CPSC 440: Machine Learning

Multi-Class Classification

Winter 2022

Last Time: Hierarchical Bayes

- We discussed a hierarchical Bayesian model for non-IID data:

$$x^i \mid z^i \sim \text{Ber}(\theta_{z^i}) \qquad \theta_j \sim \text{Beta}(\alpha, \beta) \qquad \alpha, \beta \sim D(p, q, m)$$

- Example are IID within their group 'z'.
- But we have a shared prior across the groups.
  - Allows predictions for new groups or groups with few examples.
- And we are often interested in learning the hyper-parameters.
  - Allows learning from data across groups.
- We talked about empirical Bayes:
  - Optimize marginal likelihood to set hyper-parameters.
  - Often need a hyper-prior to avoid weird results.
- Or we can take Bayesian approach:
  - Integrate over parameters and hyper-parameters (usually with Monte Carlo).

METHODS article

Front. Environ. Sci., 09 March 2021 | https://doi.org/10.3389/fenvs.2021.491636

Evaluating the Benefits of Bayesian Hierarchical Methods for Analyzing Heterogeneous Environmental Datasets: A Case Study of Marine Organic Carbon Fluxes

observations from 407 sampling locations spanning eight biomes across the global ocean. We fit a global scale Bayesian hierarchical model that describes the vertical profile of organic carbon flux with depth. Profile parameters within a particular biome are assumed to share a common deviation from the global mean profile. Individual station-level parameters are then modeled as deviations from the common biome-level profile. The hierarchical approach is shown to have several benefits over simpler and more common data aggregation methods. First, the hierarchical approach avoids statistical complexities introduced due to unbalanced sampling and allows for flexible incorporation of spatial heterogeneitites in model parameters. Second, the hierarchical approach uses the whole dataset simultaneously to fit the model parameters which shares information across datasets and reduces the uncertainty up to 95% in individual profiles. Third, the Bayesian approach incorporates prior scientific information about model parameters; for example, the non-negativity of chemical concentrations or mass-balance, which we apply here. We explicitly quantify each of these properties in turn. We emphasize the generality of the hierarchical Bayesian approach for diverse environmental applications and its increasing feasibility for large datasets due to recent developments in Markov Chain Monte Carlo algorithms and easy-to-use high-level software implementations.

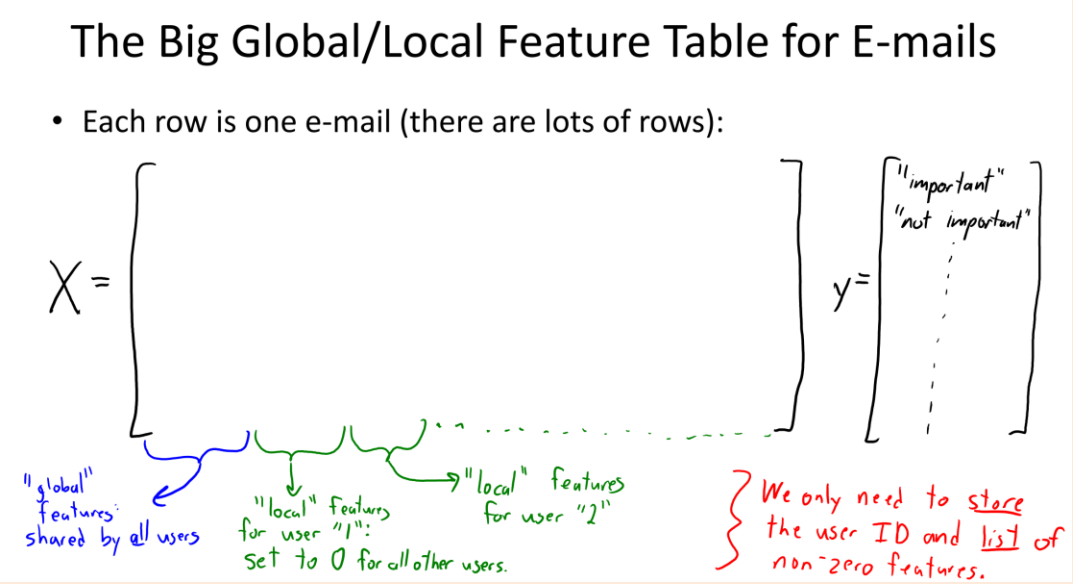
We will cover MCMC later

Parameter(s) for each group
→ Groups with many samples inform other groups
→ All data is used
→ Prior and hyper-prior can have domain knowledge
→ Monte Carlo used to approximate ugly integrals

https://www.frontiersin.org/articles/10.3389/fenvs.2021.491636/full

Discussion of Hierarchical Bayes

- "We finally have an elegant mathematical way to do…"
  - Frequently used as a justification for hierarchical Bayesian methods.
  - We will see some influential and/or neat examples later in the course.
- But often you can find a simple less-elegant solution:
  - 340 slide giving features addressing similar issues to hospital example.
    - Just features and gradients, no hyper-priors or integrals.



Next Topic: Multi-Class Classification

Multi-Class Classification

- Consider classification with categorical features and labels:

X =

Cough	Fever	Shortness
1	Low	0
1	High	1
0	High	0
0	Low	0
1	Medium	0

y =

Diagnosis
Cold
Pneumonia
Covid
Covid
Cold

- Recall our previous binary classification methods:
  - Naïve Bayes.
  - Tabular probabilities.
  - Logistic regression.
  - Neural networks.

Product of Categoricals and Multi-Class Naïve Bayes

- We could consider multivariate categorical density estimation:
  - Input: 'n' IID samples of categorical vectors $x^1, x^2, x^3, \ldots, x^n$ from population.
  - Output: model giving probability for any assignment of values $x_1, x_2, \ldots, x_d$.

X =

SNP 1	SNP 2	SNP 3	SNP 4	SNP 5	SNP 6	SNP 7	SNP 8	SNP 9
A	C	C	T	T	T	A	G	C
A	C	C	G	T	T	A	G	G
A	C	C	T	T	T	A	G	C
A	A	C	T	T	T	C	G	G

$p(x_1 = A, x_2 = C, x_3 = C, x_4 = T, x_5 = T, x_6 = T, x_7 = A, x_8 = G, x_9 = C) = 0.11$

(Estimate probability for all 49 possible values)

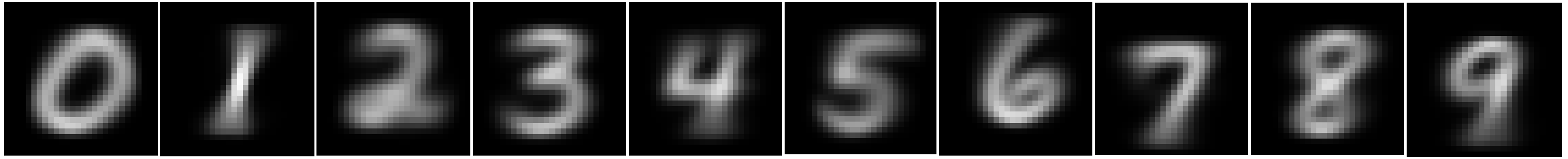
- Similar to product of Bernoullis, we could use product of categoricals:
  - Assums x_j are mutually independent (strong assumption, easy computation).

$$p(x_1 = c_1, x_2 = c_2, \ldots, x_d = c_d) = p(x_1 = c_1)p(x_2 = c_2) \cdots p(x_d = c_d) = \theta_{1c_1}\theta_{2c_2} \cdots \theta_{dc_d}$$

  - We have a parameter θ_{jc} representing probability that features 'j' is in category 'c'.
- If we use product of categoricals within a generative classifier for modeling p(x,y).
  - We get a version of naïve Bayes for categorical data.
  - If we used posterior predictive for predictions we would get a "Bayesian naïve Bayes" method.
    - It is called naïve "Bayes" because it uses Bayes rule, not because it uses Bayesian inference.

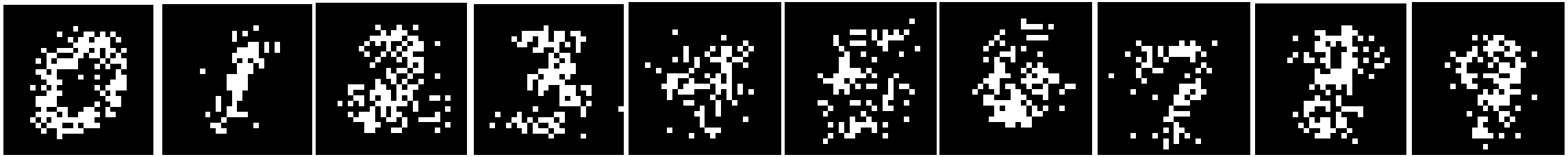
Multi-Class Naïve Bayes on MNIST Digits

- Consider fitting multi-class naïve Bayes to binarized MNIST digits.
  - Visualizing the conditional Bernoulli parameter for each class:



  - The class probabilities are all ~1/10.

  - Generating a sample from each class:



Tabular Probabilities for Categorical Data

- Tabular parameterization (2-features and 3-categories per feature/label):
  - $p(y = 1 \mid x_1 = 1, x_2 = 1) = \theta_{111}$.
  - $p(y = 2 \mid x_1 = 1, x_2 = 1) = \theta_{211}$.
  - $p(y = 3 \mid x_1 = 1, x_2 = 1) = \theta_{311}$.
  - $p(y = 1 \mid x_1 = 1, x_2 = 0) = \theta_{110}$.
  - $p(y = 2 \mid x_1 = 1, x_2 = 0) = \theta_{210}$.
  - $p(y = 3 \mid x_1 = 1, x_2 = 0) = \theta_{310}$.
  - …. (enumerate all combinations of labels and features).
    - Could reduce parameters by define $\theta_{311} = 1 - \theta_{111} - \theta_{211}$ because of "sum to 1" over 'y' values.

- MLE has simple closed-form solution: $\hat{\theta}_{111} = n_{111}/n_{11}$.
  - (number of times y=1 when x_1=1 and x_2=1)/(number of times x_1=1 and x_2=1).
  - Can add a Dirichlet prior and do MAP.
  - Could integrate over Θ to do Bayesian inference.

Decision Theory

- We may also have a cost for different predictions:

Predict\True	Cold	Pneumonia	Covid
Cold	0	15	2
Pneumonia	10	0	5
Covid	5	15	0

- In the above example:
  - Cost for correct prediction is zero.
  - Cost for missing pneumonia is high.
  - Cost for falsely declaring pneumonia or covid is relatively high.
    - Need to take antibiotics or isolate.

Decision Theory

- We may also have a cost for different predictions:

Predict\True	Cold	Pneumonia	Covid
Cold	0	15	2
Pneumonia	10	0	5
Covid	5	15	0

- Instead of most probable label, take $\hat{y}$ minimizing expected cost:
  - $\mathbb{E}[C(\hat{y}, \tilde{y})] = \sum_c p(\tilde{y} = c \,|\tilde{x}, \widehat{\Theta}) C(\hat{y}, c)$.
    - Probability that true label is ‘c’, times cost of predicting $\hat{y}$ when true label is ‘c’.
      - Marginalized over possible values of ‘c’.
    - In the above example, if all probabilities are equal then you predict pneumonia.
      - Mis-diagnosing as pneumonia has the smallest “cost” in this example.

Bayesian Decision Theory

- Unfortunately, we get sub-optimal decisions using MLE/MAP $\hat{\Theta}$.
  - Relying on decoding ("point estimate") can miss important information.

- Bayesian decision theory gives optimal actions:
  - Minimize expected cost using posterior predictive estimate for class 'c'.
    - $\mathbb{E}[C(\hat{y}, \tilde{y})] = \sum_c p(\tilde{y} = c \mid \tilde{x}, X, y, \mathrm{A}) C(\hat{y}, c)$.

The posterior predictive for discriminative models (condition on training data $\{X, y\}$ and test features $\tilde{x}$)

$$\int p(\tilde{y} = c \mid \tilde{x}, \theta) p(\theta \mid \alpha) d\theta$$

This is posterior predictive distribution we use for supervised learning.

Linear Parameterization of Conditionals

- Tabular parameterization will overfit when you have many features.
  - If each features has 'k' categories and 'y' has 'k' categories:
    - Total number of parameters is k^{d+1}.
  - Fully-expressive, but really only useful with a small number of features.

- Similar to logistic regression:
  - We can use parameterizations based on linear combinations of features.
    - Have a weight w_c for each class 'c', giving $z_c = w_c^T x$ for each class 'c'.
      - Allows us to have continuous features.
    - To turn these into a probability, we typically use functions of the form:

$$p(y = c \mid z_1, z_2, \dots, z_k) = f_c(z_1, z_2, \dots, z_k)$$

Convert from these 'k' real numbers into a categorical probability

But first…

- How do we use categorical features in regression models?
- Standard approach is to convert to a set of binary features:
  - "1 of k" ("one hot") encoding.

Age	City	Income
23	Van	22,000.00
23	Bur	21,000.00
22	Van	0.00
25	Sur	57,000.00
19	Bur	13,500.00
22	Van	20,000.00

→

Age	Van	Bur	Sur	Income
23	1	0	0	22,000.00
23	0	1	0	21,000.00
22	1	0	0	0.00
25	0	0	1	57,000.00
19	0	1	0	13,500.00
22	1	0	0	20,000.00

  - What if you get a new category in the test data?
    - Common approach: set all three variables to 0.

Softmax Function and Unnormalized Probabilities

- We want to map from the 'k' real values of z_c to probabilities.
  - To do this, we typically use the softmax function:

$$f_c(z_1, z_2, \dots, z_k) = \frac{\exp(z_c)}{\sum_{c'=1}^{k} \exp(z_{c'})} \propto \underbrace{\exp(z_c)}_{\tilde{\theta}_c}$$

- This is similar to when we used unnormalized probabilities:
  - We converted unnormalized (but positive) values $\tilde{\theta}_c$ to probabilities.

$$p(y = c \mid \tilde{\theta}_1, \tilde{\theta}_2, \dots, \tilde{\theta}_k) = \frac{\tilde{\theta}_c}{\sum_{c'=1}^{k} \tilde{\theta}_{c'}} \propto \tilde{\theta}_c$$

  - Softmax is similar but uses exponentiation since the z_c may be negative.
    - You could use other operations, but exponential function has many nice properties.

Softmax Function and Binary Logistic Regression

- We want to map from the 'k' real values of z_c to probabilities.
  - To do this, we typically use the softmax function:

$$f_c(z_1, z_2, \ldots, z_k) = \frac{\exp(z_c)}{\sum_{c'=1}^{K} \exp(z_{c'})} \propto \underbrace{\exp(z_c)}_{\tilde{\theta}_c}$$

- We obtain the sigmoid function as a special case:
  - With two classes and $z_2=0$: $f_1(z_1, 0) = \frac{\exp(z_1)}{\exp(z_1)+\exp(0)} = \underbrace{\frac{1}{1+\exp(-z_1)}}_{\text{sigmoid}}$
  - So linearly-parameterized softmax generalizes logistic regression.

Inference in Multi-Class Logistic Regression

- Using $z_c = w_c^T x$ in the softmax function as probabilities gives:

$$p(y=c \mid x, w_1, w_2, \ldots, w_k) = \frac{\exp(w_c^T x)}{\sum_{c'=1}^{k} \exp(w_{c'}^T x)} \propto \exp(w_c^T x)$$

  - This is the likelihood for multi-class logistic regression.

- To do inference in the model, first compute the $z_c = w_c^T x$ values:

$$Wx = \begin{bmatrix} — & w_1^T & — \\ — & w_2^T & — \\ & \vdots & \\ — & w_k^T & — \end{bmatrix} x = \begin{bmatrix} w_1^T x \\ w_2^T x \\ \vdots \\ w_k^T x \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix}$$

  - Cost of this is O(dk): need to do 'k' dot products with 'd' elements.

- Plug the z_c values into softmax to get probabilities.
  - And then you can do inference as if it was a categorical distribution.

Review: Softmax Loss and its Gradient

- Take negative log-likelihood ('n' IID examples) to obtain softmax NLL:

$$f(w_1, w_2, \dots, w_k) = \sum_{i=1}^{n} \left[-w_{y^i}^T x^i + \log\left(\sum_{c=1}^{k} \exp(w_c^T x^i) \right) \right]$$

  - Softmax loss is equivalent to what people call the cross-entropy.
  - This is a convex and differentiable, so we can use gradient descent.
    - Convexity of the second term may not be obvious.
  - We often add a regularizer such as an L2-regularizer.

- The gradient has a special form:

$$\nabla_{w_c} f(w_1, w_2, \dots, w_k) = -\sum_{i=1}^{n} I[y^i = c] x^i + \sum_{i=1}^{n} \frac{\exp(w_c^T x^i)}{\sum_{c'=1}^{k} \exp(w_{c'}^T x^i)} x^i = \sum_{i=1}^{n} \left(p(y^i = c \mid x^i, w) - I[y^i = c] \right) x^i$$

"MLE (where gradient is 0) wants probabilities to match indicator functions on average"

$p(y^i = c \mid x^i, w_1, w_2, \dots, w_k)$

- If you do not like probabilities, can alternately use multi-class SVM losses ("discriminant function").

Next Topic: Neural Networks for Multi-Class

Multi-Class Linear Classification

- We are now discussing multi-class classification:

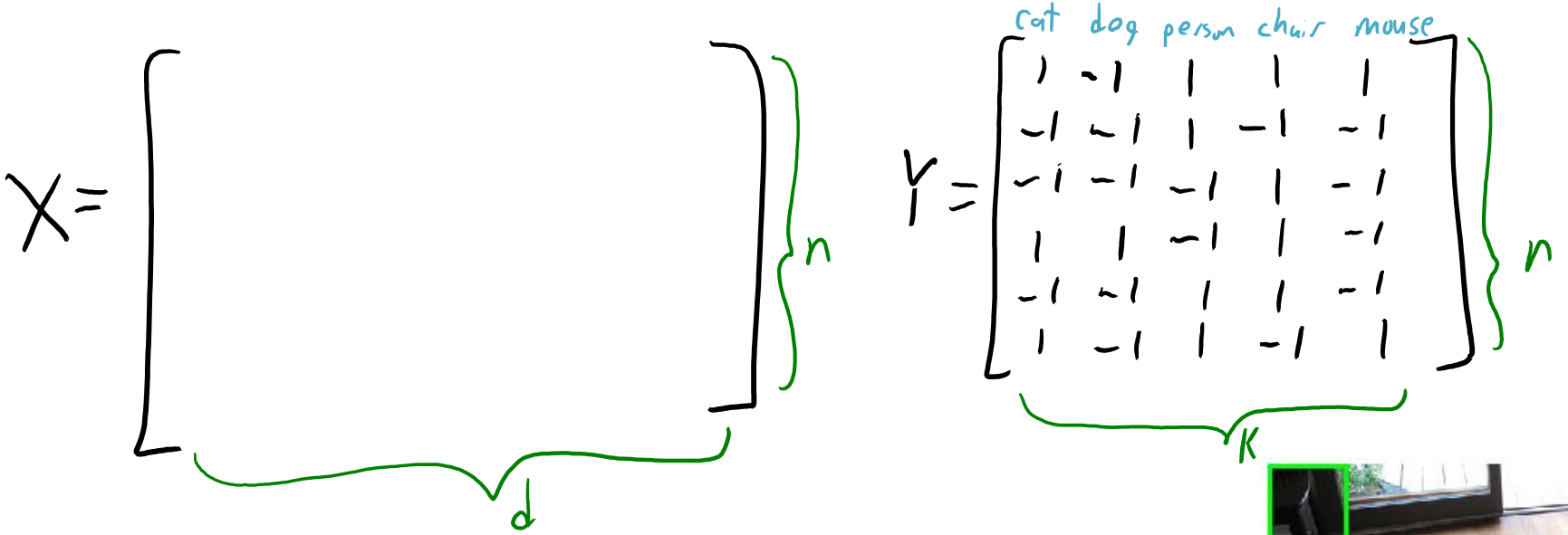
$$X = \begin{bmatrix} & \\ & \end{bmatrix} \qquad y = \begin{bmatrix} 27 \\ 16 \\ 8 \\ 7 \\ 21 \\ 5 \end{bmatrix}$$

- For example, classify image as “cat”, “dog”, or “person”.
  - There is one correct label.

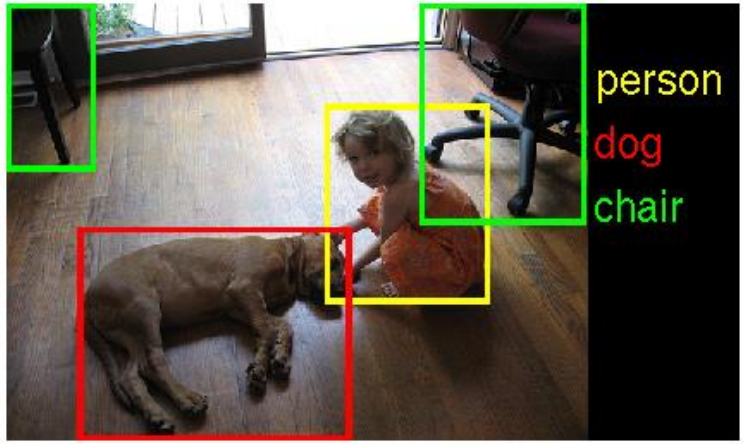
https://www.youtube.com/watch?v=tntOCGkgt98

Previously: Multi-Label Classification

- We previously saw neural networks for multi-label classification:



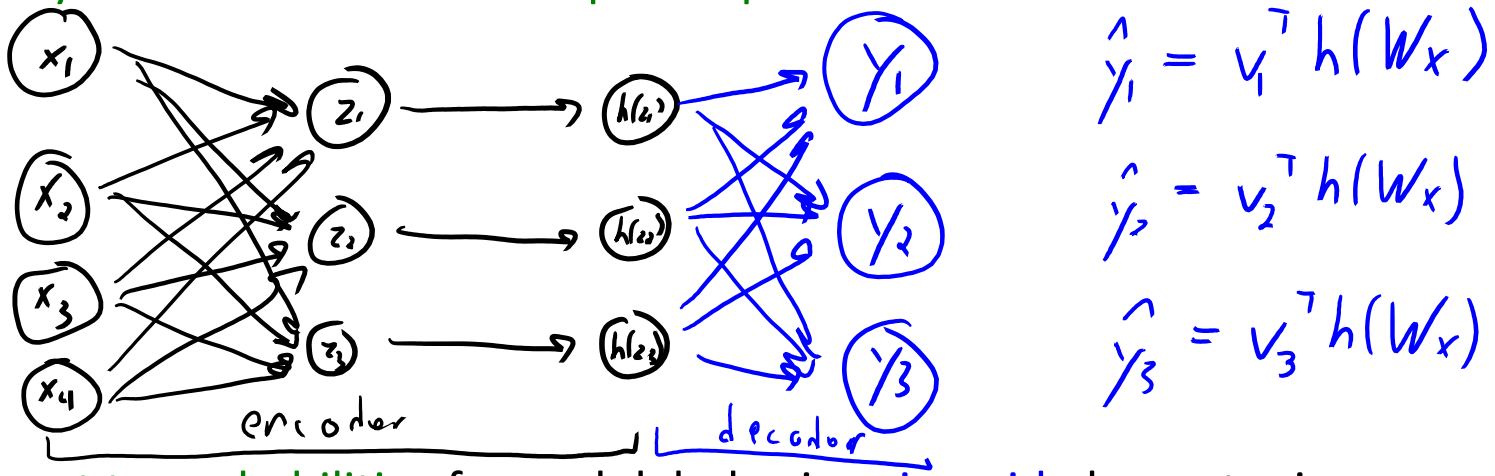
- Which of the 'k' objects are in this image?
  - There may be more than one "correct" class label.



http://image-net.org/challenges/LSVRC/2013/

Neural Network for Multi-Label Classification

- Multi-label classification a neural network:
  - Input is connected to a hidden layer.
  - Hidden layer is connected to multiple output units.



$$\hat{y}_1 = v_1^T h(Wx)$$

$$\hat{y}_2 = v_2^T h(Wx)$$

$$\hat{y}_3 = v_3^T h(Wx)$$

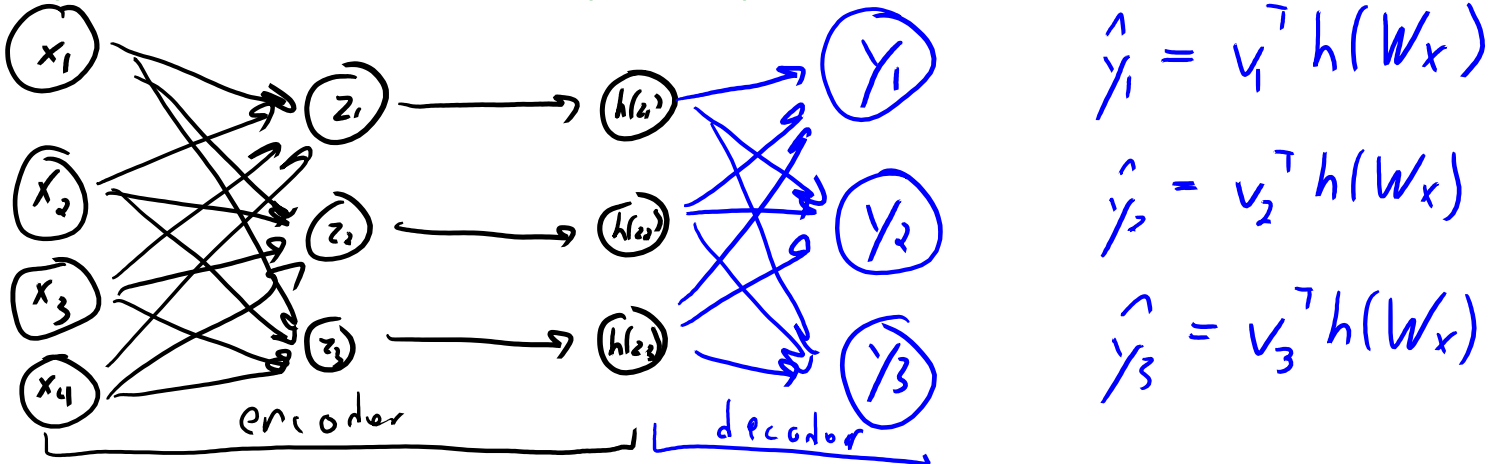
  - We convert to probabilities for each label using sigmoid element-wise:

$$p(y_c \mid x, W, V) = \frac{1}{1+\exp(-y_c \hat{y}_c)}$$

  - We predict by maximizing $p(y_c \mid x, W, V)$ for $y_c = +1$ and $y_c = -1$ for each 'c' (predict each class).
  - We train by minimizing sum of negative logs of these probabilities over 'c'.

Neural Network for Multi-Class Classification

- Multi-class classification a neural network:
  - Input is connected to a hidden layer (same as multi-label).
  - Hidden layer is connected to multiple output units (same as multi-label).



$$\hat{y}_1 = v_1^T h(Wx)$$

$$\hat{y}_2 = v_2^T h(Wx)$$

$$\hat{y}_3 = v_3^T h(Wx)$$

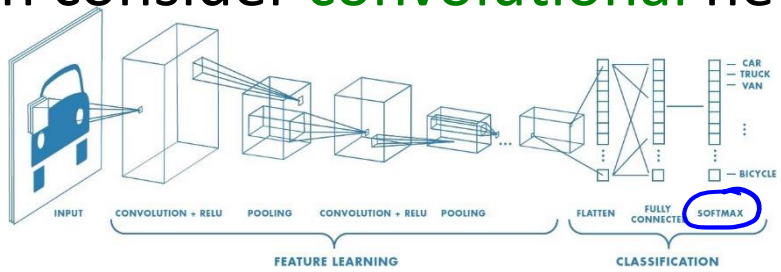
  - We convert to probabilities for each class using softmax to the $\hat{y}_c$ values.

$$p(y = c \mid x, W, V) = \frac{\exp(\hat{y}_c)}{\sum_{c'=1}^{K} \exp(\hat{y}_{c'})}$$

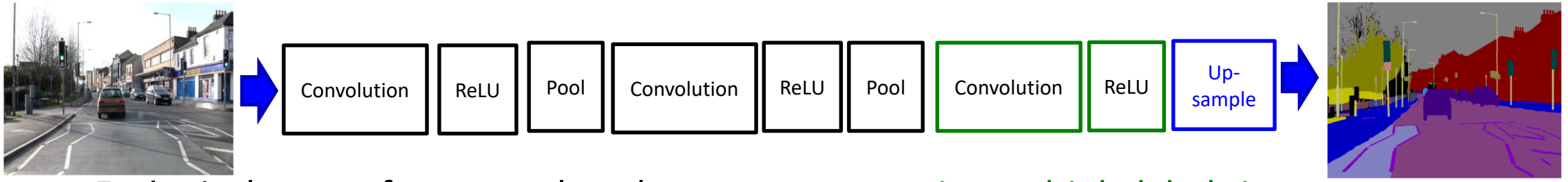
  - We predict by maximizing p(y | x, W, V) over all each 'c' (one prediction across classes).
  - We train by minimizing negative log of this probability (summing across examples).

Discussion: Multi-Class Neural Networks

- Binary versus multi-label versus multi-class neural networks:
  - Network can be the same except for the last layer (same encoding steps).
    - Training issues/challenges/tricks are the same.
  - We often only need to change last layer for new problems.
    - You can pre-train multi-class neural network on multi-label (or go in the opposite direction).
- As in Part 1, we can consider convolutional neural networks:



- For pixel labeling, we can again use fully-convolutional networks.



  - Each pixel comes from one class, but we can recognize multiple labels in same image.

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Summary

- Multi-class classification:
  - Supervised learning with categorical labels.
- Tabular probabilities for conditional categorical features/labels:
  - One parameter for each combination of label and features.
  - Fully-expressive but has an exponential number of parameters.
- Bayesian decision theory:
  - Given a cost function, optimize expected cost under posterior predictive.
- Softmax function:
  - Converts 'k' real numbers into a probability.
- Multi-class logistic regression:
  - Take linear combination for each class, use softmax to get a probability.
  - Differentiable, convex, and MLE "matches probabilities to labels".
- Multi-class neural networks:
  - Same as binary and multi-label neural networks, just use categorical likelihood is last layer.

- Next time: are CNNs learning something sensible?

Softmax NLL vs. Cross-Entropy

- Multi-class objective often written as minimizing cross-entropy:

$$f(W,V) = \sum_{i=1}^{n} \sum_{j=1}^{c} I[y^i = c](-\log p(y^i = c \mid X, W, V))$$

- The indicator function is zero except for true label y^i:

$$f(W,V) = -\sum_{i=1}^{n} \log p(y^i \mid X, W, V)$$

- When we plug in the softmax likelihood, we get the softmax NLL.
  - So cross-entropy is the softmax NLL with extra terms that do nothing.
    - Cross-entropy would make more sense if training data had "soft" assignments to classes.

Loss vs. Objective vs. Error vs. Cost

loss function, cost function, objective function

What are the differences between loss/cost/objective/error function?

lectures

~ An instructor (Mark Schmidt) thinks this is a good question ~

edit · undo good question | 1 Updated 8 minutes ago by (Anon. Mouse to classmat

the instructors' answer, *where instructors collectively construct a single answer*

Some of these terms are often used interchangeably, or may be synonyms depending what notation people use. In this course, I am trying to use the following:

Loss: the function measuring how well you fit a given data point. In this course, this is the negative of the log-likelihood (NLL), but there exist models (like SVMs) that use other measures.

Objective: the function that are optimizing in terms of a set of parameters. In this course, this will usually be the sum of the NLLs over all training examples.

Error: a measure of how you have fit the data. This might be the NLL, or for classification be the total number of mistakes you make in prediction on a given data set.

Cost: the penalty you get for making a given prediction, based on what the correct value is. The error corresponds to the special case where all errors have the same cost, but more generally you could have different costs for different types of errors.