

CS440/ECE448 Lecture 14: Exam 1 Review

Mark Hasegawa-Johnson, 2/2022

Slides may be redistributed under [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/).

Exam 1 Review

- Mechanics: How to take the exam
- Review topics
- Sample review problem

Mechanics: How to take the exam

- No calculators, computers, or textbooks
- You may bring one 8.5x11 page of notes, handwritten, front & back

If you're taking the exam in person:

- Show up to class on Monday, ready to take an exam.

Mechanics: How to take the exam

If you are taking it online:

- You will receive a zoom URL by e-mail by Sunday night. If you do not receive it by Sunday night, please let us know.
- Log on to that zoom URL before 1:00pm on Monday.
- Turn on your webcam. Your webcam needs to show your hands, face, and workspace.
- At 1:00pm, if everybody has their webcam on, the TA will post the exam PDF in the zoom chat window.
- You can write your answers electronically or on paper.
- At exactly 1:50pm, the TA will tell you to stop working, and to start photographing your answers, and uploading them to Gradescope.

How to study: Recommended approach

1. Review the quiz problems and MPs: for each quiz, there is now a 0-point “practice quiz” that you can re-take as often as you like.
2. Review the lecture slides – try to extract a few key bullet points or equations from each lecture.
3. Do the sample exam problems. Do them yourself, then compare your answer with your friend’s answer, then check the solutions.
4. Write your cheat sheet (8.5x11, both sides if you wish).

Exam 1 Review

- Mechanics: How to take the exam
- Review topics
- Sample review problem

Topics on the exam

Broadly, there are just two topics:

- Probability
- Machine learning

Topics on the exam

Narrowing down a little bit:

- Probability (lecture 2: definition)
 - Lecture 2: conditional probability, axioms of probability, independence
 - Lecture 3: Bayesian classifier, naïve Bayes
 - Lecture 4: recall & precision, train, dev, & test corpora, Laplace smoothing
- Machine learning (lecture 5: definition)
 - Lecture 6: perceptron
 - Lecture 7: linear regression
 - Lecture 8: logistic regression
 - Lectures 9-11: multilayer networks, back-propagation
 - Lecture 12: computer vision (pinhole camera equations, convolution)

The axioms of probability

Axiom 1: every event has a non-negative probability.

$$P(A = T) \geq 0$$

Axiom 2: If an event always occurs, we say it has probability 1.

$$\Omega = \begin{cases} T & \text{always} \\ F & \text{never} \end{cases}$$

$$P(\Omega = T) = 1$$

Axiom 3: probability measures behave like set measures.

$$P(A \vee B = T) = P(A = T) + P(B = T) - P(A \wedge B = T)$$

Joint and Conditional distributions of random variables

- $P(X, Y)$ is the **joint probability distribution** over all possible outcomes $P(X = x, Y = y)$.
- $P(X|Y)$ is the **conditional probability distribution** of outcomes $P(X = x|Y = y)$.

$$P(X = x|Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)}$$

MPE = MAP using Bayes' rule

Suppose your goal is to minimize the probability of error:

$$f(x) = \operatorname{argmin} P(\text{Error}|X = x)$$

...but the only things you know are the prior $P(Y = y)$, and the likelihood $P(X = x|Y = y)$. Well, presto! Using Bayes' rule, we can prove that the MPE decision rule is:

$$f(x) = \operatorname{argmax}_y P(Y = y)P(X = x|Y = y)$$

MPE = MAP using naïve Bayes

Using naïve Bayes, the MPE decision rule is:

$$f(x) = \operatorname{argmax}_y P(Y = y) \prod_{i=1}^n P(W = w_i | Y = y)$$

Laplace Smoothing for Naïve Bayes

- The basic idea: add k “unobserved observations” to the count of every unigram
 - If a word occurs 2000 times in the training data, Count = 2000+k
 - If a word occur once in training data, Count = 1+k
 - If a word never occurs in the training data, then it gets a pseudo-Count of k
- Estimated probability of a word that occurred Count(w) times in the training data: =

$$P(w) = \frac{\text{Count}(w) + k}{\sum_w \text{Count}(w) + k(1 + \sum_w 1)}$$

- Estimated probability of a word that never occurred in the training data (an “out of vocabulary” or OOV word):

$$P(OOV) = \frac{0 + k}{\sum_w \text{Count}(w) + k(1 + \sum_w 1)}$$

- Notice that

$$P(OOV) + \sum_w P(w) = 1$$

Bayes Error Rate

The “Bayes Error Rate” is the smallest possible error rate of any classifier with labels y and features x :

$$\text{Error Rate} = \sum_x P(X = x) \min_y P(Y \neq y | X = x)$$

It’s called the “Bayes error rate” because it’s the error rate of the Bayesian classifier.

How IR and AI summarize confusions

Precision:

$$P = P(Y = 1 | f(X) = 1) = \frac{TP}{TP + FP}$$

Recall = Sensitivity = TPR:

$$R = P(f(X) = 1 | Y = 1) = \frac{TP}{TP + FN}$$

Classified As:

	0	1
Correct Label:	0	1
0	TN	FP
1	FN	TP

Training vs. development test vs. evaluation test corpora

Training Corpus = a set of data that you use in order to optimize the parameters of your classifier (for example, optimize which features you measure, what are the weights of those features, what are the thresholds, and so on).

Development Test (DevTest or Validation) Corpus = a dataset, separate from the training dataset, on which you test 200 different fully-trained classifiers (trained, e.g., using different training algorithms, or different features) to find the best.

Evaluation Test Corpus = a dataset that is used only to test the ONE classifier that does best on DevTest. From this corpus, you learn how well your classifier will perform in the real world.

A mathematical definition of learning

- **Environment**: there are two random variables, $x \sim X$ and $y \sim Y$, that are jointly distributed according to

$$P(X = x, Y = y)$$

- **Data**: $P(X, Y)$ is unknown, but we have a sample of training data

$$\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

- **Objective**: We would like a function f such that $f(x) \approx y$
- **Definition of learning**: Learning is the task of estimating the function f , given knowledge of nothing other than \mathcal{D} .

Training a Multi-Class Perceptron

First, classify a training token, $f(\vec{x}) = \operatorname{argmax}_{c=1}^V (\vec{w}_c^T \vec{x})$. Then:

- If $f(\vec{x}) = y$ then do nothing
- If $f(\vec{x}) \neq y$ then
 - **Add** x to the vector that **should** have been the winner:

$$\vec{w}_y = \vec{w}_y + \eta \vec{x}$$

- **Subtract** x from the vector that **shouldn't** have won, but did:

$$\vec{w}_{f(x)} = \vec{w}_{f(x)} - \eta \vec{x}$$

- Don't change any of the other classes

Linear Regression

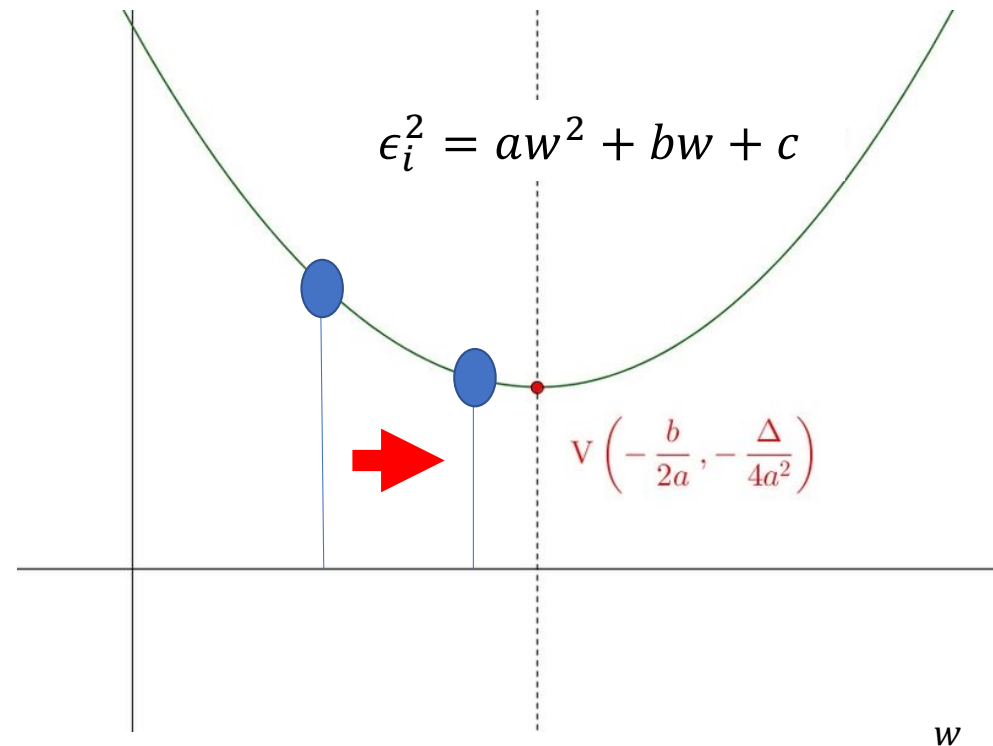
$$\epsilon_i^2 = (\vec{x}_i^T \vec{w} - y_i)^2$$

If we differentiate that, we discover that:

$$\nabla_{\vec{w}} \epsilon_i^2 = 2\epsilon_i \vec{x}_i$$

So the stochastic gradient descent algorithm is:

$$\vec{w} \leftarrow \vec{w} - \eta \epsilon_i \vec{x}_i$$



Perceptron versus logistic regression

Remember that for the perceptron, we have

$$f(\vec{x}_i) = \begin{bmatrix} f_1(\vec{x}_i) \\ \vdots \\ f_V(\vec{x}_i) \end{bmatrix}, \quad f_c(\vec{x}_i) = \begin{cases} 1 & c = \operatorname{argmax} \vec{w}_c^T \vec{x} \\ 0 & \text{otherwise} \end{cases}$$

For logistic regression, we have

$$f(\vec{x}_i) = \begin{bmatrix} f_1(\vec{x}_i) \\ \vdots \\ f_V(\vec{x}_i) \end{bmatrix}, \quad f_c(\vec{x}_i) = \frac{e^{\vec{w}_c^T \vec{x}}}{\sum_{k=1}^V e^{\vec{w}_k^T \vec{x}}}$$

Some details: Cross entropy

- The loss function is called “cross entropy,” because it is similar in some ways to the entropy of a thermodynamic system in physics.
- When you implement this in software, it’s a good idea to normalize by the number of training tokens, so that the scale is easier to understand:

$$\mathcal{L} = -\frac{1}{n} \ln P(\mathcal{D}|W) = -\frac{1}{n} \sum_{i=1}^n \ln P(C = c_i | X = \vec{x}_i)$$

The gradient of the cross-entropy of a softmax

$$\begin{aligned}\nabla_{\vec{w}_c} \mathcal{L}_i &= -\nabla_{\vec{w}_c} \ln P(C = c_i | X = \vec{x}_i) \\ &= \nabla_{\vec{w}_c} \left(\ln \sum_{k=1}^V e^{\vec{w}_k^T \vec{x}_i} \right) - \nabla_{\vec{w}_c} (\vec{w}_{c_i}^T \vec{x}_i) \\ &= \frac{e^{\vec{w}_c^T \vec{x}_i}}{\sum_{k=1}^V e^{\vec{w}_k^T \vec{x}_i}} \vec{x}_i - y_{i,c} \vec{x}_i = \epsilon_{i,c} \vec{x}_i\end{aligned}$$

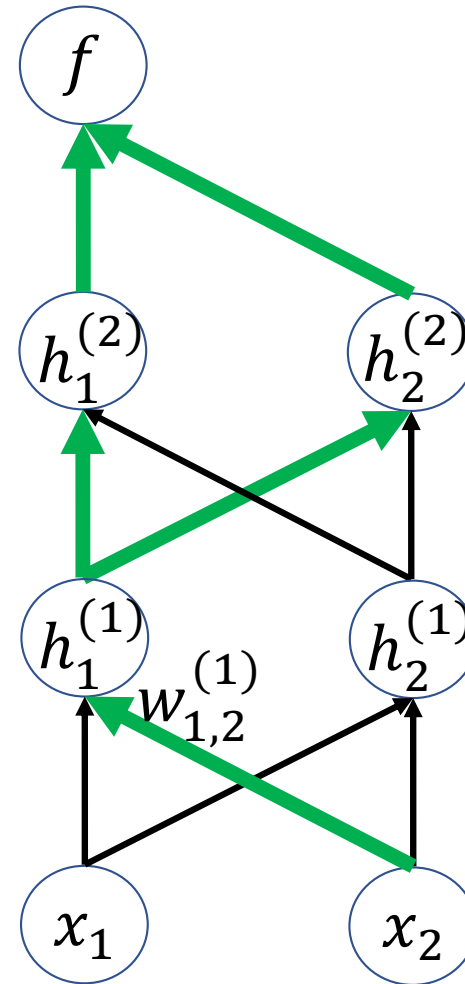
...where:

$$\epsilon_{i,c} = f_c(\vec{x}_i) - y_{i,c} = \begin{cases} \frac{e^{\vec{w}_c^T \vec{x}_i}}{\sum_{k=1}^V e^{\vec{w}_k^T \vec{x}_i}} - 1 & c_i = c \\ \frac{e^{\vec{w}_c^T \vec{x}_i}}{\sum_{k=1}^V e^{\vec{w}_k^T \vec{x}_i}} - 0 & \text{otherwise} \end{cases}$$

Back-propagation

The key idea of back-propagation is to calculate $\frac{\partial \mathcal{L}(\vec{w})}{\partial w_{j,k}^{(l)}}$, for every layer l , for every pair of nodes j and k , as follows:

- Start at the output node.
- Apply the chain rule of calculus backward, layer-by-layer, from the output node backward toward the input.



Back-propagation: splitting it up into excitation and activation

- **Activation to excitation**: here, the derivative is pre-computed. For example, if $g=\text{ReLU}$, then $g'=\text{unit step}$:

$$h_j^{(l)} = \text{ReLU}(\xi_j^{(l)}) \implies \frac{\partial h_j^{(l)}}{\partial \xi_j^{(l)}} = u(\xi_j^{(l)})$$

- **Excitation to activation**: here, the derivative is just the network weight!

$$\xi_j^{(l)} = b_j^{(l)} + \sum_k w_{j,k}^{(l)} h_k^{(l-1)} \implies \frac{\partial \xi_j^{(l)}}{\partial h_k^{(l-1)}} = w_{j,k}^{(l)}$$

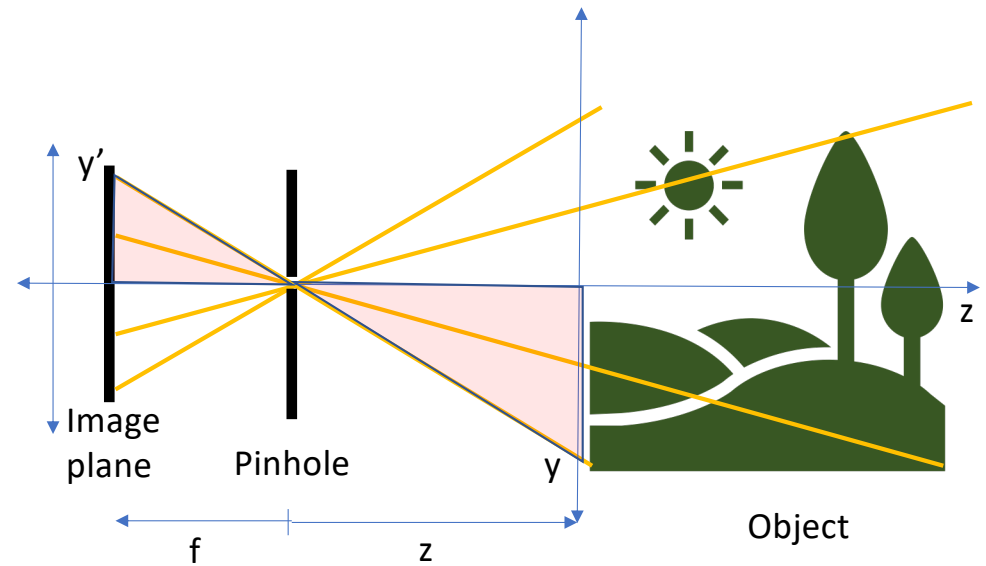
The pinhole camera equations

- These are similar triangles! So

$$\frac{y'}{f} = \frac{-y}{z}, \quad \frac{x'}{f} = \frac{-x}{z}$$

- Solving for (x', y') , we get the pinhole camera equations:

$$x' = \frac{-fx}{z}, \quad y' = \frac{-fy}{z}$$



- A **convolution** is exactly the same thing as a **weighted local average**. We give it a special name, because we will use it very often. It's defined as:

$$y[n] = \sum_m g[m]f[n - m] = \sum_m g[n - m]f[m]$$


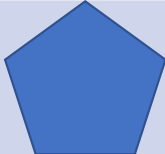
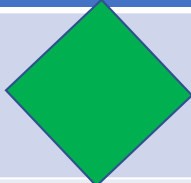
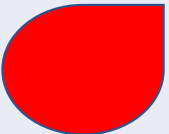
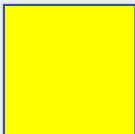
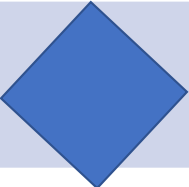
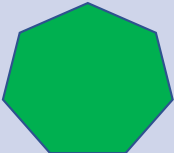
- We use the symbol $*$ to mean “convolution:”

$$y[n] = g[n] * f[n] = \sum_m g[m]f[n - m] = \sum_m g[n - m]f[m]$$

Exam 1 Review

- Mechanics: How to take the exam
- Review topics
- Sample review problem








Sample review problem

Boofs	Tizzles	?
 Yellow pacman	 Blue pentagon	 Green diamond
 Red teardrop	 Yellow square	
 Blue diamond	 Green heptagon	

Sample review problem

Let's express these as feature vectors.

$x_1 = \# \text{ points}$, $x_2 = \text{redness}$,
 $x_3 = \text{greenness}$, $x_4 = \text{blueness}$.

Boofs		Tizzles		?	
	Yellow pacman		Blue pentagon		Green diamond
	Red teardrop		Yellow square		
	Blue diamond		Green heptagon		

$$Y=\text{Boof: } \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 4 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$Y=\text{Tizzle: } \begin{bmatrix} 5 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 7 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$








$$Y=? : \begin{bmatrix} 4 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Sample review problem

$$Y=\text{Boof: } \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 4 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$








$$Y=\text{Tizzle: } \begin{bmatrix} 5 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 7 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$Y=? : \begin{bmatrix} 4 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Boofs		Tizzles		?	
	Yellow pacman		Blue pentagon		Green diamond
	Red teardrop		Yellow square		
	Blue diamond		Green heptagon		

- **Naïve Bayes:** If a naïve Bayes classifier is trained using these training data, how does it classify this test token?
- **Perceptron:** Assume we start with all-zero weights and biases, and $\eta = 1$. What are \vec{w} and b after the first 4 training tokens?
- **Logistic regression:** Assume we start with all-zero weights and biases, and $\eta = 1$. What are \vec{w}_{boof} , b_{boof} , \vec{w}_{tizzle} , and b_{tizzle} after the first 1 training token?
- **Two-layer neural net:** Assume we start with all-zero weights and biases, and $\eta = 1$. Find $b_2^{(2)}$, the second bias term in the second layer, after the first 1 training token.

Naïve Bayes

Boofs		Tizzles		?	
	Yellow pacman		Blue pentagon		Green diamond
	Red teardrop		Yellow square		
	Blue diamond		Green heptagon		

$$P(Y = \text{boof} | X) = \frac{P(Y = \text{boof}, X)}{P(X)}$$

$$P\left(Y = \text{boof}, X = \begin{bmatrix} 4 \\ 0 \\ 1 \\ 0 \end{bmatrix}\right) = P(Y = \text{b})P(x_1 = 4 | Y = \text{b}) \cdots P(x_4 = 0 | Y = \text{b})$$








$$= \binom{1}{2} \left(\frac{1+k}{3+4k}\right) \left(\frac{1+k}{3+3k}\right) \left(\frac{1+k}{3+3k}\right) \left(\frac{2+k}{3+3k}\right)$$

$$P\left(Y = \text{tizzle}, X = \begin{bmatrix} 4 \\ 0 \\ 1 \\ 0 \end{bmatrix}\right) = \binom{1}{2} \left(\frac{1+k}{3+4k}\right) \left(\frac{2+k}{3+3k}\right) \left(\frac{2+k}{3+3k}\right) \left(\frac{2+k}{3+3k}\right)$$

Perceptron

Boof: $Y = -1$, Tizzle: $Y = +1$






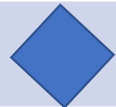

Start with $\vec{w} = [0,0,0,0]^T, b = 0$

Boofs		Tizzles		?	
	Yellow pacman		Blue pentagon		Green diamond
	Red teardrop		Yellow square		
	Blue diamond		Green heptagon		

- First training token: $X = \text{Yellow pacman} = [2,1,1,0]^T, Y = -1$. Token is misclassified (classifier output is undefined), so $\vec{w} \leftarrow \vec{w} - \vec{x} = [-2, -1, -1, 0]^T$ and $b \leftarrow b - 1 = -1$.
- Second training token: $X = \text{Blue pentagon} = [5,0,0,1]^T, Y = +1$. Token is misclassified ($\vec{w}^T \vec{x} + b < 0$), so $\vec{w} \leftarrow \vec{w} + \vec{x} = [3, -1, -1, 1]^T$ and $b \leftarrow b + 1 = 0$.
- Third training token: $X = \text{Red teardrop} = [1,1,0,0]^T, Y = -1$. Token is misclassified ($\vec{w}^T \vec{x} + b > 0$), so $\vec{w} \leftarrow \vec{w} - \vec{x} = [2, -2, -1, 1]^T$ and $b \leftarrow b - 1 = -1$.
- Fourth training token: $X = \text{Yellow square} = [4,1,1,0]^T, Y = +1$. Token is classified correctly ($\vec{w}^T \vec{x} + b > 0$), so \vec{w} and b are unchanged.

Logistic Regression

Boof: $Y = [1,0]^T$, Tizzle: $Y = [0,1]^T$
 Start with $\vec{w}_{\text{boof}} = [0,0,0,0]^T$, $b_{\text{boof}} = 0$,
 $\vec{w}_{\text{tizzle}} = [0,0,0,0]^T$, $b_{\text{tizzle}} = 0$

Boofs		Tizzles		?	
	Yellow pacman		Blue pentagon		Green diamond
	Red teardrop		Yellow square		
	Blue diamond		Green heptagon		

First training token: $X = \text{Yellow Pacman} = [2,1,1,0]^T$, $Y = [1,0]^T$.

$$f_{\text{boof}}(\vec{x}) = \frac{e^{\vec{w}_{\text{boof}}^T \vec{x} + b_{\text{boof}}}}{e^{\vec{w}_{\text{boof}}^T \vec{x} + b_{\text{boof}}} + e^{\vec{w}_{\text{tizzle}}^T \vec{x} + b_{\text{tizzle}}}} = 0.5, \quad f_{\text{tizzle}}(\vec{x}) = 0.5$$

$$\varepsilon_{i,\text{boof}} = 0.5 - 1 = -0.5, \quad \varepsilon_{i,\text{tizzle}} = 0.5 - 0 = 0.5$$

$$\vec{w}_{\text{boof}} \leftarrow \vec{w}_{\text{boof}} - \varepsilon_{i,\text{boof}} \vec{x} = \begin{bmatrix} 1 \\ 0.5 \\ 0.5 \\ 0 \end{bmatrix},$$






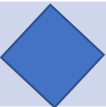

$$\vec{w}_{\text{tizzle}} \leftarrow \vec{w}_{\text{tizzle}} - \varepsilon_{i,\text{tizzle}} \vec{x} = \begin{bmatrix} -1 \\ -0.5 \\ -0.5 \\ 0 \end{bmatrix}$$

$$b_{\text{boof}} \leftarrow b_{\text{boof}} - \varepsilon_{i,\text{boof}} = 0.5,$$

$$b_{\text{tizzle}} \leftarrow b_{\text{tizzle}} - \varepsilon_{i,\text{tizzle}} = -0.5$$

Two-Layer Network

Assume two hidden nodes h_1 and h_2 with ReLU activation functions, and two output nodes f_1 and f_2 with softmax activation function. All weights and biases start at zero.

Boofs		Tizzles		?	
	Yellow pacman		Blue pentagon		Green diamond
	Red teardrop		Yellow square		
	Blue diamond		Green heptagon		

First training token: $X = \text{Yellow Pacman} = [2, 1, 1, 0]^T$, $Y = [1, 0]^T$.

$\xi_1^{(1)} = 0 + \sum_{k=1}^4 0 \cdot x_k = 0$. $h_1 = \text{ReLU}(0) = 0$. Likewise $h_2 = 0$, likewise

$\xi_1^{(2)} = 0$ and $\xi_2^{(2)} = 0$, so $f_1(\vec{x}) = \frac{e^{\xi_1^{(2)}}}{e^{\xi_1^{(2)}} + e^{\xi_2^{(2)}}} = 0.5$ and $f_2(\vec{x}) = 0.5$. If we use

cross-entropy as our loss function, then

$$b_2^{(2)} \leftarrow b_2^{(2)} - \frac{\partial \mathcal{L}}{\partial \xi_2^{(2)}} \frac{\partial \xi_2^{(2)}}{\partial b_2^{(2)}} = 0 - (f_2(\vec{x}) - y_{1,2})(1) = -0.5$$

Exam 1 Review

- Mechanics: How to take the exam
- Review topics
 - Probability and Machine Learning
- Sample review problem
 - Any of these four classifiers can solve this problem!
 - The equations are a bit more complex for the two-layer network...