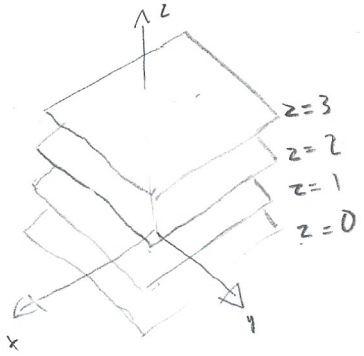
6.036 - Machine Learning

Hollis Holmes.

Planes & Hyperplanes review:



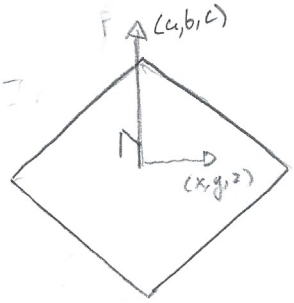
Linear eqn in 3 vars x, y, z is of the form:

$$ax + by + cz = d$$

Examples for intuition:

- equation $z=0$ defines the x-y plane.
- if d is constant, $z=d$ is plane parallel to x-y at $z=d$.
- equation $x+y+z=1$ defines slanted plane in $\mathbb{R}^3$ that goes through points $(1,0,0), (0,1,0), (0,0,1)$

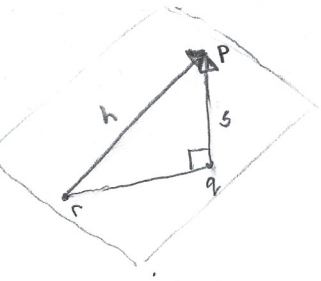
Intercepts can be found by setting other coef's to 0:
ex: z-intercept found → $0x + 0y + cz = d \Rightarrow z = d/c$.



Normal Vectors

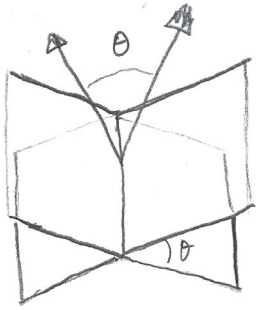
If P is the plane in $\mathbb{R}^3$ defined by $ax + by + cz + d$ then:
$n = (a,b,c)$ is normal to this plane.

Distance From Point to Plane



- There is always 1 point on plane that is closest to the point.
  This point, q, is the projection
- Project the vector h (from any point on the plane, r, to point P) onto the normal vector from the plane.

$$|s| = |h \cdot u|, \text{ where } u \text{ is the unit normal vector}$$



Angle Between 2 planes (dihedral angle)

This is the angle between the plane's normal vectors.
This can be found from a dot product of the two vectors.

Ex $v=(1,0,1)$, $w=(1,1,1)$

$$v \cdot w = |v||w| \cos\theta$$
$$2 = \sqrt{2}\sqrt{3}\cos\theta \Rightarrow \theta = \cos^{-1}\left(\frac{2}{\sqrt{2}\sqrt{3}}\right) \approx 35^\circ$$

Note for data representation & numpy:

- 2D arrays used to represent both matrices and vectors

Machine Learning

coursera

Types:

Supervised learning - gave a data set with "answers", ex data set w/ housing price (answer) for square footage (data point)

Types: regression, classification,

Unsupervised Learning - data has no "right" answer, instead just looks for structure in the data.

Market segmenting,

Models:

Linear regression: fitting predicted curve for real-value output.

Size of house -> h -> price estimate

$h = \theta_0 + \theta_1 x$ $\quad h(x) = \theta_0 + \theta_1 x$

Notation: m = # training examples

(x, y) = (input, output)

$x^{(i)}$ -> i^{th} training example.

Cost Function: how do we pick $h(x)$? create a cost function to quantify how bad the model and minimize it.

squared error example. $J(\theta_0, \theta_1) = \underset{\theta_0, \theta_1}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)}\right)^2$, square diff between output and estimate.

Gradient Descent: Algorithm to minimize the cost function and thereby, solve for $h(x)$.

- walk in direction that has the most negative gradient until getting to minimum.

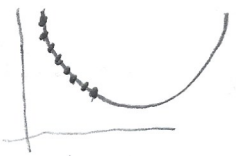
until at min:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j=0 \text{ and } j=1)$$

new = current − α · change

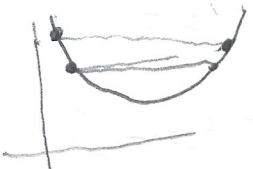
, be careful to update θ_0 and θ_1 simultaneously

what about alpha (α)? Too small:



too long to get there.

Too big!



overshoot or may never converge.

accidentally start at a local min? grad. descent will leave your value unchanged good!

As you approach a min, generally gradient will decrease, and you will automatically take smaller steps.

Week 2

Multiple Features notation $x_j^{(i)}$ j^{th} feature of i^{th} example

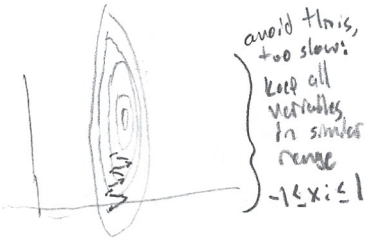
Feature scaling: x_1: size (0 - 2000)

x_2: # bedrooms (1-5)

change to

$x_1 = \text{size}/2000$

$x_2 = \text{bedrooms}/5$.



avoid this, too slow: keep all variables in similar range

$-1 \le x_i \le 1$

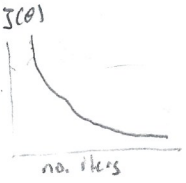
Mean normalization: shift values to be centered about 0 ex: size 0-2000

$x_1 = \frac{\text{size} - 1000}{2000}$

$x_i = \frac{x_i - \mu_i}{s_i}$ (μ_i → mean, s_i → range)

Learning Rate [α, alpha]

to help debug, can plot cost against # iterations



$J(\theta)$ should decrease after every iteration.

Declare convergence if $J(\theta)$ decreases by less than ε on some iter ex: $\varepsilon = 10^{-3}$

, for sufficiently small α, $J(\theta)$ should decrease on every iter.

but if α is too small, grad. descent can be slow to converge.

Polynomial Regression feature scaling becomes very important.

Normal Equation: Alternative to gradient descent, solves for optimal θ

- $\frac{\partial}{\partial \theta_j} J(\theta) = 0 \quad \forall j$, all partial derivatives are 0 at minimum.

$$\theta = (X^T X)^{-1} X^T y$$

Feature scaling not necessary.

Grad. Descent	Normal Eqn
• Works well when n is large.	• No need to choose α • No need to iterate. • Have to compute $(X^T X)^{-1}$ ↳ $n \times n$ $O(n^3)$

$X^T X$ is rarely not invertible
↳ pseudo inverse will still give right answer if not invertible.

Why not invertible, (singular, not full rank row/col)
↳ two features are related.
↳ too many features ($m \leq n$)

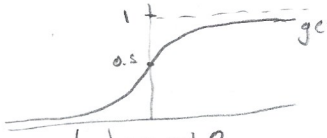
Week 3.

Classification: $\{0, 1\}$ ex: email spam/not spam, tumor malignant/benign.
negative class, positive class

regression isn't right to use here.

Logistic Regression:

$$h_\theta(x) = g(\theta^T x), \quad g(z) = \frac{1}{1+e^{-z}}$$



z centered around 0.
$z=0 \rightarrow g(z) = 1/2$ so 50%
$z \rightarrow \infty, g(z) \rightarrow 1$
$z \rightarrow -\infty, g(z) \rightarrow 0$.

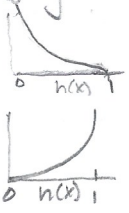
Logistic Regression want $0 \leq h(x) \leq 1$

so: $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ Interpretation: $h_\theta(x)$ = estimated probability $y=1$ for this x.

$h_\theta(x) = P(y=1 \mid x; \theta)$ → probability $y=1$, given x, parameterized by θ.

Cost function: want a convex cost function to guarantee finding a local min

for logistic regression: $Cost(h(x), y) = \begin{cases} -\log(h(x)) & \text{if } y=1 \\ -\log(1-h(x)) & \text{if } y=0 \end{cases}$



$y=1$ and $h(x)=1 \rightarrow$ no co
$y=1$ and $h(x)=0 \rightarrow$ cost to
$y=0$ and $h(x)=0 \rightarrow$ no co
$y=0$ and $h(x)=1 \rightarrow$ cost to

- cost greater than 0
- increases further away you are from output.

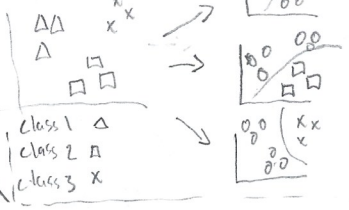
Cost function: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$
↳ sum individual cost for all points.

can be re-written: $J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^{m} \underbrace{y^{(i)} \log h_\theta(x^{(i)})}_{\text{if } y=1} + \underbrace{(1-y^{(i)}) \log(1-h_\theta(x^{(i)}))}_{\text{if } y=0} \right]$

now take derivative for gradient descent:

$$\theta_j = \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

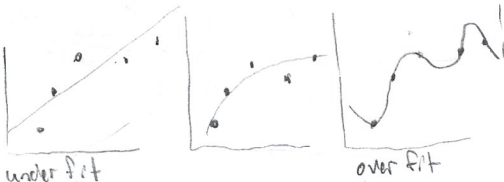
Multiclass Classification: one vs all



Solve a multiclass classification problem by solving k binary classification problems.

class i train logistic classifier $h_\theta^{(i)}(x)$
for each class i to predict prob. $y=i$
then for new x, pick class s.t. $\max_i h_\theta^{(i)}(x)$

Overfitting



- overfit models has a very low cost function but fails to generalize to new examples.

Address overfitting: 1) Reduce # features. 2) Regularization, keep features but reduce magnitude/values of parameters θ_j

Regularization: penalize making parameters θ_j big.

linear regression: $J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \underbrace{\lambda \sum_{j=1}^{n} \theta_j^2}_{\text{regularization}} \right]$

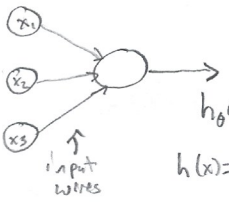
λ too high → underfitting as parameters are too heavily pena

don't penalize θ_0

logistic regression: $J(\theta) = -\left[\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)})) \right] + \lambda/2m \sum_{j=1}^{n} \theta_j^2$

Week 4

Neural Networks

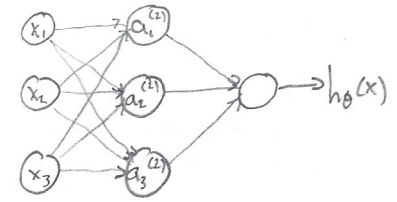


$a_i^{(j)}$ = "activation" of unit i in layer j

$\theta^{(j)}$ = matrix of weights controlling function mapping from layer j to $j+1$

Sigmoid (logistic) activation function.

$$h(x) = \frac{1}{1+e^{-\theta^T x}}$$



Layer 1 — Input Layer; Layer 2 — Hidden Layer; Layer 3 — Output Layer.

Neural Network

neuron

$z_1^{(2)} \Rightarrow a_1^{(2)} = g(z_1^{(2)})$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$\vdots$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$z_3^{(2)}$

} # in next layer

in previous layer + 1

$\Rightarrow \Theta^{(j)} = s_{j+1} \times (s_j + 1)$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

Cost Function (classification problems)

L = # layers, s_k = # units in layer k

$h_\Theta(x) \in \mathbb{R}^K$, k classifier problem

$h_\Theta(x)_i$ = i^{th} output.

$$J(\Theta) = \frac{-1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{n} y_k^{(i)} \log (h_\Theta(x^{(i)}))_k + (1-y_k^{(i)}) \log (1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$$

don't sum over the bias (x_0) terms.

- logistic regression cost over all training examples for the output node
- if multiclass, have to sum over all in the output layer.

Backpropagation

$\delta_j^{(l)}$ = error of node j in layer l.

backprop calculates the direction to move for the gradient descent, similar to what the partial derivatives do

$\delta_j^{(4)} = a_j^{(4)} - y_j$ have error term now work your way back

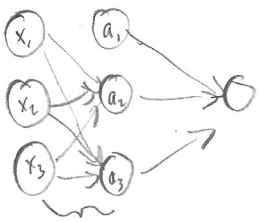
$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$$

Gradient Checking

catch subtle errors in back propagation algorithm. $\frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon}$ as an approximation to check gradient.

Initializing Θ



since everything going from layer 1 -> layer 2 is the same (0)

$\Rightarrow a_1^{(2)} = a_2^{(2)}$ and $\delta_1^{(2)} = \delta_2^{(2)}$ causes weights corresponding to the same input will all stay the same

To combat this initialize all values in Θ as rand values in some range

↳ breaks symmetry (good) also can't set all to the same non-zero value

Picking Architecture

- # of input units = dimension of features $x^{(i)}$
- # of output units = # of classes
- Reasonable default: 1 hidden layer or if > 1 hidden layer have same # of units in each hidden layer
  ↳ usually the more the better.

Training a Neural Network: (may find local optima (non-convex))

1) Randomly initialize weights
2) Implement forward propagation to get $h_\Theta(x^{(i)})$ for all $x^{(i)}$
3) Implement code to compute cost function $J(\Theta)$
4) Implement backprop to compute partial derivatives $\frac{\partial}{\partial \Theta_{jk}^{(l)}}$
5) Use gradient checking to compare its estimate to $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$ From backprop
   ↳ then disable gradient checking code.
6) Use gradient descent or other algo w/ backpropagation to try and minimize $J(\Theta)$ as

Wk 6| Debugging / choosing a good implementation.

Imagine getting unreasonably large errors in your model's predictions

Machine learning diagnostic: test to gain insight into what is/isn't working.

Evaluate A hypothesis predicted by model

split data set into a training set and test set on which trained model will be tested.

↳ Learn θ from training (minimize $J(\theta)$) then compute test set error.

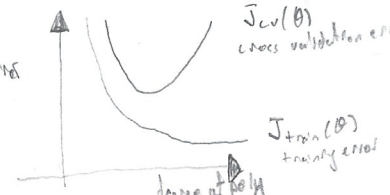
Model Selection: need to pick degree of polynomial, "d" } • Much like how $J(\theta)$ on training set doesn't indicate if θ is a good choice because it was trained on this
• We calibrate d on test set, so we need to test on a different data set to see if it generalizes well

So now we need to configure θ and "d".

So, split data into 3 parts (training, test, cross-validation)
60% 20% 20%

1) Get set of θ's for all degree polynomials running on training set
2) select d by checking error running each degree on cross validation set
3) Test this model on test set

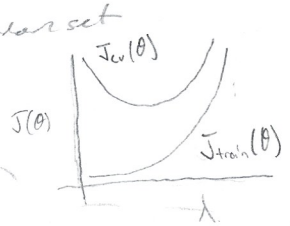
Diagnosing Bias Vs. Variance: if your model doesn't work as good as expected, you often have bias or variance issue



Bias (underfit): $J_{train}(\theta)$ is low, $J_{cv}(\theta) \approx J_{train}(\theta)$

Variance (overfit): $J_{train}(\theta)$ is low, $J_{cv}(\theta) >> J_{train}(\theta)$

Choosing Regularization Parameter (λ), check a bunch of lambda values on cross validation set
↳ then test on test set.



Learning Curves: good for sanity checks or performance of model

High bias: error does not improve w/ more data

High variance: getting more data can help.

Recommended Approach for new problem
1) start w/ very simple model that can be implemented and tested quickly → great for deciding what to work on next.
2) plot learning curves to decide next approach: more data? more features, etc
3) Error analysis: manually examine examples classified incorrectly (in cross validation set) to try and spot a pattern in prob

Skewed Classes: Tumor classification, test set has 99% benign, 1% malignant tumors.
Boasting 99% accuracy isn't good, because this could be accomplished by classifying everything as benign
Need a new metric.

Actual 1	Actual 0
True positive	False Positive
False Negative	True Negative

Predicted 1 / Predicted 0

Precision: $\frac{\text{True Positives}}{\text{Total Positives}}$, of all patients who tested positive, what percentage are actually positive

Recall: $\frac{\text{True Positives}}{\#\text{Actual Positives}}$, of all patients that actually have cancer, what fraction did we correctly detect as having cancer.

Trading off Precision and Recall

Predict 1 if $h_\theta(x) \geq 0.8$ } high precision
Predict 0 if $h_\theta(x) \leq 0.8$ }
tag fewer wrong ones

Predict 1 if $h_\theta(x) \geq 0.3$ } high recall
Predict 0 if $h_\theta(x) \leq 0.3$ }
don't miss any.

	Precision	Recall
Algo 1	0.5	0.4
Algo 2	0.7	0.1
Algo 3	0.02	1.0

How do we evaluate these?
↳ Average? $(P+R)/2$ → BAD
↳ predict 0 or 1 all the time give recall or precision all the time but that is a useless classify

Use the Fscore: $2\frac{PR}{P+R}$
similar to average but gives the lowe of precision and recall a lower w

Data in machine Learning: having more data can often improve model accuracy
ensure model has enough features to solve problem
Good sanity check is whether a human expert could do it with the info given to the model.
good ex: Fill in this sentence: I have ___ to assess. bad ex: housing price w/ only footage^2, no info about area or ammenities.

HW 2 - Perceptron

1.1) a) first a mistake -> [1, -1], data 2 is correct, 3 is wrong so add. t. -> [-0.5, -2]
↳ Now correct -> 2 mistakes.

b) [2, 3]

c) [0, -1] then correct so 1 mistake. d) [1]

1.2) a) $x^{(3)} = [-10, -1]$ $y^{(3)} = 1$

[1, -1] ① , 2 good, 3 wrong [-9, -2] ②, 1 wrong [-8, -3] ③, 2 right, 3 right, 1 wrong [-7, -4] ④,
2 right, 3 right, 1 wrong [-6, -5] ⑤, 2 right, 3 right, 1 wrong [-5, -6] ⑥, all right

-> 6 mistakes

b) [0, -1], all right -> 1 mistake

2) θ is initialized to the first mistake in the data set large θ => large $\|x^{(i)}\|$ => large R.
mistakes is a function of R^2 => more mistakes

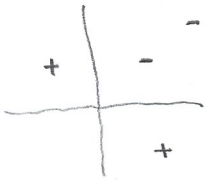
2.2) start $\bar{w}$ a correct answer different than result from 1 => [1, -10]

3) $[2(-3) - 4(-1) - 2(-1) - 2(1), 2(2) - 4(1) - 2(-1) - 1(2)] = [-2, 0]$ $\theta_6 = -5$

4) a) $\begin{cases} 1 & \text{if } (1,1,1) \\ 0 & \text{o.w} \end{cases}$ b) not seperable through origin. c) [1, 1, 1, -2.5]

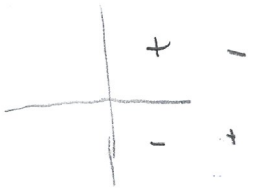
Do the types of classifyers work?
1 circle -> no
2 circle centered anywhere -> yes -> circle around -ve points (1.5, 1.5, 1)
3 line through origin -> no
4 line $\bar{w}$ offset -> (-1, -1, 1)

4.2)



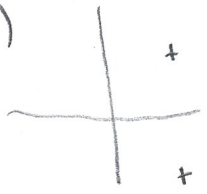
Which are lin-classifyers? -> [3, 4]

5.1)

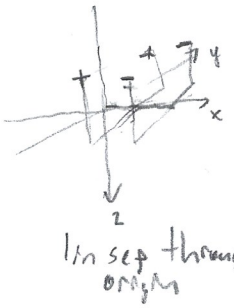


not lin seperable

5.2)



lin seperable. $\bar{w}$ offset



lin seperable through origin

lin sep $\bar{w}$ offset
-> lin sep $\bar{w}$/o offset
add extra dimension to each point using some non-zero #

6) R - bound on element magnitude, a) [1, 0.25] b) [1, 0.5] c) [1, 0] d) [10, 2]
γ - margin of dataset.
Mistakes bounded by $\left(\frac{R}{\gamma}\right)^2$

Rest Coded

Week 3 - Features

Exercises.

1) One-hot rep

lin sep $\bar{w}$: $|x|$, x^4, x^{2k}

2)

3)

$x^2 - 0.25 > 0$

$|x| > 0.5$

-0.5, 0.5 both on the classifier.

Homework

1A) done in code 1B) $\left(\frac{R}{\gamma}\right)^2 \leq 8,910,670$ E) $\left(\frac{1,2806}{0,00005}\right)^2 = 18,222,220 \Rightarrow$ more mistakes

F) $\left(\frac{R}{\gamma}\right)^2 \rightarrow R = 1,51 \Rightarrow \left(\frac{R}{0}\right)^2 \simeq 32$

4.2B) The most impactfull paramaters are cylinders (one-hot) and weight (standard). If two paramaters could yield similar accuracy it would be these 2.

5.2) Reviews analysis. Need the 10 words with highest positive influence

→ Word that has highest value in θ

We need indices of θ that have highest value → np.argmax()

5.2C) Found the highest and lowest reviews. (optional) What is interesting is that they were very long. I think this is important. Though the sentiment of these reviews is generally positive/negative. the length of the reviews allow the margin to compile resulting in an extremely positive/negative review. It may be an improvement to account for the length of the review to get a better idea of something like the "density" of positives/negatives in the review. In this case I believe it is also important to remove stop-words, because these will unfairly decrease the magnitude of the sentiment in these longer reviews.

I later filtered the reviews for ones less than 500 characters long and got much more extreme reviews. This leads me to believe that this 'density' property is important.

6.2F) Having a balanced data set is very important. For instance you could guess with >90% accuracy whether someone has cancer simply by guessing no for every person on Earth. However, this does not mean its a good test. The MINST data set is balanced.

Week 4 - Margin Maximization

Execises: $\theta = (1, 1, -4)$: $(3,2) \to \frac{1}{\|\theta\|}$, $(1,1) \to \frac{2}{\|\theta\|}$, $(4,2) \to \frac{-2}{\|\theta\|}$

→ Note θ_0 not included in $\|\theta\|$

Lab - Gradient Descent

$x^{(i)} = x^{(0)} - \eta \nabla_x f(x^{(0)})$ $\quad f(x) = (2x+3)^2$

local min $f'(x) = 0$ $\quad f'(x) = 2(2x+3)\cdot 2 \Rightarrow 2x+3 = 0 \Rightarrow x = \frac{-3}{2}$

1D) $f'(x) = 8x + 12$

$x^{(k+1)} = x^{(k)} - \eta(8x+12)$

written in these terms

$$\Leftarrow \begin{cases} x^{(k+1)} + 3/2 = x^{(k)} - 8x^{(k)}\eta + \eta(12) + 3/2 \\ = x^{(k)}(1-8\eta) + (\frac{3}{2} - 12\eta) \\ x^{(k+1)} + 3/2 = (1-8\eta)(x + \frac{3}{2}) \end{cases}$$

$z^{(k+1)} = \alpha z^{(k)}$

where $z = x + 3/2$ and $\alpha = (1-8\eta)$

→ This tells us some interesting stuff about the starting point of our gradient descent

$|\alpha| > 1 \to \alpha < -1$ or $\alpha > 1$ diverges; $\eta < 0$

this makes sense because we would be moving opposite the dir. calculated if we have -ve η

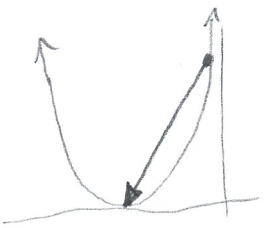
$\alpha = 1 \Rightarrow$ no gradient descent

this makes sense no steps taken because $\eta = 0$

$\alpha = -1 \Rightarrow$ oscillates endlessly between 2 nums

$1 > \alpha \geq 0$ converges w/o oscillation

$-1 < \alpha < 0$ converges w oscillation.



solves

Cool! The maximum step size that converges w/o oscillation gradient descent in 1 step.

1G) $\eta = 0.1 \Rightarrow \alpha = 1 - 0.8 = 0.2 \Rightarrow$ converges w/o oscillation,

1H) $\eta = 0.1 \Rightarrow \alpha = 0.2$ ←

$\eta = 0.11 \Rightarrow \alpha = 0.12$

$\eta = 0.12 \Rightarrow \alpha = 0.04$ } converge fast

$\eta = 0.13 \Rightarrow \alpha = -0.04$

$\eta = 0.14 \Rightarrow \alpha = -0.12$

$\eta = 0.15 \Rightarrow \alpha = -0.2$ ← converge slow

2A) n friends at location l_i, party location p

$$\mathcal{L}(p) = \sum_{i=1}^{n} (l_i - p)^2$$

$$\mathcal{L}'(p) = \sum_{i=1}^{n} 2(l_i - p)$$

set to 0

$$\to 0 = \sum_{i=1}^{n} l_i - np$$

$$\to p = \frac{1}{n} \sum_{i=1}^{n} l_i$$

- minimum loss when p is placed in the average of all house locations.
- linear equation → has 1 solution ↳ no local optimas.

4BS) hinge loss

↳ data isn't seperable but still want to encourage margins.

$$\mathcal{L}_h\left(\frac{\gamma(x,y,\theta,\theta_0)}{\gamma_{ref}}\right) = \begin{cases} 1 - (\gamma(x,y,\theta,\theta_0)/\gamma_{ref}) & \text{if } \gamma \leq \gamma_{ref} \\ 0 & \text{else.} \end{cases}$$

Loss is 0 if $\gamma > \gamma_{ref}$ otherwise it is proportional to γ/γ_{ref}

Adding regularizer and letting $\gamma_{ref} = \frac{1}{\|\theta\|}$ we get objective function.

$$J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} L_h\left(y^{(i)}(\theta^T x^{(i)} + \theta_0)\right) + \lambda \|\theta\|^2$$

writing the training objective as an average:

$$\frac{1}{n} \sum_{i=1}^{n} \left[L_h(y^{(i)} \theta \cdot x^{(i)}) + \frac{\lambda}{2} \|\theta\|^2 \right]$$

where $L_h(y\theta x) = \max\{0, 1-y$

5A) considering one point where hinge loss is positive this becomes: $J'_\lambda(\theta) = 1 - y(\theta \cdot x) + \frac{\lambda}{2}\|\theta\|^2$

derive wrt θ and set to 0 for $\hat{\theta} \to \frac{\partial J}{\partial \theta} = -yx + \lambda\|\theta\| = 0 \Rightarrow \hat{\theta} = \frac{yx}{\lambda}$

Week 4 continued...

5B We have: $\hat{\theta} = \frac{yx}{\lambda}$ when hinge loss is non-zero.

general objective function for 1 point is: $J'_\lambda(\theta) = [1 - y(\theta \cdot x)] + \frac{\lambda}{2}\|\theta\|^2$ ($L_n(y(\theta,x))$)

smallest $\hat{\theta}$ for which $L_n(y(\theta \cdot x)) = 0$

$y(\hat{\theta} x) = 1 \Rightarrow \hat{\theta} \cdot x = \frac{1}{y}$, $\hat{\theta}^T x = \frac{1}{y}$ $\Rightarrow$ $\hat{\theta}^T x x^T = \frac{x^T}{y}$

$\hat{\theta}^T = \frac{x^T}{\|x\|^2 y} \Rightarrow \hat{\theta} = \frac{x}{\|x\|^2 y}$

5C) $\hat{\theta} = \hat{\theta}(\lambda)$ minimizes $J'_\lambda(\theta)$

$y(\hat{\theta} \cdot x) = y\left(\frac{yx}{\lambda}\right) \cdot x = \frac{\text{norm}(x)^2}{\lambda}$ → This term is always positive → can't be misclassified.

5D) single training element on the boundary, hence $1 - y(\theta x) = 0$

point needs to be on the boundary

① hence: $y \cdot (\theta \cdot x) = 1$

we also need θ that minimizes our objective function, we solved for this $\hat{\theta}$

② hence: $\theta = \hat{\theta} = \frac{yx}{\lambda}$

② → ① $y\left[\left(\frac{yx}{\lambda}\right) \cdot x\right] = 1$

$y \cdot y = 1$ $\quad x \cdot x = \lambda \Rightarrow \lambda = \|x\|^2$.

7.2) SVM Gradient

$J(\theta, \theta_0) = \frac{1}{n}\left[\sum_{i=1}^{n} \begin{cases} 1 - y(\theta x + \theta_0) \\ 0 \end{cases} + \lambda\|\theta\|^2\right]$

$\frac{\partial J(\theta,\theta_0)}{\partial \theta \theta_0} = \begin{bmatrix} \frac{1}{n}(-yx + 2\lambda\theta) \\ \frac{1}{n}(-y) \end{bmatrix}$

Week 5 - Regression Hwk, written work

1)

$$J(\theta,\theta_0) = \frac{1}{n}\sum_{i=1}^{n} L(x^{(i)}, y^{(i)}, \theta, \theta_0) + \lambda R(\theta,\theta_0)$$

$$L_s(x^{(i)}, y^{(i)}, \theta, \theta_0) = (\theta^T x^{(i)} + \theta_0 - y^{(i)})^2$$

$$\frac{dL_s}{d\theta} = 2(\theta^T x^{(i)} + \theta_0 - y^{(i)}) \cdot x^{(i)}, \quad \frac{dL_s}{d\theta_0} = 2(\theta^T x^{(i)} + \theta_0 - y^{(i)})$$

That was for individual points, now using X, Y which are the whole data sets.

X: dxn, Y: $1xn$, θ: dxn

$$J(X,Y,\theta,\theta_0) = (\underbrace{\theta^T X}_{1xd\ \ dxn} + \theta_0 - Y)(\theta^T X + \theta_0 - Y)^T$$

$1xn$ $\quad$ $1xn$ $\quad$ $1xn$

$1xn$ $\quad$ $1x1$

need to square & sum

$$\frac{1}{n}\|(\theta^T X + \theta_0 - Y)\|^2$$

$$\frac{2}{n}(\underbrace{\theta^T X + \theta_0 - Y}_{1xn}) \cdot \underbrace{X^T}_{nxd}$$

this is $1xd$ -> need $dx1$

so: transpose the whole thing

$$\frac{dJ}{d\theta} = \underbrace{X}_{dxn}(\underbrace{\theta^T X + \theta_0 - Y}_{nx1})^T + X^T(\underbrace{\theta^T X + \theta_0 - Y}_{1xn})$$

$dx1$

2) Sources of error

1) Structural: hypothesis class cannot represent the data (data too good)
2) Estimation: not enough data to accurately build a hypothesis. (model too good)

3) $0 = \frac{2}{n} \cdot Z^T \cdot (Z\theta - T)$

$$0 = Z^T Z\theta - Z^T T$$

$$Z^T Z\theta = Z^T T$$

$$\theta = (Z^T Z)^{-1} Z^T T$$

$$\text{is } (XX^T)^{-1} XY^T$$

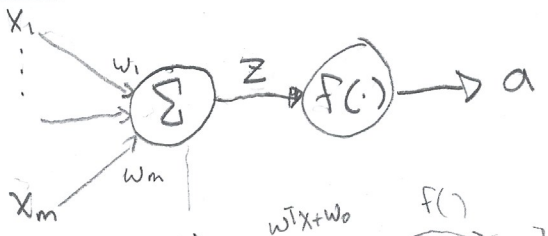
5)
- minimizing over
- dataset
- predictor used
- regularization?

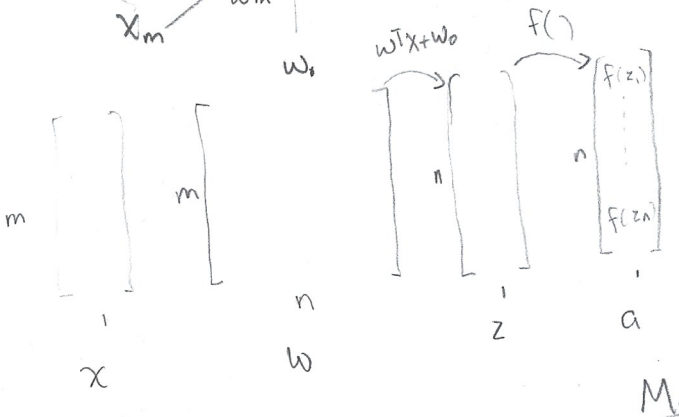
Testing a data set don't need a regularizer, that is only for training a classifier.

Week 6 - Neural Networks

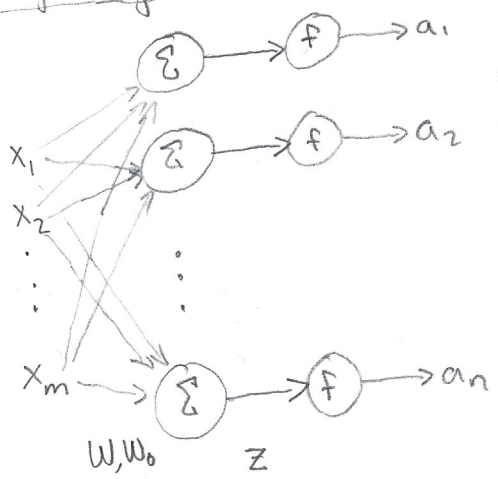
Basic Element

$$a = f(z) = f(w^T x + x_0)$$

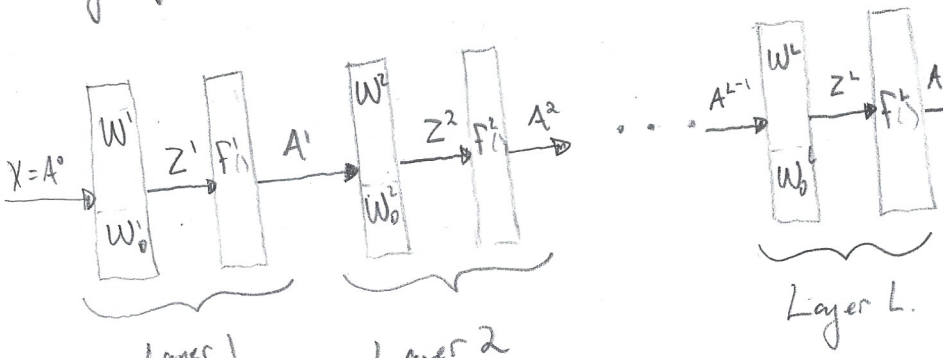




Single Layer



Many Layers



Back Propagation

In a neural net, the parameters are w, so that is always what we need to calculate how much we need to change, and what we are calculating the gradient w.r.t. i.e: $\frac{\partial Loss}{\partial w}$. How much we need to change weights in order to reduce the loss incurred on this particular example.

First, looking at how loss depends on the final layer, using chain rule:

$$\frac{\partial Loss}{\partial W^L} = \underbrace{\frac{\partial Loss}{\partial A^L}}_{\text{derivative of Loss function}} \cdot \underbrace{\frac{\partial A^L}{\partial Z^L}}_{f^{L}(\cdot)} \cdot \underbrace{\frac{\partial Z^L}{\partial W^L}}_{\substack{Z^L = (W^L)^T X^L \\ \Rightarrow \frac{\partial Z}{\partial W} = X^L = A^{L-1}}} \Rightarrow \frac{\partial Loss}{\partial W^L} = A^{L-1}\left(\frac{\partial Loss}{\partial Z^L}\right)^T$$

Week 6 - Homework

1.1) $z = w^T x$

$a = z$

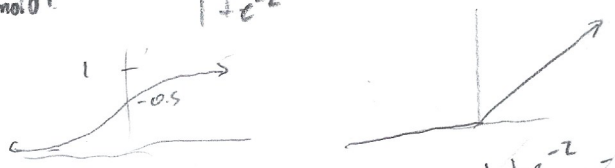
$\mathcal{L} = \begin{cases} 0 & \text{if } y a > 1 \\ 1 - y a & \text{else} \end{cases}$

$\mathcal{L} = \begin{cases} 0 & \text{if } w^T x(y) > 1 \\ 1 - y(w^T x) & \text{ow} \end{cases}$ $\quad \frac{d\mathcal{L}}{dw} = \begin{cases} 0 & w^T x(y) > \\ -y(x) & \text{ow} \end{cases}$

1.2) Log loss

Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$ Relu: $\sigma(z) = \max(0, z)$

$\frac{\partial \text{sigmoid}}{\partial z} = \frac{e^{-z}}{(1+e^{-z})^2}$



b) $1 - \frac{1}{1+e^{-z}} = \frac{1+e^{-z}-1}{1+e^{-z}} = \frac{e^{-z}}{1+e^{-z}}$ $\quad \frac{\partial(sig)}{\partial z} = sig(1-sig)$

D) for a single point. $NLL(a, y) = y \cdot \log(a) + (1-y) \cdot \log(1-a)$

$\frac{\partial NLL}{\partial w} = \frac{\partial NLL}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial w} =$

① ② ③

$NLL = -y \log(a) - (1-y) \log(1-a)$

$\frac{\partial NLL}{\partial w} = -y \frac{a(1-a)}{a} x_1 + (1-y) \frac{a(1-a)}{1-a} x_1$

$= -y(1-a) x_1 + (1-y) a x_1$

$= [-y + ay + a - ay] x_1$

$= (a - y) x_1$

E) vectorize.

2) Multiclass classification. $a = SM(z)$

Softmax takes n_L pre-activation values (z^L) returns a vector with each components probability where $\sum_j a_j^L = 1$, prob dist. over the categories

$a_j = \frac{e^{z_j}}{\sum_{k=1}^{n^L} e^{z_k}}$ (your $e^{\text{pre-activation}}$ / sum(all $e^{\text{pre-activation}}$))

a) $SM([-1, 0, 1]) = \frac{[e^{-1} \; e^0 \; e^1]}{e^{-1} + e^0 + e^1}$ $\quad [0.09, 0.24, 0.66]$

b) $NLLM = -\sum_{j=1}^{n^L} y_i \ln a_j^L$

sum along the length of the vectors multiplying matching components only 1 should be there though

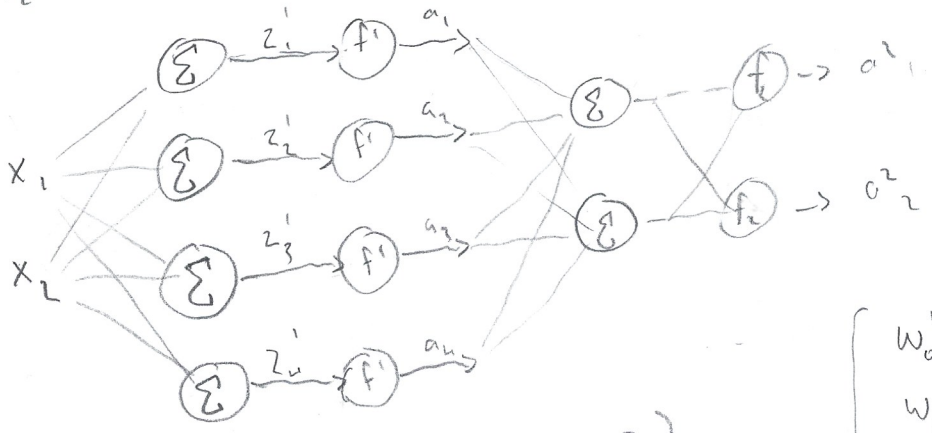
$\frac{\partial NLLM}{\partial w_{kj}^L} = x_k (a_j^L - y_i)^T$

HW6 continued...

3) Neural Networks

f_1 activation - ReLU = max(0, z)

f_2 activation - Softmax



$x_1 = 3$

$x_2 = 14.$

$$\begin{bmatrix} w^1_{11} & w^1_{12} & w^1_{13} & w^1_{14} \\ w^1_{21} & w^1_{22} & w^1_{23} & w^1_{24} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} w^1_{01} \\ w^1_{02} \\ w^1_{03} \\ w^1_{04} \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} w^2_{11} & w^2_{12} \\ w^2_{21} & w_{22} \\ w^2_{31} & w_{32} \\ w^2_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}, \quad \begin{bmatrix} w^2_{01} \\ w^2_{02} \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

$$z^1 = \begin{bmatrix} z^1_1 \\ z^1_2 \\ z^1_3 \\ z^1_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 14 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ 13 \\ -4 \\ -15 \end{bmatrix} \xrightarrow{\max(0,z)} a = f^1(z^1) = \begin{bmatrix} 2 \\ 13 \\ 0 \\ 0 \end{bmatrix}$$

$$w^{1T} x + w^1_0$$

$$z^2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 13 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 15 \\ -13 \end{bmatrix}$$

$2 \times 4 \quad 4 \times 1$

$a_2 = \text{softmax}$

$$a = \begin{bmatrix} \frac{e^{15}}{e^{15}+e^{-13}} \\ \frac{e^{-13}}{e^{15}+e^{-13}} \end{bmatrix} \simeq \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Week 6 Exercise Rough Work

Exercises

$$X = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 1 & 2 \end{bmatrix}$$

$d \times m$ (2 × 3)

$n \times m$ (2 × 3)

W (1 × 2) ← 1 layer, 2 mapping to.

↑ 2 parameters coming from

W^T extended X

$$\begin{bmatrix} 1 & -0.5 \\ 0 & 1.5 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.5 & 1.5 \\ 1.5 & 0.5 & -0.5 \end{bmatrix}$$

$$\rightarrow f(z) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad y \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$ } AND Truth table.

2) $z = w_{1,1}' x_1 + w_{2,1}' x_2 + w_{0,1}'$ $\quad y = -1$

$= 1(1) + 1(2) + 1$

$= 4$

$f(z) = 1 \neq y$

↳ 2c.

$z = w_1 x_1 + w_2 x_2 + w_0$

$= 0.5(1) + 0(1) + 0.5$

$= 1$

$\frac{\partial L}{\partial a} = 1 \quad \frac{\partial a}{\partial z} = 1$

$\frac{\partial L}{\partial w_1} = [1 \ 2 \ 1]$

$w = [0.5, 0, 0.5] - 0.5[1, 2, 1]$

$= [0, -1, 0]$

new output -2

new loss $\max(0, 1 - (-1)(-2))$

$= 0$

↳ no new updates

Training $L_h(v) = \max(0, 1-v)$

$\frac{\partial Loss}{\partial w} = \frac{\partial L}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial w}$

$Loss = 1 - v = 3 \quad \frac{\partial (1 - ya)}{\partial a} = -ya$

1) $\frac{\partial Loss}{\partial a} = 1$

2) $a = f(z) = \quad a = z \Rightarrow \boxed{\frac{\partial a}{\partial z} = 1}$

3) $z = w^T x$

$= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \frac{\partial z}{\partial w} = x$

$\boxed{\frac{\partial z}{\partial w_1} = 1 \quad \frac{\partial z}{\partial w_2} = 2 \quad \frac{\partial z}{\partial w_0} = 1}$

$\frac{\partial Loss}{\partial w_1} = (-1)(1)(1) = -1 \quad \frac{\partial Loss}{\partial w_2} = (-1)(1)(2) = -2$

$\frac{\partial Loss}{\partial w_0} = -1$

$w = [1, 1, 1] - 0.5[1, 1, 2] = [0.5, 0.5, 0$

Week 6 Lab- Rough Work

1.1 A) $z = w^T x$

$a = \frac{1}{1+e^{-z}}$

Ans: Linear

Seperator is $a = 0.5$

$0.5 = \frac{1}{1+e^{-z}}$

$0.5(1+e^{-z}) = 1$

$e^{-z} = 1 \Rightarrow z = 0$

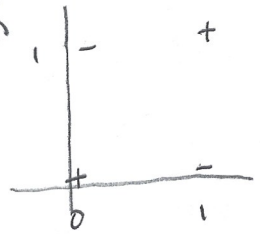
$0 = w^T x$

↳ This is a linear system of equations, hence the operator is linear.

B) yes, increasing the iters by a factor of 10 yields consistent results.

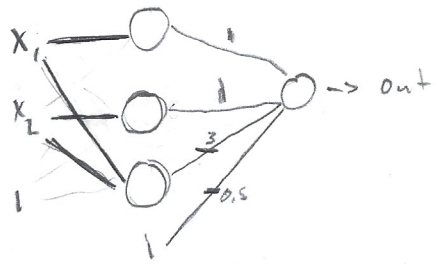
1.2) Not XOR

a)



No, the network above would not work as its hypothesis class is linear functions, but this data-set is not linearly seperable.

b)



A layer w/ 3 hidden units can be used to build a classifier as described below.

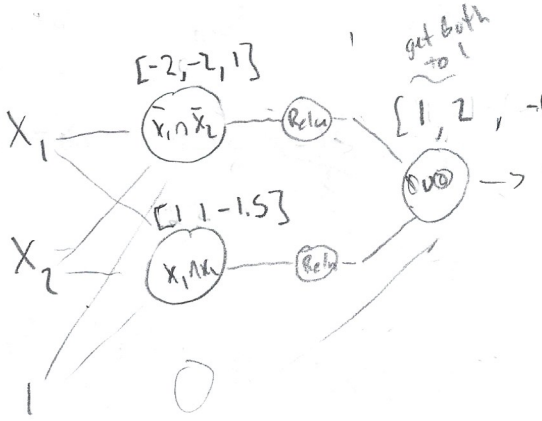
top unit represents if x_1 is a one $\begin{bmatrix}1\\0\\0\end{bmatrix}$

middle represents if x_2 is a one $\begin{bmatrix}0\\1\\0\end{bmatrix}$

bottom represents if x_1 and x_2 are one $\begin{bmatrix}1\\1\\-1.5\end{bmatrix}$

$\begin{bmatrix}1\\1\\-3\end{bmatrix}$ the weights coming out of these require at least 1 of the top 2 to be 1 or the top 2 to be on, but is <0 if both are on with a large negative

$\begin{bmatrix}-0.5\end{bmatrix}$



$y = (\bar{x}_1 \cap \bar{x}_2) \wedge (x_1 \wedge x_2)$

represent each of these in the hidden layer, then and in the last.

WAIT We can do Better!

$y = (\bar{x}_1 \cap \bar{x}_2) \cup (x_1 \cap x_2)$

Now only 2 hidden units (and the x_0) are needed!

Week 6 Lab cntd'...

1.3) Hard data set.
I was able to get a reliable 95% accuracy with parameters hu = 18, iters = 5000, lr = 0
However, this value set was tough to find because of the small how close some
of the +ve and -ve points are.

b) No I don't think its a good idea to find a perfect separator. By only maximizing
training accuracy you will likely over-fit the data and the model will perfo
poorly on other data sets.

2) #(1) output units, activation(2) function on output units, and loss function(3)
↳ should all be chosen jointly.

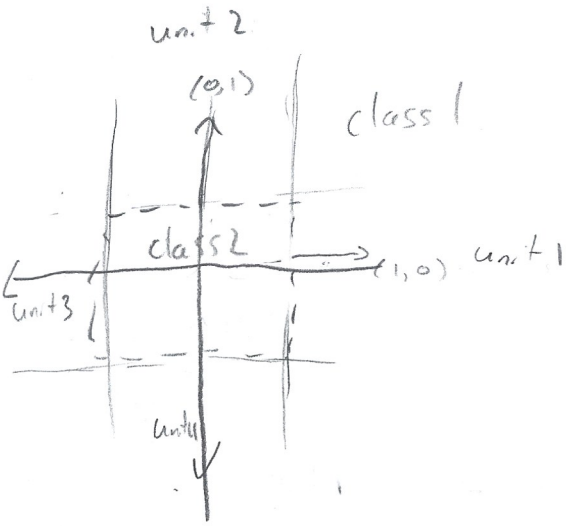
A) Words on front page NY Times → change in stock market.
(1) one output unit, magnitude or percent change in price.
(2) ReLU - bunch of parameters, not probabilistic
(3) squared loss, care about the magnitude difference between prediction and o.

B) Satellite image → probability it will rain in next 4 hours.
(1) one output unit, probability
(2) sigmoid, map to a probability, binary classification.
(3) Neg. log. likelyhood - maximize probability over the data set.

C) Words in an email → folder it should go in.
(1) # of folders that can be chosen, one-hot classification.
(2) softmax - multiclass classification
(3) NLL - maximize probability over the whole data set

D) Words of a document → vector of topics, each index is 1 if topic is adressed
↳ not strictly classification because vector can have multiple 1's
If there are T topics, this seems like T-copies of a binary classifica
since each topic is either mentioned or not-mentioned
(1)
(2)
(3)

HW6 cont'd...



3.2) Unit decision boundaries.

$[1,0]\,[-1]$ unit 1 $x_1 - 1 = 0$

$[0,1]\,[-1]$ unit 2 $x_2 - 1 = 0$

$[-1,0]\,[-1]$ unit 3 $-x_1 - 1 = 0$

$[0,-1]\,[-1]$ unit 4 $-x_2 - 1 = 0$

The intuition here is that each row in W^1 is its own linear separator.

Now outputs of the whole system, the classifiers.

Class 2

$f^1(z_1^1) + f^1(z_2^1) + f^1(z_3^1) + f^1(z_4^1) = 0$

$z_1^2 = f(z_1^1) + f(z_2^1) + f(z_3^1) + f(z_4^1) + 0 = 0$

$z_2^2 = -(\quad\quad\quad\quad\quad\quad) + 2 = 2$

$a_1^2 = \frac{e^0}{e^0+e^2} =$

$a_2^2 = \frac{e^2}{e^0+e^2} =$

In this case ReLU's are all 0. So point is in none of the unit zones → it is in the middle 1x1

Boundary

$f^1(z_1^1) + f^1(z_2^1) + f^1(z_3^1) + f^1(z_4^1) = 1$

$z_1^2 = 1 + 0 = 1$ $\quad a_1^2 = \frac{e^1}{e^1+e^1} = \frac{1}{2}$

$z_2^2 = -1 + 2 = 1$ $\quad a_2^2 = \frac{e^1}{e^1+e^1} = \frac{1}{2}$

over one of the boundaries by or just over 2 of them sum to 1.

on the boundary, equal prob of both classes

Class 1

$f^1(z_1^1) + f^1(z_2^1) + f^1(z_3^1) + f^1(z_4^1) = 3$

$z_1^2 = 3 + 0 = 3$ $\quad a_1^2 = \frac{e^3}{e^3+e^1} \simeq 0.98$

$z_1^2 = -3 + 2 = -1$ $\quad a_2^2 = \frac{e^{-1}}{e^3+e^{-1}} \simeq 0.02$

Way out in one of the zones, far away from the origin. $\bar{w}$ high probab

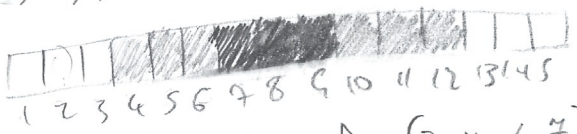
class 1 seems to just be are you outside of the 1x1.

class 2 is are you pretty close to the origin.

↳ cool how a point can stay station but with only magnitudes of weights changing probability/confidence of model prediction changes.

Lab 8 - Convolutional NN's

2A) $x = [0,0,0,2,2,2,4,4,4,2,2,2,0,0,0]$



as greg scale

2B $x_3 - x_1 > 1$ $(-1, 0, 1, -1)$ true for: $[3, 4, 6, 7]$

$x_1 - x_3 > 1$ $(1, 0, -1, -1)$ $[7, 8, 10, 11]$

3A) $f_1 \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ - detects vertical lines

$f_2 \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$ - detects horizontal lines.

$f_3 \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ - detects standout pixels that are brighter than its surroundings.

Decreasing the 6. (making it more... means the feature being detected needs to be stronger for it to be detected

D) changing coef's on the array other than bias.

if bias is 0: this has no effect

if bias is non-0: increasing coefficient reduces the affect of bias

decreasing coef increases " " " "

ex setting coef to 0 → all pixels are just determined by bias.

E) series of 0's is an object we are detecting. 0 1 0 | 0 1 0 | 0 0 1 0 1 1 0 ...

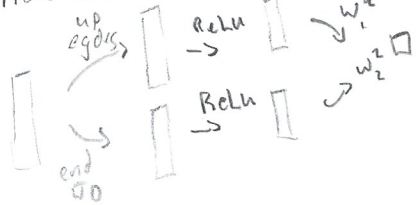
how do we detect this?

every change up 0->1 means you came from a 0. $(-1, 1)$, - 0.5

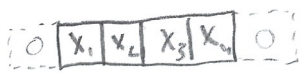
you then need to check if it ends in a 0. $(-1, -1, -1, \ldots, -k)$ 0.5

↳ positive unless there is a 1 anywhere.

now sum these in fully connected network w/ bias 0.



$w_1^2 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$ $w_2^2 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$ these will both just sum. exp edges + ends w/ 0.

4A) 

Notice how calculations 'slide' and are reused multiple times in the matrix. This is parameter

$6X_1 + 1X_2 - 0X_3 + 0X_4 = z_1$

$5X_1 + 6X_2 + 1X_3 + 0X_4 = z_2$

$0X_1 + 5X_2 + 6X_3 + 1X_4 = z_3$

$0X_1 + 0X_2 + 5X_3 + 6X_4 = z_4$

4B) $\frac{d - (k-1) + 2 \cdot p}{s}$

Week 9 Lab

1) State space: {dirty, clean, painted}
Action space: {wash, paint, eject}

$$T(s_t, \text{wash}, s_{t+1}) = \begin{bmatrix} 0.1 & 0.9 & 0 \\ 0.1 & 0.9 & 0 \\ 0.1 & 0.9 & 0 \end{bmatrix}, \quad T(s_t, \text{paint}, s_{t+1}) = \begin{bmatrix} 1 & 0 & 0 \\ 0.1 & 0.1 & 0.8 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R(s,a) = \begin{cases} 10 & \text{if } (s = \text{painted}, a = \text{eject}) \\ 0 & \text{if } (s \neq \text{painted}, a = \text{eject}) \\ -3 & \text{if } (a \neq \text{eject}) \end{cases}$$

b) if dirty → wash, if clean → paint, if painted → eject

c) now you only 'survive' with probability 0.1, so rewards now are heavily valued.
All plates start dirty so cleaning and painting earns:

$$-(3) - (0.1)(3) + (0.1)^2 10 = -3.2.$$

→ Doesn't even make sense to paint plates, just eject immediately for 0 reward.

d) If horizon is 2 can never eject a clean plate so just eject the dirty ones.

$$\begin{bmatrix} & * \\ \$ & \end{bmatrix}$$

2) A) $ is 100, * is 50

B) states find each reward square. $ is worth more given its higher one-time reward, and states haven't had a chance to reclaim * multiple times.

c) States that were pointing towards * have now been able to reach $ but not able to get back to * twice. So the π^* for these squares is shifting back towards $.

d) These states are now claiming * multiple times and π^* shifting to going towards *

e) Almost all values have been able to claim * multiple times.

f) All values will point towards * and their values will keep climbing.

3)
$$\left.\begin{aligned} y_t &= (1) y_{t-1} - (2) y_{t-2} + (3) y_{t-3} \\ s_t &= f_1(W^{ss} s_{t-1} + W^{sx} x_t + W_0^{ss}) \\ y_t &= f_2(W^{o} s_t + W_0^{o}) \end{aligned}\right\}$$

Week 10 Lab

2A) With Q-learning, the 'learning' doesn't start until the robots first hits the target square. Until then, all squares have 0 value (depending on implementation). With value iteration, target states are given a value on the first iteration and the values propegate outwards.

B) The robot has not come across the target-state yet. So, the value of all values is 0.

C) One square is yellow because the goal state has been found. After it has been found, robots position is re-initialized to a random point on the grid. No more updates are made until the robot stumbles upon a square adjacent to the target square.

d) Once there are a few squares w values, the robot stumbles on these positions more and more quickly. This in turn, allows q-values to propagate out more quickly.

f) Choosing a random action instead of the first in case of a tie.

g) Directions are random and value of all squares are plummet Because each update the value gets reduced in the SSP mode

h) As opposed to target-oriented, almost all values on the grid has valu changing very often. This is because in goal-oriented only 1 square has a reward, but in SSP, all but 1 have a reward.

I) I think this depends on your use. Goal-oriented can take a very long time to get started, but ended with a more complete sol than SSP in this exampl

3) This is a tradeoff between exploration vs. exploitation.

Exploitation acts exactly in accordance with the current Q-table. However, you could imagine that if you haven't spent enough time exploring you will concentrate on a poor solution. Conversely, at some point you would like to shift your efforts towards focusing on areas you found to be important.

Epsilon search: • random choice w/ probability ε (exploration)
• best according to current Q w/ prob $(1-\varepsilon)$ (exploi

With $\varepsilon = 0$: purely exploitation, risks never finding the optimal policy. Puts too much faith in results w/ very little data

With $\varepsilon = 1$: purely exploration will find a good policy but never use it because it chooses at random always.

$\varepsilon = 0.5$: a mix of both, some exploration, some exploitation.

None of these guarantee optimal behaviour during learning.