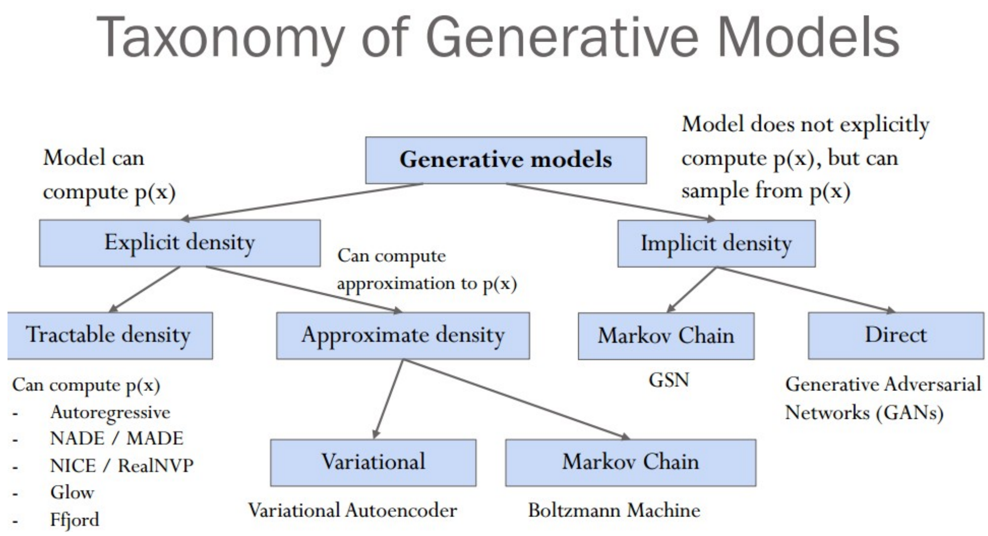
- Deep Learning Ingredient
  - Algorithm
  - Data
  - Computation
- Deep (Machine) Learning
  - Representation learning
  - Neural Networks
  - Deep Learning
    - Hierarchical Compositionality
    - End-to-End learning
      - Distributed Representation
      - Complicated Function
      - issues
        - Non-Convex
        - may have multiple local minima
        - Lack of interpretability
      - Reproducibility
- Predictive/supervised modeling
  - Classification → Discrete target
  - Regression → continuous targeT
- Unsupervised learning
  - Clustering
  - Data compression
  - Dimensionality Reduction, Manifold learning
  - Data distribution learning
- Self-supervised or predictive learning
  - Self supervision
  - Pretext tasks
- Reinforcement learning → Rewards in sequential Environment
- Active Learning
  - Human in loop
  - most important data to learn from
- Vision Challenges
  - Viewpoint variation, illumination ,Deformation,Occlusion
  - Background clutter ,Intraclass variation
- Instance based learn
  - Expensive, inefficient, Curse of Dimensionality
- Loss function → measure how much the good the predictive function is
  - SVM loss – hinge loss
  - Cross Entropy loss → Softmax function
- Regularization
  - Prevent model from doing too well on the training data
  - Simple: L1(lasso) , L2 (ridge), Elastic Net (L1+L2)
  - Complex: Dropout, Batch Normalization, Fraction Pooling, early stopping
- Optimization
  - Gradient Descent
    - Numerical → slow, approximate, easy to write
    - Analytic→ Fast, exact, error-prone
- Neural Network

- Perceptron
  - Binary classifier
  - perceptron algorithm
- Multilayer perceptron
    - Requires non-linearity→ else a linear class
      - Activation functions
        - Sigmoid→ saturation, not zero centered, computational heavy
        - tanh → zero centered
          - not to use too many
        - ReLU → easy to use
          - not zero centered
          - negative crushing
        - ELU, Maxout, leaky ReLU
    - Gradient Descent
- Back-propagation training algorithm→ localgradient*upstreamgradient
- Convolutional neural network
  - parameter sharing across different location
    - Convolution kernel and size K
    - Convolve (slide) over spatial location
  - multiple feature maps – channel
  - CNN → pooling → Activation function → Normalization
  - Stacking Convolution
  - padding P to solve the shrinking with each layer
    - Add zeros around the input
    - Edge gets similar calculation as the insides
  - Receptive Fields
    - What one points see
    - kxk immediate next layer
    - 1+L(k-1) with L layers
  - Downsampling to increase the Receptive field
  - Stride Convolution S→ Filter shift
  - Output size (W-K+2P)/S+1
  - Pooling layer
      - Nonlinear Downsampling
      - reduce feature size
      - Control overfitting
    - Max pooling
    - average pooling
  - Batch Normalization
      - zero mean, unit variance
      - Differentiable function
    - reduces Internal covariant shift
    - other control parameters
    - Test time: Running average/variance during the training time
      - a linear operation during test time
    - usually in between the CNN/FC and activation function
    - May behave different in test (A bug)
    - Theoretically not well understood!!
    - layer normalization, instance normalization

    - Most expensive part
        - Convolutional parts (most memory)
        - Most parameter (flatten layer to FC layers)
    - Some architecture
        - AlexNet
        - ZFNet- big alexnet
        - VGG net
            - uses smaller kernel but deep kernel
            - reduces parameters
        - Google net
                - Efficiency
                - no flattening to FC layer at the end
            - inception layer
            - auxiliary classification (gradient vanishing issues)
            - Aggressive Stem network
            - Global average pooling
        - ResNet
            - Residual network
            - Solve optimization problem → Vanishing gradient issues
            - identity mapping by skip connection
            - Basic block and bottleneck block
            - Pre-activation, BN modification
            - Squeeze and excitation network
- Training Neural network
    - Go with traditional architecture
    - Go with relu activation
    - Data preprocessing
        - normalize it → covariant shift
    - Weight initialization
        - never all zero – symmetric issues
        - initialize randomly → Xavier initialization
    - use regularization
        - L1, L2, Elastic net
        - Drop out (only during training)
            - Prevent coadaptation
            - Drop connection
        - Data augmentation
            - augmentation
            - mix up
        - Batch normalization
        - Factorial max pooling
    - Optimization
        - SGD → long time required
            - too much noise
            - local minima, saddle point
        - SGD with momentum
        - Second order optimization → inverse of hassian (Jacobian of Gradient)
    - One time setup→ architecture, regularization
    - Training dynamics

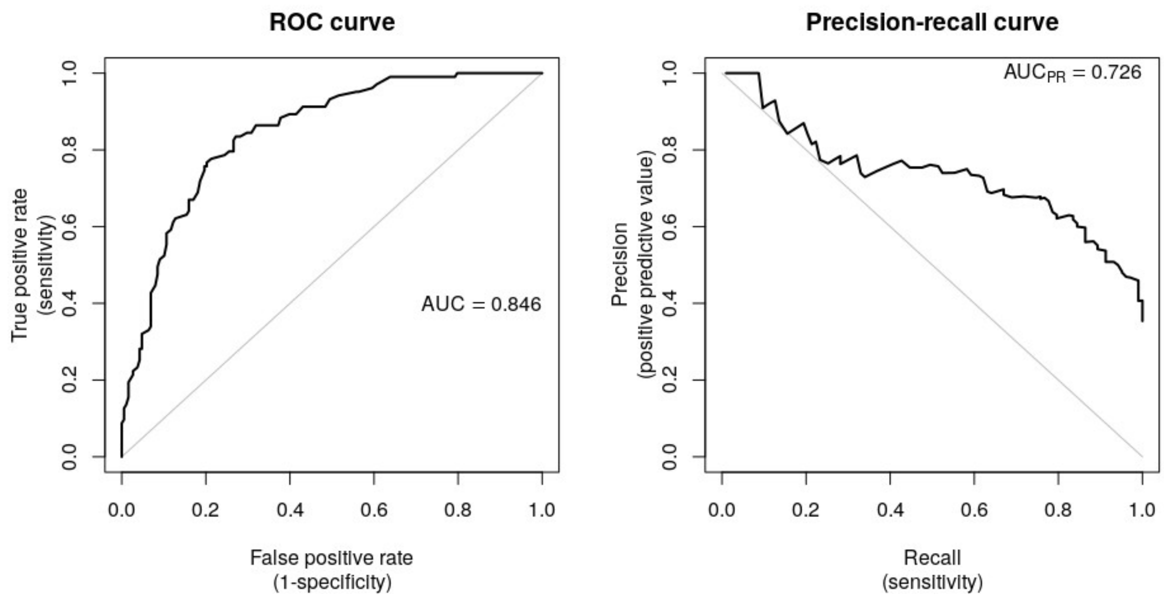
                - learning rate schedule: Decay
                - hyperparameters optimization
                    - Check initial losses
                    - overfit small data
                    - Find LR
                    - Refine and train longer
                - Observe learning curves
                    - Look for overfit, underfit, goodfit, good learning
            - After training→ ensembles, transfer learning
        - Recurrent Neural Networks (Elman RNN)
            - Time series predictions
            - sequence to sequence
            - Requires a lot of memory
            - vanishing/exploding gradient → Solution: Gradient clipping
        - LSTM
        - Attention
        - Computer vision tasks
            - Classification
            - Semantic Segmentation
            - Object detection
                - Multitask learning loss function
                - Region Proposal Network
                - R-CNN: Region Based CNN → too much training cost
                - Intersection over union
                - Mean Average precision
                - Fast R-CNN
                    - Two stage methods
                    - Region of interest selection
                - Faster R-CNN→ Region Proposal Network
                - YOLO
                - single shot detection
            - Instance Segmentation
                - Mask R-CNN → Extends Faster R-CNN

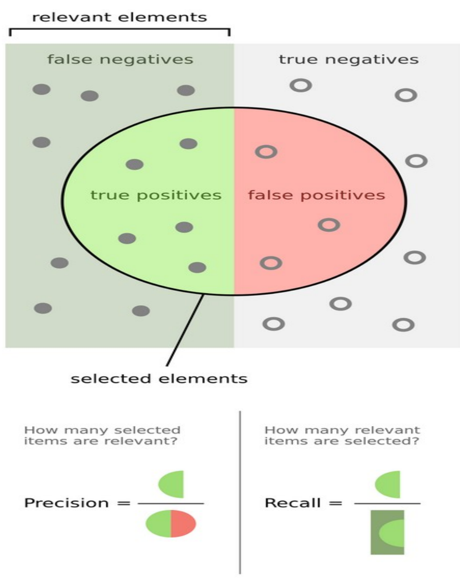
- Generative Adversarial Networks



- Unsupervised
  - k-Means Clustering
  - PCA dimension reduction
  - Feature learning auto-encoder
- Learning the distribution of the data
- Autogressive model → pixelRNN, pixelCNN
- Variational autoencoder → Maximize the lower bound of p(data)
- GAN
  - Vanishing gradient problem
  - DCGAN
  - WGAN → Better gradient and prevent mode collapse
  - LSGAN
  - BigGAN
  - Pix2Pix
  - CycleGAN

- RL (https://smartlabai.medium.com/reinforcement-learning-algorithms-an-intuitive-overview-904e2dff5bbc)
  - https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html
- Random Forest
  - https://sebastianraschka.com/faq/docs/bagging-boosting-rf.html#boosting
  - https://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics
  - https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205
- p-value → https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/p-value/#:~:text=The%20p%20value%20is%20the,value%20of%200.0254%20is%202.54%25.
  - https://www.investopedia.com/terms/p/p-value.asp
- Machine learning Glossary (https://developers.google.com/machine-learning/glossary)
- Accuracy https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-average-performance-in-a-multiclass-classification-settin

- 

- 

- **True Class**

Acquired Class	T	F
Y	True Positives (TP)	False Positives (FP)
N	False Negatives (FN)	True Negatives (TN)

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

$$\text{Accuracy} \quad (ACC) = \frac{TP + TN}{TP + FP + TN + FN}$$

Predicted Class

Actual Class		Positive	Negative	
	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

- svd /pca (https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca)