activation spatial location works like the "filtering operation" or "convolution"
- The neighborhood window is the filter window.
- The weight connection is called "convolution filter"
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i'j'c'} w_{i-i',j-j',c',c}$



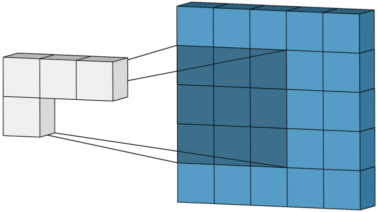
2D convolution

- Using the same weight connections for each activation spatial location works like the "filtering operation" or "convolution"
- The neighborhood window is the filter window.
- The weight connection is called "convolution filter"
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i'j'c'} \textcolor{red}{w_{i-i',j-j',c',c}}$



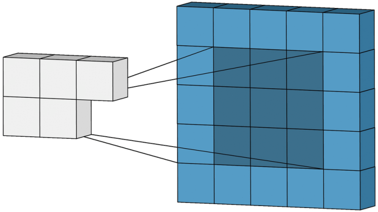
2D convolution

- Using the same weight connections for each activation spatial location works like the "filtering operation" or "convolution"
- The neighborhood window is the filter window.
- The weight connection is called "convolution filter"
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i'j'c'} w_{i-i',j-j',c',c}$



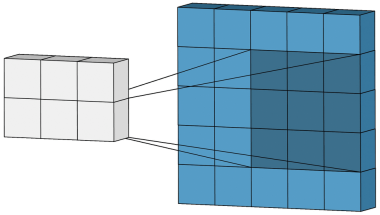
2D convolution

- Using the same weight connections for each activation spatial location works like the “filtering operation” or “convolution”
- The neighborhood window is the filter window.
- The weight connection is called “convolution filter”
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i'j'c'} w_{i-i',j-j',c',c}$



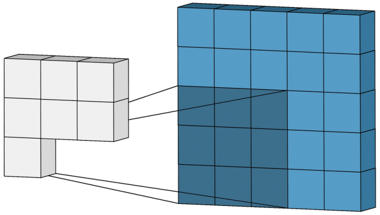
2D convolution

- Using the same weight connections for each activation spatial location works like the "filtering operation" or "convolution"
- The neighborhood window is the filter window.
- The weight connection is called "convolution filter"
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i'j'c'} w_{i-i',j-j',c',c}$



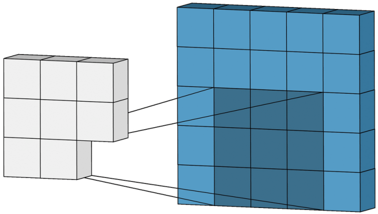
2D convolution

- Using the same weight connections for each activation spatial location works like the "filtering operation" or "convolution"
- The neighborhood window is the filter window.
- The weight connection is called "convolution filter"
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i'j'c'} \textcolor{red}{w_{i-i',j-j',c',c}}$



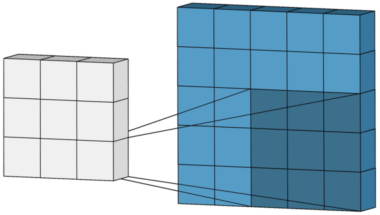
2D convolution

- Using the same weight connections for each activation spatial location works like the "filtering operation" or "convolution"
- The neighborhood window is the filter window.
- The weight connection is called "convolution filter"
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i'j'c'} \textcolor{red}{w_{i-i',j-j',c',c}}$



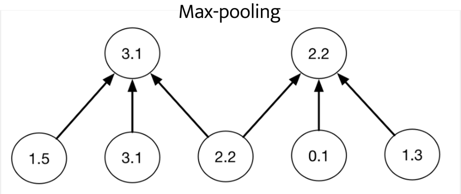
2D convolution

- Using the same weight connections for each activation spatial location works like the "filtering operation" or "convolution"
- The neighborhood window is the filter window.
- The weight connection is called "convolution filter"
- $z_{i,j,c} = \sum_{i' \in [i \pm k], j' \in [j \pm k], c'} x_{i'j'c'} \color{red}{w_{i-i',j-j',c',c}}$



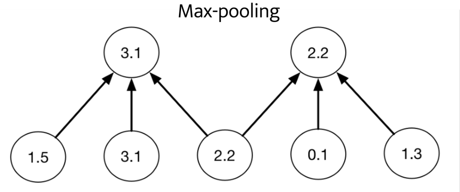
Pooling

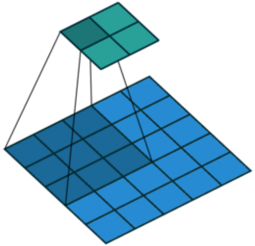
- Need to summarize global information more efficiently.
- Pooling reduces image / activation dimensions.
- Max-pooling or average-pooling



Pooling

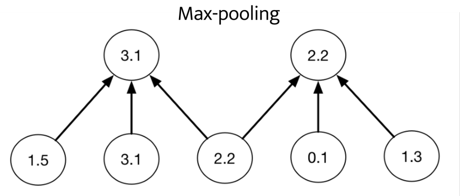
- Need to summarize global information more efficiently.
- Pooling reduces image / activation dimensions.
- Max-pooling or average-pooling
- You can also perform a "strided" convolution by jumping multiple steps.

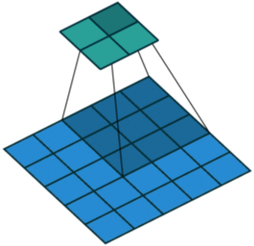




Pooling

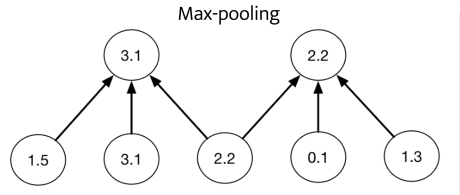
- Need to summarize global information more efficiently.
- Pooling reduces image / activation dimensions.
- Max-pooling or average-pooling
- You can also perform a "strided" convolution by jumping multiple steps.

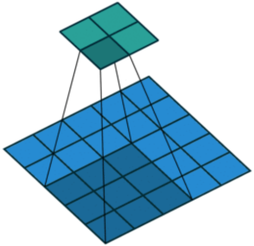




Pooling

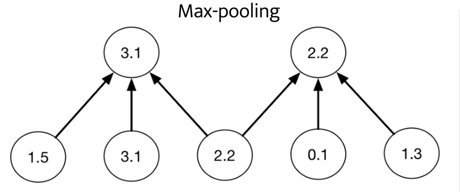
- Need to summarize global information more efficiently.
- Pooling reduces image / activation dimensions.
- Max-pooling or average-pooling
- You can also perform a “strided” convolution by jumping multiple steps.

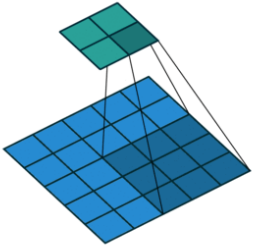




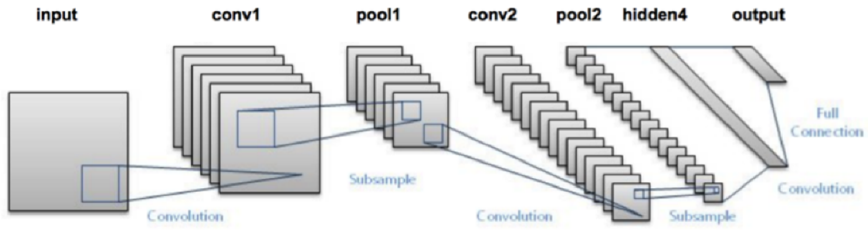
Pooling

- Need to summarize global information more efficiently.
- Pooling reduces image / activation dimensions.
- Max-pooling or average-pooling
- You can also perform a "strided" convolution by jumping multiple steps.





Assembling together: LeNet



- Used by USPS to read post code in the 90s.

Historical development

- LeNet has worked and being put to practice in the 1990s.

Historical development

- LeNet has worked and being put to practice in the 1990s.
- Neural networks for images start to dominate in the last 10 years (starting 2012) for understanding general high resolution natural images.

Historical development

- LeNet has worked and being put to practice in the 1990s.
- Neural networks for images start to dominate in the last 10 years (starting 2012) for understanding general high resolution natural images.
- During the years:
  - Neural networks were difficult to work
  - People focused on feature engineering
  - Then apply SVM or random forest (e.g. AdaBoost face detector)
  - What has changed?

Gradient learning conditioning

Optimization challenges

- Larger images require deeper networks (more stages of processing at different resolutions)
- Optimizing deeper layers of networks is not trivial.
- Loss often stalls or blows up.
- Why?

Optimization challenges

- Larger images require deeper networks (more stages of processing at different resolutions)
- Optimizing deeper layers of networks is not trivial.
- Loss often stalls or blows up.
- Why?
  - Backpropagation: multiplying the Jacobian $\frac{\partial y}{\partial x}$ by each layer.
  - If the maximum singular value of each layer of Jacobian is less than 1: then the gradient will converge to 0 with more layers.
  - If the greater than 1: then the gradient will explode with more layers.
  - The bottom (input) layer may get 0 or infinite gradients.

Weight initialization

- Even with a few layers ($>$3), optimization is still hard.
- If weight initialization is bad (too small or too big), then optimization is hard to kick off.
- Consider the distribution of whole dataset in the activation space.
  - Intuition: upon initialization, the variance of the activations should stay the same across every layer.

Kaiming Initialization

- Suppose each neuron and weight connection are sampling from a random distribution.

[1] He et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet. ICCV, 2015.

Kaiming Initialization

- Suppose each neuron and weight connection are sampling from a random distribution.
- At l-th layer, $Var[z_l] = n_l Var[w_l x_l]$ (n_l = num. input neurons to l-th layer)

[1] He et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet. ICCV, 2015.

Kaiming Initialization

- Suppose each neuron and weight connection are sampling from a random distribution.
- At l-th layer, $Var[z_l] = n_l Var[w_l x_l]$ (n_l = num. input neurons to l-th layer)
- If we suppose that ReLU is used as the activation, and w_l is symmetric and zero-mean, $x_{l+1} = \frac{1}{2} Var[z_l]$.

[1] He et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet. ICCV, 2015.

Kaiming Initialization

- Suppose each neuron and weight connection are sampling from a random distribution.
- At l-th layer, $Var[z_l] = n_l Var[w_l x_l]$ (n_l = num. input neurons to l-th layer)
- If we suppose that ReLU is used as the activation, and w_l is symmetric and zero-mean, $x_{l+1} = \frac{1}{2} Var[z_l]$.
- Putting altogether, $x_{l+1} = \frac{1}{2} n_l Var[w_l] Var[x_l]$.

[1]He et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet. ICCV, 2015.

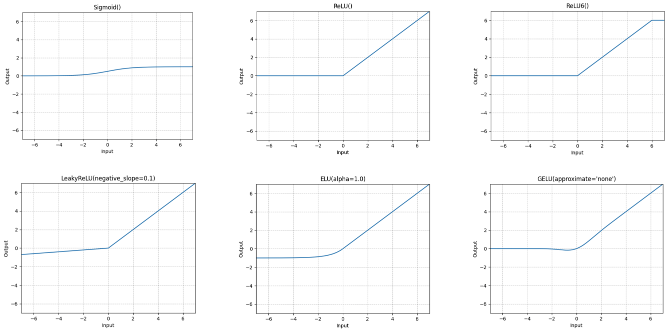
Kaiming Initialization

- Suppose each neuron and weight connection are sampling from a random distribution.
- At l-th layer, $Var[z_l] = n_l Var[w_l x_l]$ ($n_l =$ num. input neurons to l-th layer)
- If we suppose that ReLU is used as the activation, and w_l is symmetric and zero-mean, $x_{l+1} = \frac{1}{2} Var[z_l]$.
- Putting altogether, $x_{l+1} = \frac{1}{2} n_l Var[w_l] Var[x_l]$.
- To make the variance constant, we need $\frac{1}{2} n_l Var[w_l] = 1$, $Std[w_l] = \sqrt{2/n_l}$[1].

[1] He et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet. ICCV, 2015.

Activation functions

- ReLU was proposed in 2009-2010[2,3], and was successfully used in AlexNet in 2012[4].
- Address the vanishing gradient issue in activations, comparing to sigmoid or tanh.

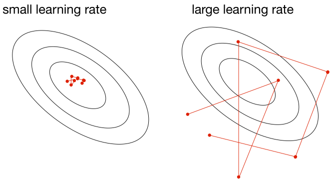


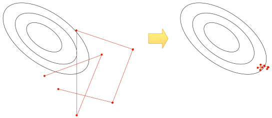
[2] Jarrett et al. What is the Best Multi-Stage Architecture for Object Recognition? ICCV, 2009.
[3] Nair & Hinton/ Rectified Linear Units Improve Restricted Boltzmann Machines. ICML, 2010.
[4] Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks. NIPS, 2012.

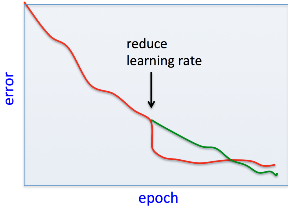
SGD Learning Rate

- In stochastic training, the learning rate also influences the fluctuations due to the stochasticity of the gradients.
- Typical strategy:
  - Use a large learning rate early in training so you can get close to the optimum
  - Gradually decay the learning rate to reduce the fluctuations

- We also need to be aware about the impact of learning rate due to the stochasticity.







RMSprop and Adam

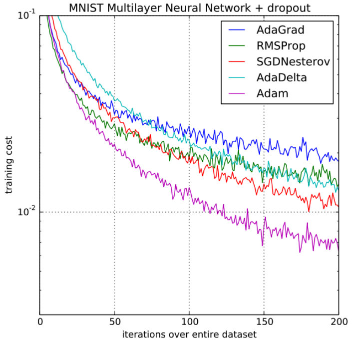
- Recall: SGD takes large steps in directions of high curvature and small steps in directions of low curvature.
- RMSprop is a variant of SGD which rescales each coordinate of the gradient to have norm 1 on average. It does this by keeping an exponential moving average s_j of the squared gradients.
- The following update is applied to each coordinate j independently:

$$s_j \leftarrow (1-\gamma)s_j + \gamma[\tfrac{\partial L}{\partial \theta_j}]^2$$

$$\theta_j \leftarrow \theta_j - \frac{\alpha}{\sqrt{s_j + \epsilon}} \frac{\partial L}{\partial \theta_j}$$

Adam optimizer

- Adam = RMSprop + momentum = Adaptive Momentum estimation
- Smoother estimate of the average gradient and gradient norm.
- m_t: exponential moving average of gradient.
- v_t: exponential moving average of gradient squared.
- $\hat{m}_t$, $\hat{v}_t$: Bias correction.
- $\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
- The "default" optimizer for modern networks.

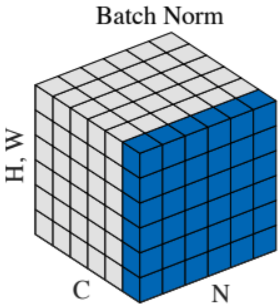


Normalization

- Weight initialization is tricky, and there is no guarantee that the distribution of activations will stay the same over the learning process.
- What if the weights keep grow bigger and activation may explode?
- We can "normalize" the activations.
- The idea is to control the activation within a normal range: zero-mean, uni-variance.

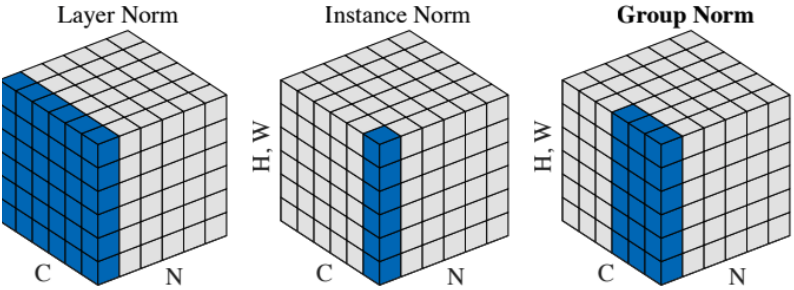
Batch Normalization (BN)

- Training image distribution -> activation distribution
- In CNNs, neurons across different spatial locations are also samples of the same feature channel.
- Batch norm: Normalize across N H W dimensions, leaving C channels.
- $\tilde{x} = \gamma \frac{x-\mu}{\sigma} + \beta$
- Test time: using the mean and variance from the entire training set.



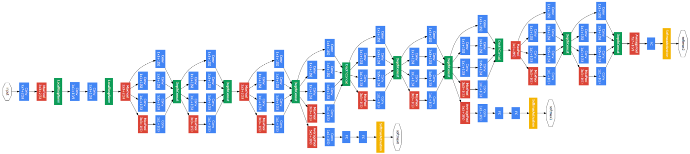
BN Alternatives

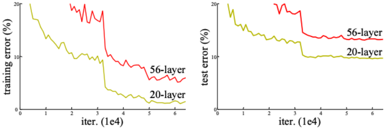
- Need a considerable batch size to estimate mean and variance correctly.
- Training is different from testing.
- Alternatives consider the C channel dimension instead of N batch dimension.



Going Deeper

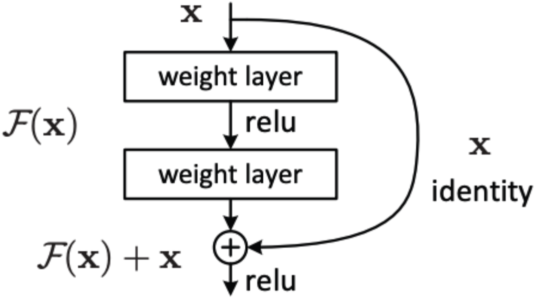
- The progress of normalization allowed us to train even deeper networks.
- The networks are no longer too sensitive with initialization.
- But the best networks were still around 20 layers and deeper results in worse performance.





Residual Networks (ResNet)

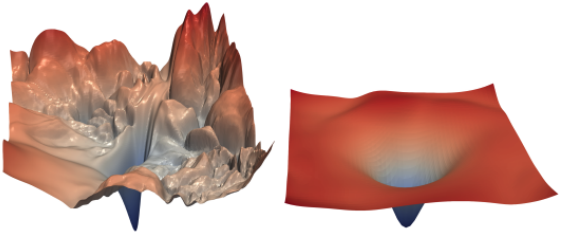
- Recall in gradient boosting, we are iteratively adding a function to the model to expand the capacity.
- Residual connection: Skip connection to prevent gradient vanishing.[5]



[5] He et al. Deep Residual Learning for Image Recognition. CVPR 2016.

ResNet Success

- Now able to train over 100 layers.
- One of the most important network design choices in the past decade.
- Prevalent in almost all network architectures, including Transformers.
- Loss landscape view: Skip connections makes loss smoother -> easier to optimize [6].

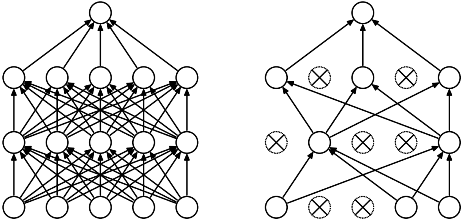


(a) without skip connections (b) with skip connections

[6] Li et al. Visualizing the Loss Landscape of Neural Nets. NIPS 2018.

Dropout[7]

- Want to reduce overfitting in neural networks.
- Stochastically turning off neurons in propagation.

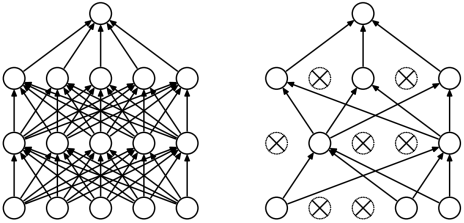


(a) Standard Neural Net (b) After applying dropout.

[7] Srivastava et al. A Simple Way to Prevent Neural Networks from Overfitting. JMLR, 2014.

Dropout[7]

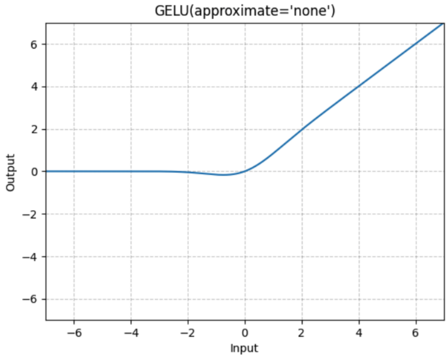
- Want to reduce overfitting in neural networks.
- Stochastically turning off neurons in propagation.
- Training to preserve redundancy.
- Test time: multiplying activations with probability. Model ensembling effect.


(a) Standard Neural Net (b) After applying dropout.

[7] Srivastava et al. A Simple Way to Prevent Neural Networks from Overfitting. JMLR, 2014.

GELU[8]

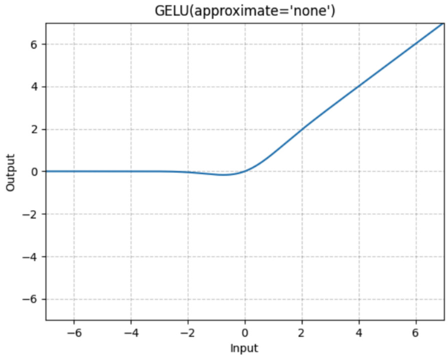
- Gaussian Error Linear Unit - A smoother activation function.
- Motivated by Dropout.



[8]Hendrycks & Gimpel. Gaussian Error Linear Unit (GELU). CoRR abs/1606.08415, 2016.

GELU[8]

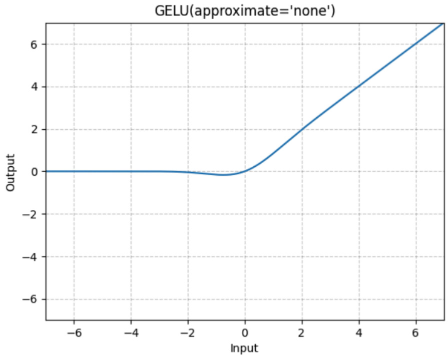
- Gaussian Error Linear Unit - A smoother activation function.
- Motivated by Dropout.
- $f(x) = \mathbb{E}[x \cdot m]$.



[8]Hendrycks & Gimpel. Gaussian Error Linear Unit (GELU). CoRR abs/1606.08415, 2016.

GELU[8]

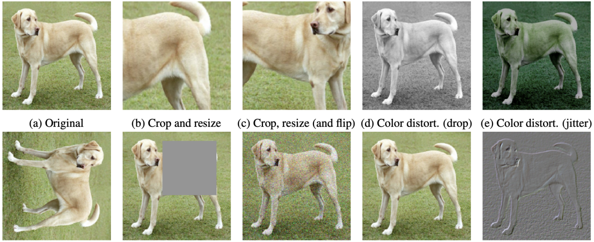
- Gaussian Error Linear Unit - A smoother activation function.
- Motivated by Dropout.
- $f(x) = \mathbb{E}[x \cdot m]$.
- $m \sim Bernoulli(\Phi(x))$.
- $\Phi(x) = P(X \leqslant x)$.
- $X \sim \mathcal{N}(0, 1)$.



[8] Hendrycks & Gimpel. Gaussian Error Linear Unit (GELU). CoRR abs/1606.08415, 2016.

Data augmentation

- Leverage the invariances of images
- Create more data points for free



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

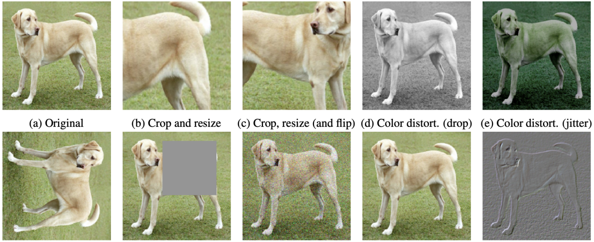
(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Image credit[9]

[9] Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. ICML 2020.

Data augmentation

- Leverage the invariances of images
- Create more data points for free
  - Random cropping



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate {90°, 180°, 270°} (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Image credit[9]

[9] Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. ICML 2020.

Data augmentation

- Leverage the invariances of images
- Create more data points for free
  - Random cropping
  - Left+right flipping

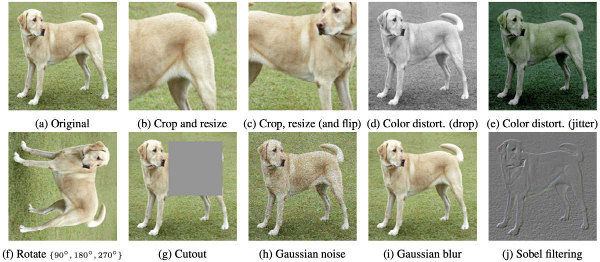
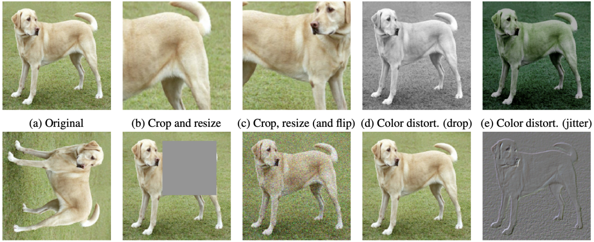


Image credit[9]

[9] Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. ICML 2020.

Data augmentation

- Leverage the invariances of images
- Create more data points for free
  - Random cropping
  - Left+right flipping
  - Random color jittering



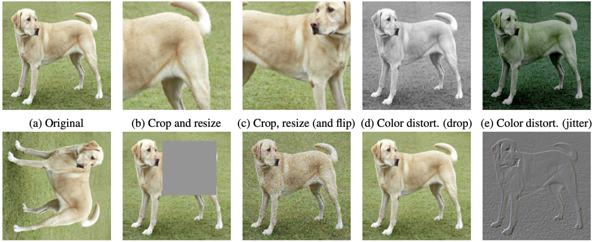
(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)
(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Image credit[9]

[9] Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. ICML 2020.

Data augmentation

- Leverage the invariances of images
- Create more data points for free
  - Random cropping
  - Left+right flipping
  - Random color jittering
  - Random blurring



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

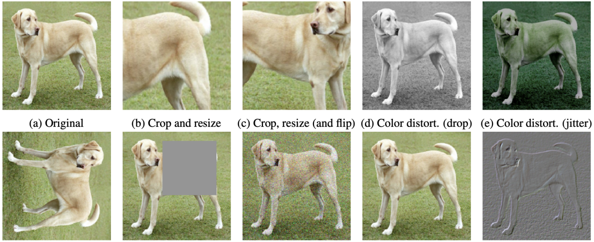
(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Image credit[9]

[9] Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. ICML 2020.

Data augmentation

- Leverage the invariances of images
- Create more data points for free
    - Random cropping
    - Left+right flipping
    - Random color jittering
    - Random blurring
    - Affine warping
    - Etc.



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate {90°, 180°, 270°} (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

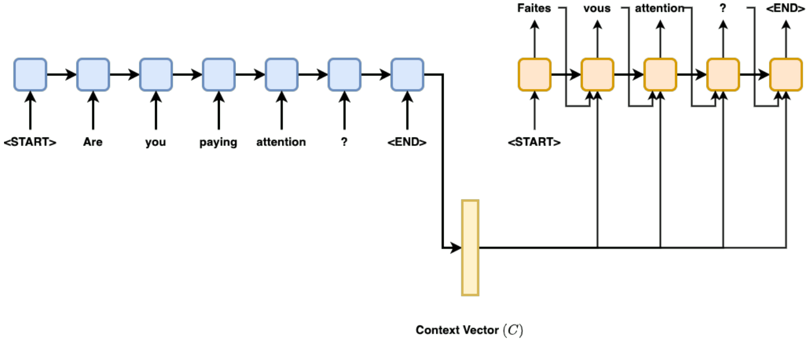
Image credit[9]

[9] Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. ICML 2020.

Language and sequential signals

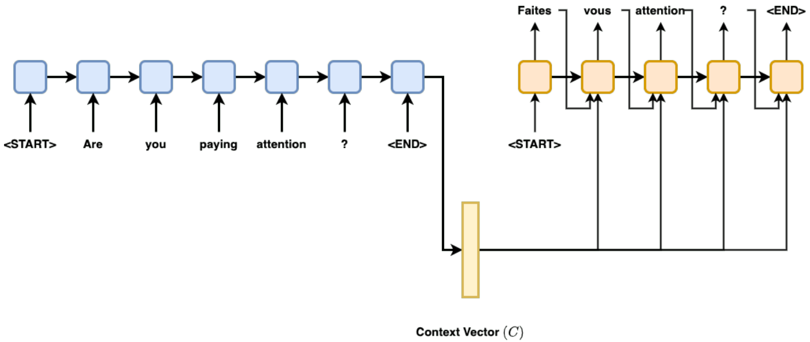
What about natural language

- Neural networks are great for dealing with naturalistic and unstructured signals.



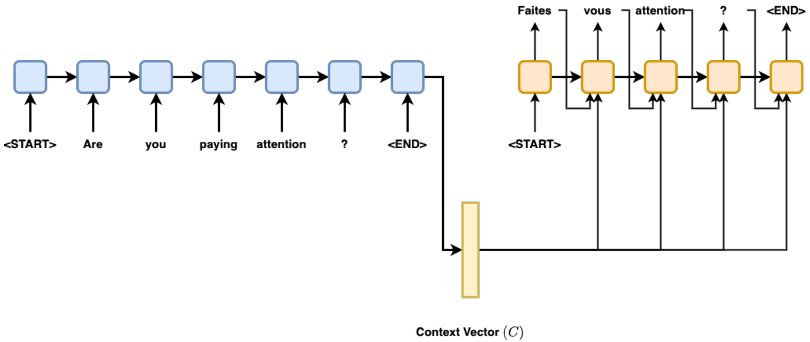
What about natural language

- Neural networks are great for dealing with naturalistic and unstructured signals.
- Past lectures: Feature functions in structured models, but still primitive.



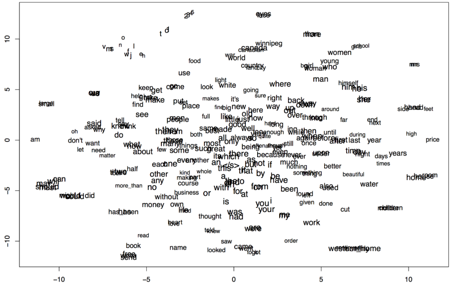
What about natural language

- Neural networks are great for dealing with naturalistic and unstructured signals.
- Past lectures: Feature functions in structured models, but still primitive.
- Design neural networks to accomodate sequential signals such as language.



Word embeddings

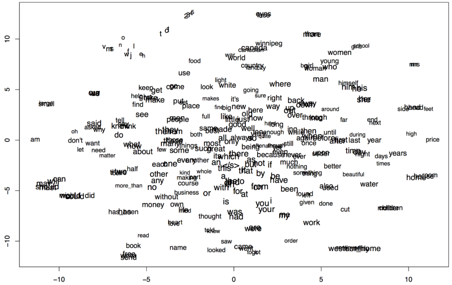
- Neural networks are best dealing with real valued vectors.

 [10]

[10] https://aelang.github.io/word-embeddings.html

Word embeddings

- Neural networks are best dealing with real valued vectors.
- Need to convert words (discrete) into vectors (continuous).

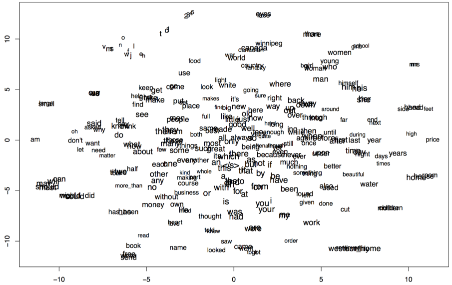


[10]

[10] https://aelang.github.io/word-embeddings.html

Word embeddings

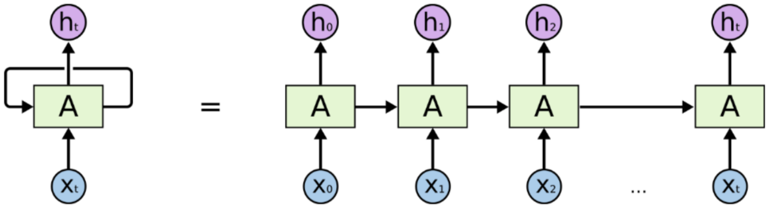
- Neural networks are best dealing with real valued vectors.
- Need to convert words (discrete) into vectors (continuous).
- A large matrix of $V \times D$. V = vocab size, D = network embedding size.



[10]

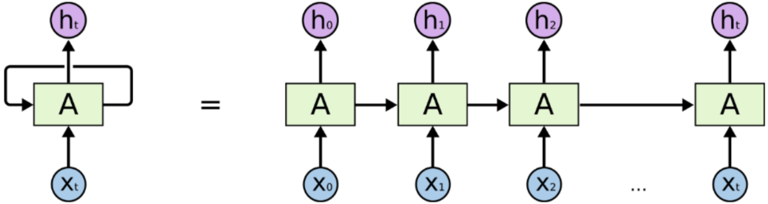
Convolutional vs. recurrent networks

- Recall in images we used the convolution operation.
- We can also use the idea of convolution for temporal signals.
- Another alternative is to use a type of network called recurrent networks.
- Two inputs: x_t is the current input, and h_t is the historical hidden state.
- We can unroll the computation graph into a direct acyclic graph (DAG).



Recurrent neural networks (RNNs)

- A simple RNN can be made similar to a standard NN with one hidden layer.
- $h_t = \tanh(Wh_{t-1} + Ux_t)$.
- $y_t = \text{Softmax}(Vh_t)$.



Gradient vanishing

- Every iteration, we multiply the hidden state h_{t-1} from the previous iteration with W.

Gating function

Attention

Transformers

Autoregressive modeling

Multi-modal learning

Visual attention

Visual question answering

Interim Summary

Interpretability

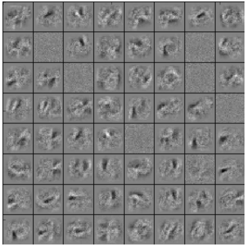
ML Interpretability

- Linear regression: Weights represent feature selection strength
- SVMs: Dual weights represent sample selection
- Bayesian methods: Model the generative process as a probabilistic model, fully transparent
- Decision trees: If-else decision making process
- Neural networks: ?

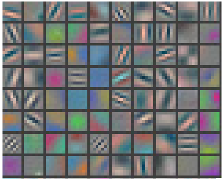
Feature Visualization

- Recall: we can understand what first-layer features are doing by visualizing the weight matrices.
- Higher-level weight matrices are hard to interpret.

Fully connected



Convolutional



Zeiler and Fergus, Visualizing and understanding convolutional networks, ECCV 2014.

- The better the input matches these weights, the more the feature activates.
  - Obvious generalization: visualize higher-level features by seeing what inputs activate them.

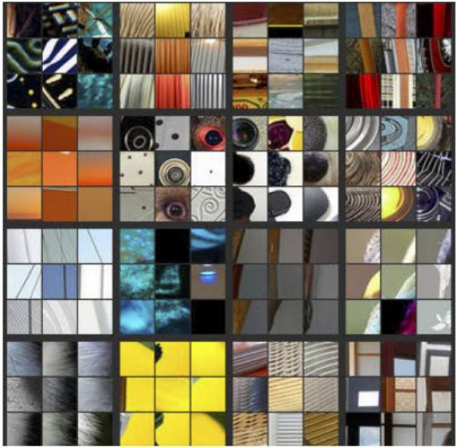
Feature Visualization

- One way to formalize: pick the images in the training set which activate a unit most strongly.
- Here's the visualization for layer 1:

Zeiler and Fergus, Visualizing and understanding convolutional networks, ECCV 2014.

Feature Visualization

- Layer 3:



Zeiler and Fergus, Visualizing and understanding convolutional networks, ECCV 2014.

Feature Visualization

- Layer 4:

Feature Visualization

- Layer 5:

Feature Visualization

- Higher layers seem to pick up more abstract, high-level information.
- Problems?
  - Can't tell what the unit is actually responding to in the image.
  - We may read too much into the results, e.g. a unit may detect red, and the images that maximize its activation will all be stop signs.
- Can use input gradients to diagnose what the unit is responding to.

Feature Visualization

- Input gradients can be hard to interpret.
- Take a good object recognition conv net (Alex Net) and compute the gradient of $\log p(y = \text{“cat”}|\mathrm{x})$:

Original image

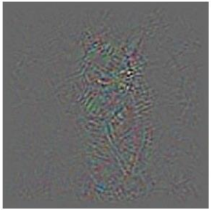
Gradient for “cat”

Feature Visualization

- Guided backprop is a total hack to prevent this cancellation.
- Do the backward pass as normal, but apply the ReLU nonlinearity to all the activation error signals.

$$y = \mathrm{ReLU}(z) \qquad \bar{z} = \begin{cases} \bar{y} & \text{if } z > 0 \text{ and } \bar{y} > 0 \\ 0 & \text{otherwise} \end{cases}$$

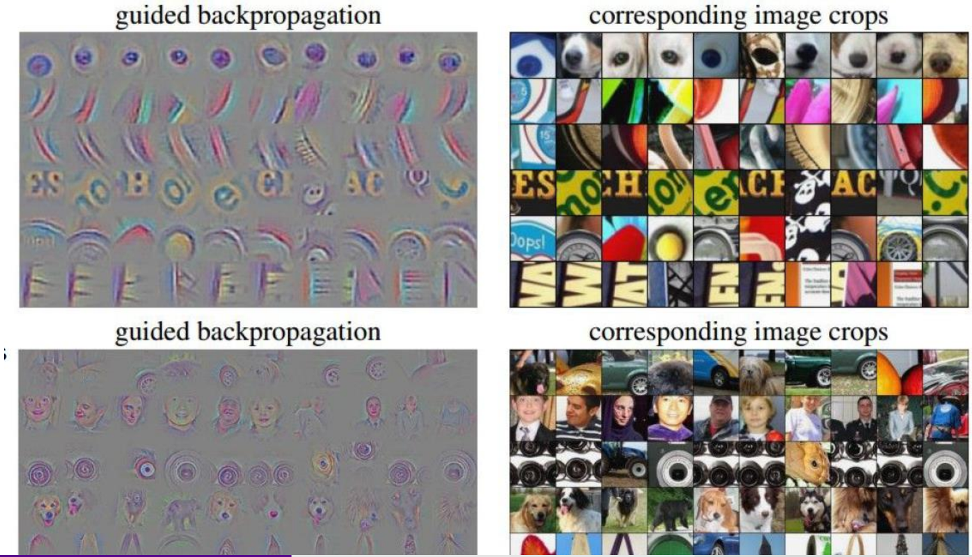
- We want to visualize what excites given unit, not what suppresses it.



Backprop

Guided Backprop

Guided Backprop

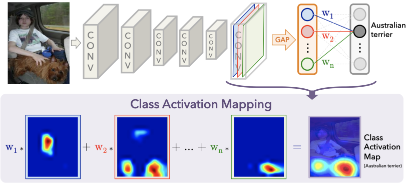


Class activation map (CAM)

Classification networks typically use global avg pooling before the final layer.

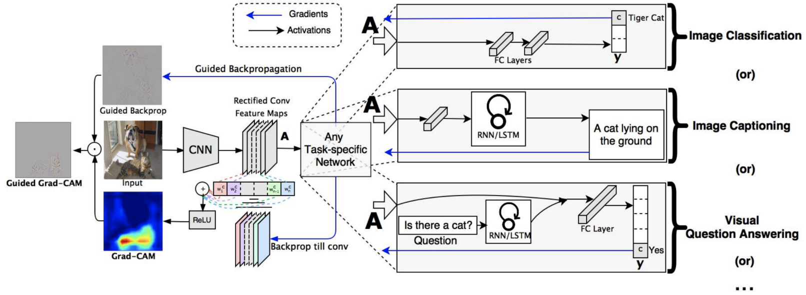
This pooling layer can already contain semantic information.

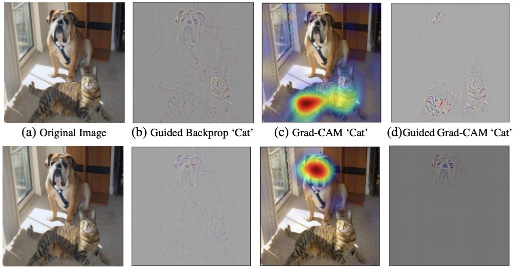
We can visualize a heat map



Zhou et al. Learning deep features for discriminative localization. CVPR 2016.

GradCAM



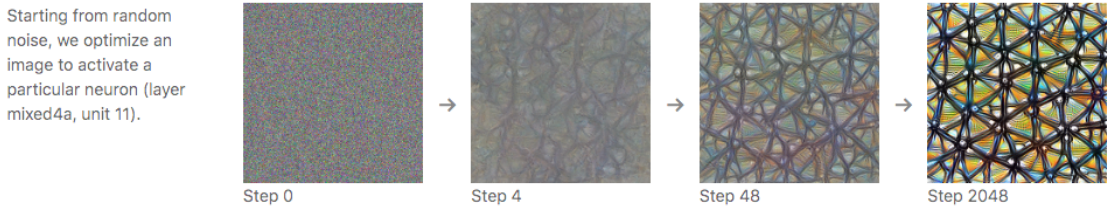


(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (d)Guided Grad-CAM 'Cat'

(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog' (j)Guided Grad-CAM 'Dog'

Gradient Ascent on Images

- Can do gradient ascent on an image to maximize the activation of a given neuron.



https://distill.pub/2017/feature-visualization/

Gradient Ascent on Images

Dataset Examples show us what neurons respond to in practice

Optimization isolates the causes of behavior from mere correlations. A neuron may not be detecting what you initially thought.

Baseball—or stripes? *mixed4a, Unit 6*

Animal faces—or snouts? *mixed4a, Unit 240*

Clouds—or fluffiness? *mixed4a, Unit 453*

Buildings—or sky? *mixed4a, Unit 492*

https://distill.pub/2017/feature-visualization/

Gradient Ascent on Images

- Higher layers in the network often learn higher-level, more interpretable representations

Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a)

https://distill.pub/2017/feature-visualization/

Gradient Ascent on Images

- Higher layers in the network often learn higher-level, more interpretable representations

Parts (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

https://distill.pub/2017/feature-visualization/

- Start with an image, and run a conv net on it.
- Change the image such that units which were already highly activated get activated even more strongly. "Rich get richer."

→

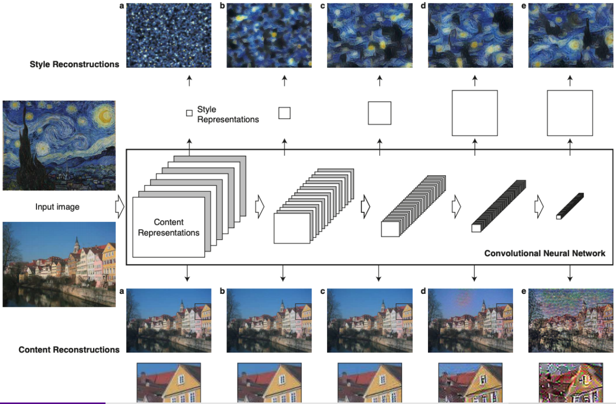
Deep dream

"Admiral Dog!" "The Pig-Snail" "The Camel-Bird" "The Dog-Fish"

Deep dream

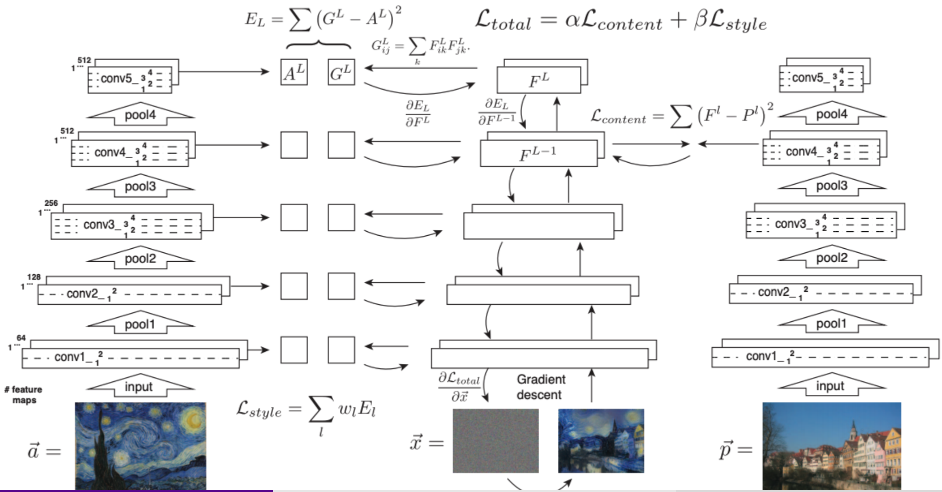
Artistic style transfer

- Activation stores content information
- Activation correlation across space stores style information and discards spatial arrangement



Artistic style transfer

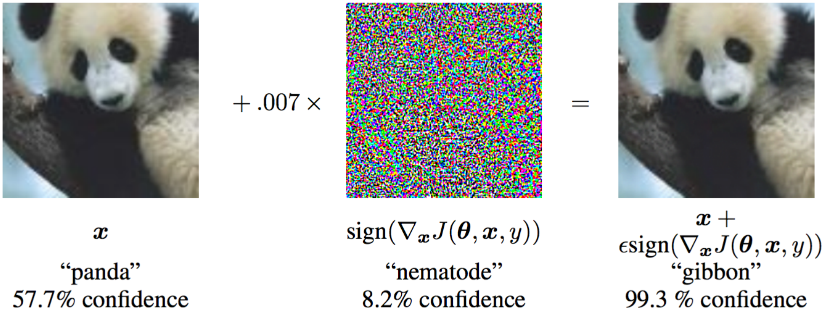
- Optimizing both content & style from random noise



Artistic style transfer

Adversarial Examples

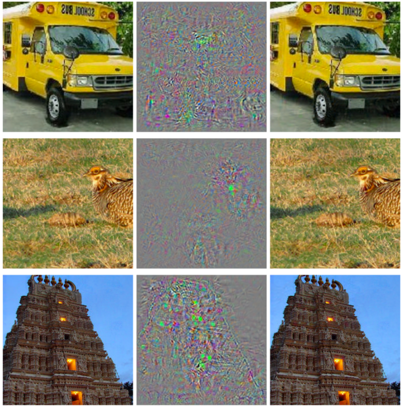
- One of the most surprising findings about neural nets has been the existence of adversarial inputs, i.e. inputs optimized to fool an algorithm.

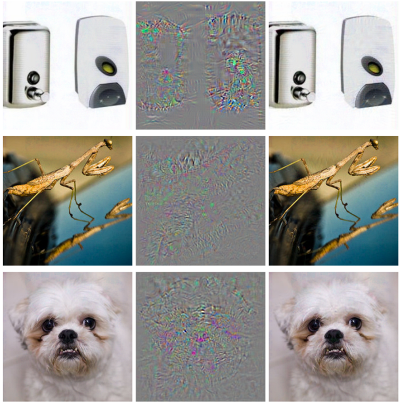


Goodfellow et al., Explaining and harnessing adversarial examples, ICLR 2015.

Adversarial Examples

- The following adversarial examples are misclassified as ostriches. ($10\times$ perturbation visualized in middle.)





Szegedy et al., Intriguing properties of neural networks, ICLR 2014.

Adversarial Examples

- You can print out an adversarial image and take a picture of it, and it still works!

(a) Printout (b) Photo of printout (c) Cropped image

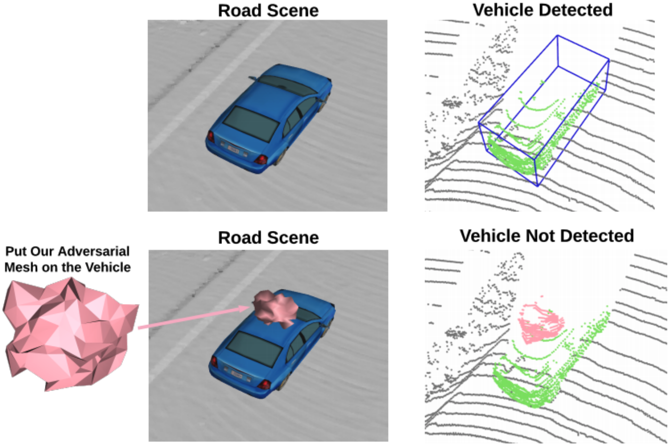
Kurakin et al., Adversarial examples in the physical world, ICLR workshop 2017.

Adversarial Examples

- An adversarial example in the physical world (network thinks it's a gun, from a variety of viewing angles!)

Adversarial Examples

- An adversarial mesh object that can hide cars from LiDAR detector



Tu et al., Physically realizable adversarial examples for LiDAR object detection, CVPR 2020.

Large Language Model Safety

Adversarial Defense

- How to defend from adversarial perturbation is still an active research area.
- One common approach is to train with millions of adversarial examples.
- Needs to train much longer, and also suffers a little from normal accuracy.

K-means Clustering

Unsupervised learning

Goal Discover interesting *structure* in the data.

Unsupervised learning

Goal Discover interesting *structure* in the data.

Formulation Density estimation: $p(x;\theta)$ (often with *latent* variables).

Unsupervised learning

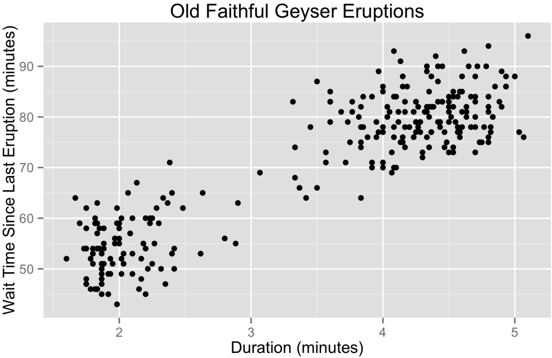
Goal Discover interesting *structure* in the data.

Formulation Density estimation: $p(x;\theta)$ (often with *latent* variables).

Examples

- Discover *clusters*: cluster data into groups.
- Discover *factors*: project high-dimensional data to a small number of "meaningful" dimensions, i.e. dimensionality reduction.
- Discover *graph structures*: learn joint distribution of correlated variables, i.e. graphical models.

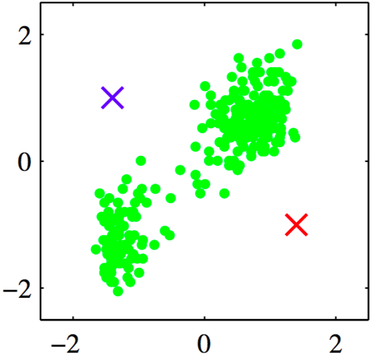
Example: Old Faithful Geyser



- Looks like two clusters.
- How to find these clusters algorithmically?

k-Means: By Example

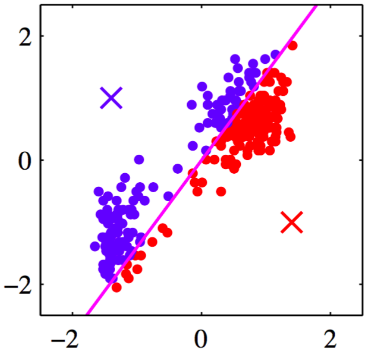
- Standardize the data.
- Choose two cluster centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(a).

k-means: by example

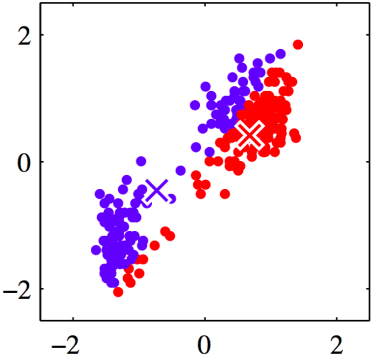
- Assign each point to closest center.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(b).

k-means: by example

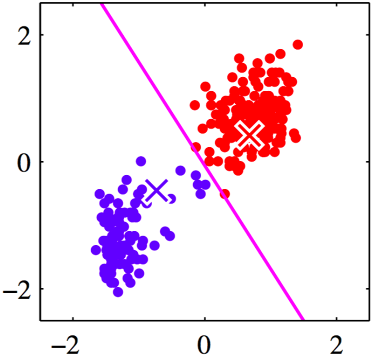
- Compute new cluster centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(c).

k-means: by example

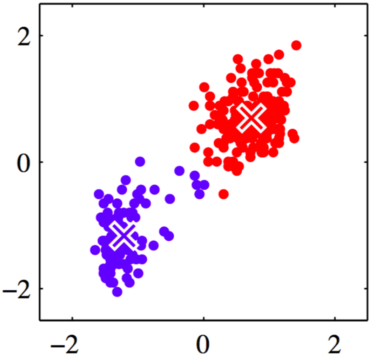
- Assign points to closest center.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(d).

k-means: by example

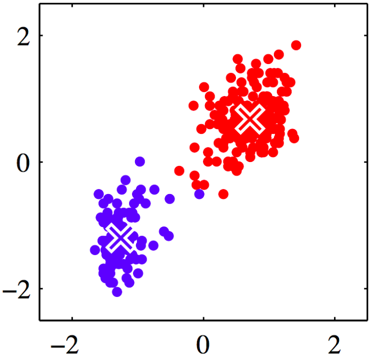
- Compute cluster centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(e).

k-means: by example

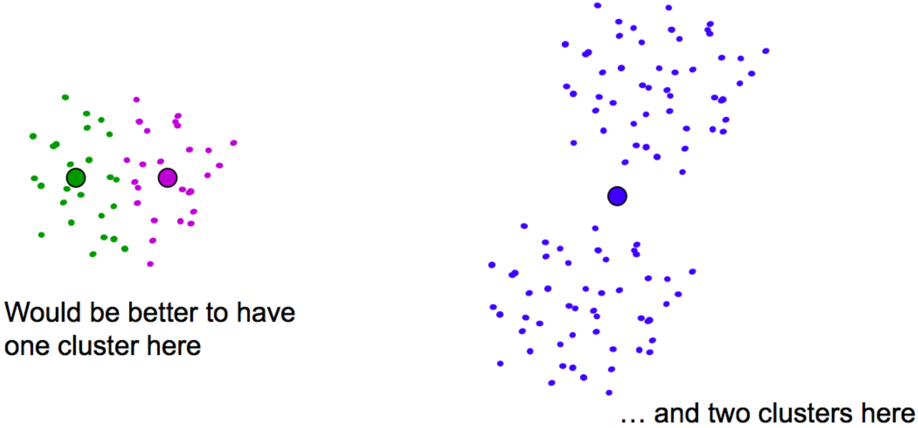
- Iterate until convergence.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(i).

Suboptimal Local Minimum

- The clustering for $k = 3$ below is a local minimum, but suboptimal:



From Sontag's DS-GA 1003, 2014, Lecture 8.

Formalize k-Means

- Dataset $\mathcal{D} = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ where $\mathcal{X} = \mathrm{R}^d$.

Formalize k-Means

- Dataset $\mathcal{D} = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ where $\mathcal{X} = \mathrm{R}^d$.
- Goal: Partition data $\mathcal{D}$ into k disjoint sets $C_1, \ldots, C_k$.

Formalize k-Means

- Dataset $\mathcal{D} = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ where $\mathcal{X} = \mathrm{R}^d$.
- Goal: Partition data $\mathcal{D}$ into k disjoint sets $C_1, \ldots, C_k$.
- Let $c_i \in \{1, \ldots, k\}$ be the cluster assignment of x_i.

Formalize k-Means

- Dataset $\mathcal{D} = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ where $\mathcal{X} = \mathrm{R}^d$.
- Goal: Partition data $\mathcal{D}$ into k disjoint sets $C_1, \ldots, C_k$.
- Let $c_i \in \{1, \ldots, k\}$ be the cluster assignment of x_i.
- The **centroid** of C_i is defined to be

$$\mu_i = \underset{\mu \in \mathcal{X}}{\arg\min} \sum_{x \in C_i} \|x - \mu\|^2. \qquad \text{mean of } C_i \tag{1}$$

Formalize k-Means

- Dataset $\mathcal{D} = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ where $\mathcal{X} = \mathrm{R}^d$.
- Goal: Partition data $\mathcal{D}$ into k disjoint sets $C_1, \ldots, C_k$.
- Let $c_i \in \{1, \ldots, k\}$ be the cluster assignment of x_i.
- The **centroid** of C_i is defined to be

$$\mu_i = \arg\min_{\mu \in \mathcal{X}} \sum_{x \in C_i} \|x - \mu\|^2. \qquad \text{mean of } C_i \tag{1}$$

- The k-means objective is to minimize the distance between each example and its cluster centroid:

$$J(c, \mu) = \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|^2. \tag{2}$$

k-Means: Algorithm

1. Initialize: Randomly choose initial centroids $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$.

k-Means: Algorithm

1. Initialize: Randomly choose initial centroids $\mu_1, \dots, \mu_k \in \mathbb{R}^d$.
2. Repeat until convergence (i.e. c_i doesn't change anymore):

k-Means: Algorithm

1. Initialize: Randomly choose initial centroids $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$.
2. Repeat until convergence (i.e. c_i doesn't change anymore):
    1. For all i, set

$$c_i \leftarrow \arg\min_j \|x_i - \mu_j\|^2. \tag{3}$$

k-Means: Algorithm

1. Initialize: Randomly choose initial centroids $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$.
2. Repeat until convergence (i.e. c_i doesn't change anymore):
    1. For all i, set

    $$c_i \leftarrow \arg\min_j \|x_i - \mu_j\|^2. \tag{3}$$

    2. For all j, set

    $$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \tag{4}$$

k-Means: Algorithm

1. Initialize: Randomly choose initial centroids $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$.
2. Repeat until convergence (i.e. c_i doesn't change anymore):
    1. For all i, set
    $$c_i \leftarrow \arg\min_j \|x_i - \mu_j\|^2. \tag{3}$$
    2. For all j, set
    $$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \tag{4}$$

- Recall the objective: $J(c, \mu) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2$.

k-Means: Algorithm

1. Initialize: Randomly choose initial centroids $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$.
2. Repeat until convergence (i.e. c_i doesn't change anymore):
   1. For all i, set

$$c_i \leftarrow \arg\min_j \|x_i - \mu_j\|^2. \qquad \text{Minimize } J \text{ w.r.t. } c \text{ while fixing } \mu \tag{3}$$

   2. For all j, set

$$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \tag{4}$$

- Recall the objective: $J(c, \mu) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2$.

k-Means: Algorithm

1. Initialize: Randomly choose initial centroids $\mu_1, \ldots, \mu_k \in R^d$.
2. Repeat until convergence (i.e. c_i doesn't change anymore):
    1. For all i, set
    $$c_i \leftarrow \arg\min_j \|x_i - \mu_j\|^2. \qquad \text{Minimize } J \text{ w.r.t. } c \text{ while fixing } \mu \tag{3}$$
    2. For all j, set
    $$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \qquad \text{Minimze } J \text{ w.r.t. } \mu \text{ while fixing } c. \tag{4}$$

- Recall the objective: $J(c, \mu) = \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|^2$.

Avoid bad local minima

k-means converges to a local minimum.

- J is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.

Avoid bad local minima

k-means converges to a local minimum.

- J is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.
- k-**means++**: choose initial centroids that spread over all data points.

Avoid bad local minima

k-means converges to a local minimum.

- J is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.
- k-**means++**: choose initial centroids that spread over all data points.
    - Randomly choose the first centroid from the data points $\mathcal{D}$.

Avoid bad local minima

k-means converges to a local minimum.

- J is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.

- k-**means++**: choose initial centroids that spread over all data points.
  - Randomly choose the first centroid from the data points $\mathcal{D}$.
  - Sequentially choose subsequent centroids from points that are farther away from current centroids:

Avoid bad local minima

k-means converges to a local minimum.

- J is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.
- k-**means++**: choose initial centroids that spread over all data points.
  - Randomly choose the first centroid from the data points $\mathcal{D}$.
  - Sequentially choose subsequent centroids from points that are farther away from current centroids:
    - Compute distance between each x_i and the closest already chosen centroids.

Avoid bad local minima

k-means converges to a local minimum.

- J is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.
- k-**means++**: choose initial centroids that spread over all data points.
    - Randomly choose the first centroid from the data points $\mathcal{D}$.
    - Sequentially choose subsequent centroids from points that are farther away from current centroids:
        - Compute distance between each x_i and the closest already chosen centroids.
        - Randomly choose next centroid with probability proportional to the computed distance squared.

Summary

We've seen

- Clustering—an unsupervised learning problem that aims to discover group assignments.
- k-means:
  - Algorithm: alternating between assigning points to clusters and computing cluster centroids.
  - Objective: minmizing some loss function by cooridinate descent.
  - Converge to a local minimum.

Summary

We've seen

- Clustering—an unsupervised learning problem that aims to discover group assignments.
- k-means:
  - Algorithm: alternating between assigning points to clusters and computing cluster centroids.
  - Objective: minmizing some loss function by cooridinate descent.
  - Converge to a local minimum.

Next, probabilistic model of clustering.

- A generative model of x.
- Maximum likelihood estimation.

Gaussian Mixture Models

Probabilistic Model for Clustering

- Problem setup:
  - There are k clusters (or **mixture components**).
  - We have a probability distribution for each cluster.

Probabilistic Model for Clustering

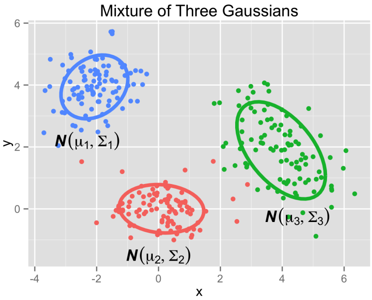
- Problem setup:
  - There are k clusters (or **mixture components**).
  - We have a probability distribution for each cluster.
- Generative story of a **mixture distribution**:
  1. Choose a random cluster $z \in \{1, 2, \ldots, k\}$.
  2. Choose a point from the distribution for cluster z.

Probabilistic Model for Clustering

- Problem setup:
  - There are k clusters (or **mixture components**).
  - We have a probability distribution for each cluster.
- Generative story of a **mixture distribution**:
  1. Choose a random cluster $z \in \{1, 2, \ldots, k\}$.
  2. Choose a point from the distribution for cluster z.

Example:

1. Choose $z \in \{1, 2, 3\}$ with $p(1) = p(2) = p(3) = \frac{1}{3}$.
2. Choose $x \mid z \sim \mathcal{N}(X \mid \mu_z, \Sigma_z)$.



Gaussian mixture model (GMM)

Generative story of GMM with k mixture components:

1. Choose cluster $z \sim \text{Categorical}(\pi_1, \ldots, \pi_k)$.
2. Choose $x \mid z \sim \mathcal{N}(\mu_z, \Sigma_z)$.

Gaussian mixture model (GMM)

Generative story of GMM with k mixture components:

1. Choose cluster $z \sim \text{Categorical}(\pi_1, \dots, \pi_k)$.
2. Choose $x \mid z \sim \mathcal{N}(\mu_z, \Sigma_z)$.

Probability density of x:

- Sum over (marginalize) the **latent variable** z.

$$p(x) = \sum_z p(x, z) \tag{5}$$

$$= \sum_z p(x \mid z) p(z) \tag{6}$$

$$= \sum_k \pi_k \mathcal{N}(\mu_k, \Sigma_k) \tag{7}$$

Identifiability Issues for GMM

- Suppose we have found parameters

$$
\begin{aligned}
\text{Cluster probabilities :} \quad & \pi = (\pi_1, \ldots, \pi_k) \\
\text{Cluster means :} \quad & \mu = (\mu_1, \ldots, \mu_k) \\
\text{Cluster covariance matrices:} \quad & \Sigma = (\Sigma_1, \ldots \Sigma_k)
\end{aligned}
$$

that are at a local minimum.

Identifiability Issues for GMM

- Suppose we have found parameters

$$\begin{aligned} \text{Cluster probabilities}: &\quad \pi = (\pi_1, \ldots, \pi_k) \\ \text{Cluster means}: &\quad \mu = (\mu_1, \ldots, \mu_k) \\ \text{Cluster covariance matrices}: &\quad \Sigma = (\Sigma_1, \ldots \Sigma_k) \end{aligned}$$

  that are at a local minimum.

- What happens if we shuffle the clusters? e.g. Switch the labels for clusters 1 and 2.

Identifiability Issues for GMM

- Suppose we have found parameters

$$\begin{aligned} \text{Cluster probabilities}: &\quad \pi = (\pi_1, \ldots, \pi_k) \\ \text{Cluster means}: &\quad \mu = (\mu_1, \ldots, \mu_k) \\ \text{Cluster covariance matrices}: &\quad \Sigma = (\Sigma_1, \ldots \Sigma_k) \end{aligned}$$

  that are at a local minimum.

- What happens if we shuffle the clusters? e.g. Switch the labels for clusters 1 and 2.
- We'll get the same likelihood. How many such equivalent settings are there?

Identifiability Issues for GMM

- Suppose we have found parameters

$$\begin{aligned} \text{Cluster probabilities}: \quad & \pi = (\pi_1, \ldots, \pi_k) \\ \text{Cluster means}: \quad & \mu = (\mu_1, \ldots, \mu_k) \\ \text{Cluster covariance matrices}: \quad & \Sigma = (\Sigma_1, \ldots \Sigma_k) \end{aligned}$$

  that are at a local minimum.

- What happens if we shuffle the clusters? e.g. Switch the labels for clusters 1 and 2.
- We'll get the same likelihood. How many such equivalent settings are there?
- Assuming all clusters are distinct, there are $k!$ equivalent solutions.

Identifiability Issues for GMM

- Suppose we have found parameters

$$
\begin{aligned}
\text{Cluster probabilities}: &\quad \pi = (\pi_1, \dots, \pi_k) \\
\text{Cluster means}: &\quad \mu = (\mu_1, \dots, \mu_k) \\
\text{Cluster covariance matrices}: &\quad \Sigma = (\Sigma_1, \dots \Sigma_k)
\end{aligned}
$$

  that are at a local minimum.

- What happens if we shuffle the clusters? e.g. Switch the labels for clusters 1 and 2.
- We'll get the same likelihood. How many such equivalent settings are there?
- Assuming all clusters are distinct, there are $k!$ equivalent solutions.
- Not a problem *per se*, but something to be aware of.

Learning GMMs

How to learn the parameters π_k, μ_k, Σ_k?

Learning GMMs

How to learn the parameters π_k, μ_k, Σ_k?

- MLE (also called maximize marginal likelihood).
- Log likelihood of data:

$$
\begin{aligned}
L(\theta) &= \sum_{i=1}^{n} \log p(x_i;\theta) && (8) \\
&= \sum_{i=1}^{n} \log \sum_{z} p(x, z;\theta) && (9)
\end{aligned}
$$

Learning GMMs

How to learn the parameters π_k, μ_k, Σ_k?

- MLE (also called maximize marginal likelihood).
- Log likelihood of data:

$$
\begin{aligned}
L(\theta) &= \sum_{i=1}^{n} \log p(x_i; \theta) && (8) \\
&= \sum_{i=1}^{n} \log \sum_{z} p(x, z; \theta) && (9)
\end{aligned}
$$

- Cannot push log into the sum... z and x are coupled.
- No closed-form solution for GMM—try to compute the gradient yourself!

Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = -\sum_{i=1}^{n} \log \left\{ \sum_{z=1}^{k} \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\} ?$$

[11] See Hosseini and Sra's Manifold Optimization for Gaussian Mixture Models for discussion and further references.

Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) \quad = \quad -\sum_{i=1}^{n} \log \left\{ \sum_{z=1}^{k} \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\} ?$$

- Can be done, in principle – but need to be clever about it.

[11] See Hosseini and Sra's Manifold Optimization for Gaussian Mixture Models for discussion and further references.

Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) \quad = \quad -\sum_{i=1}^{n} \log \left\{ \sum_{z=1}^{k} \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\} ?$$

- Can be done, in principle – but need to be clever about it.

- For example, each covariance matrix $\Sigma_1, \ldots, \Sigma_k$ has to be positive semidefinite.

[11] See Hosseini and Sra's Manifold Optimization for Gaussian Mixture Models for discussion and further references.

Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) \quad = \quad -\sum_{i=1}^{n} \log \left\{ \sum_{z=1}^{k} \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\}?$$

- Can be done, in principle – but need to be clever about it.
- For example, each covariance matrix $\Sigma_1, \ldots, \Sigma_k$ has to be positive semidefinite.
- How to maintain that constraint?

[11] See Hosseini and Sra's Manifold Optimization for Gaussian Mixture Models for discussion and further references.

Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = -\sum_{i=1}^{n} \log \left\{ \sum_{z=1}^{k} \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\}?$$

- Can be done, in principle – but need to be clever about it.
- For example, each covariance matrix $\Sigma_1, \ldots, \Sigma_k$ has to be positive semidefinite.
- How to maintain that constraint?
  - Rewrite $\Sigma_i = M_i M_i^T$, where M_i is an unconstrained matrix.

[11] See Hosseini and Sra's Manifold Optimization for Gaussian Mixture Models for discussion and further references.

Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) \quad = \quad -\sum_{i=1}^{n} \log \left\{ \sum_{z=1}^{k} \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\} ?$$

- Can be done, in principle – but need to be clever about it.
- For example, each covariance matrix $\Sigma_1, \ldots, \Sigma_k$ has to be positive semidefinite.
- How to maintain that constraint?
  - Rewrite $\Sigma_i = M_i M_i^T$, where M_i is an unconstrained matrix.
  - Then Σ_i is positive semidefinite.

[11] See Hosseini and Sra's Manifold Optimization for Gaussian Mixture Models for discussion and further references.

Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = -\sum_{i=1}^{n} \log \left\{ \sum_{z=1}^{k} \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\}?$$

- Can be done, in principle – but need to be clever about it.
- For example, each covariance matrix $\Sigma_1, \ldots, \Sigma_k$ has to be positive semidefinite.
- How to maintain that constraint?
  - Rewrite $\Sigma_i = M_i M_i^T$, where M_i is an unconstrained matrix.
  - Then Σ_i is positive semidefinite.
- Even then, pure gradient-based methods have trouble.[11]

[11] See Hosseini and Sra's Manifold Optimization for Gaussian Mixture Models for discussion and further references.

Learning GMMs: observable case

Suppose we observe cluster assignments z. Then MLE is easy:

$$n_z = \sum_{i=1}^{n} \mathbb{1}[z_i = z] \qquad \text{\# examples in each cluster} \tag{10}$$

$$\hat{\pi}(z) = \frac{n_z}{n} \qquad \text{fraction of examples in each cluster} \tag{11}$$

$$\hat{\mu}_z = \frac{1}{n_z} \sum_{i:z_i=z} x_i \qquad \text{empirical cluster mean} \tag{12}$$

$$\hat{\Sigma}_z = \frac{1}{n_z} \sum_{i:z_i=z} (x_i - \hat{\mu}_z)(x_i - \hat{\mu}_z)^T. \qquad \text{empirical cluster covariance} \tag{13}$$

Learning GMMs: inference

The inference problem: observe x, want to know z.

$$ \tag{16} $$

Learning GMMs: inference

The inference problem: observe x, want to know z.

$$p(z=j \mid x_i) = p(x, z=j)/p(x) \tag{14}$$

$$\tag{16}$$

Learning GMMs: inference

The inference problem: observe x, want to know z.

$$
\begin{aligned}
p(z = j \mid x_i) &= p(x, z = j)/p(x) && (14)\\
&= \frac{p(x \mid z = j)p(z = j)}{\sum_k p(x \mid z = k)p(z = k)} && (15)\\
& && (16)
\end{aligned}
$$

Learning GMMs: inference

The inference problem: observe x, want to know z.

$$p(z=j \mid x_i) = p(x, z=j)/p(x) \tag{14}$$

$$= \frac{p(x \mid z=j)p(z=j)}{\sum_k p(x \mid z=k)p(z=k)} \tag{15}$$

$$= \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_k \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)} \tag{16}$$

Learning GMMs: inference

The inference problem: observe x, want to know z.

$$\begin{aligned}
p(z=j \mid x_i) &= p(x, z=j)/p(x) && (14)\\
&= \frac{p(x \mid z=j)p(z=j)}{\sum_k p(x \mid z=k)p(z=k)} && (15)\\
&= \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_k \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)} && (16)
\end{aligned}$$

- $p(z \mid x)$ is a *soft assignment*.
- If we know the parameters μ, Σ, π, this would be easy to compute.

EM for GMM

Let's compute the cluster assignments and the parameters iteratively.

EM for GMM

Let's compute the cluster assignments and the parameters iteratively.

The expectation-minimization (EM) algorithm:

1. Initialize parameters μ, Σ, π randomly.
2. Run until convergence:

EM for GMM

Let's compute the cluster assignments and the parameters iteratively.

The expectation-minimization (EM) algorithm:

1. Initialize parameters μ, Σ, π randomly.
2. Run until convergence:
    1. E-step: fill in latent variables by inference.
        - compute soft assignments $p(z \mid x_i)$ for all i.

EM for GMM

Let's compute the cluster assignments and the parameters iteratively.

The expectation-minimization (EM) algorithm:

1. Initialize parameters μ, Σ, π randomly.
2. Run until convergence:
   1. E-step: fill in latent variables by inference.
      - compute soft assignments $p(z \mid x_i)$ for all i.
   2. M-step: standard MLE for μ, Σ, π given "observed" variables.
      - Equivalent to MLE in the observable case on data weighted by $p(z \mid x_i)$.

M-step for GMM

- Let $p(z \mid x)$ be the soft assignments:

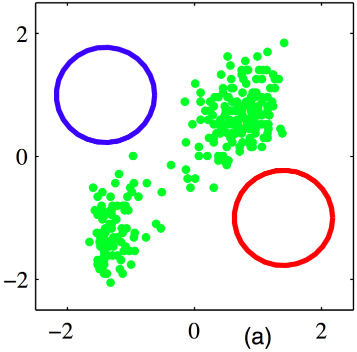
$$\gamma_i^j = \frac{\pi_j^{\text{old}} \mathcal{N}\left(x_i \mid \mu_j^{\text{old}}, \Sigma_j^{\text{old}}\right)}{\sum_{c=1}^{k} \pi_c^{\text{old}} \mathcal{N}\left(x_i \mid \mu_c^{\text{old}}, \Sigma_c^{\text{old}}\right)}.$$

- Exercise: show that

$$\begin{aligned}
n_z &= \sum_{i=1}^{n} \gamma_i^z \\
\mu_z^{\text{new}} &= \frac{1}{n_z} \sum_{i=1}^{n} \gamma_i^z x_i \\
\Sigma_z^{\text{new}} &= \frac{1}{n_z} \sum_{i=1}^{n} \gamma_i^z \left(x_i - \mu_z^{\text{new}}\right)\left(x_i - \mu_z^{\text{new}}\right)^T \\
\pi_z^{\text{new}} &= \frac{n_z}{n}.
\end{aligned}$$

EM for GMM

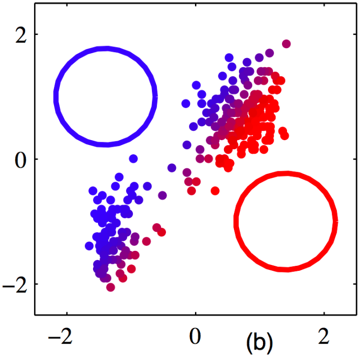
- Initialization



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

EM for GMM

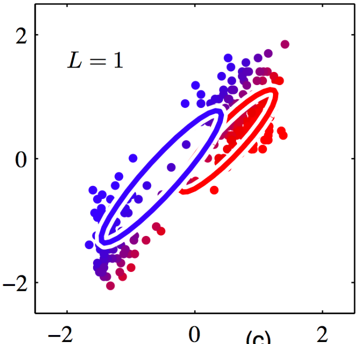
- First soft assignment:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

EM for GMM

- First soft assignment:

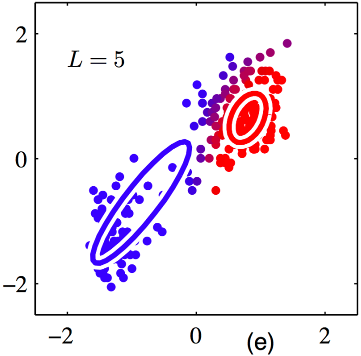


(c)

From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

EM for GMM

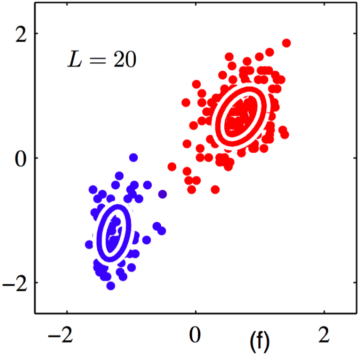
- After 5 rounds of EM:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

EM for GMM

- After 20 rounds of EM:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

EM for GMM: Summary

- EM is a general algorithm for learning latent variable models.
- *Key idea*: if data was fully observed, then MLE is easy.
  - E-step: fill in latent variables by computing $p(z \mid x, \theta)$.
  - M-step: standard MLE given fully observed data.
- Simpler and more efficient than gradient methods.
- Can prove that EM monotonically improves the likelihood and converges to a local minimum.
- k-means is a special case of EM for GMM with *hard assignments*, also called hard-EM.

Latent Variable Models

General Latent Variable Model

- Two sets of random variables: z and x.
- z consists of unobserved **hidden variables**.
- x consists of **observed variables**.

General Latent Variable Model

- Two sets of random variables: z and x.
- z consists of unobserved **hidden variables**.
- x consists of **observed variables**.
- Joint probability model parameterized by $\theta \in \Theta$:

$$p(x, z \mid \theta)$$

General Latent Variable Model

- Two sets of random variables: z and x.
- z consists of unobserved **hidden variables**.
- x consists of **observed variables**.
- Joint probability model parameterized by $\theta \in \Theta$:

$$p(x, z \mid \theta)$$

Definition

A **latent variable model** is a probability model for which certain variables are never observed.

General Latent Variable Model

- Two sets of random variables: z and x.
- z consists of unobserved **hidden variables**.
- x consists of **observed variables**.
- Joint probability model parameterized by $\theta \in \Theta$:

$$p(x, z \mid \theta)$$

Definition

A **latent variable model** is a probability model for which certain variables are never observed.

e.g. The Gaussian mixture model is a latent variable model.

Complete and Incomplete Data

- Suppose we observe some data $(x_1, \ldots, x_n)$.

Complete and Incomplete Data

- Suppose we observe some data $(x_1, \ldots, x_n)$.
- To simplify notation, take x to represent the entire dataset

$$x = (x_1, \ldots, x_n),$$

and z to represent the corresponding unobserved variables

$$z = (z_1, \ldots, z_n).$$

- An observation of x is called an **incomplete data set**.
- An observation (x, z) is called a **complete data set**.

Our Objectives

- **Learning problem**: Given incomplete dataset x, find MLE

$$\hat{\theta} = \arg\max_{\theta} p(x \mid \theta).$$

Our Objectives

- **Learning problem**: Given incomplete dataset x, find MLE

$$\hat{\theta} = \underset{\theta}{\arg\max}\, p(x \mid \theta).$$

- **Inference problem**: Given x, find conditional distribution over z:

$$p(z \mid x, \theta).$$

Our Objectives

- **Learning problem**: Given incomplete dataset x, find MLE

$$\hat{\theta} = \arg\max_{\theta} p(x \mid \theta).$$

- **Inference problem**: Given x, find conditional distribution over z:

$$p(z \mid x, \theta).$$

- For Gaussian mixture model, learning is hard, inference is easy.
- For more complicated models, inference can also be hard. (See DSGA-1005)

Log-Likelihood and Terminology

- Note that

$$\arg\max_{\theta} p(x \mid \theta) = \arg\max_{\theta} \left[\log p(x \mid \theta)\right].$$

Log-Likelihood and Terminology

- Note that

$$\arg\max_{\theta} p(x \mid \theta) = \arg\max_{\theta} \left[\log p(x \mid \theta)\right].$$

- Often easier to work with this "**log-likelihood**".

Log-Likelihood and Terminology

- Note that
$$\arg\max_{\theta} p(x \mid \theta) = \arg\max_{\theta} \left[\log p(x \mid \theta)\right].$$
- Often easier to work with this "**log-likelihood**".
- We often call $p(x)$ the **marginal likelihood**,
  - because it is $p(x, z)$ with z "marginalized out":
$$p(x) = \sum_{z} p(x, z)$$

Log-Likelihood and Terminology

- Note that
$$\arg\max_{\theta} p(x \mid \theta) = \arg\max_{\theta} \left[\log p(x \mid \theta)\right].$$
- Often easier to work with this "**log-likelihood**".
- We often call $p(x)$ the **marginal likelihood**,
  - because it is $p(x, z)$ with z "marginalized out":
$$p(x) = \sum_{z} p(x, z)$$
- We often call $p(x, z)$ the **joint**. (for "joint distribution")

Log-Likelihood and Terminology

- Note that
$$\arg\max_{\theta} p(x \mid \theta) = \arg\max_{\theta} \left[\log p(x \mid \theta)\right].$$
- Often easier to work with this "**log-likelihood**".
- We often call $p(x)$ the **marginal likelihood**,
  - because it is $p(x, z)$ with z "marginalized out":
$$p(x) = \sum_{z} p(x, z)$$
- We often call $p(x, z)$ the **joint**. (for "joint distribution")
- Similarly, $\log p(x)$ is the **marginal log-likelihood**.

EM Algorithm

Intuition

Problem: marginal log-likelihood $\log p(x;\theta)$ is hard to optimize (observing only x)

Observation: complete data log-likelihood $\log p(x,z;\theta)$ is easy to optimize (observing both x and z)

Idea: guess a distribution of the latent variables $q(z)$ (soft assignments)

Maximize the **expected complete data log-likelihood**:

$$\max_{\theta} \sum_{z \in \mathcal{Z}} q(z) \log p(x,z;\theta)$$

EM assumption: the expected complete data log-likelihood is easy to optimize

Why should this work?

Math Prerequisites

Jensen's Inequality

Theorem (Jensen's Inequality)

*If $f : \mathbb{R} \to \mathbb{R}$ is a **convex** function, and x is a random variable, then*

$$\mathbb{E}f(x) \geqslant f(\mathbb{E}x).$$

Jensen's Inequality

Theorem (Jensen's Inequality)

*If $f : \mathrm{R} \to \mathrm{R}$ is a **convex** function, and x is a random variable, then*

$$\mathbb{E}f(x) \geqslant f(\mathbb{E}x).$$

*Moreover, if f is **strictly convex**, then equality implies that $x = \mathbb{E}x$ with probability 1 (i.e. x is a constant).*

Jensen's Inequality

Theorem (Jensen's Inequality)

If $f : \mathrm{R} \to \mathrm{R}$ is a ***convex*** *function, and x is a random variable, then*

$$\mathbb{E}f(x) \geqslant f(\mathbb{E}x).$$

Moreover, if f is ***strictly convex****, then equality implies that $x = \mathbb{E}x$ with probability 1 (i.e. x is a constant).*

- e.g. $f(x) = x^2$ is convex. So $\mathbb{E}x^2 \geqslant (\mathbb{E}x)^2$. Thus

$$\mathrm{Var}(x) = \mathbb{E}x^2 - (\mathbb{E}x)^2 \geqslant 0.$$

Kullback-Leibler Divergence

- Let $p(x)$ and $q(x)$ be probability mass functions (PMFs) on $\mathcal{X}$.
- How can we measure how "different" p and q are?

Kullback-Leibler Divergence

- Let $p(x)$ and $q(x)$ be probability mass functions (PMFs) on $\mathcal{X}$.
- How can we measure how "different" p and q are?
- The **Kullback-Leibler** or **"KL" Divergence** is defined by

$$\mathrm{KL}(p\|q) \quad = \quad \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}.$$

(Assumes $q(x)=0$ implies $p(x)=0$.)

Kullback-Leibler Divergence

- Let $p(x)$ and $q(x)$ be probability mass functions (PMFs) on $\mathcal{X}$.
- How can we measure how "different" p and q are?
- The **Kullback-Leibler** or **"KL" Divergence** is defined by

$$\mathrm{KL}(p\|q) \quad = \quad \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}.$$

  (Assumes $q(x)=0$ implies $p(x)=0$.)

- Can also write this as

$$\mathrm{KL}(p\|q) \quad = \quad \mathbb{E}_{x \sim p} \log \frac{p(x)}{q(x)}.$$

Gibbs Inequality ($\mathbf{KL}(p\|q) \geqslant 0$ and $\mathbf{KL}(p\|p) = 0$)

Theorem (Gibbs Inequality)

Let $p(x)$ and $q(x)$ be PMFs on $\mathcal{X}$. Then

$$KL(p\|q) \geqslant 0,$$

with equality iff $p(x) = q(x)$ for all $x \in \mathcal{X}$.

Gibbs Inequality ($\mathrm{KL}(p\|q) \geqslant 0$ and $\mathrm{KL}(p\|p) = 0$)

Theorem (Gibbs Inequality)

Let $p(x)$ and $q(x)$ be PMFs on $\mathcal{X}$. Then

$$KL(p\|q) \geqslant 0,$$

with equality iff $p(x) = q(x)$ for all $x \in \mathcal{X}$.

- KL divergence measures the "distance" between distributions.

Gibbs Inequality ($\mathbf{KL}(p\|q) \geqslant 0$ and $\mathbf{KL}(p\|p) = 0$)

Theorem (Gibbs Inequality)

Let $p(x)$ and $q(x)$ be PMFs on $\mathcal{X}$. Then

$$KL(p\|q) \geqslant 0,$$

with equality iff $p(x) = q(x)$ for all $x \in \mathcal{X}$.

- KL divergence measures the "distance" between distributions.
- Note:
  - KL divergence **not a metric**.
  - KL divergence is **not symmetric**.

Gibbs Inequality: Proof

$$\mathrm{KL}(p\|q) \quad = \quad \mathbb{E}_p\left[-\log\left(\frac{q(x)}{p(x)}\right)\right]$$

Gibbs Inequality: Proof

$$\begin{aligned}\mathrm{KL}(p\|q) &= \mathbb{E}_p\left[-\log\left(\frac{q(x)}{p(x)}\right)\right] \\ &\geqslant -\log\left[\mathbb{E}_p\left(\frac{q(x)}{p(x)}\right)\right] \qquad \text{(Jensen's)}\end{aligned}$$

Gibbs Inequality: Proof

$$\begin{aligned}
\mathrm{KL}(p\|q) &= \mathbb{E}_p\left[-\log\left(\frac{q(x)}{p(x)}\right)\right] \\
&\geqslant -\log\left[\mathbb{E}_p\left(\frac{q(x)}{p(x)}\right)\right] \qquad \text{(Jensen's)} \\
&= -\log\left[\sum_{\{x|p(x)>0\}} p(x)\frac{q(x)}{p(x)}\right]
\end{aligned}$$

Gibbs Inequality: Proof

$$
\begin{aligned}
\mathrm{KL}(p\|q) &= \mathbb{E}_p\left[-\log\left(\frac{q(x)}{p(x)}\right)\right] \\
&\geqslant -\log\left[\mathbb{E}_p\left(\frac{q(x)}{p(x)}\right)\right] \qquad \text{(Jensen's)} \\
&= -\log\left[\sum_{\{x|p(x)>0\}} p(x)\frac{q(x)}{p(x)}\right] \\
&= -\log\left[\sum_{x\in\mathcal{X}} q(x)\right]
\end{aligned}
$$

Gibbs Inequality: Proof

$$
\begin{aligned}
\mathrm{KL}(p\|q) &= \mathbb{E}_p\left[-\log\left(\frac{q(x)}{p(x)}\right)\right] \\
&\geqslant -\log\left[\mathbb{E}_p\left(\frac{q(x)}{p(x)}\right)\right] \qquad \text{(Jensen's)} \\
&= -\log\left[\sum_{\{x|p(x)>0\}} p(x)\frac{q(x)}{p(x)}\right] \\
&= -\log\left[\sum_{x\in\mathcal{X}} q(x)\right] \\
&= -\log 1 = 0.
\end{aligned}
$$

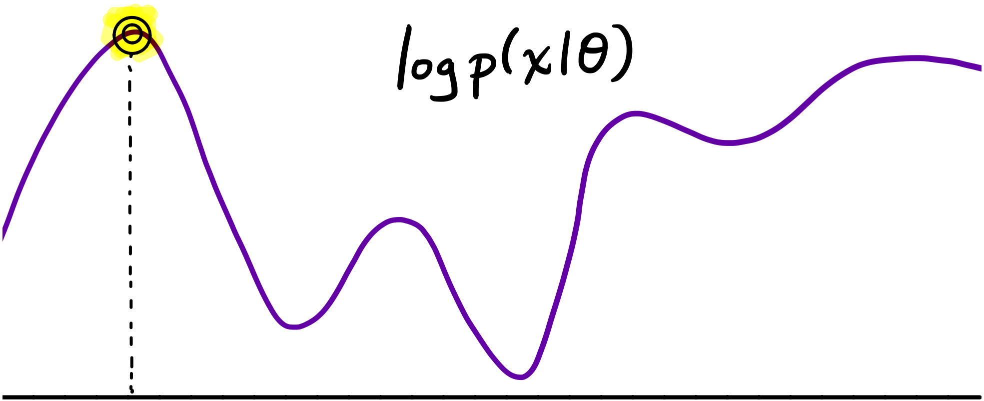
Gibbs Inequality: Proof

$$
\begin{aligned}
\mathrm{KL}(p\|q) &= \mathbb{E}_p\left[-\log\left(\frac{q(x)}{p(x)}\right)\right] \\
&\geqslant -\log\left[\mathbb{E}_p\left(\frac{q(x)}{p(x)}\right)\right] \qquad \text{(Jensen's)} \\
&= -\log\left[\sum_{\{x|p(x)>0\}} p(x)\frac{q(x)}{p(x)}\right] \\
&= -\log\left[\sum_{x\in\mathcal{X}} q(x)\right] \\
&= -\log 1 = 0.
\end{aligned}
$$

- Since $-\log$ is strictly convex, we have strict equality iff $q(x)/p(x)$ is a constant, which implies $q = p$.

The ELBO: Family of Lower Bounds on $\log p(x \mid \theta)$

The Maximum Likelihood Estimator



$$\theta^* = \arg\max_{\theta} \left[\log p(x|\theta) \right]$$

θ

Lower bound of the marginal log-likelihood

$$\log p(x;\theta) \quad = \quad \log \sum_{z \in \mathcal{Z}} p(x,z;\theta)$$

Lower bound of the marginal log-likelihood

$$
\begin{aligned}
\log p(x;\theta) &= \log \sum_{z \in \mathcal{Z}} p(x, z;\theta) \\
&= \log \sum_{z \in \mathcal{Z}} q(z) \frac{p(x, z;\theta)}{q(z)}
\end{aligned}
$$

Lower bound of the marginal log-likelihood

$$\begin{aligned}
\log p(x;\theta) &= \log \sum_{z\in\mathcal{Z}} p(x,z;\theta) \\
&= \log \sum_{z\in\mathcal{Z}} q(z)\frac{p(x,z;\theta)}{q(z)} \\
&\geqslant \sum_{z\in\mathcal{Z}} q(z)\log \frac{p(x,z;\theta)}{q(z)}
\end{aligned}$$

Lower bound of the marginal log-likelihood

$$\begin{aligned}
\log p(x;\theta) &= \log \sum_{z\in\mathcal{Z}} p(x,z;\theta) \\
&= \log \sum_{z\in\mathcal{Z}} q(z)\frac{p(x,z;\theta)}{q(z)} \\
&\geqslant \sum_{z\in\mathcal{Z}} q(z)\log \frac{p(x,z;\theta)}{q(z)} \\
&\stackrel{\text{def}}{=} \mathcal{L}(q,\theta)
\end{aligned}$$

- **Evidence**: $\log p(x;\theta)$
- **Evidence lower bound (ELBO)**: $\mathcal{L}(q,\theta)$
- q: chosen to be a family of tractable distributions
- Idea: *maximize the ELBO* instead of $\log p(x;\theta)$

MLE, EM, and the ELBO

- The MLE is defined as a maximum over θ:

$$\hat{\theta}_{\text{MLE}} = \underset{\theta}{\arg\max}\,[\log p(x \mid \theta)]\,.$$

- For any PMF $q(z)$, we have a lower bound on the marginal log-likelihood

$$\log p(x \mid \theta) \geqslant \mathcal{L}(q, \theta).$$

- In EM algorithm, we maximize the lower bound (ELBO) over θ and q:

$$\hat{\theta}_{\text{EM}} \approx \underset{\theta}{\arg\max}\left[\max_{q} \mathcal{L}(q, \theta)\right]$$

MLE, EM, and the ELBO

- The MLE is defined as a maximum over θ:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \left[\log p(x \mid \theta)\right].$$

- For any PMF $q(z)$, we have a lower bound on the marginal log-likelihood

$$\log p(x \mid \theta) \geqslant \mathcal{L}(q, \theta).$$

- In EM algorithm, we maximize the lower bound (ELBO) over θ and q:

$$\hat{\theta}_{\text{EM}} \approx \arg\max_{\theta} \left[\max_{q} \mathcal{L}(q, \theta)\right]$$

- In EM algorithm, q ranges over all distributions on z.

EM: Coordinate Ascent on Lower Bound

- Choose sequence of q's and θ's by "**coordinate ascent**" on $\mathcal{L}(q, \theta)$.

EM: Coordinate Ascent on Lower Bound

- Choose sequence of q's and θ's by "**coordinate ascent**" on $\mathcal{L}(q, \theta)$.
- EM Algorithm (high level):
  1. Choose initial θ^{old}.
  2. Let $q^* = \arg\max_q \mathcal{L}(q, \theta^{\text{old}})$

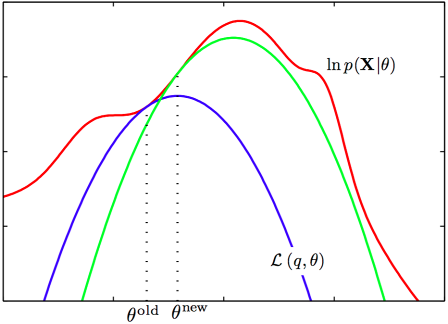
EM: Coordinate Ascent on Lower Bound

- Choose sequence of q's and θ's by "**coordinate ascent**" on $\mathcal{L}(q, \theta)$.
- EM Algorithm (high level):
  1. Choose initial θ^{old}.
  2. Let $q^* = \arg\max_q \mathcal{L}(q, \theta^{\text{old}})$
  3. Let $\theta^{\text{new}} = \arg\max_\theta \mathcal{L}(q^*, \theta)$.

EM: Coordinate Ascent on Lower Bound

- Choose sequence of q's and θ's by "**coordinate ascent**" on $\mathcal{L}(q, \theta)$.
- EM Algorithm (high level):
  1. Choose initial θ^{old}.
  2. Let $q^* = \arg\max_q \mathcal{L}(q, \theta^{\text{old}})$
  3. Let $\theta^{\text{new}} = \arg\max_\theta \mathcal{L}(q^*, \theta)$.
  4. Go to step 2, until converged.
- Will show: $p(x \mid \theta^{\mathbf{new}}) \geqslant p(x \mid \theta^{\mathbf{old}})$
- **Get sequence of θ's with monotonically increasing likelihood.**

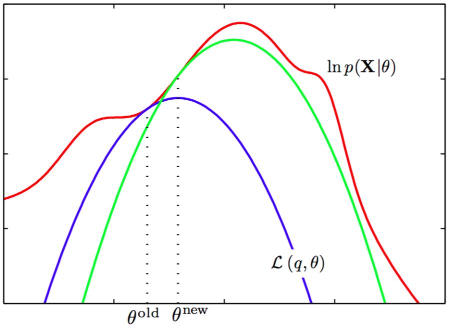
EM: Coordinate Ascent on Lower Bound



1. Start at θ^{old}.

From Bishop's *Pattern recognition and machine learning*, Figure 9.14.

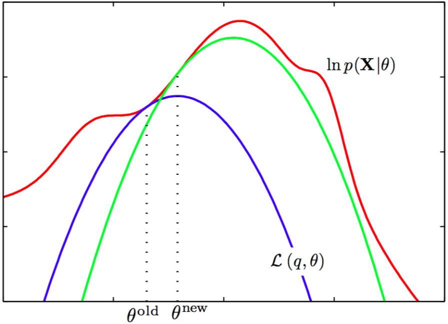
EM: Coordinate Ascent on Lower Bound



1. Start at θ^{old}.
2. Find q giving best lower bound at $\theta^{\text{old}} \Longrightarrow \mathcal{L}(q,\theta)$.

From Bishop's *Pattern recognition and machine learning*, Figure 9.14.

EM: Coordinate Ascent on Lower Bound



1. Start at θ^{old}.
2. Find q giving best lower bound at $\theta^{\text{old}} \Longrightarrow \mathcal{L}(q,\theta)$.
3. $\theta^{\text{new}} = \arg\max_{\theta} \mathcal{L}(q,\theta)$.

From Bishop's *Pattern recognition and machine learning*, Figure 9.14.

Is ELBO a "good" lowerbound?

$$\begin{aligned}
\mathcal{L}(q,\theta) &= \sum_{z\in\mathcal{Z}} q(z)\log\frac{p(x,z\mid\theta)}{q(z)} \\
&= \sum_{z\in\mathcal{Z}} q(z)\log\frac{p(z\mid x,\theta)p(x\mid\theta)}{q(z)} \\
&= -\sum_{z\in\mathcal{Z}} q(z)\log\frac{q(z)}{p(z\mid x,\theta)} + \sum_{z\in\mathcal{Z}} q(z)\log p(x\mid\theta) \\
&= -\mathrm{KL}\left(q(z)\|p(z\mid x,\theta)\right) + \underbrace{\log p(x\mid\theta)}_{\text{evidence}}
\end{aligned}$$

- **KL divergence**: measures "distance" between two distributions (not symmetric!)
- $\mathrm{KL}\left(q\|p\right) \geqslant 0$ with equality iff $q(z) = p(z\mid x)$.
- ELBO = evidence - KL $\leqslant$ evidence

Maximizing over q for fixed θ.

- Find q maximizing

$$\mathcal{L}(q,\theta) = -\mathrm{KL}[q(z), p(z \mid x,\theta)] + \log p(x \mid \theta)$$

Maximizing over q for fixed θ.

- Find q maximizing

$$\mathcal{L}(q,\theta) = -\mathrm{KL}[q(z), p(z \mid x,\theta)] + \underbrace{\log p(x \mid \theta)}_{\text{no } q \text{ here}}$$

Maximizing over q for fixed θ.

- Find q maximizing

$$\mathcal{L}(q,\theta) = -\mathrm{KL}[q(z), p(z \mid x, \theta)] + \underbrace{\log p(x \mid \theta)}_{\text{no } q \text{ here}}$$

- Recall $\mathrm{KL}(p \| q) \geqslant 0$, and $\mathrm{KL}(p \| p) = 0$.

Maximizing over q for fixed θ.

- Find q maximizing

$$\mathcal{L}(q,\theta) \quad = \quad -\mathrm{KL}[q(z), p(z \mid x,\theta)] + \underbrace{\log p(x \mid \theta)}_{\text{no } q \text{ here}}$$

- Recall $\mathrm{KL}(p\|q) \geqslant 0$, and $\mathrm{KL}(p\|p) = 0$.
- Best q is $q^*(z) = p(z \mid x,\theta)$ and

Maximizing over q for fixed θ.

- Find q maximizing

$$\mathcal{L}(q,\theta) = -\mathrm{KL}[q(z), p(z \mid x,\theta)] + \underbrace{\log p(x \mid \theta)}_{\text{no } q \text{ here}}$$

- Recall $\mathrm{KL}(p \| q) \geqslant 0$, and $\mathrm{KL}(p \| p) = 0$.
- Best q is $q^*(z) = p(z \mid x,\theta)$ and

$$\mathcal{L}(q^*,\theta) = -\underbrace{\mathrm{KL}[p(z \mid x,\theta), p(z \mid x,\theta)]}_{=0} + \log p(x \mid \theta)$$

Maximizing over q for fixed θ.

- Find q maximizing

$$\mathcal{L}(q,\theta) = -\mathrm{KL}[q(z), p(z \mid x,\theta)] + \underbrace{\log p(x \mid \theta)}_{\text{no } q \text{ here}}$$

- Recall $\mathrm{KL}(p\|q) \geqslant 0$, and $\mathrm{KL}(p\|p) = 0$.
- Best q is $q^*(z) = p(z \mid x,\theta)$ and

$$\mathcal{L}(q^*,\theta) = -\underbrace{\mathrm{KL}[p(z \mid x,\theta), p(z \mid x,\theta)]}_{=0} + \log p(x \mid \theta)$$

- Summary:

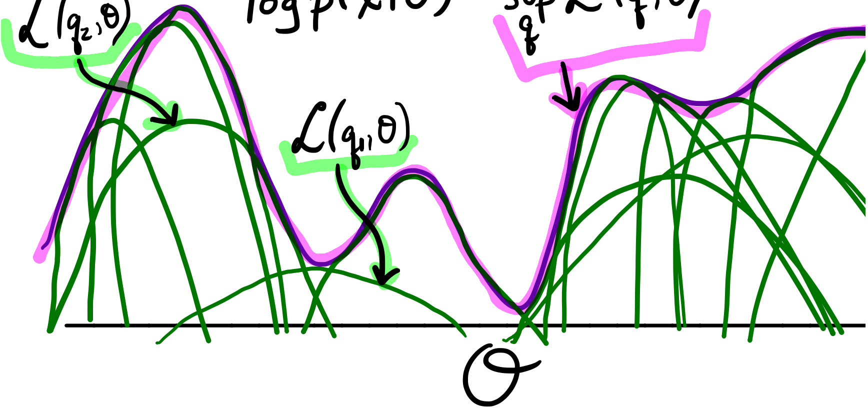
$$\log p(x \mid \theta) = \sup_q \mathcal{L}(q,\theta) \qquad \forall\theta$$

- For any θ, **sup is attained** at $q(z) = p(z \mid x,\theta)$.

Marginal Log-Likelihood **IS** the Supremum over Lower Bounds

sup is over all distributions on z

$$\log p(x|\theta) = \sup_q \mathcal{L}(q,\theta)$$



Summary

Latent variable models: clustering, latent structure, missing lables etc.

Parameter estimation: maximum marginal log-likelihood

Challenge: directly maximize the **evidence** $\log p(x;\theta)$ is hard

Solution: maximize the **evidence lower bound**:

$$\text{ELBO} = \mathcal{L}(q,\theta) = -\text{KL}\left(q(z)\|p(z \mid x;\theta)\right) + \log p(x;\theta)$$

Why does it work?

$$\begin{aligned} q^*(z) &= p(z \mid x;\theta) \quad \forall\theta \in \Theta \\ \mathcal{L}(q^*,\theta^*) &= \max_{\theta} \log p(x;\theta) \end{aligned}$$

EM algorithm

Coordinate ascent on $\mathcal{L}(q, \theta)$

1. Random initialization: $\theta^{\text{old}} \leftarrow \theta_0$
2. Repeat until convergence
   1. $q(z) \leftarrow \arg\max_q \mathcal{L}(q, \theta^{\text{old}})$

$$\textbf{Expectation} \text{ (the E-step):} \quad q^*(z) = p(z \mid x; \theta^{\text{old}})$$
$$J(\theta) = \mathcal{L}(q^*, \theta)$$

   2. $\theta^{\text{new}} \leftarrow \arg\max_\theta \mathcal{L}(q^*, \theta)$

$$\textbf{Maximization} \text{ (the M-step):} \quad \theta^{\text{new}} \leftarrow \arg\max_\theta J(\theta)$$

EM Algorithm

1. **Expectation Step**
   - Let $q^*(z) = p(z \mid x, \theta^{\text{old}})$. [$q^*$ gives best lower bound at θ^{old}]

EM Algorithm

1. **Expectation Step**
   - Let $q^*(z) = p(z \mid x, \theta^{\text{old}})$. [$q^*$ gives best lower bound at θ^{old}]
   - Let

$$J(\theta) := \mathcal{L}(q^*, \theta) = \underbrace{\sum_z q^*(z) \log \left(\frac{p(x, z \mid \theta)}{q^*(z)} \right)}_{\textbf{expectation} \text{ w.r.t. } z \sim q^*(z)}$$

EM Algorithm

1. **Expectation Step**
   - Let $q^*(z) = p(z \mid x, \theta^{\text{old}})$. [$q^*$ gives best lower bound at θ^{old}]
   - Let

$$J(\theta) := \mathcal{L}(q^*, \theta) = \underbrace{\sum_z q^*(z) \log\left(\frac{p(x, z \mid \theta)}{q^*(z)}\right)}_{\textbf{expectation} \text{ w.r.t. } z \sim q^*(z)}$$

2. **Maximization Step**

$$\theta^{\text{new}} = \underset{\theta}{\arg\max}\, J(\theta).$$

EM Algorithm

1. **Expectation Step**
   - Let $q^*(z) = p(z \mid x, \theta^{\text{old}})$. [$q^*$ gives best lower bound at θ^{old}]
   - Let

$$J(\theta) := \mathcal{L}(q^*, \theta) = \underbrace{\sum_z q^*(z) \log\left(\frac{p(x, z \mid \theta)}{q^*(z)}\right)}_{\textbf{expectation} \text{ w.r.t. } z \sim q^*(z)}$$

2. **Maximization Step**

$$\theta^{\text{new}} = \underset{\theta}{\arg\max}\, J(\theta).$$

[Equivalent to maximizing expected complete log-likelihood.]

EM Algorithm

1. **Expectation Step**
   - Let $q^*(z) = p(z \mid x, \theta^{\text{old}})$. [$q^*$ gives best lower bound at θ^{old}]
   - Let

$$J(\theta) := \mathcal{L}(q^*, \theta) = \underbrace{\sum_z q^*(z) \log\left(\frac{p(x, z \mid \theta)}{q^*(z)}\right)}_{\textbf{expectation} \text{ w.r.t. } z \sim q^*(z)}$$

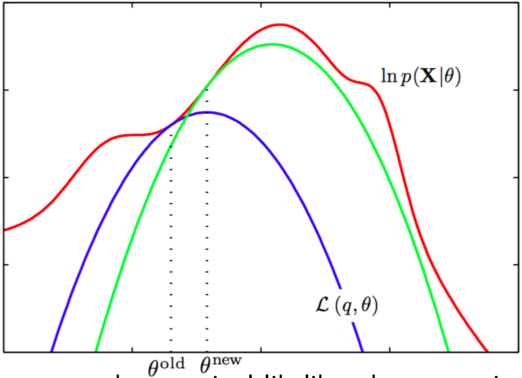
2. **Maximization Step**

$$\theta^{\text{new}} = \arg\max_\theta J(\theta).$$

   [Equivalent to maximizing expected complete log-likelihood.]

EM puts no constraint on q in the E-step and assumes the M-step is easy. In general, both steps can be hard.

Monotonically increasing likelihood



Exercise: prove that EM increases the marginal likelihood monotonically

$$\log p(x;\theta^{\text{new}}) \geqslant \log p(x;\theta^{\text{old}}) \ .$$

Does EM converge to a global maximum?

Variations on EM

EM Gives Us Two New Problems

- The "E" Step: Computing

$$J(\theta) := \mathcal{L}(q^*, \theta) = \sum_z q^*(z) \log\left(\frac{p(x, z \mid \theta)}{q^*(z)}\right)$$

EM Gives Us Two New Problems

- The "E" Step: Computing

$$J(\theta) := \mathcal{L}(q^*, \theta) = \sum_z q^*(z) \log\left(\frac{p(x, z \mid \theta)}{q^*(z)}\right)$$

- The "M" Step: Computing

$$\theta^{\text{new}} = \arg\max_\theta J(\theta).$$

EM Gives Us Two New Problems

- The "E" Step: Computing

$$J(\theta) := \mathcal{L}(q^*, \theta) = \sum_z q^*(z) \log\left(\frac{p(x, z \mid \theta)}{q^*(z)}\right)$$

- The "M" Step: Computing

$$\theta^{\text{new}} = \arg\max_\theta J(\theta).$$

EM Gives Us Two New Problems

- The "E" Step: Computing

$$J(\theta) := \mathcal{L}(q^*, \theta) = \sum_z q^*(z) \log\left(\frac{p(x, z \mid \theta)}{q^*(z)}\right)$$

- The "M" Step: Computing

$$\theta^{\text{new}} = \arg\max_{\theta} J(\theta).$$

- Either of these can be too hard to do in practice.

Generalized EM (GEM)

- Addresses the problem of a difficult "M" step.

Generalized EM (GEM)

- Addresses the problem of a difficult "M" step.
- Rather than finding

$$\theta^{\text{new}} = \arg\max_{\theta} J(\theta),$$

find **any** θ^{new} for which

$$J(\theta^{\text{new}}) > J(\theta^{\text{old}}).$$

Generalized EM (GEM)

- Addresses the problem of a difficult "M" step.
- Rather than finding
$$\theta^{\text{new}} = \arg\max_{\theta} J(\theta),$$
find **any** θ^{new} for which
$$J(\theta^{\text{new}}) > J(\theta^{\text{old}}).$$
- Can use a standard nonlinear optimization strategy
  - e.g. take a gradient step on J.

Generalized EM (GEM)

- Addresses the problem of a difficult "M" step.
- Rather than finding
$$\theta^{\text{new}} = \arg\max_{\theta} J(\theta),$$
find **any** θ^{new} for which
$$J(\theta^{\text{new}}) > J(\theta^{\text{old}}).$$
- Can use a standard nonlinear optimization strategy
  - e.g. take a gradient step on J.
- We still get monotonically increasing likelihood.

EM and More General Variational Methods

- Suppose "E" step is difficult:
  - Hard to take expectation w.r.t. $q^*(z) = p(z \mid x, \theta^{\text{old}})$.

EM and More General Variational Methods

- Suppose "E" step is difficult:
  - Hard to take expectation w.r.t. $q^*(z) = p(z \mid x, \theta^{\text{old}})$.
- Solution: Restrict to distributions $\mathcal{Q}$ that are easy to work with.

EM and More General Variational Methods

- Suppose "E" step is difficult:
  - Hard to take expectation w.r.t. $q^*(z) = p(z \mid x, \theta^{\text{old}})$.
- Solution: Restrict to distributions $\mathcal{Q}$ that are easy to work with.
- Lower bound now looser:

$$q^* = \underset{q \in \mathcal{Q}}{\arg\min} \, \mathrm{KL}[q(z), p(z \mid x, \theta^{\text{old}})]$$

Today's Summary

- Motivation: Unsupervised learning
- K-means: A simple algorithm for discovering clusters
- Making k-means probabilistic: Gaussian mixture models
- More generally: Latent variable models
- Learning of latent variable models: EM
- Underlying principle: Maximizing ELBO