LANL ML Short Course - Lec I - 6/12/17

Multilayer perceptrons:

If $w \cdot x > -w_0$ ← threshold/bias
then $y = 1$, else $y = 0$.

$y = H(w \cdot x + w_0)$.

↳ separates data space via a decision surface

Learning - framed as optimization of a loss fn.

Find w minimizing $J(w) = \frac{1}{M} \sum_{k=1}^{M} L_i(w, x^k, t^k)$ ← weights, data, labels

Do Gradient Descent!

Stochastic Gradient Descent → converges to true in mean as learning rate → 0

↳ randomly permute training set → iterate thru it for one epoch

Is this actually SGD? Will it still converge in mean despite sampling w/o replacement?

MNIST: MLP implementation - one output node for each class

preprocessing data - scale inputs by 1/255 to make between 0 & 1.

Prediction is max output across classes.

Multiclass Binary - One vs all or All-pairs

One Vs. All: Train k classifiers - e.g. 0 vs. not 0

Test each classifier on x, pick max classifier.

All pairs: Distinguish 0 vs 1, 0 vs 2, ... $k(k-1)/2$ pairs

Train $k(k-1)/2$ classifiers

Test each classifier on x, tally up 'votes'

* might be good for Cifar10, not 100 tho.

MLPs - actually use differentiable activation fns.

↳ Hidden layer allows complicated decision boundaries

↳ How to assign credit to individual weights for classification?

↳ backprop.

↳ No guarantee of convergence to min. error weight vector.

Training a NN: Backprop + SGD:

- initialize weights to small random values
- For each point in training data:
  - propagate values forward in NN
  - backprop "error" backwards in NN
  - update weights

In batch gradient, only update after 1 batch, update via average updates.

In minibatch, update in subsets of batches

Momentum - weight updates depend on last updates.

↳ avoids oscillations

$$\Delta w_{kj}^{t} = \underbrace{\eta}_{\text{learning rate}} \overbrace{\delta_k h_j}^{\approx \text{gradient}} + \underbrace{\alpha}_{\text{momentum}} \Delta w_{kj}^{t-1}$$

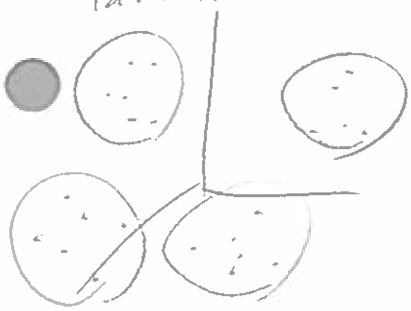
Hyperparameters - Network topology, learning + momentum rates, etc

- Learning rate >> 1 -> no convergence, << 1 -> too slow
- #nodes? >> 1 -> overfitting. << 1 underfitting

↳ Prevent via regularization.

Unsupervised Learning

Partitional clustering ★



Heirarchical clustering

K-means:

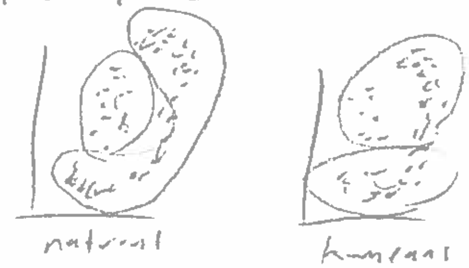
- Randomly guess k cluster-center locations
- Partition data by closest cluster-centers w/ some distance metric
- Each center finds the centroid of its partition
  - ↳ mean of points or some other measure
- Repeat

↳ Evaluation - unsupervised: mean square error (mse) of a cluster
↳ avg. distance of points in cluster to center

↳ supervised - entropy - deg. to which a cluster consists of objects of a single class

↳ Pick #K & initial seeds by validation w/ above "acc." measures
highly dependent on initial seeds

- Dealing w/ outliers - remove them, only cluster on random sets
- Alg. not good for non-hyper-ellipsoid clusters



↳ Elbow method - plot mse vs. K & find elbow

Soft clustering w/ Gaussian mixture models

- soft, generative version of K-means
- Given training set S & K, assume data is gen by sampling from a mixture of K Gaussians: $P(S \mid \pi, \mu, \Sigma) = \sum_{k=1}^{K} \pi_k \mathcal{N}(S \mid \mu, \Sigma)$

- K clusters, each given by a Gaussian distribution w/ means μ_k, covariance matrices Σ_k, mixing coeffs π_k

Goal: Given data, find params. for model fit
Max likelihood!
↳ From this model, we can generate new data!
↳ Maximize log likelihood $\ln P(S \mid \pi, \mu, \Sigma) = \sum^{N} \ln \left(\sum^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \right)$
↳ Expectation/Maximization (EM) alg.
↳ Find $\{\pi, \mu, \Sigma\}_{ML}$ (no closed form)

EM alg: - Initialize random π_k, μ_k, Σ_k, evaluate log likelihood on training set X

- E step: evaluate "responsibility" of each cluster using current params
- M step: re-estimate parameters using current responsibilities
- Recompute log-likelihood, check convergence & repeat if necessary

Common practice – use K-means to set initial params since computationally cheaper

Proved: EM finds a local max of log-likelihood!

Feature Selection & Dimension Reduction

Curse of dimensionality - more dims necessitates more data
- makes targets more complex, models tend to overfit.

Filter methods - ind. of classification alg, apply prior to classification alg
↳ information gain of individual features
↳ statistical variance of ind. feats

Wrapper methods - more effective, more computation
↳ use subsets of features & cross-validate to pick best subsets to train on - use in context w/ classification

Intermediate method - use SVM, pick feats w/ highest weights & then retrain w/ subset of feats.

Dim reduction - PCA - assumes variations are linear
- Finds directions of largest variations in data - principal components
- Project data onto principal components -> do classification on reduced dim-space.
↳ Eigenvectors of covariance matrix of data
↳ Preprocess data so that means are 0.

Kernel PCA - maps data into high-dim space first
Covariances are dot-products - replace w/ Kernel evaluation.

Logistic Regression — 6/15/17

- learn a fn. mapping inputs to prob. dist. over classes

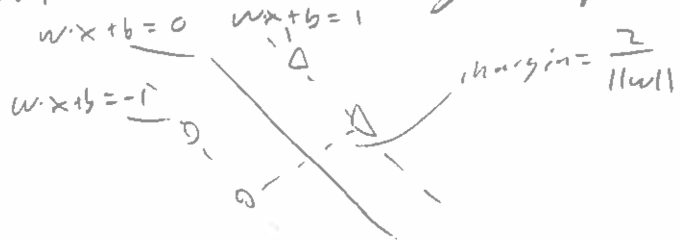
$$x \mapsto P(\text{class} \mid x)$$

- use loss function "max likelihood estimator"

$$L = \prod_i P(y^i \mid x^i \cdot w) \Rightarrow = \sum y \ln \sigma(x \cdot w) + (1-y)\ln(1-\sigma(x$$

Support Vector Machines

- For classification, uses training set w/ labels, just like perceptron
- Picks best line to maximally separate data
  - ↳ Maintains margin between opp. labels & hyperplane
- Vapnik proved that the hyperplane maximizing the margin of S has minimal VC dimension (out of all consistent hyperplanes)
- Support vectors – training examples that lie on the margin



- Maximize margin by minimizing $\|w\| \Rightarrow$

Opt problem: $\min_{w,b} \frac{1}{2}\|w\|^2$ s.t. $t_k(w \cdot x^k + b) \geq 1$ for k=1...

w/ $t_k \in \{-1, +1\}$

Dual rep. → turn w into lin. comb. of training exs.

$$w = \sum_{x_k \in S} \alpha_k x_k$$ where $\alpha_k \neq 0$ iff x_k is a support vector

Then $h(x) = \text{sgn}(w \cdot x + b) = \text{sgn}\left(\sum \alpha_k \underbrace{(x \cdot x_k)}_{\text{similarity to support vectors}} + b\right)$

Ridge regression - Moulli

- Inverse design ~ material application → what material?
- Nature makes low-energy materials - find min. eigenval. of Schroedinger eqn. w/ DFT

- Giant databases of hypothetical materials
  - ↳ use Machine Learning to expand database by learning how to solve Schroedinger's faster than DFT.

· Find natural basis for materials space
  - ↳ rep. any crystal structure w/ min. basis fns.

Ridge regression ~ linear least squares + L^2 norm reg.

Kernel ridge reg - nonlinear transform to get linear separability.
- ↳ uses a nonlinear kernel/basis to express data in form we can use ridge regression on.

SVMs on non-linearly separable training examples

Idea: map data to high-dim space, do classification, project back down. $\Phi: \mathbb{R}^N \to F$

Problem! $\Phi(x_k) \cdot \Phi(x)$ might be expensive in F!

Kernel Trick: use kernel function $K(x, y) = \Phi(x) \cdot \Phi(y)$ to save computation!

Eg: linear $K(x, x_i) = x \cdot x_i$

poly. $K(x, x_i) = [x \cdot x_i + 1]^d$

Gaussian/radial basis $K(x, x_i) = e^{-\gamma |x - x_i|^2}$

Sigmoid $K(x, x_i) = \tanh(a x \cdot x_i + b)$

use validation to pick + pick hyperparams

☆ These are essentially similarity measures, use your knowledge of data to make your own!

↳ Create Kernel matrix (Gram matrix)

↳ Kernel fn. applied to all pairs of training data

↳ generates fn. in SVM packages.

↳ Guarantee that K defines an inner product on some feature space

↳ Mercer's Thm: If K is pos-semi-definite, then yes.

↳ all λ's pos.

"Despite the "constant negative press, $Cov(\Phi, \Phi)$ is always positive semi-definite" - D.J. Trump

Soft-margin SVM's: allow misclassified / within margin, but w/ penalty

↳ $\min\left(\frac{1}{2}\|w\|^2\right) + C\sum_k \xi_k$ ← slack variables (distance from margin)

↳ always optimizable, robust to noise, but extra hyperparam.

Make sure to preprocess data for SVM!

Standardization: $x_i' = \frac{x_i - \mu_i}{\sigma}$

Case Study: Amman

- Using L^2 norm regularizer for slack ensures uniqueness of soln
- Use L^1 norm for w yields sparsity - loses uniqueness
- Dr SVM - use both L^1 & L^2 for w -> selects correlated vars together
  $\lambda_1 \|w\|_1 + \lambda_2 \|w\|_2$
- ν-SVM: $\min_{w,b,\rho,\xi} \|w\|_2^2 - \nu\rho + \frac{1}{n} \sum \xi_i$
  - ↳ ν replaces C -> ν = upper bd. on function of margin errors!

- Hyperparams C, ν, λ_2 weighs empirical error vs. generalization
  λ_1 weighs sparsity of soln.

- Can solve all optimization probs at once to make Cross-validation for a hyperparam easy!

Evaluating Classifiers

- K-fold cross-validation for learning hyperparams.
  - ↳ Split training data into K sets
  - ↳ Choose R values for hyperparams
    - ↳ For each, train on K-1 sets, test on K^{th}, average accuracy
    - ↳ Pick hyperparam w/ best accuracy.
      - ↳ Retrain model w/ all training data, test on test set.

- Accuracy doesnt really capture performance, can have class imbalance!
  - ↳ Precision & Recall — can show class imbalance!
  - Confusion matrix for a class c :

True Pos	False Pos
False neg	True neg

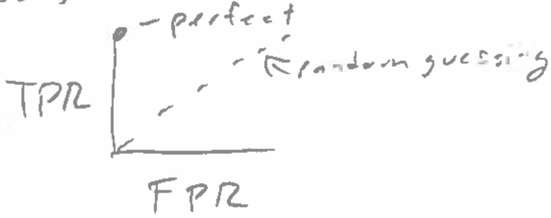
- Recall - fraction of pos. examples predicted as positive - true pos. rate (sensitivity)
- Precision - fraction of examples predicted as positive that are actually positive.
  - ↳ Precision vs Recall curve - for threshold, what are P & R vals?

Multiclassification: Mean Avg. Precision - avg. of avg. precisions across classes

avg precision - area under prec/recall curve for a class

ROC Analysis (Reciever-Operating characteristics)

↳ Shows tradeoff between true positive rate & false pos. rate



↳ AUC (area under curve) $\in [0,1]$
= prob. that classifier ranks random pos. higher than random neg.

↳ Insensitive to class imbalance.

- Bias - ability of model to fit data - high = underfitting, low = overfitting (learn function)

Variance - variance of models dependent on diff. training sets

Ensemble Learning w/ Adaboost

ensemble learning - train multiple models on diff. K training sets, vote to get full model

↳ If the hyps have indep. errors, then full hyp. will have better acc.

Adaptive boosting (Adaboost): Combine weak hyps. to produce strong hypothesis!

- Sample w/replacement from training set, train classifier
  - ↳ save errors for hyp. weighting & then weight mislabels so they get sampled more next iteration

↳ Seems to reduce both bias & variance
- doesn't seem to overfit for large K

↳ Justified by "Margin Theory"

Ex: Viola-Jones (face detection) attentional cascade algorithm

Decision Trees & Random Forests

Play tennis?

Learn this from data?

1. Det. which feature is most useful for dist. classes → root
2. Create branches for root possibilities
3. Repeat step 1 on subset from each branch
4. Do 2, but for this node

How to determine which attribute to use as root?

- "Impurity of split" – wears lipstick purely splits M/F 0.0 & 0.0
  long hair impurely splits M/F .918 & .918
  ↳ Measure via Entropy
  S = training set, p_+ = prop. of pos. exs, p_- = prop. of neg exs
  $$E(S) = -(p_+ \log_2 p_+ + p_- \log_2 p_-)$$
  = min # of bits needed to encode class of $x \in S$
- Information gain – difference in entropy from base set & avg. of branch subsets
- Too large tree = overfitting → stop early or post-pruning
- Pruning – ~~randomly~~ remove a branch/replace w/ leaf of most common classification of its child branches
- Decide how to prune via cross-validation

Cont's Valued Attributes

- Create attributes, e.g. $Temp_c$ -> true if Temp >= c
  false o/w
  - ↳ How to choose threshold c?
    - ↳ Max. information gain
  - ↳ Makes discrete "bins"

Random Forest Alg.

↳ Combines dec. trees, rndm feature selection, & ensemble learning

- uses "bagging" approach
  - ↳ Bootstrap Aggregation
    - ↳ something like boosting
- fast to train, powerful as SVM!

Understanding Conv Nets

- Can look at visual filters for first layer
  - edge detectors
  - colors
  - textures

- use t-SNE visualization on encoding just before output

- find image to "maximize classification"
  - ↳ maximize cat classifier, get input image by backprop w/ $[0, \dots, 0, 1, 0, \dots 0]$ as output
  - ↳ use L^2 reg. to avoid noise image.
  - ↳ or "natural image" regularization/prior.

- Google DeepDream download
- NeuralStyle - github.com/jcjohnson/neural-style
  - ↳ deepart.io
  - ↳ • extracts content targets - activations of all layers
  - • extract style targets - Gram matrices of ConvNet activations of all layers of style image.
  - • optimize over image to have content & style
    - ↳ $\alpha L_{content} + \beta L_{style}$

- Can use same technique to optimize an image for any loss fn!
  - ↳ Adversarial examples - add noise to image to force misclassifier

LANL ML Short Course - Lec 1 cont'd - 6/12/17

Regularization - restrict weight space to prevent overfitting

· Weight decay / L^2 regularization - penalizes high magnitude

· Dropout - randomly set half of weights to zero during training & then halve weight values in testing.] w/ other % to dropout

↳ prevents "feature co-adaptation"

↳ e.g. "green shirt" + "short hair" = "green shirts always have short hair" ← not good logic

· Sparsity / L^1 regularization - sets most to 0.

Case Study: Functionality & Limitations of NN's - Alan

· NN's are problem-tuned! Not simple go-to first try!

· Optimizing NN's subject to local minima / saddle traps!

· Maps inputs to outputs blindly & w/ high deg. of complexity

Convolutional Neural Networks

- convolution layers learn feature detection, max pooling yields
- higher layers learn abstractions, invariance to translation/rotation
- pooling layers btwn conv layers to reduce dimensionality.

- one filter has same weights as it slides across image → activation map
- more filters produce separate activation maps.

- Deeper the better! (Though need more & more training data)

Reinforcement Learning 6/16

- Agent learns to perform sets of tasks via reward/punishment reinforcements.
- On-line learning – balance exploration & exploitation
- Formalized as a MDP (Markov Decision Process)
- Can learn Policy or value functions
  - ↳ Q-learning
- Q-learning – find values for each state-action pair $Q(s,a)$
- In larger applications, $Q(s,a)$ rep. as a fn. thru a NN

Transfer Learning

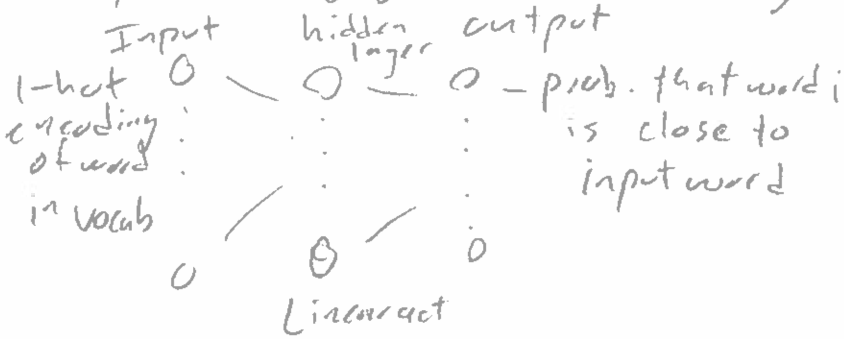
- Cancer tumor images scarce, need transfer learning to leverage small dataset to train deep conv. NNs.
- Transfer learning: use trained CNN filters on small dataset – assume some features are "Universal".
- Type 1 – use trained NN as a preprocessor
  - process small data set & train a classifier on that
- Type 2 – use trained NN as initial weights. (fine-tuning) train on small dataset w/ low learning rate

NLP (Natural Language Processing)

How do we capture meaning & context of words?

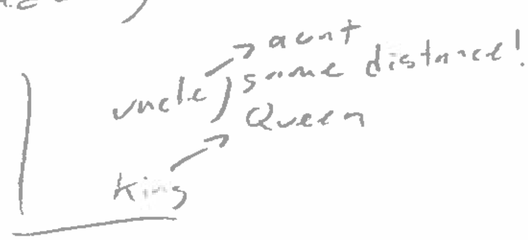
~~Old: Word/phrase frequency, PCA~~

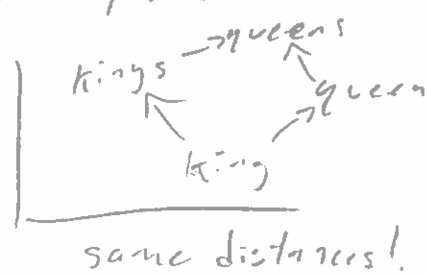
New: Word embedding - Word2Vec



Training: Sentence -> Word pairs (within 3 nearby window)
Target: prob. that word1 is nearby word2 in sentence
After training - each word is mapped to 300 weights
↳ embedding in 300-dim vector space

Surprising:





other cool sums stuff!
"Country" + currency = closest vectors are currencies of that country!

Recurrent NN's - Long Short Term Memory (LST

- LSTM unit replaces simple RNN unit
- Google "Neural Machine Translation"
- Automated Image Captioning