图形用户界面, 应用程序, 表格

描述已自动生成**Course 1: Neural Networks and Deep Learning - Week 1: Supervised Learning with Neural Networks**

**文本, 信件

描述已自动生成**文本

描述已自动生成文本

描述已自动生成文本

描述已自动生成Structure data vs. unstructured data/performance of neural network (amount of labeled data, scale)/what drives deep learning: data, computation, algorithms/vanishing gradient: sigmoid -> ReLU/progress of building: idea (faster)-> code -> experiment (faster)

**文本

描述已自动生成**图示

描述已自动生成**Course 1 Week 2: Binary Classification, Problem**: Binary Classification, **model**: logistic regression, **calculation**: Computation graph: Forward propagation, One step of backward propagation on a computation graph yields derivative of final output variable. /**code:** Vectorization and broadcasting **Course 1 Week 3: Shallow Neural Network** 1. tanh (z) is better for sigmoid (z) for most cases, except for output layer. Reason is mean to 0. 2. use sigmoid for output layer 3. ReLU (rectified linear unit), for normal pick. reason is overcoming gradient descent. 4. leaky ReLU. Faster, 5. Random weight initialization: [0, 0.01], biases = 0 **Course 1 Week 4: Deep Neural Network** Deep vs. fully connected: O(log n) and O(2^n)/Parameter: weights, biases/hyperparameters: num of iterations, num of hidden layers, num of hidden units, activation functions, minibatch size, regularization (determine the value of parameter)

**报纸上的文字

描述已自动生成文本

描述已自动生成文本, 信件

描述已自动生成文本

描述已自动生成报纸上的文字

描述已自动生成文本

描述已自动生成文本, 信件

描述已自动生成文本

描述已自动生成文本, 信件

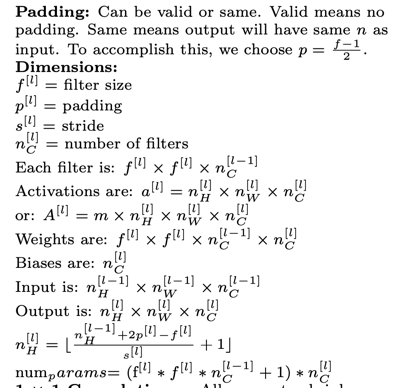
描述已自动生成Course 2: Improving DNN: Hyperparameter Tuning,** **Regularization and** **Optimization, Train/dev/test ratio:** small data (100-10000):60-20-20, large data (1000000): 98-1-1**, Regularization: 1. orthogonalization:** optimize cost function J (gradient descent…) /do not overfitting (regularization…) **2. Drop out reason**: cannot rely on one feature/could be different for different layers (overfitting layer)**, Optimization: 1. Normalization input:** apply (x-u)/(sigma^2) for train, dev, and test (symmetric, smaller gradients, slower learning), **2. Exploding gradients:** W > I (identity matrix), **3. vanishing gradients:** W < I (identity matrix), **method: weight initialization, gradient check (does not work with dropout) Optimization algorithms:** 1. Mini-batch gradient descent: X{t}, Y{t}, J{t}, minibatch size = 64, 128, 256, 512, 1024 power of 2 (Faster learning while maintaining some advantage from vectorization. Doesn’t always converge, but can reduce learning rate when getting close. Fit into CPU/GPU), 2. Batch gradient descent (small training set, <=2000), 3. Stochastic GD. **- Problem of local optima:** 1. more likely to get to the saddle point. 2. Sometimes plateaus can make learning slow when gradient is close to zero for a long time, but small random perturbations can help along with algorithms like momentum, RMSprop or Adam. -**Hyperparameter tuning: *1. Most important to tune is alpha. Then mini-batch size, number of hidden units, and beta. Then number of layers and learning decay rate. We almost never tune Adam optimizer parameters.*** 2. Try random sampling instead of grid, 3. Use coarse to fine sampling scheme. 4.↵ logarithmic scale of 10: a use [10^(-4), 1], beta use 1 – 10^(rand\_num). 5. Babysit (train one model)/caviar (train many models in parallel). **-Batch normalization:** in between z and a (faster). Don’t rely on batch normalization for regularization. It’s for normalization of hidden units, activations, speeding up learning. Train time: for each mini-batch, test time: estimated mean and variance = weighted average across the mini-batches, **Course 3: Structuring ML projects – week 1 why ML strategy, Chain of assumptions in supervised learning systems:** **Steps: *1. Fit train set well on cost function (near human level performance if possible). (knob: bigger network, adam, train longer, NN architecture/parameter search) 2. Fit dev set well on cost function.(knob: regularization, bigger train set, NN architecture/parameter search) 3. Fit test set well on cost function. (knob: bigger dev set) 4. Performs well in real world.*** ***(change dev set/cost function).* Method: 1. dev + single number evaluation metric (faster):**  Precision=TP/(TP+FP), recall=TP/ (TP + FN), F1 score = 2 / [(1/P) + (1/R)] (better to combine these metrics). 2. choosing one optimizing metric+decide that other metrics are satisficing (threshold). 3. If good for metric and not good for application, change metric and/or dev and test sets. **2. Choose Train/Dev/Test Set:** Dev and test sets from the same distribution, define the future and importance, dev set+validation metric define the target. **3. Build your first system quickly then iterate 4. Measure:** human error>model error >= bayes error, (method: improving avoidable bias/bias: get labeled data from humans, manual error analysis, better analyze bias/variance.), variance = dev error – training error. **ML surpass human:** online advertising, product recommendation, logistics, loan approvals/improving, speech recognition, image recognition, medical. **Data mismatch:** train set is different with dev/test. (method: manual error analysis, similar train/dev/test dataset, artificial data synthesis) **Error analysis:** ceiling, percentage, time/goal analysis. **Clean incorrect label data:** percentage, same dev/test distribution. **How to train and test different distribution:** train + dev + test (5:2.5:2.5) **Bias-variance analysis: *1.humanlevel error/train set error avoidable bias: train error – human error/2. Train-dev set(variance = traindev error – training error)/3. dev error(data mismatch=Dev error-traindev error)/4. test error (degree of overfitting to dev set=test error-dev error)/more general analysis (compare two tasks)* Transfer learning** (when: same input x, less data for new task, low level features, small dataset: freeze previous train softmax, medium dataset: train final 1-2 layers, large dataset: train full layers, Hyperparameters are: layers to keep, layers to add, and parameters to freeze or re-tune.) **Multi-task learning** (when: shared lower-level features, amount of data similar, train a big neural network) **End To End Deep learning** (Data vs. hand design, choose x -> y mapping) **Course 4: Convolutional Neural Networks** filter: sobel filter, scharr filter, prewitt filter, learn the filter, cross-correlation: flip the filter, choose other's hyperparameter: 1. nh, nw decreases, 2. nc increases, 3. c-p-c-p-fc-fc-fc-softmax. Three reason: 1. Parameter sharing, 2. Sparse connection, 3. translation invariance (shift of image), example: LeNet-5 (sigmoid/tanh), AlexNet (Relu), VGG-16 (double the filter num), ResNet: a[l+2] = g(z[l+2]+a[l]) by short cut/skip connection, learn identity 文本

描述已自动生成文本

描述已自动生成文本

描述已自动生成文本

低可信度描述已自动生成**报纸上的文字

描述已自动生成**function, train deeper networks, 1 by 1 convolutions (networks in networks)

文本, 信件

描述已自动生成

文本, 信件

描述已自动生成文本, 信件

描述已自动生成Programming summary

文本, 信件

描述已自动生成文本

描述已自动生成文本

描述已自动生成图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成Sum vertically: axis=0, sum horizontally: axis=1/Initialization: np.random.randn(5, 1) (do not use rank1 array)