CPSC 440: Advanced Machine Learning

End to End Learning

Mark Schmidt

University of British Columbia

Winter 2022

Last Time: Bayesian Logistic Regression

- We discussed Bayesian inference in L2-regulairzed logistic regression,

$$p(y^i \mid x^i, w) = \frac{1}{1 + \exp(-y^i w^T x^i)}, \quad w_j \sim \mathcal{N}\left(0, \frac{1}{\lambda}\right).$$

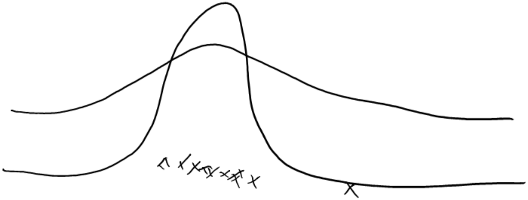
  - Prior is not conjugate so posterior does not have a nice form.
  - We could use Monte Carlo for inference, but difficult to sample from posterior.

- We discussed rejection sampling to sample complicated distributions.
  - Samples from a simple distribution q and accepts/rejects samples to look like p.
  - But requires knowing a bound on $q(x)/p(x)$ and may reject almost all samples.

- We discussed importance sampling to approximate expectations.
  - Samples from q but reweights $f(x)$ by $q(x)/p(x)$ in Monte Carlo estimate.

Importance Sampling

- Importance sampling is only efficient if q is close to p.
- Otherwise, weights will be huge for a small number of samples.
    - Even though unbiased, variance can be huge.
- Can be problematic if q has lighter "tails" than p:
    - You rarely sample the tails, so those samples get huge weights.



- As with rejection sampling, does not tend to work well in high dimensions.
    - Though there is room to cleverly design q.
        - Like "alternate between sampling two Gaussians with different variances".

Overview of Bayesian Inference Tasks

- Bayesian inference requires computing expectations with respect to posterior,

$$E[f(\theta)] = \int_{\theta} f(\theta)p(\theta \mid x)d\theta.$$

- Examples:
  - If $f(\theta) = p(\tilde{x} \mid \theta)$, we get posterior predictive.
  - If $f(\theta) = \mathbb{I}(\theta \in S)$ we get probability of S (e.g., marginals or conditionals).
  - If $f(\theta) = 1$ and we use $\tilde{p}(\theta \mid x)$, we get marginal likelihood.

- But posterior often doesn't have a closed-form expression.
  - We don't just want to flip coins and multiply Gaussians.

- Our two main tools for aproximate inference:
  1. Monte Carlo methods.
  2. Variational methods.
- Classic ideas from statistical physics, that revolutionized Bayesian stats.

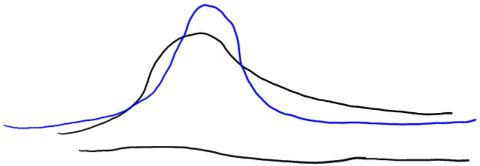
Approximate Inference

Two main strategies for approximate inference:

1. Monte Carlo methods:
   - Approximate p with empirical distribution over samples,
     $$p(x) \approx \frac{1}{n}\sum_{i=1}^{n} \mathcal{I}[x^i = x].$$
   - Turns inference into sampling.
2. Variational methods:
   - Approximate p with "closest" distribution q from a tractable family,
     $$p(x) \approx q(x).$$
     - E.g., Gaussian, product of Bernoulli, or other models with easy inference.
   - Turns inference into optimization.

Variational Inference Illustration

- Approximate non-Gaussian p by a Gaussian q:



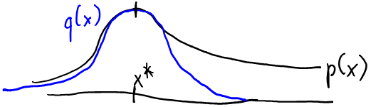
- Variational methods try to find simple distribution q that is closest to target p.
  - Unlike Monte Carlo, does not converge to solution.
    - A Gaussian may not be able to perfectly model posterior.
  - But variational methods quickly give approximation solution.
    - Which sometimes is all we need.

Laplace Approximation

- A classic variational method is the Laplace approximation.
  1. Find an x that maximizes $p(x)$,
  $$x^* \in \underset{x}{\operatorname{argmin}}\{-\log p(x)\}.$$
  2. Computer second-order Taylor expansion of $f(x) = -\log p(x)$ at x^*.
  $$-\log p(x) \approx f(x^*) + \underbrace{\nabla f(x^*)}_{0}{}^T(x - x^*) + \frac{1}{2}(x - x^*)^T \nabla^2 f(x^*)(x - x^*).$$
  3. Find Gaussian distribution q where $-\log q(x)$ has same Taylor expansion at x^*.
  $$-\log q(x) = f(x^*) + \frac{1}{2}(x - x^*)\nabla^2 f(x^*)(x - x^*),$$
  so q follows a $\mathcal{N}(x^*, \nabla^2 f(x^*)^{-1})$ distribution.
     - This is the same approximation used by Newton's method in optimization.

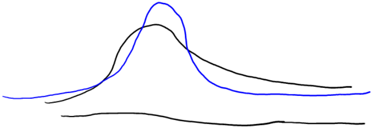
Laplace Approximation

- So Laplace approximation replaces complicated p with Gaussian q.
  - Centered at mode and agreeing with 1st/2nd-derivatives of log-likelihood:



- Now you only need to compute Gaussian integrals (linear algebra for many f).
  - Very fast: just solve an optimization (compared to super-slow Monte Carlo).
  - Bad approximation if posterior is heavy-tailed, multi-modal, skewed, and so on.

- It might not even give you the "best" Gaussian approximation:

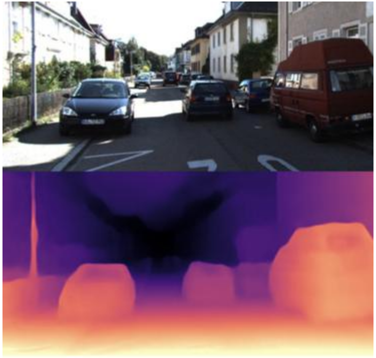


- We will discuss more-fancy variational methods later.

Outline

Motivating Problem: Depth Estimation from Images

- We want to predict "distance to car" for each pixel in an image.



https://paperswithcode.com/task/3d-depth-estimation

- We might consider using fully-convolutional networks.
  - But we now have multiple continuous labels.

Neural Network with Continuos Outputs

- Standard neural network with multiple continuous outputs (3 hidden layers):

$$\hat{y}^i = Vh(W^3h(W^2h(W^1x^i))), \quad \text{so} \quad \hat{y}_c^i = v_c^T h(W^3h(W^2h(W^1x^i))).$$

- Standard training objective is to minimize squared error,

$$f(W^1, W^2, W^3, V) = \frac{1}{2}\sum_{j=1}^{n}\sum_{c=1}^{k}(y_c^i - \hat{y}_c^i)^2.$$

- This corresponds to MLE in a network that outputs the mean of a Gaussian,

$$y^i \sim \mathcal{N}(\hat{y}^i, I).$$

- As usual, we only need to change the last layer to change output type.

Neural Networks with Covariances

- The neural network could also parameterize the variance,

$$y^i \sim \mathcal{N}(\hat{y}^i, S(W^3 h(W^2 h(W^1 x^i)))),$$

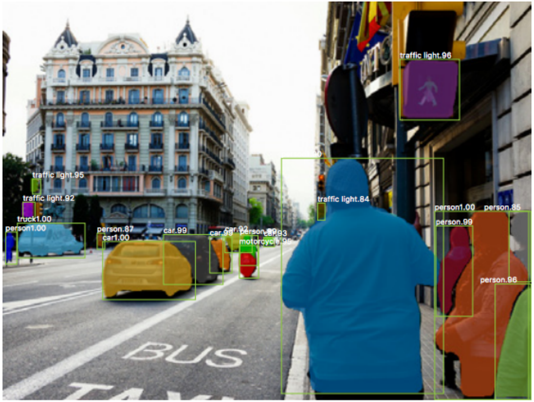
where the function S transforms the hidden layer into a positive-definite matrix.
  - So inferences over multiple variables will capture the label's pairwise correlations.
    - For depth estimation, neighbouring pixels are likely to have similar depths.

- Common choices for S:
  - S parameterizes a diagonal matrix D (may output $\log(\sigma_c)$ values to make positive).
  - S parameterizes a square root matrix A, such that $\Sigma = AA^T$.

- We could also consider Bayesian neural networks.
  - Where you might use a Laplace approximation of the posterior.
    - Though the matrix $\nabla^2 f(W^3, W^2, W^1, V)$ may be too large and will be singular.

Object Localization

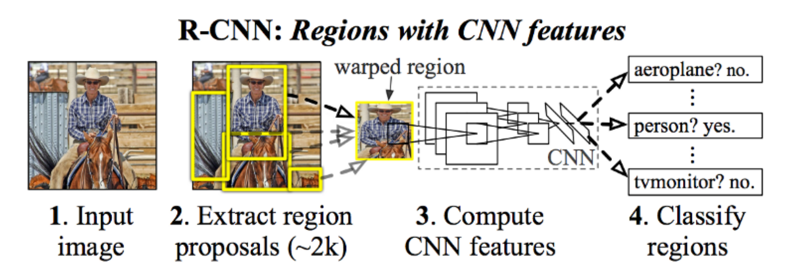
- Object localization is task of finding locations of objects:
  - Input is an image.
  - Output is a bounding box for each object (among predefined classes).



https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

Region Convolutional Neural Networks: "Pipeline" Approach

- Early approach (region CNN) resemble classic computer vision "pipelines":
  1. Propose a bunch of potential boxes (based on segmenting image in various ways).
  2. Compute features of each box using a CNN (after re-shaping box to standard size).
  3. Classify boxes using SVMs (max pool among regions with high overlap).
  4. Refine each box using linear regression on CNN features.
     - 4 continuous outputs: center x-coordinate, center y-coordinate, log-width, log-height.

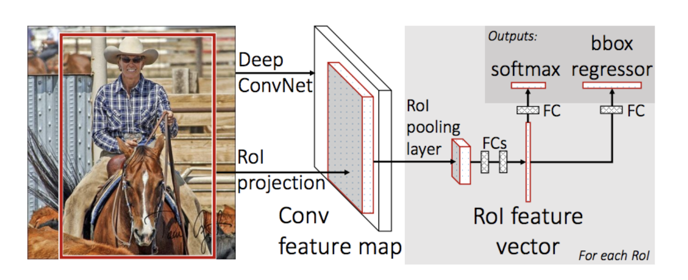


https://arxiv.org/pdf/1311.2524.pdf

- Improved on state of the art, but slow and there are 4 parts to train.

Fast R-CNNs

- R-CNN was quickly replaced by fast R-CNN:
  - Propose a bunch of potential bounding boxes (same as before).
  - Apply CNN to whole image, then get features of bounding boxes.
    - Faster than applying CNN to 2000 candidate regions.
  - Make softmax (over $k+1$ classes) and bounding box regression part of network.
    - More accurate since are parts are trained together.

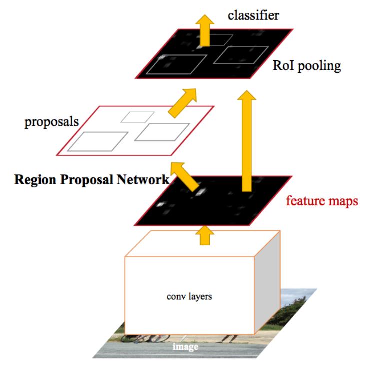


https://arxiv.org/pdf/1504.08083.pdf

- Most parts trained together, but bounding box proposals do not use encoding.

Faster R-CNNs

- Faster R-CNNs made generating bounding boxes part of the network.
  - Uses region-proposal network as part of network to predict potential bounding boxes.
  - Many implementation details required to get it working.



https://arxiv.org/pdf/1506.01497.pdf

- With all steps being part of one network, this called an end-to-end model.

YOLO: You Only Look Once

- A more-recent variant that further speeds things up is YOLO:

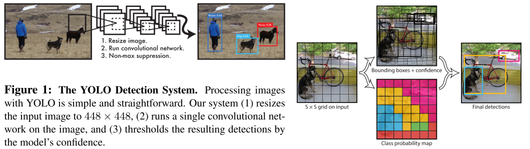


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

https://arxiv.org/pdf/1506.02640.pdf

  - Divides image into grid.
  - Directly predict properties for a fixed number of bounding boxes for grid box:
    - Probability that box is an object (for pruning set of possible boxes).
    - Box x-coordinate, y-coordinate, width, height.
    - Class of box (no separate phase of "proposing boxes" and "classifying boxes").
  - Max pooling ("non-max suprresion").
- Reasonably-accurate real-time object detection (with fancy-enough hardware).

Instance Segmentation and Pose Estimation

- Can add extra predictions to these networks.
- For example, mask R-CNNs add instance segmentation and/or pose estimation:

https://arxiv.org/pdf/1703.06870.pdf

- Instance segmentation applies binary mask to bounding boxes (pixel labels).
- Pose estimation predicts continuous joint keypoint locations.

End-to-End Computer Vision Models

- Key ideas behind end-to-end systems:
  1. Write each step as a differentiable operator.
  2. Train all steps using backpropagation and stochastic gradient.

- Has been called differentiable programming.

- There now exist end-to-end models for all the standard vision tasks.
  - Depth estimation, pose estimation, optical flow, tracking, 3D geometry, and so on.
  - A bit hard to track the progress at the moment.
  - A survey of ≈ 200 papers from 2016:
    - http://www.themtank.org/a-year-in-computer-vision

- Pose estimation video: https://www.youtube.com/watch?v=pW6nZXeWlGM
- Making 60-fps high-resolution colour version of videos from 120 year ago:
  - https://www.youtube.com/watch?v=YZuP41ALx_Q

End of Part 3 ("Gaussian Variables"): Key Concepts

- We discussed continuous density estimation with multivariate Gaussians.
  - Parameterized by mean vector and positive definite covariance matrix.
  - Assumes distribution is uni-modal, no outliers, untruncated.
    - And symmetric around principle axes.
  - "Gaussianity" is preserved under many operations.
    - Addition, marginalization, conditionining, product of densities.

- We discussed conditional independence in Gaussians.
  - Models correlations between variables where $\Sigma_{ij} \neq 0$.
    - Diagonal covariance corresponds to assuming variables all variables are independent.
  - We define a graph based on the Θ_{ij} values.
    - If variables are blocked in graph, implies conditional independence.

End of Part 3 ("Gaussian Variables"): Key Concepts

- We discussed several methods for sampling and/or Monte Carlo:
  - Inverse transform method uses inverse of CDF to sample continuos densities.
  - Rejection sampling rejects samples from a simpler distribution.
  - Importance sampling reweights samples from a simpler distribution.

- We discussed learning in Gaussians.
  - Closed-form MLE given by data's mean and variance.
  - Conjugate prior for mean in Gaussian.
  - Adding a scaled identity matrix to MLE gives positive-definite estimate.
  - Graphical Lasso allows learning sparse conditional independence graph.

- Gaussian discriminant analysis is generative classifer with Gaussian classes.
  - Does not need naive Bayes assumption.

End of Part 3 ("Gaussian Variables"): Key Concepts

- We discussed regression.
  - Supervised learning with continuous outputs.
  - Least squares with L2-regularization assumes Gaussian likelihood and prior.

- We discussed Bayesian linear regression.
  - Gives confidence in predictions.
  - Empirical Bayes can be used to set many hyper-parameters.
    - Automatic relevance determination: prefers simpler models that fit data well.
  - Laplace approximation can be used in non-conjugate settings.
    - Special case of a variational inference method (approximate with simpler distribution).

- We discussed end-to-end learning.
  - Try to write each step as a differentiable operation.
  - Train entire network with backprop and SGD.
    - We illustrated this with evolution of object localization in vision.

Summary

- Variational methods approximate p with a simpler distribution q.
  - You then do inference with q as an approximation to using p.
- Laplace approximation simple variationl inference method.
  - Use Gaussian centered at MAP that agrees with first two derivatives of NLL.
- Neural networks with continous output:
  - Typically trained using squared error, corresponding to Gaussian likelihood.
- End to end models: use a neural network to do all steps.
  - Write each step in a vision "pipeline" as a differentiable operator.
  - Train entire network using SGD.

- Next time: the exponential family.