Image and Text as Data in Social Science Research

Andrea Ciccarone

Columbia University

October 28, 2024

Outline

Why do we need ML?[1]

Let's start by thinking of text and images as data X that we can use for some task:

1. (Raw) Data representation:
   - **Text:** An observation is an array of words from a vocabulary
   - **Image:** An observation is a matrix of pixel values
2. Tasks:
   - **Supervised:** Predict Y from X
   - **Unsupervised:** Extract latent signal from X

- Text and Image are intrinsically **high dimensional**: even for familiar tasks ($Y = \beta X + \epsilon$) we can't use "standard" techniques
- Machine Learning is really a collection of tools for prediction (not coefficient estimation): observe train data $\rightarrow$ make predictions about unseen test data

[1]The slides on ML are partially based on slides by Suresh Naidu.

Outline

ML: Improving from OLS?

$$Y_i = \beta X_i + \epsilon_i$$

- β_{OLS} is BLUE (best linear unbiased estimator)
  - Smallest variance among estimators with 0 bias
  - But can we improve predictions with more bias and smaller variance?
- With many parameters, OLS is unfeasible:
  - With $k > n$, $X'X$ is not even invertible
  - But even if we can find β_{OLS}, we are going to have overfitting
  - This is where ML starts coming up (Ridge, Lasso, Random Forests, Neural Networks...)
- If X_i is a text document, it will have **at least** thousands of entries (the dimension of the vocabulary)

Improving from OLS: Ridge

- **Ridge** estimator shrinks coefficients towards 0, reducing sensitivity of single data points
- It introduces biased, but lowers variance

$$\begin{aligned}\beta_{Ridge}(\lambda) &= \text{argmin}_\beta ||y - X\beta||_2^2 + \lambda||\beta||_2^2 \\ &= (X'X + \lambda I)^{-1} X'y\end{aligned}$$

- How do we set λ? **Cross-validation:**
    1. Split train data into K folds
    2. Loop over values of λ
    3. Train model on $K-1$ folds and use the Kth fold for validation
    4. λ^* minimizes average prediction error across folds

Improving from OLS: Lasso

- **Lasso** estimator also helps with overfitting but the L_1 norm shrinks some (superflous) coefficients at 0 $\implies$ good for **feature selection**

$$\beta_{LASSO}(\lambda) = \text{argmin}_\beta ||y - X\beta||_2^2 + \lambda||\beta||_1$$

- Unlike Ridge, LASSO does not have a closed form solution (L1-norm component non differentiable) $\implies$ use numerical solutions via eg. LARS
- λ set, again, with cross-validation

Capturing Complex Relationships: Neural Networks

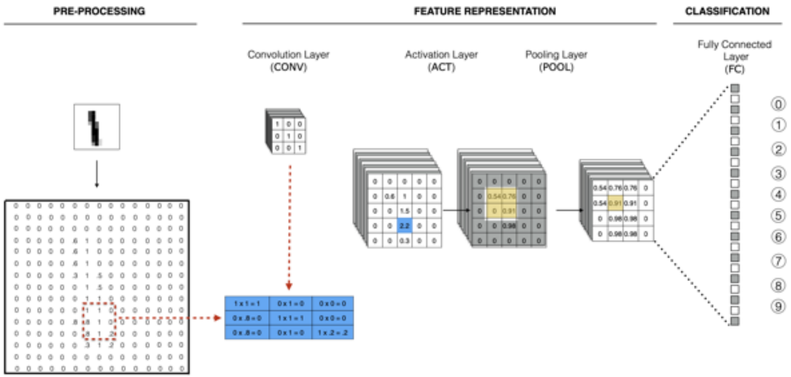
So far we have used linear models... but when working with text and particularly images we need more complex, non-linear, approaches!

- A **Neural Network** is a model:

$$g(X, \theta) = f_k(f_{k-1}(\dots f_1(X, \theta_1), \theta_2, \dots), \theta_{k-1}, \theta_k)$$

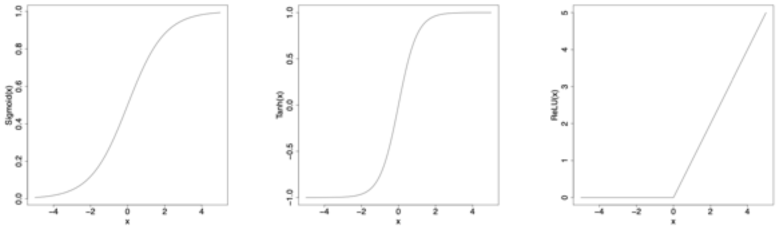
  - Input X is successively transformed through a series of functions (layers) $f_1, f_2, \dots f_k$
  - Each θ_i is a vector or matrix of weights for layer i (this is the **learning** part)
  - A *typical* f_i may look like $f_i(X, \theta_i) = \sigma(\theta_i X + b_i)$, where $\sigma(x)$ is often $\tanh(x), \text{logit}(x)$ or $\max(0, x)$
  - Output layer can be a simple linear transformation $f_k(X, \theta_k) = \theta_k X + b_k$ (for continuous outcomes) or a softmax (for discrete classification)
  - Our goal is to find the parameters θ that minimize some loss function $\mathcal{L}$

Picture of a (Convolutional) Neural Network



Source: Torres and Cantù (2022)

Examples of Activation Functions σ



(a) $Sigmoid(x) = \frac{1}{1+e^{-x}}$ (b) $Tanh(x) = \frac{2}{1+e^{-2x}}$ (c)$ReLU(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{otherwise.} \end{cases}$

Source: Torres and Cantù (2018)

How to Estimate a Neural Network

Forward Pass:

1. Given a starting θ, NN takes input X and sequentially applies each function f_i: $a_1 = f_1(X, \theta_1)$, $a_2 = f_2(a_2, \theta_2), \ldots g(x, \theta) = a_k = f_k(a_{k-1}, \theta_k)$
2. Once we have $g(X, \theta)$, we can compute the loss function $\mathcal{L}(g(x, \theta), Y)$

Backpropagation:

1. From the last layer, chain rule to compute:
$$\frac{\delta \mathcal{L}}{\delta \theta_k} = \frac{\delta \mathcal{L}}{\delta a_k} \cdot \frac{\delta a_k}{\delta \theta_k}$$
and (recall a_{k-1} is the input of the final layer)
$$\frac{\delta \mathcal{L}}{\delta a_{k-1}} = \frac{\delta \mathcal{L}}{\delta a_k} \cdot \frac{\delta a_k}{\delta a_{k-1}}$$
2. We can then *backpropagate* the gradients through each layer, eg.:
$$\frac{\delta \mathcal{L}}{\delta \theta_{k-1}} = \frac{\delta \mathcal{L}}{\delta a_{k-1}} \cdot \frac{\delta a_{k-1}}{\delta \theta_{k-1}}$$

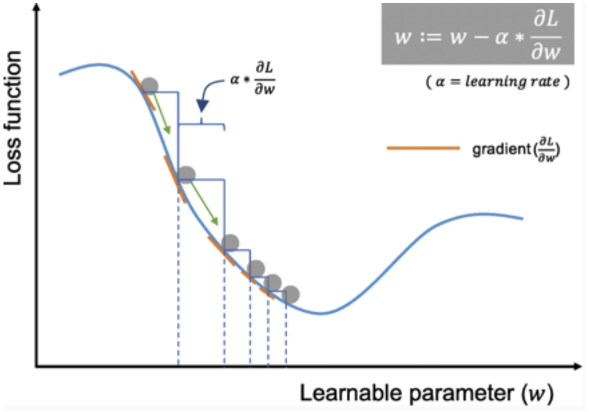
Gradient Descent

- With backpropagation, we can compute each $\frac{\delta \mathcal{L}}{\delta \theta_i}$
- We update parameters θ by moving on the opposite direction of the gradient $\nabla_\theta \mathcal{L}$, because this is the direction of steepest descent
- At each step of the algorithm we update the parameters so that

$$\theta^{t+1} = \theta^t - \alpha \nabla_\theta \mathcal{L}$$

  where α is called the *learning rate*
- We iterate the algorithm until loss is lower than some threshold

Gradient Descent


$w := w - \alpha * \frac{\partial L}{\partial w}$
(α = learning rate)
$\alpha * \frac{\partial L}{\partial w}$
gradient ($\frac{\partial L}{\partial w}$)
Loss function
Learnable parameter (w)

Source: Yamashita et al. (2018)

Outline

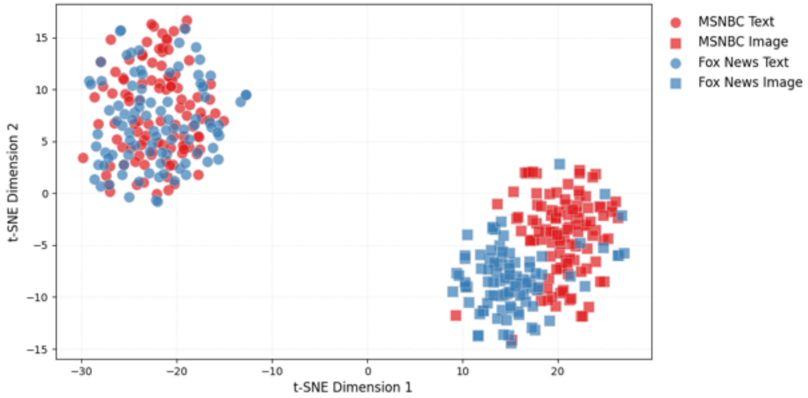
Unsupervised Tasks: Reducing Dimensionality

- In supervised tasks, we wanted to make our model predictive of some target Y, which we observed
- ML allows us to extract latent structures from unstructured multi-dimensional data (text and images!)
- All of these methods are essentially about **reducing dimensionality**
- There's different ways to represent text and image data as numerical vectors (more later):
    - Bag of Words, Visual Bag of Words (not as common nowadays)
    - Take a trained CNN, remove the last layer
    - Word Embeddings, Image Embeddings
- Once we have that, we can cluster our data (PCA, k-means...)
- Later: topic models

Unsupervised Tasks: PCA

- PCA finds a lower-dimensional representation that retains as much of the variance as possible
- Find projection function $f(\lambda) = \mu + V_q\lambda$ where V_q maps p (target dimension) into q (original dimension)
- Solve $\min_{\lambda, V_q, \mu} \sum_{i=1}^{N} ||x_i - \mu - V_q\lambda_i||^2$
- To perform this, perform Singular Value Decomposition on $X \implies X = UDV^T$
- The columns of V contain the eigenvectors of X^TX and give the directions in the original feature space - ordered by explained variance
- The first component is the direction capturing the the most variance so if we project the data onto this direction we retain the largest amount of variability
- PCA is the simplest way to perform dimension reduction. Other sophisticated alteratives exist (eg. t-SNE)

Component Reduction: Example



Unsupervised Tasks: k-means clustering

- Another useful technique is k-means clustering, used to group observations into clusters
- Within each cluster, points are as similar as possible to each other
- We solve:

$$\min_{S} \sum_{j=1}^{K} \sum_{i \in C_j} ||x_i - S_j||^2$$

- We are minimizing within-cluster variance (K is the number of clusters), where S_j is the cluster center
- So we are essentially finding the cluster centers, which also provides representative observations for each cluster

Image Clusters Representation: Fox News vs CNN

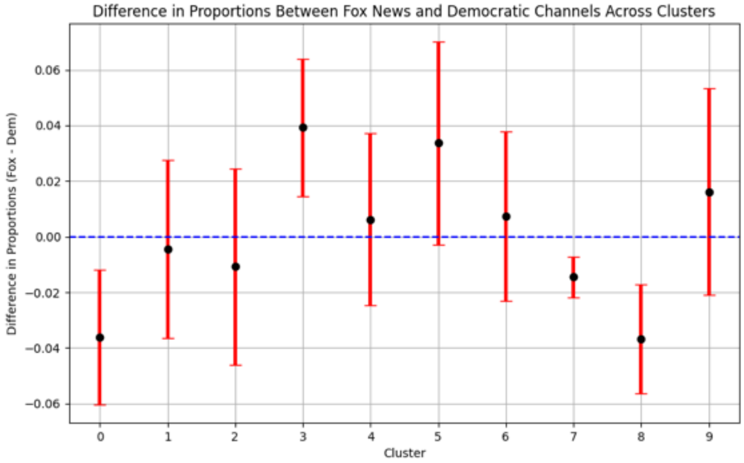
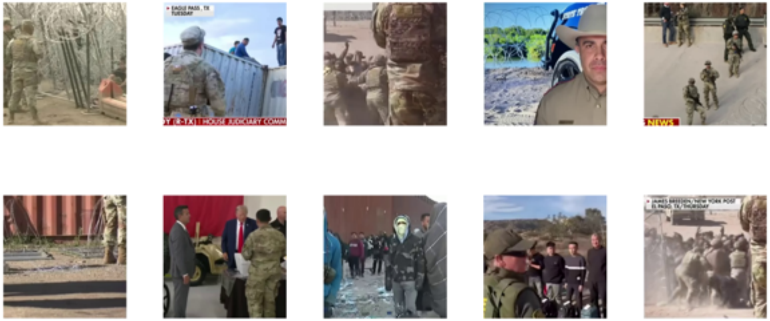


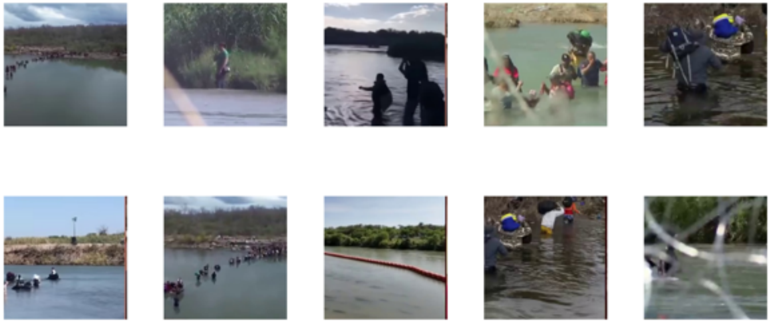
Image Clusters Representation: Fox News vs CNN


Images from Cluster 3 - Fox News

EAGLE PASS, TX TUESDAY
(R-TX) | HOUSE JUDICIARY COMM
NEWS
JAMES BREEDEN/NEW YORK POST EL PASO, TX/THURSDAY

Image Clusters Representation: Fox News vs CNN

Images from Cluster 0 - CNN



Outline

Outline

Steps in Text Analysis

- Most text analysis applications have 3 steps:
  1. Represent raw text $\mathcal{D}$ as a numerical array X;
  2. Map X to predicted values $\hat{Y}$ of unknown outcomes Y;
  3. Use $\hat{Y}$ in subsequent descriptive or causal analysis (not covered)
- Eg. With sentiment analysis, we can use the numerical representation of text in newspaper articles to estimate the true unknown sentiment Y and then use $\hat{Y}$ in subsequent analysis.
- Our **corpus** is a collection of raw text documents $d \in \mathcal{D}$, which we want to represent as a numerical array X

Step 1: Representing text as data

- Each document d has N words chosen from a vocabulary V
- We usually first reduce the number of features:
  1. remove punctuation, numbers, HTML tags, proper names, and so on;
  2. remove "stop words" as they probably do not convey much meaning;
  3. *stemming*: replacing words with their root.
- The simplest way to represent documents as numerical arrays is a BOW matrix c_w
  - Each entry represents the count of word $w \in V$ in d
  - d becomes a vector whose length is equal to $|V|$, c_{wd} is the number of times word w occurs in document d
  - Often we reweight using tf-idf:

$$\text{TF-IDF}(w, d) = c_{wd} \cdot \log\left(\frac{|D|}{d_w}\right)$$

Step 2: Statistical Methods

- Now that we have our representation (eg. document token-matrix) X we have 3 methods:
  1. Dictionary-based methods: $\hat{y}_i = f(x_d)$ for some known f;
     - Eg. sentiment analysis using lexicon of words to which a score is assigned (generally 1, 0, -1)
  2. Supervised: use other dataset to generate relation between words and outcome Y, then forecast $\hat{Y}$
     - LASSO, NN etc...
  3. Unsupervised: map words into latent variables $\hat{\theta}$, eg., the topics in each document
     - Generative models

Outline

Generative Language Models - Topic Models

- A topic model organizes text into interpretable topics (LDA is the most common)
- **Input:** BoW representation of text documents + number of topics + tuning parameters
- **Idea:** Vector of words c_i drawn from the vocabulary of words according to some document-specific probability vector on words $q_i = [q_{i1}, \dots q_{iv}]'$. Given document length m_i:
  - $c_i \sim MN(\mathbf{q}_i, m_i)$
  - In turn, q_i depends on v_i, which is a distribution over topics: $q_i = v_{i1}\beta_1 + \dots v_{ik}\beta_k$
  - A topic is the probability vector over words β_l. The latent attribute θ_i is a distribution over topics.
- **Output:** Distribution over words for each topic, distribution over topics for each document
- **Estimation:** Parameters fitting approximation of the likelihood function (eg. ELBO)

Application: Estimating Cultural Types in Organizations

- **Question:** How can we identify cultural types in firms?
- By using employee's reviews, we can extract types as conceptualized by employees, but a good topic model would need **at least** $\sim$ 20 topics
- **Idea:** Integrate clusters of topics within the generative process to extract interpretable clusters of topics
- **Generative Model:**

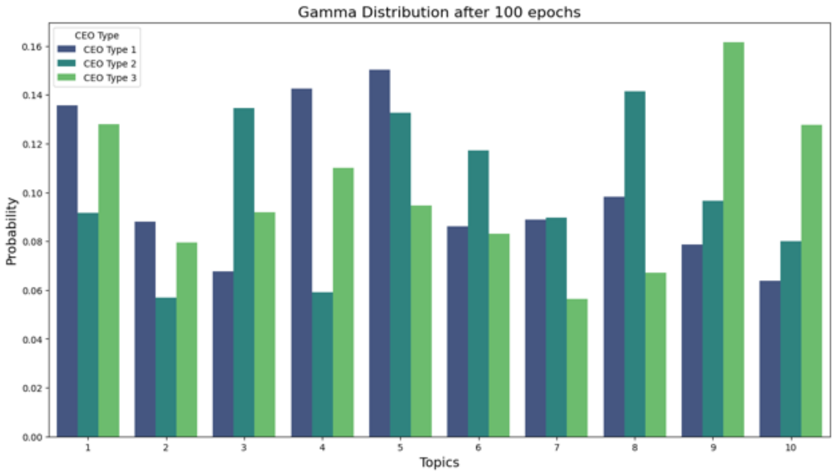
$$\begin{aligned} z_{ij} &\sim \text{Cat}\left(\pi\left(\theta_i + \gamma^T \cdot c_i\right)\right) \\ w_{ij} &\sim \text{Cat}\left(\pi\left(\beta_z\right)\right) \end{aligned}$$

  - Global latent variables: β (topics: distribution over words) and γ (types: distribution over topics)
  - Local latent variables: c_i (document-type intensity) and θ_i (document-topic intensity)

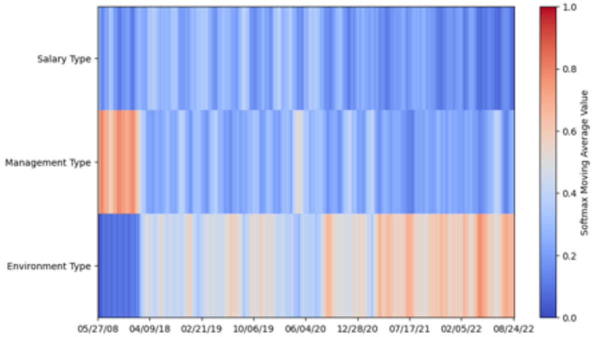
Cultural Types in Organizations: Topics

- Topic 1: home, benefit, paid
- Topic 2: job, easy, money
- Topic 3: team, management, environment
- Topic 4: pay, flexible, hour
- Topic 5: benefit, balance, salary
- Topic 6: place, manager, group
- Topic 7: opportunity, career, lot
- Topic 8: company, culture, employee
- Topic 9: environment, friendly, coworkers
- Topic 10: place, nice, fun

Cultural Types in Organizations: Types



Cultural Types in Organizations: CEO Example

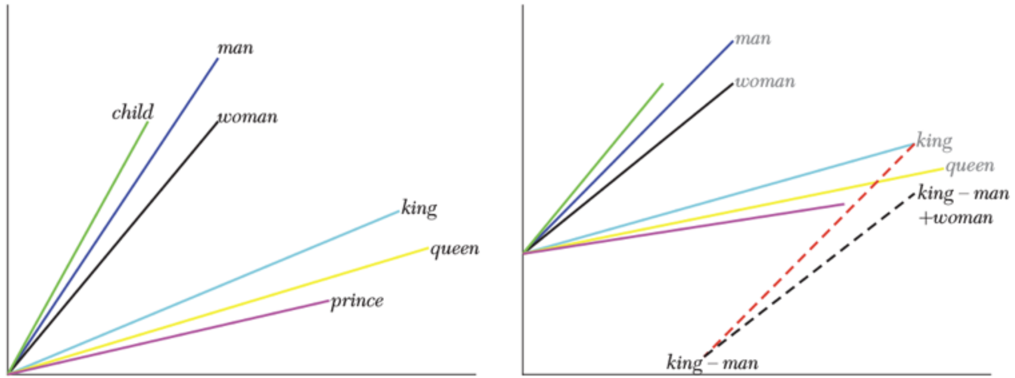


Outline

Word Embeddings

- BoW representation throws away a lot of information in word order, synonyms and general syntactical richness
- **Word Embeddings** represent words as vectors in some embeddings space $\mathbb{R}^K$
- Examples: Word2Vec, GloVe, FastText, BERT
- Semantic similarity between words is captured by the distance between embeddings in $\mathbb{R}^K$
- Representation in vector space allows for algebra to analyze relationships between words
    - eg. Ash et al. (2022) uses word embeddings to measure distance between court opinions and economics language
- They also allow to study the **directionality** of language: how is language changing? (really at the frontier)
- Training: predicts context words given a target word

Graphical Representation of Word Embeddings



Source: Gentzkow, Kelly & Taddy, 2019

Outline

Final Remarks on Text

- Textual analysis has been expanding rapidly in social sciences research within the past decade
- Text is high dimensional $\implies$ ML is necessary - a lot of advancement in NLP research still being done (think about LLMs)
- Advanced instruments for text representation are open source - easily available online
- Use it with precaution (but you can use it!)
- For political science: very important literature on partisanship measurement via text (Gentzkow and Shapiro 2010, Gentzkow et al. 2019, Vafa et al. 2021)

Outline

Outline

Images and Social Science: a Long Way to Go

- Political and social scientists have long understood the importance of visual data
- Nonetheless, before the advent of machine learning, analysis consisted mainly in manual collection and annotation of data
- Advancements in machine learning have enabled researchers with types of analysis that were not available before
- Furthermore, the expansion of Internet and social media has produced a huge amount of visual data for scientists
    - X (Twitter) only produces $\sim$ 3000 images **per second**
- Still, not a lot of work using images, especially in economics

Text vs Image

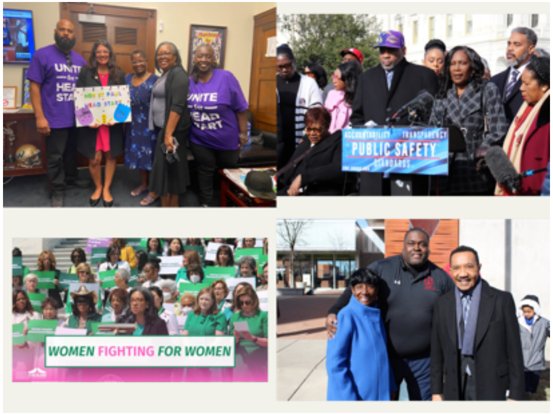
- Visual data is significantly different from text data, and thus requires different techniques for analysis:
  - If text is high dimensional (atomic elements are words) image is even higher dimensional (an array of pixels, each containing three numerical RGB components)
  - Text is **small size** - Image is **big size**
  - Text is **language specific** - Image is **universal**
  - In general, image analysis requires a great deal of work just to be on par with the starting point of text analysis

Outline

Supervised Tasks: Image Classification/Recognition

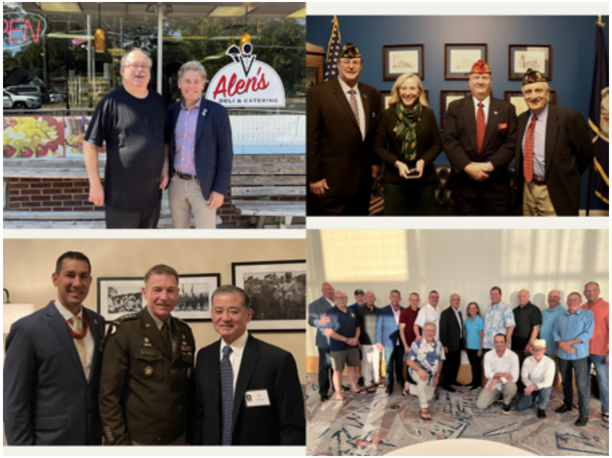
- **Image classification**
  - Essentially a $Y = f(X)$ prediction problem, as before
  - Assigns a label y_i to the image according to a prespecified label set (Eg. in binary classification $y_i \in \{1, 0\}$)
  - The object being recognized does not have to be a solo autonomous entity (scene or concept recognition)
- **Object detection**
  - Localize object instances and classify their categories
  - More complex
  - Two step procedure: first find "candidate" objects, then classify image to the candidates

Image Classification: Measuring partisanship?



Original label: democratic
Predicted probability of democratic: 0.988, 0.982, 0.986, 0.992

Image Classification: Measuring partisanship?



Original label: democratic
Predicted probability of democratic: 0.056, 0.052, 0.040, 0.046

Supervised Tasks: Face Analysis

- **Face detection**
  - Identifies location of faces in an image
  - Can be posed as a binary classification problem where the classifier is required to determine whether each image sub-region contains a human face or not
- **Face recognition**
  - Classifies the identity from a facial image
- **Face analysis**
  - Predicts general features of faces (eg. gender, age, race, emotions)

Supervised Tasks: Visual Sentiment

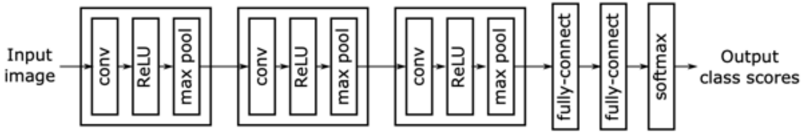
- Labeling expressed emotions or evoked emotion is subjective and hardly generalizable
- Need to personally label images: good idea to let many subjects do the task and take an average
- **Expressed Emotion**
  - Is an individual in the image sad? Happy? Angry?
- **Evoked Emotion**
  - Does the image make the viewer feel sad? Happy? Angry?

Outline

Convolutional Neural Networks

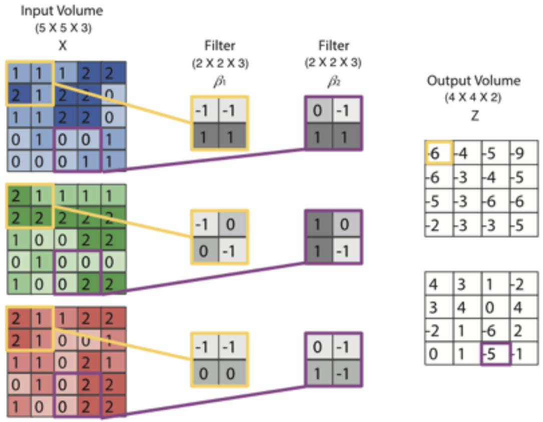
- Most of these tasks traditionally rely on Convolutional Neural Networks (CNN)
- These are Neural Networks characterized by an operation called convolution
- The convolution layer reduces the images' dimensions without losing important features
- Convolution is a linear operation where an array of numbers, called a kernel, is applied across the input, which is an array of numbers, called a tensor
- An element-wise product between each element of the weight kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor

CNN Architecture - An Example



Source: Joo and Steinert-Threlkeld (2018)

Convolution Layer - Example

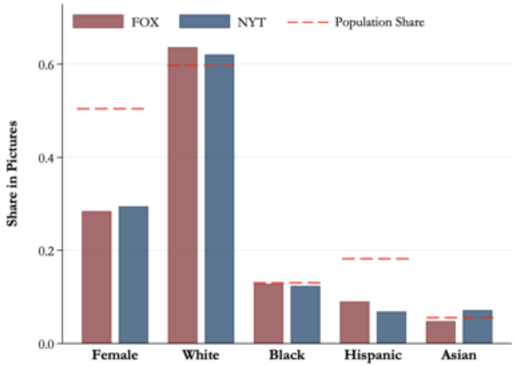


Source: Webb Williams, Casas and Wilkerson (2020)

How do we actually train a CNN?

- In general, we don't have to reinvent the wheel and design our own CNN architecture
- Also, training a CNN requires a lot of computing power (and time)
- We have alternatives:
    - **Off-the-shelf** open source libraries, eg. DeepFace, OpenCV, OpenFace (honestly, these are **very** unreliable)
    - Computer Vision **APIs** (eg. Microsoft Azure, Face++, Amazon Recognition)
    - **Fine-tuning** (this is the best!)
- Fine-tuning consists in adapting a pre-trained CNN (eg. ResNet) to our needs
    - Just alter and adapt the last fully connected layer i.e. we train existing models by just updating the parameters in the last layer
- A **simple useful technique** is to remove the last layer and represent images by their feature representation vector resulting from the forward pass - you can then do analysis using the vector representations of images

CNN Application in Political Science



Notes: This figure shows the shares of different gender and ethnic groups in the pictures of Fox and NYT. The red lines indicate the share of each group in the U.S. population.

Source: Ash et al. (2022)

Outline

Generating Images? GANs

- Generative Adversarial Networks (GANs) are a class of machine learning frameworks used to generate new data samples
- GANs consist of two neural networks, a Generator and a Discriminator, competing in a zero-sum game
- The **Generator** creates synthetic data to resemble the training data, while the **Discriminator** tries to distinguish real from generated samples.
- The goal is to reach an equilibrium where the Discriminator can no longer tell real from generated data, indicating high-quality synthetic data generation.

How GANs Work: The Adversarial Process

- The two networks, Generator G and Discriminator D, play an adversarial game:
  - **Generator** $G(z)$: Given random noise z, it generates an image $G(z)$ aimed at being indistinguishable from real images
  - **Discriminator** $D(x)$: Takes an image x as input and predicts whether it is real (from the training set) or fake (from the Generator)
- Both networks optimize opposing objectives, where:

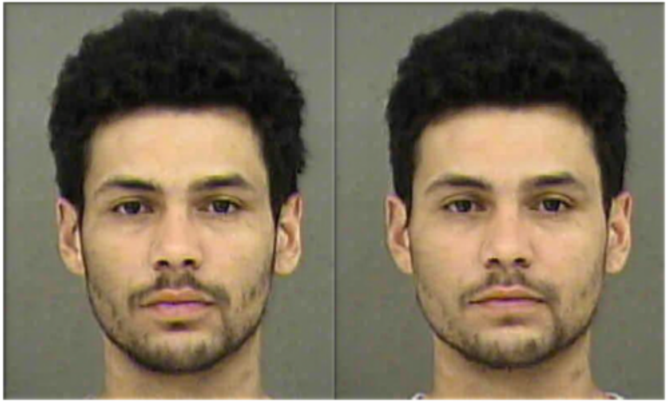
$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

- As the Generator improves, it "fools" the Discriminator, creating increasingly realistic images
- GAN can also be fine-tuned from existing models (eg. StyleGAN)

GANs in Action: Generating Counterfactuals

- GANs and CNNs can be used together to generate counterfactual images varying only on some features
- **Application to Judicial Decisions**:
  - Ludwig and Mullainathan (2024) uses GAN + CNN to create counterfactual images of defendants, varying only in features that influenced the likelihood of indictment.
  - Isolating the visual elements in mugshots that correlate with judicial outcomes, model uncovers latent patterns in judge decision-making
- The GAN-CNN pipeline can allow the generation of counterfactuals based on patterns that are not immediately apparent or difficult to study directly
- **Potential Applications**: Counterfactual newspapers images along the partisan dimension?

GANs in Action: Generating Counterfactuals



(a) Side-by-side mugshot detention morphs with detention probabilities of 0.41 and 0.13 respectively

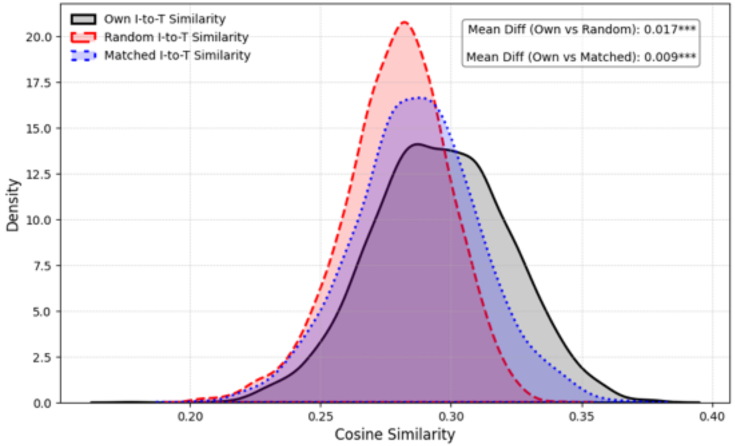
Source: Ludwig and Mullainathan (2024)

Outline

Some Remarks before Concluding

- Currently, a lot of ML research on images
- The progress is very fast! A new breakthrough is found every couple of years
- Ludwig and Mullainathan (2024) is one of the few papers using GANs and they are already considered "outdated" by computer scientists
- **New Frontier:** Diffusion Models, LLMs. This is what is used by state-of-the art text to image models (Dall-E, Stable Diffusion etc.)
- DMs rely on models like CLIP, which are trained to project images and text into the same embedding space
    - Computer scientists are not very interested in comparing text and images, but for us the comparison may represent an important question

Text and Image Similarity in Political Ads via CLIP



Back

Conclusion and Reflections

- Advances in ML allow us to analyze images and text as complex data sources
- These tools have expanded the toolkit for social scientists, enabling new insights into phenomena that were previously difficult to quantify
- Modern ML models can do more than predict: they generate interpretable hypotheses by isolating features with implicit social relevance
- High-dimensional data brings interpreability challenges - transparency remains an essential question
- An other crucial question is how to use NLP and Computer Vision for causal inference (see eg. Naoki Egami's work)
- One last "technical" recommendation: **use Python!**