Lecture 5

Dimensionality Reduction – Group Data, Remove redundant features – Benefits: Reduces time complexity: Less computation • Reduces space complexity: Fewer parameters • Saves the cost of observing the feature • Simpler models are more robust on small datasets • More interpretable; simpler explanation • Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions / solutions: Projection – map from higher to lower dimension. Often variability is in fewer dimensions with constant in others. Manifold: The manifold assumption is often accompanied by another implicit assumption: • The task at hand (e.g., classification or regression) will be simpler if expressed in the lower-dimensional space of the manifold • This implicit assumption does not always hold / Forward selection is faster than Backward Selection (why?)

PCA: Principal component analysis centers the sample and then rotates the axes to line up with the directions of the highest variance. If the variance on z2 is too small, it can be ignored and we have dimensionality reduction from two to one / NOTE: Singular Value Decomposition (SVD) can decompose the training set matrix X into the matrix multiplication of three matrices which contain the unit vectors that define all the principal components that we are looking for / Sum of first k principle components (PC)divided by sum of all PCs. Calculate for k = 1, k=2, until value is > 0.9 🡺 you have explained 90% of the variability.

Unsupervised learning: Clustering - Clustering is Used in a Wide Variety of Applications Customer segmentation • Data analysis • Dimensionality reduction • Anomaly detection (also called outlier detection) • Semi-supervised learning • Search engines • Image segmentation/ Anomaly Detection / Density Estimation / Active Learning - When a human expert interacts with the learning algorithm, providing labels for specific instances when the algorithm requests them. Can use min, max, or average to define cluster

k-Means / The K-Means algorithm does not behave very well when the blobs have very different diameters because all it cares about when assigning an instance to a cluster is the distance to the centroid. / 1. Ask user how many clusters they’d like. (e.g. k=5) 2. Randomly guess k cluster Center locations 3. Each datapoint finds out which Center it’s closest to. 4. Each Center finds the centroid of the points it owns / K-Means Works If Clusters are spherical • Clusters are well separated • Clusters are of similar volumes • Clusters have similar numbers of points / not guaranteed to be optimal / one approach is place first point randomly and then place subsequent points far away rather than randomly / choosing the number k is hard, BIC = log(m)p – 2(log(l-hat)) and AIC = 2p – 2(log(l-hat)) where m is number of instances, p is number of parameters and l-hat is maximized likelihood value / mini batch is 3-4 times faster its inertia is worse / use silhouette score to determine number of clusters – between -1 and 1 / +1 inside cluster, 0 on boundary, -1 wrong cluster / need to run multiple times / need to scale the input for k-means

Expectation Maximization – EM – Repeat until convergence: E-compute expected values of unobserved variables, M – Compute new parameter values / one approach, assume best student got them all right , or assume that on average the class got each question right

Lecture 6

Perceptron training rule / w sub I gets w sub I + delta w sub I where delta w sb I gets n(t – o) x sub I with t = c(x), target value, o is perceptron output, n is small constant called learning rate / it will converge if data is linearly separable, no noise, and n is sufficiently small. Gradient descent is better

Text

Description automatically generated

Until termination criteria is met, do, Initialize each delta wi to zero, for each <x vec, t> in training examples, do, input x vec to unit and compute o, for each weight w sub i, do delta w sub i gets delta w sub i + n (t – o) x sub I / for each linear weight w sub I =, do, w sub I gets w sub I plus delta w sub I, momentum – add alpha times delta w sub I for time t minus 1. This speeds up or slows down the convergence. Adaptive learning rate. Batch mode vs Stochastic Gradient descent.

Logic gates: AND w0 - -1.5, w1 - +1, W2 - +1 – y = s(x1+x2-1.5), OR w0 - -0.5, w1 - +1, w2 - +1 – y = s(x1+x2-0.5), XOR needs a hidden layer

Backpropagation, Forward pass, make a prediction, measure the error, measure the error contribution from each connection (reverse pass), and tweak the connection weights to reduce the error (gradient descent pass), each pass is called an epoch

Table

Description automatically generated

Training can be slow but using the network is fast. Boolean functions: everyone can be represented with one hidden layer buy may require a count of elements that is an exponential of the inputs / Continuous Functions – bounded can be approximated to an arbitrary small error with one layer / any function with arbitrary accuracy with two layers. / deep networks can use exponentially fewer neurons than shallow ones

API: Keras, PyTorch / Library: TensorFlow, CNTK (Microsoft), Theano / The softmax function then generates a vector of (normalized) probabilities with one value for each possible class. Proability of each class is the value of that class divided by the sum of values for all the classes. Results in a percentage. / model = tf.keras.Sequential([ tf.keras.layers.Flatten (input\_shape=[28, 28]), tf.keras.layers.Dense(300, activation="relu"), tf.keras.layers.Dense(100, activation="relu"), tf.keras.layers.Dense(10, activation="softmax") ])

Transfer learning. Reuse hidden layers. Usually lower ones. Speeds up training and reduces training data needs / learning rate too high will diverge, too low will converge but slowly. Early stopping to avoid overfitting, L2 to constrain weight, L1 many 0s, sparse, Dropout, dropout between 10 and 50%, 20-30 for RNN, 40-50 for CNN, generalizes better, dropout increases number of iterations but each takes less time / batch – adjust weights after all patterns, Online – adjust weights after each pattern, Epoch – a complete pass over all patterns / vanishing gradient – multiplying many small numbers gets to zero / MLP Regressor – output does not use an activation function so it can generate any value

Lecture 7

RNNs take in a sequence rather than a fixed set. Can link one to many (picture to cations), many to one (people to sentiment), or many to many (language translation) / RNNs have a state h sub t that is update at each time step as a sequence is processed. H sub t = f sub w (h sub t-1, x sub t) / same function and parameters are used at each step. Unroll to show progress across time.

Diagram

Description automatically generated

tf.keras.layers.SimpleRNN(rnn\_units) / output at time t is based on the h sub t and inputs at time t. The h is a memory. / can ignore all the outputs except for the last / embeddings convert words to vectors / we need memory from the long past to accurately predict the next word / can add multiple layers of RNNs / BPTT – Back Propagation through time – unroll it and then use normal backpropagation / Vanishing gradient – becomes small causes slow or stalled learning / Unstable gradient – values are unstable making convergence difficult / many causes – bad parameters, bad optimization, bad architecture can use a saturating function such as tanh / may need to monitor the gradients / can use relu to avoid vanishing gradients / can initialize withs to the identity matrix / gated cells such as LSTM or GRU / over time RNN forgets old information, LSTM can address this / Four functions Forget sigmoid, Store sigmoid and tanh, selectively Update combine, and Output / cell stat C sub t carries memory / back propagation from c t to c t-1 requires only elementwise multiplication – no matrix means no vanishing gradient.

GRU – Gated Recurrent Unit is a simplified LSTM / single state vector h sub t / single gate controller / LSTM and GRU cells are key to the success of RNNs

Nucleus Sampling – top k characters, Beam Search – k most promising sentences predict next word – only works well on short sentences / Augmenting model – more GRU layers

Stateless – each iteration the parameters are 0, stateful – maintain the parameters between iterations / After training, the stateful model can only be used for predictions with batches of the same size as those used during training. To alleviate this restriction, an identical stateless model can be created, and the stateful model's weights can be copied to this model.

Diagram

Description automatically generated

Encoder / Decoder for translation / during training feed the decoder the expected output shifted by one word. During use feed the output of the decoder – it will be shifted by one word / bidirectional – run two networks at once – one reading left to right and one reading right to left.

Table

Description automatically generated with medium confidence

Attention paper used 6 layers, GPT uses 12 / BERT / MLM 15% chance of being masked

Autoencoders – bottle neck hidden layer – reconstruction loss – denoising – variational – probabilistic – generative – injects randomness

GAN – Generator creates a fake, discriminator tries to distinguish real from fake

Markov decision process / bellmans equations / q values – sum of discounted future rewards – it is an off policy algorithm – approximate Q learning / agent / actions / environment /observations – discounted ttal reward / balance between exploiting and exploring / credit assignment problem / Deep Q Networks DQN / AverageReturnMetric computes the sum of undiscounted rewards for each episode / discrete vs continuous action spaces / loss = -log P (a sub t | s sub t)R sub t – R sub t is reward at time t