CS 760: Machine Learning
Neural Networks III

Ilias Diakonikolas

University of Wisconsin-Madison

Oct. 13, 2022

Outline

- **Review & Regularization**
  - Forward/backwards Pass, Views, L1/L2 Effects
- **Other Forms of Regularization**
  - Data Augmentation, Noise, Early Stopping, Dropout
- **Convolutional Neural Networks**
  - Convolution Operation, Intuition

Outline

- **Review & Regularization**
  - Forward/backwards Pass, Views, L1/L2 Effects
- **Other Forms of Regularization**
  - Data Augmentation, Noise, Early Stopping, Dropout
- **Convolutional Neural Networks**
  - Convolution Operation, Intuition

Review: Backprop

- Forward pass:

$$L(f_{\text{network}}(x), y)$$

- Let's unwrap this:

$$L(r^k(W^k r^{k-1}(W^{k-1} \cdots r^2(W^2 r^1(W^1 x)) \cdots)), y)$$

Activation function **Layer k**

Linear transformation, **Layer k**

Activation function **Layer 1**

Linear transformation, **Layer 1**

Review: Forward/Backward Passes

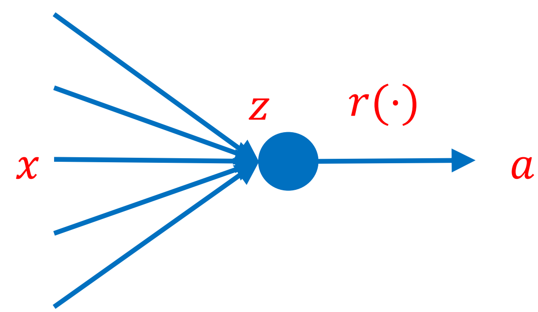
- Forward pass:

$$L(r^k(W^k r^{k-1}(W^{k-1} \cdots r^2(W^2 r^1(W^1 x)) \cdots)), y)$$

- For convenience,

$$a^j = r^j(W^j r^{j-1}(W^{j-1} \cdots r^2(W^2 r^1(W^1 x)) \cdots))$$

$$z^j = W^j r^{j-1}(W^{j-1} \cdots r^2(W^2 r^1(W^1 x)) \cdots)$$



Review: Backward Pass

- Backward pass. Say we compute gradient w.r.t. x

$$\frac{\partial L}{\partial a^k}\frac{\partial a^k}{\partial z^k}\frac{\partial z^k}{\partial a^{k-1}}\frac{\partial a^{k-1}}{\partial z^{k-1}}\frac{\partial z^{k-1}}{\partial a^{k-2}}\cdots\frac{\partial a^1}{\partial z^1}\frac{\partial z^1}{\partial x}$$

- Can write this with matrix notation
  - Writing it forward, this is equivalent

$$\nabla_x L = (W^1)^T(r^1)'\cdots(W^{k-1})^T(r^{k-1})'(W^k)^T(r^k)'\nabla_{a^k}L$$

Linear derivative

Activation function derivative

Review: Backpropagation

- Backward pass. Say we compute gradient w.r.t. x

$$\nabla_x L = (W^1)^T (r^1)' \cdots (W^{k-1})^T (r^{k-1})' (W^k)^T (r^k)' \nabla_{a^k} L$$

- Let's write this recursively:

$$\delta^j = (r^j)' (W^{j+1})^T \cdots (W^{k-1})^T (r^{k-1})' (W^k)^T (r^K)' \nabla_{a^k} L$$

- Easy to set up a recursion (start at k, go down) :

Start at j layer here

$$\delta^{j-1} = (r^{j-1})' (W^j)^T \delta^j$$

Review: Backpropagation

- Let's write this recursively:

$$\delta^j = (r^j)'(W^{j+1})^T \cdots (W^{k-1})^T (r^{k-1})'(W^k)^T (r^K)' \nabla_{a^k} L$$

- Easy to set up a recursion (start at k, go down) :

$$\delta^{j-1} = (r^{j-1})'(W^j)^T \delta^j$$

- How do we get our gradients for weights?

$$\nabla_{W^j} L = \delta^j (a^{j-1})^T$$

Review: Regularization, Bayesian Prior View

- Recall our MAP version of training. Bayes law:

$$p(\theta \mid \{x_i, y_i\}) = \frac{p(\theta)p(\{x_i, y_i\}|\theta)}{p(\{x_i, y_i\})}$$

- MAP:

$$\max_\theta \log p(\theta \mid \{x_i, y_i\}) = \min_\theta - \underbrace{\log p(\theta)}_{\textbf{Regularization}} - \underbrace{\log p(\{x_i, y_i\} \mid \theta)}_{\textbf{MLE loss}}$$

- L2: Corresponds to normal $p(x \mid y, \theta)$, **normal prior** $p(\theta)$

Choice of View?

- Typical choice for optimization: soft-constraint

$$\min_{\theta} \hat{L}_R(\theta) = \hat{L}(\theta) + \lambda R(\theta)$$

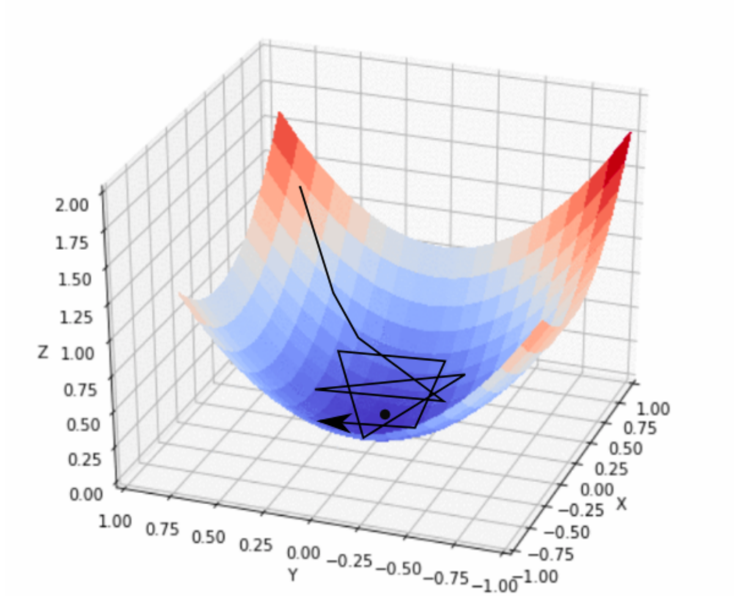
- Hard constraint / Bayesian view: conceptual / for derivation
- Hard-constraint preferred if
  - Know the explicit bound $R(\theta) \leq r$
- Bayesian view preferred if
  - Domain knowledge easy to represent as a prior

Examples: L2 Regularization

- Again,

$$\min_{\theta} \hat{L}_R(\theta) = \hat{L}(\theta) + \frac{\lambda}{2}||\theta||_2^2$$

- Questions: what are the
  - Effects on (stochastic) gradient descent?
  - Effects on the optimal solution?



L2 Regularization: **Effect on GD**

- Gradient of regularized objective

$$\nabla \hat{L}_R(\theta) = \nabla \hat{L}(\theta) + \lambda\theta$$

- Gradient descent update

$$\theta \leftarrow \theta - \eta\nabla \hat{L}_R(\theta) = \theta - \eta\,\nabla \hat{L}(\theta) - \eta\lambda\theta$$

$$= (1 - \eta\lambda)\theta - \eta\,\nabla \hat{L}(\theta)$$

- In words, **weight decay**

L2 Regularization: **Effect on Optimal Solution**

- Consider a quadratic approximation around θ^*

$$\hat{L}(\theta) \approx \hat{L}(\theta^*) + (\theta - \theta^*)^T \nabla \hat{L}(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T H (\theta - \theta^*)$$

- Since θ^* is optimal, $\nabla \hat{L}(\theta^*) = 0$

$$\hat{L}(\theta) \approx \hat{L}(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T H (\theta - \theta^*)$$

$$\nabla \hat{L}(\theta) \approx H(\theta - \theta^*)$$

L2 Regularization: **Effect on Optimal Solution**

- Gradient of regularized objective: $\nabla \hat{L}_R(\theta) \approx H(\theta - \theta^*) + \lambda\theta$

- On the optimal θ_R^*: $0 = \nabla \hat{L}_R(\theta_R^*) \approx H(\theta_R^* - \theta^*) + \lambda\theta_R^*$

$$\theta_R^* \approx (H + \lambda I)^{-1} H \theta^*$$

- H has eigendecomp. $H = Q\Lambda Q^T$, assume $(\Lambda + \lambda I)^{-1}$ exists:

$$\theta_R^* \approx (H + \lambda I)^{-1} H \theta^* = Q(\Lambda + \lambda I)^{-1} \Lambda Q^T \theta^*$$

- Effect: **rescale along eigenvectors of $\boldsymbol{H}$**

L2 Regularization: **Effect on Optimal Solution**

Effect: rescale along eigenvectors of H

Visual Example:

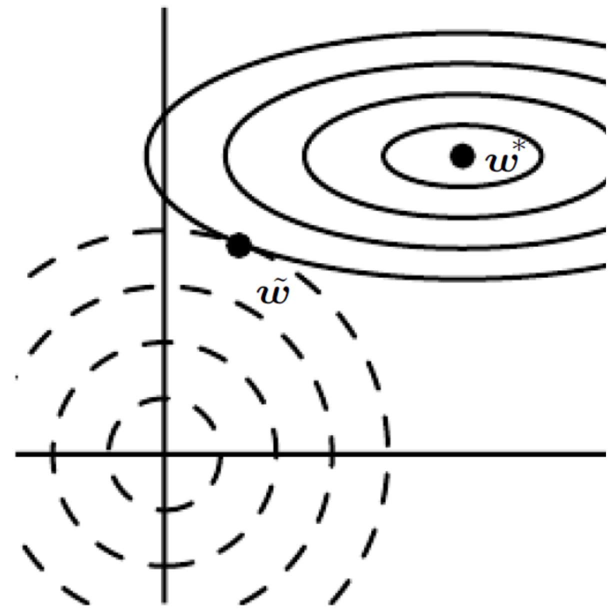


Figure from *Deep Learning*,
Goodfellow, Bengio and Courville

L1 Regularization: **Effect on GD**

$$\min_{\theta} \hat{L}_R(\theta) = \hat{L}(\theta) + \lambda ||\theta||_1$$

- Effect on (stochastic) gradient descent:
- Gradient of regularized objective

$$\nabla \hat{L}_R(\theta) = \nabla \hat{L}(\theta) + \lambda \text{sign}(\theta)$$

where **sign** applies to each element in θ

- Gradient descent update

$$\theta \leftarrow \theta - \eta \nabla \hat{L}_R(\theta) = \theta - \eta \nabla \hat{L}(\theta) - \eta \lambda \text{sign}(\theta)$$

L1 Regularization: **Effect on Optimal Solution**

- Again,

$$\hat{L}(\theta) \approx \hat{L}(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T H(\theta - \theta^*)$$

- Further assume that H is diagonal and positive $(H_{ii} > 0, \forall i)$
    - **not true in general** but assume for getting some intuition

- The regularized objective is (ignoring constants)

$$\hat{L}_R(\theta) \approx \sum_i \frac{1}{2} H_{ii}(\theta_i - \theta_i^*)^2 + \lambda|\theta_i|$$

L1 Regularization: **Effect on Optimal Solution**

- The regularized objective is (ignoring constants)

$$\hat{L}_R(\theta) \approx \sum_i \frac{1}{2} H_{ii}(\theta_i - \theta_i^*)^2 + \lambda|\theta_i|$$

- The optimal θ_R^*

$$(\theta_R^*)_i \approx \begin{cases} \max\left\{\theta_i^* - \frac{\lambda}{H_{ii}}, 0\right\} & \text{if } \theta_i^* \geq 0 \\ \min\left\{\theta_i^* + \frac{\lambda}{H_{ii}}, 0\right\} & \text{if } \theta_i^* < 0 \end{cases}$$

- Compact expression for the optimal θ_R^*

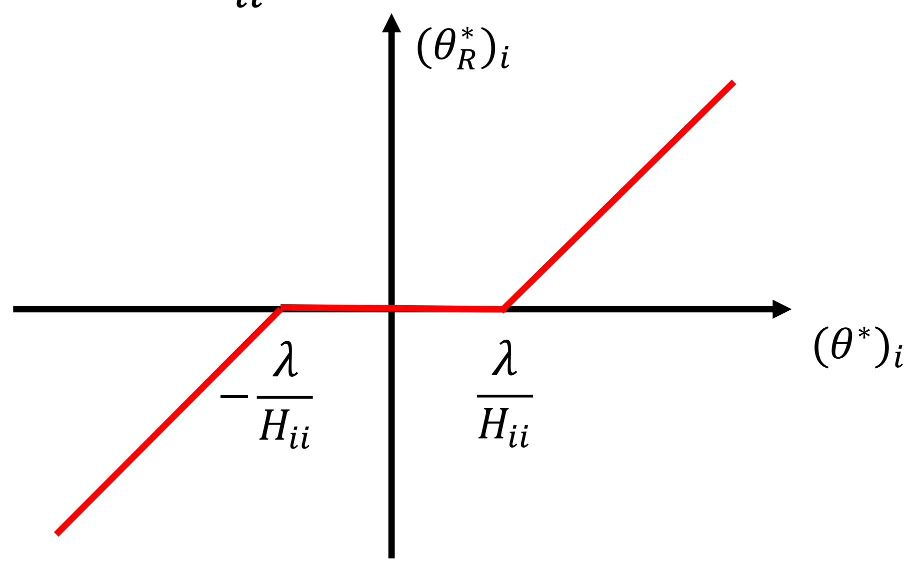
$$(\theta_R^*)_i \approx \operatorname{sign}(\theta_i^*) \max\{|\theta_i^*| - \frac{\lambda}{H_{ii}}, 0\}$$

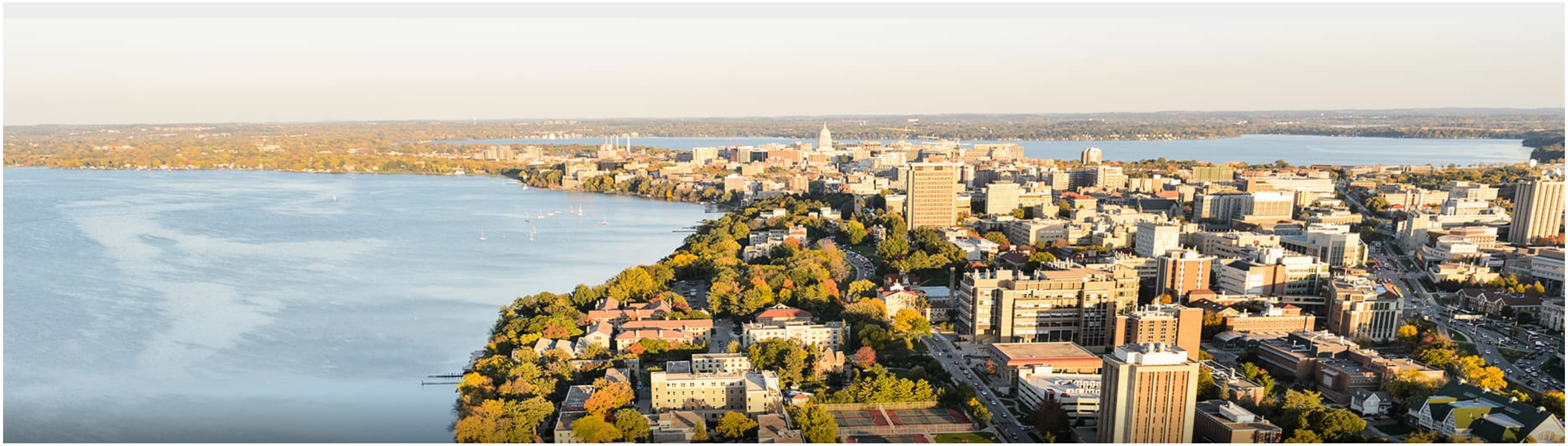
L1 Regularization: **Effect on Optimal Solution**

- The optimal θ_R^*

$$(\theta_R^*)_i \approx \begin{cases} \max\left\{\theta_i^* - \frac{\lambda}{H_{ii}}, 0\right\} & \text{if } \theta_i^* \geq 0 \\ \min\left\{\theta_i^* + \frac{\lambda}{H_{ii}}, 0\right\} & \text{if } \theta_i^* < 0 \end{cases}$$

- Effect: **induces sparsity**





Break & Quiz

Q1-2: Which of the following statement(s) is(are) TRUE about regularization parameter λ ?

A. *λ is the tuning parameter that decides how much we want to penalize the flexibility of our model.*
B. *λ is usually set using cross validation.*

1. True, True
2. True, False
3. False, True
4. False, False

Q1-2: Which of the following statement(s) is(are) TRUE about regularization parameter λ ?

A. *λ is the tuning parameter that decides how much we want to penalize the flexibility of our model.*
B. *λ is usually set using cross validation.*

1. True, True ⬅
2. True, False
3. False, True
4. False, False

- The optimization problem can be viewed as following:

$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}) + \lambda\ \text{complexity}(\text{Model}))$$

- If the regularization parameter is large then it requires a small model complexity
- We have learned how to use cross validate to set hyperparameters including regularization parameters.

Q2-1: Select the correct option about regression with L2 regularization (also called *Ridge Regression*).

A. *Ridge regression technique prevents coefficients from rising too high.*

B. *As $\lambda \rightarrow \infty$, the impact of the penalty grows, and the ridge regression coefficient estimates will approach infinity.*

1. Both statements are true.
2. Both statements are false.
3. Statement A is true, Statement B is false.
4. Statement B is true, Statement A is false.

Q2-1: Select the correct option about regression with L2 regularization (also called *Ridge Regression*).

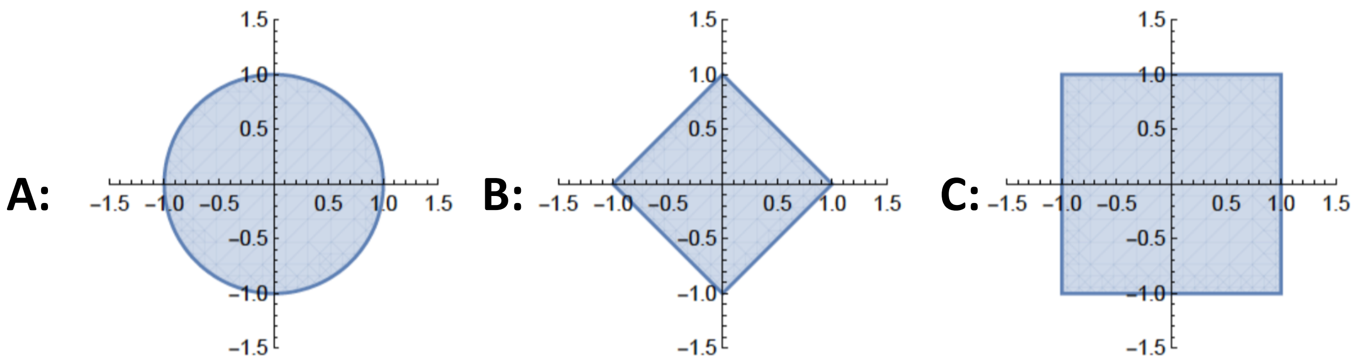
A. *Ridge regression technique prevents coefficients from rising too high.*

B. *As λ→∞, the impact of the penalty grows, and the ridge regression coefficient estimates will approach infinity.*

1. Both statements are true.
2. Both statements are false.
3. Statement A is true, Statement B is false.
4. Statement B is true, Statement A is false.

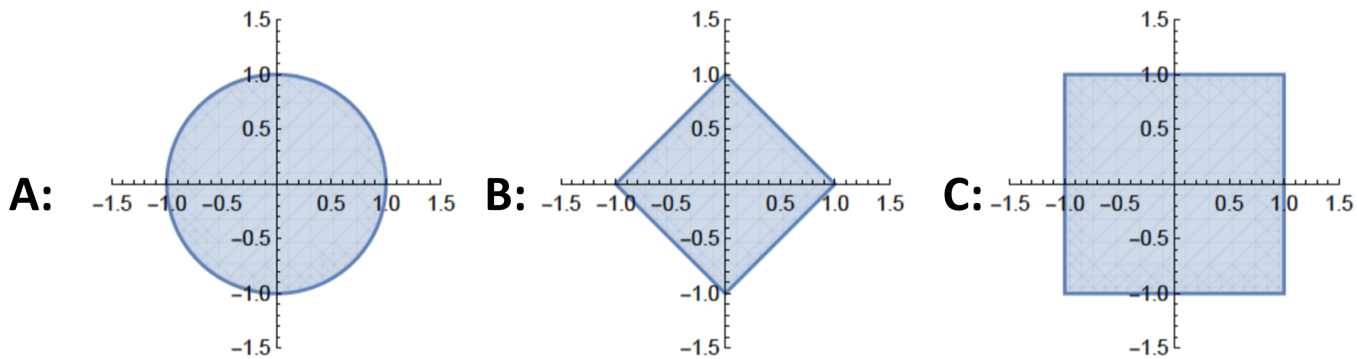
As λ→∞, the impact of the penalty grows, and the ridge regression coefficient estimates will approach zero.

Q3-1: Following figure shows 3-norm sketches: $||x||_p<1$ for p = 1, 2, ∞. Recall that $||x||_\infty = \max\{|x_i| \text{ for all } i\}$



1. A: 1, B: 2, C: ∞
2. A: 2, B: 1, C: ∞
3. A: 2, B: ∞, C: 1
4. A: ∞, B: 2, C: 1

Q3-1: Following figure shows 3-norm sketches: $||x||_p<1$ for p = 1, 2, ∞.
Recall that $||x||_\infty = \max\{|x_i| \text{ for all } i\}$



1. A: 1, B: 2, C: ∞
2. A: 2, B: 1, C: ∞
3. A: 2, B: ∞, C: 1
4. A: ∞, B: 2, C: 1

Outline

- **Review & Regularization**
  - Forward/backwards Pass, Views, L1/L2 Effects
- **Other Forms of Regularization**
  - Data Augmentation, Noise, Early Stopping, Dropout
- **Convolutional Neural Networks**
  - Convolution Operation, Intuition

Data Augmentation

Augmentation: transform + add new samples to dataset

- Transformations: based on domain
- Idea: build **invariances** into the model
  - **Ex**: if all images have same alignment, model learns to use it
- Keep the label the same!

Data Augmentation: Examples

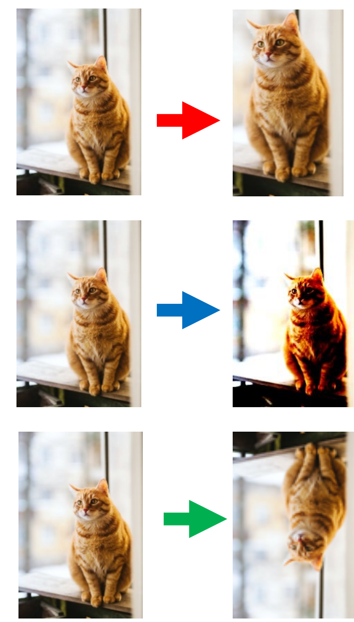
Examples of transformations for images

- **Crop** (and zoom)
- **Color** (change contrast/brightness)
- **Rotations+** (translate, stretch, shear, etc)

Many more possibilities. Combine as well!

Q: how to deal with this at **test time**?

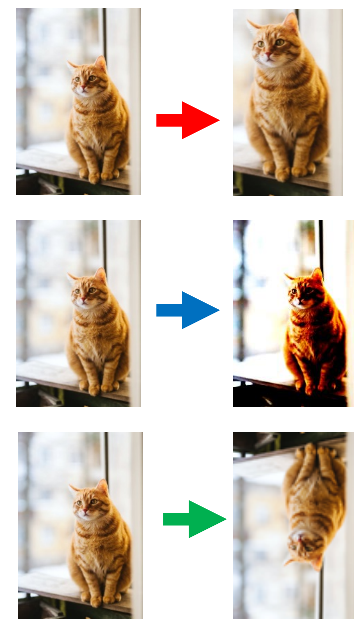
- A: transform, test, average



Combining & Automating Transformations

One way to automate the process:

- Apply every transformation and combinations
- **Downside:** most don't help…



Want a good policy, ie, 

- Active area of research: search for good policies

1. **Ratner et al**: "Learning to Compose Domain-Specific Transformations for Data Augmentation"
2. **Cubuk et al**: "AutoAugment: Learning Augmentation Strategies from Data"

Data Augmentation: Other Domains

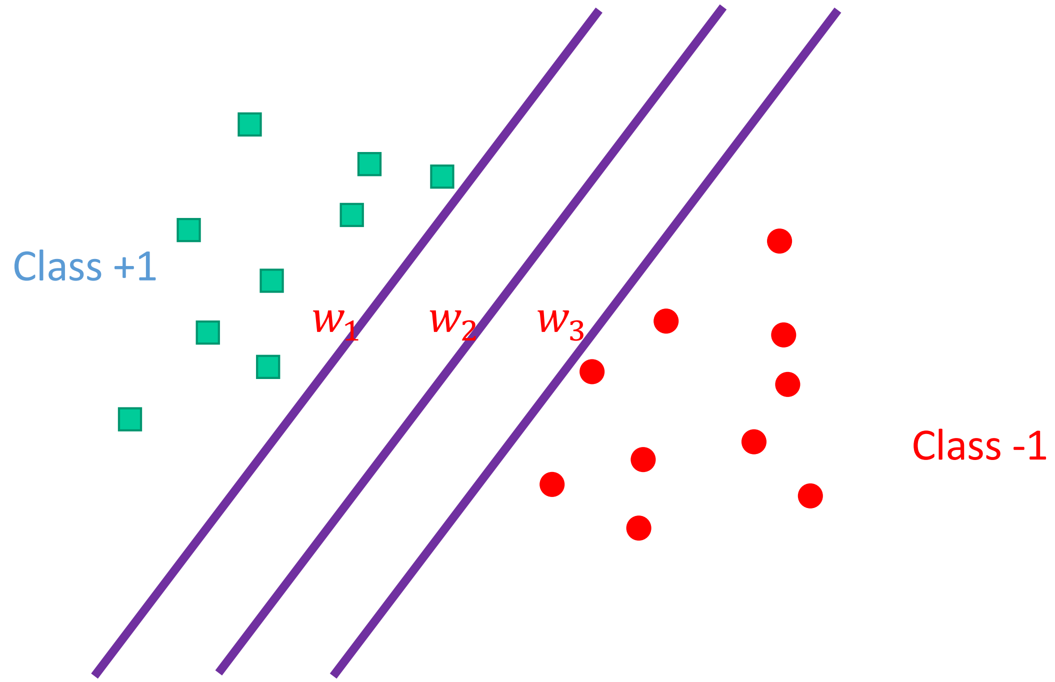
Not just for image data. For example, on text:

- Substitution
  - E.g., “It is a **great** day” ➔ “It is a **wonderful** day”
  - Use a thesaurus for particular words
  - Or, use a model. Pre-trained word embeddings, language models
- Back-translation
  - “Given the low budget and production limitations, this movie is very good.” ➔ “There are few budget items and production limitations to make this film a really good one”

Xie **et al**: “Unsupervised Data Augmentation for Consistency Training”

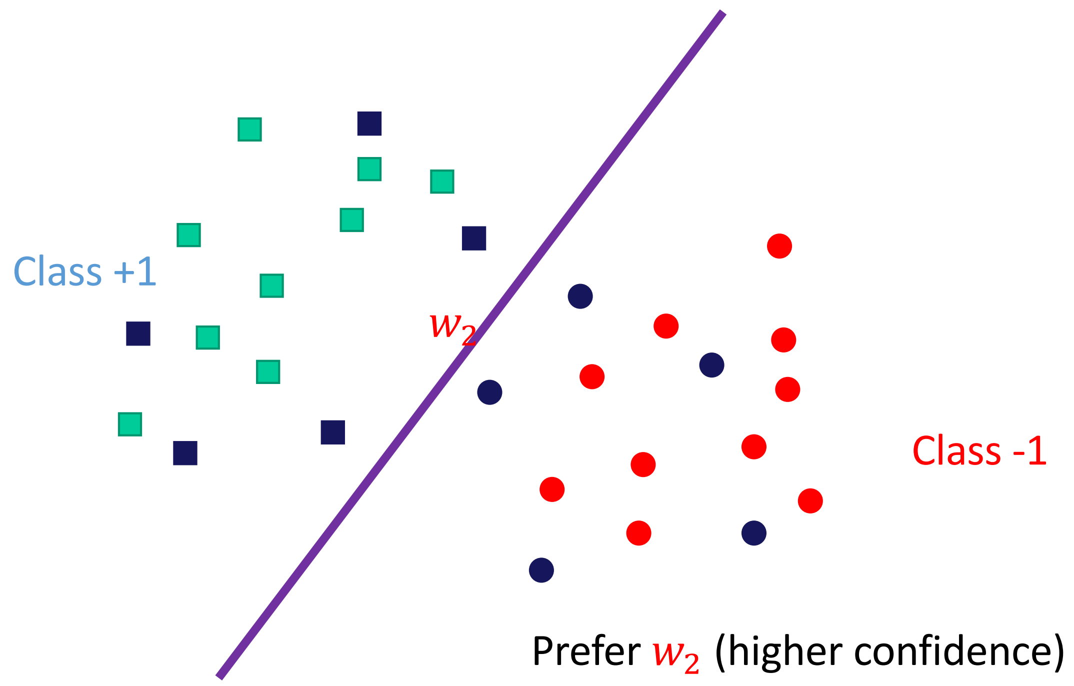
Adding Noise

- What if we have many solutions?



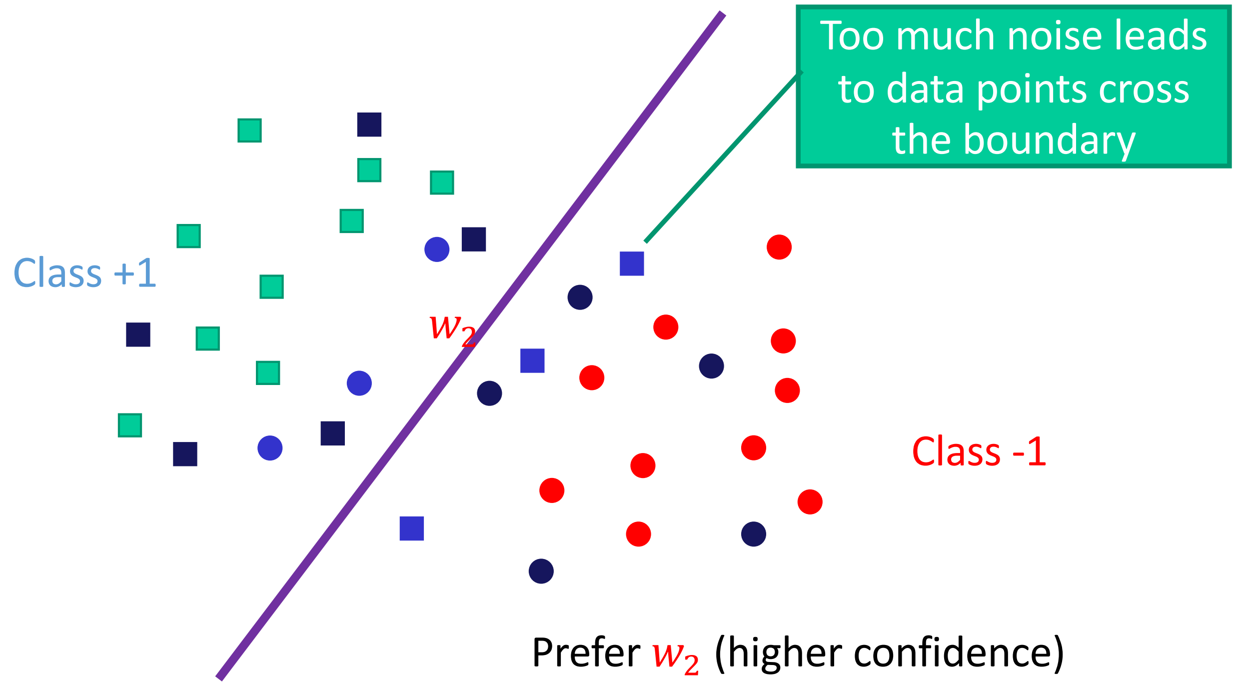
Adding Noise

- Adding some amount of noise helps us pick solution:



Adding Noise

- Too much: hurts instead



Adding Noise: Equivalence to Weight Decay

- Suppose the hypothesis is $f(x) = w^T x$, noise is $\epsilon \sim N(0, \lambda I)$
- After adding noise, the loss is

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f(x+\epsilon) - y]^2 = \mathbb{E}_{x,y,\epsilon}[f(x) + w^T\epsilon - y]^2$$

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f(x) - y]^2 + 2\mathbb{E}_{x,y,\epsilon}[w^T\epsilon(f(x) - y)] + \mathbb{E}_{x,y,\epsilon}[w^T\epsilon]^2$$

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f(x) - y]^2 + \lambda\left||w|\right|^2$$

Early Stopping

- **Idea**: don't train the network to too small training error
  - Larger the hypothesis class, easier to find a hypothesis that fits the difference between the two
  - So: do not push the hypothesis too much; use validation error to decide when to stop

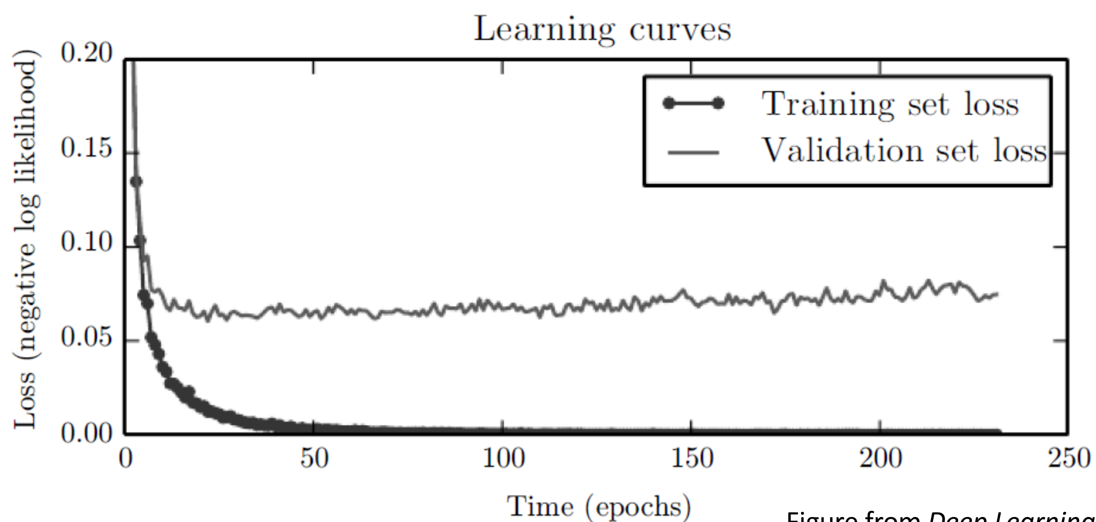


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Early Stopping

- Practically: when training, also output validation error
  - Every time validation error improved, store a copy of the weights
  - When validation error not improved for some time, stop
  - Return the copy of the weights stored

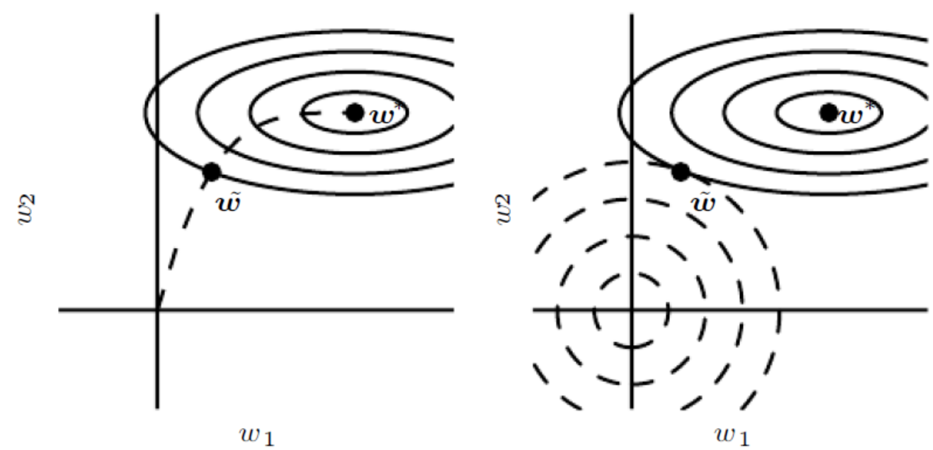


Figure from *Deep Learning*,
Goodfellow, Bengio and Courville

Dropout

- **Basic idea**: randomly select weights to update

- In each update step
  - Randomly sample a different binary mask to all the input and hidden units
  - Multiply the mask bits with the units and do the update as usual

- Typical dropout prob: 0.2 for input and 0.5 for hidden units

Dropout

- Closely related to bagging:
  - Ensembling many models

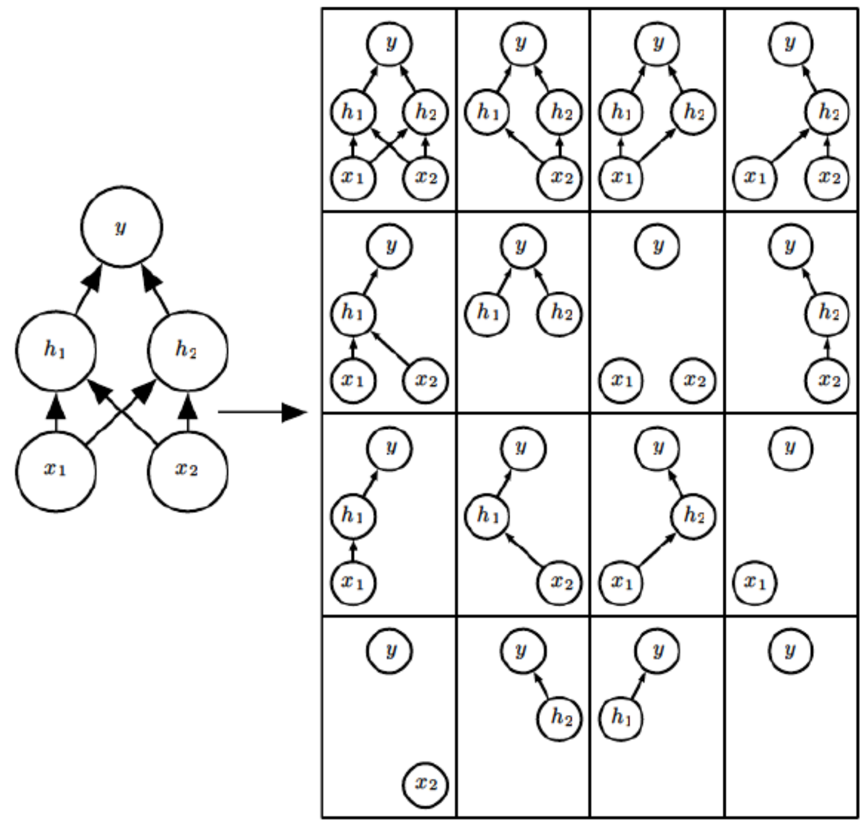


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Batch Normalization

- If outputs of earlier layers are uniform or change greatly on one round for one mini-batch, then neurons at next levels can't keep up: they output all high (or all low) values

- Next layer doesn't have ability to change its outputs with learning-rate-sized changes to its input weights

- We say the layer has "saturated"

Batch Normalization

- Algorithm:

- (i)-(iii) like standardization of input data, but w.r.t. only the data in mini-batch. Can take derivative and incorporate the learning of last step parameters into backpropagation.

- Note last step can completely un-do previous 3 steps

- But if so this un-doing is driven by the *later* layers, not the *earlier* layers; later layers get to "choose" whether they want standard normal inputs or not

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

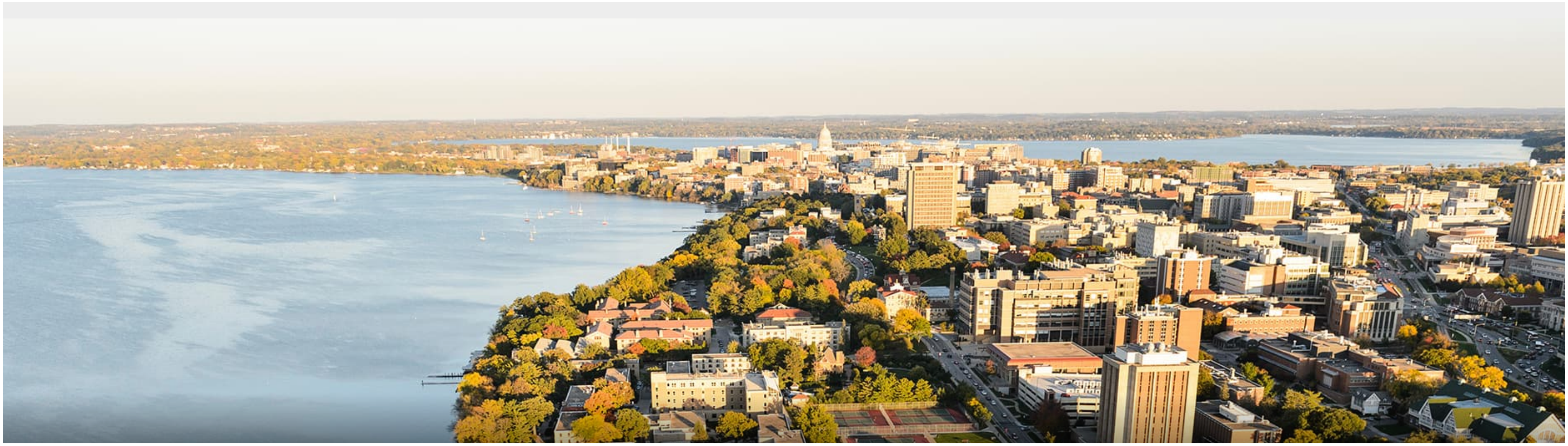
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.



Break & Quiz

Q2-2: Are these statements true or false?
(A) We need validation data to decide when to early stop.
(B) We can think early stopping as a regularization to limit the volume of parameter space reachable from the initial parameter.

1. True, True
2. True, False
3. False, True
4. False, False

Q2-2: Are these statements true or false?
(A) We need validation data to decide when to early stop.
(B) We can think early stopping as a regularization to limit the volume of parameter space reachable from the initial parameter.

1. True, True ⬅
2. True, False
3. False, True
4. False, False

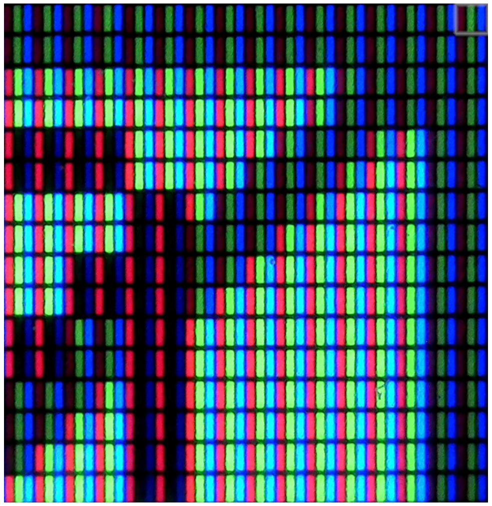
(A) As is shown in the lecture.
(B) That's true. Early stopping will limit the training time and thus potentially limit the space the training can search.

Outline

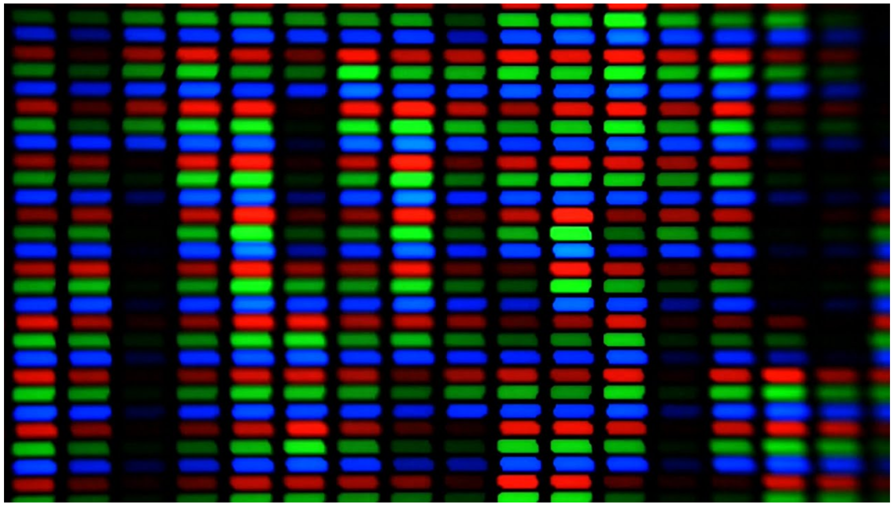
- **Review & Regularization**
  - Forward/backwards Pass, Views, L1/L2 Effects
- **Other Forms of Regularization**
  - Data Augmentation, Noise, Early Stopping, Dropout
- **Convolutional Neural Networks**
  - Convolution Operation, Intuition

Images as Input?

- We could use the feed-forward fully-connected layers we have so far…
  - Kind of big though…
  - Also, if our images move, should the weights change?



Microsoft



PixelArt4K

Convolution Operation

- Given array u_t and w_t, their convolution is a function s_t

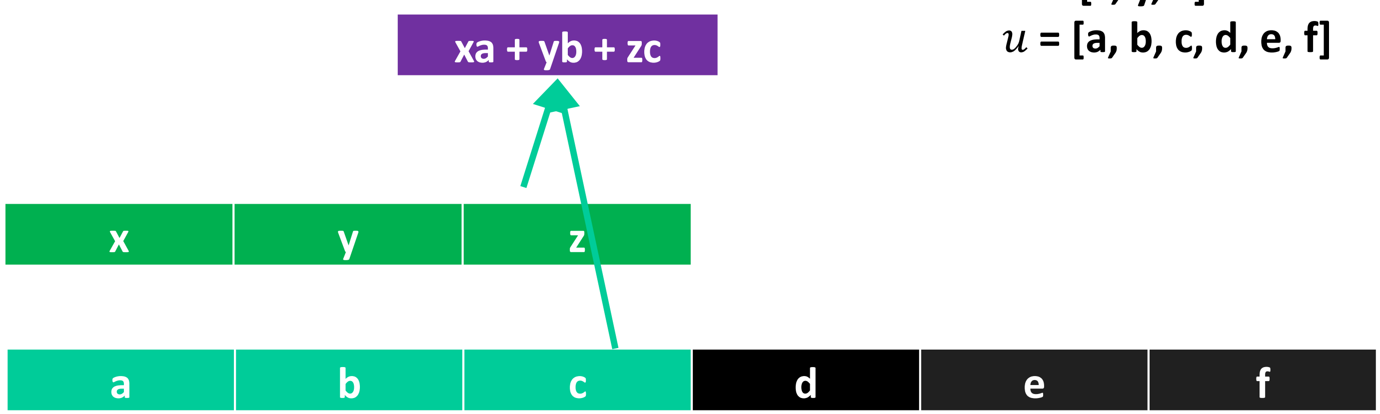
$$s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a}$$

- Written as $s = (u * w) \quad \text{or} \quad s_t = (u * w)_t$
- When u_t or w_t is not defined, assumed to be 0

Convolution Operation

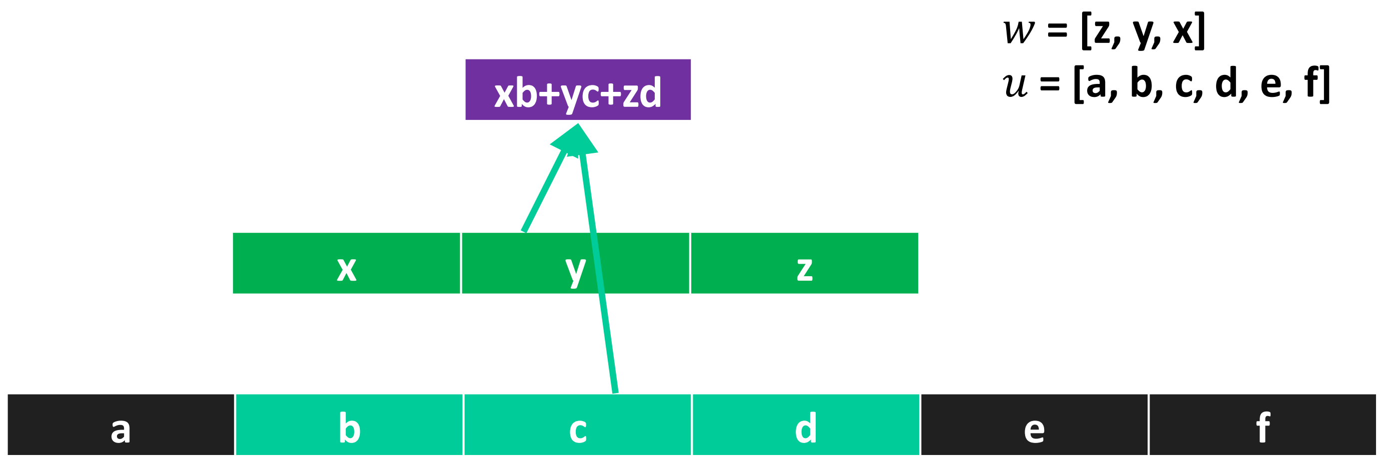
- Example:

w = [z, y, x]
u = [a, b, c, d, e, f]



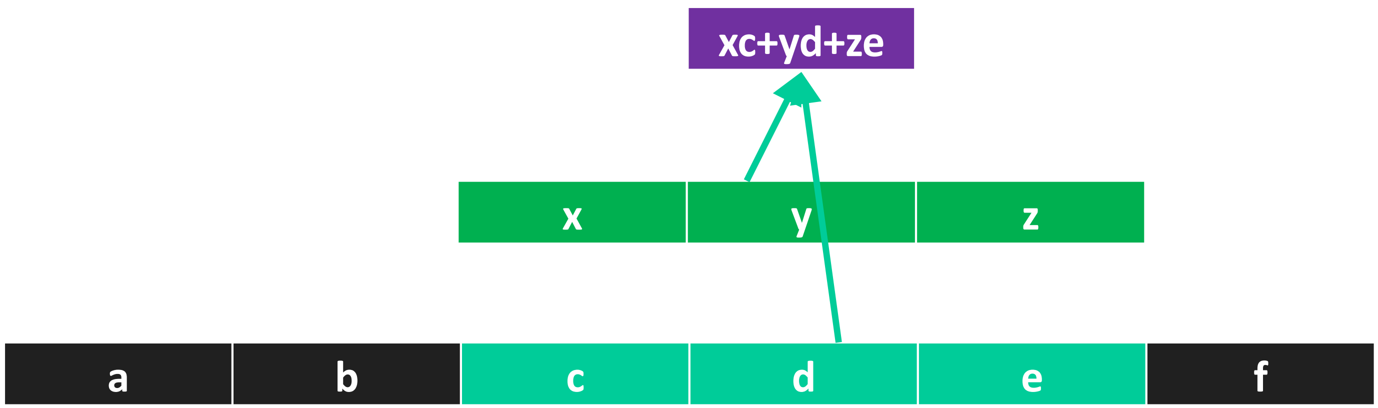
Convolution Operation

- Example:



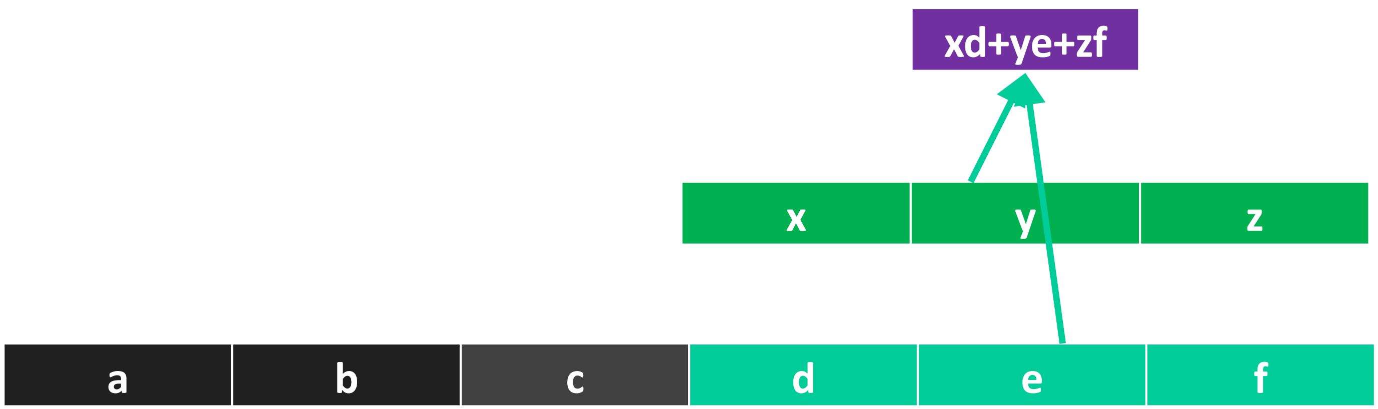
Convolution Operation

- Example:



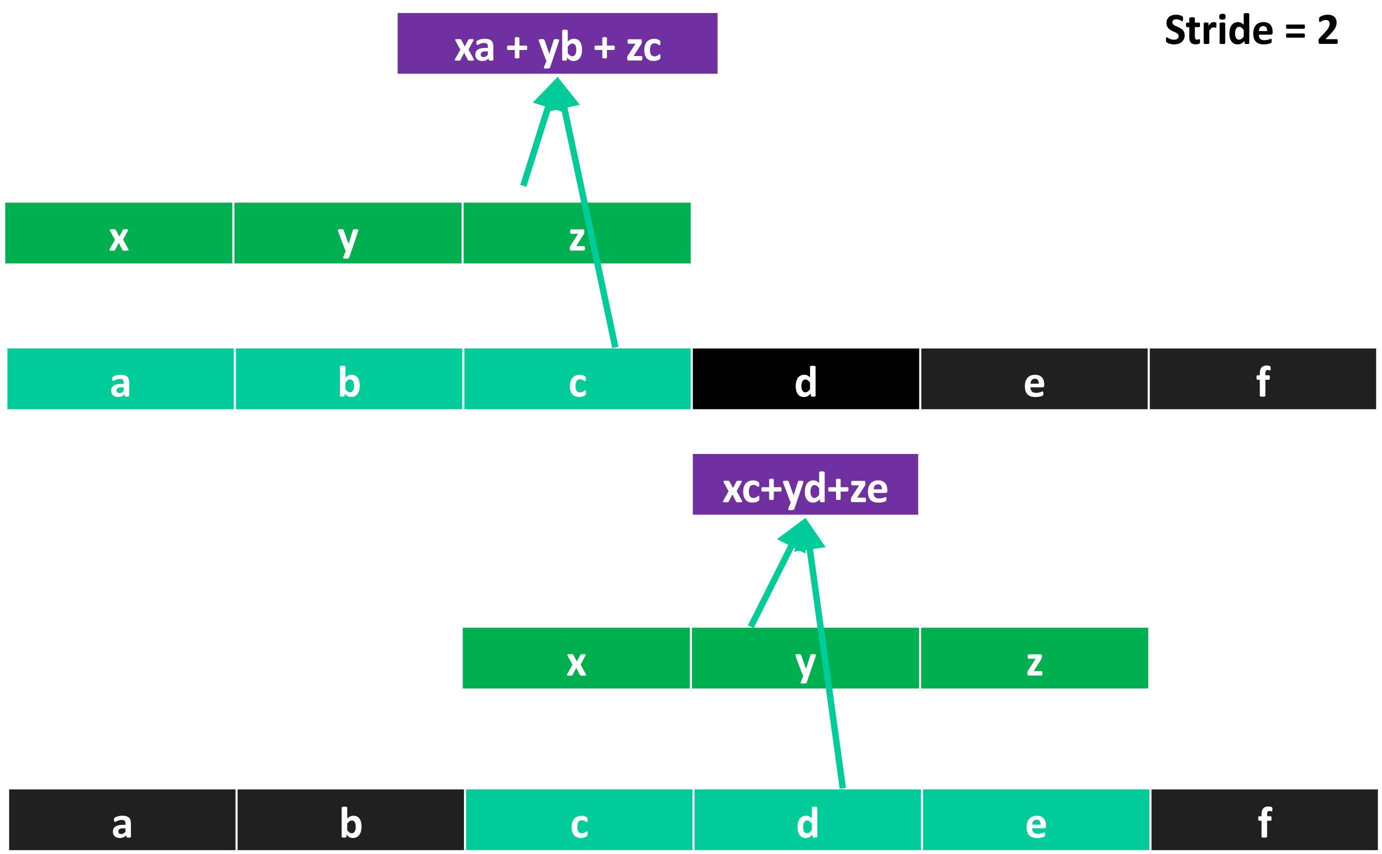
Convolution Operation

- Example:



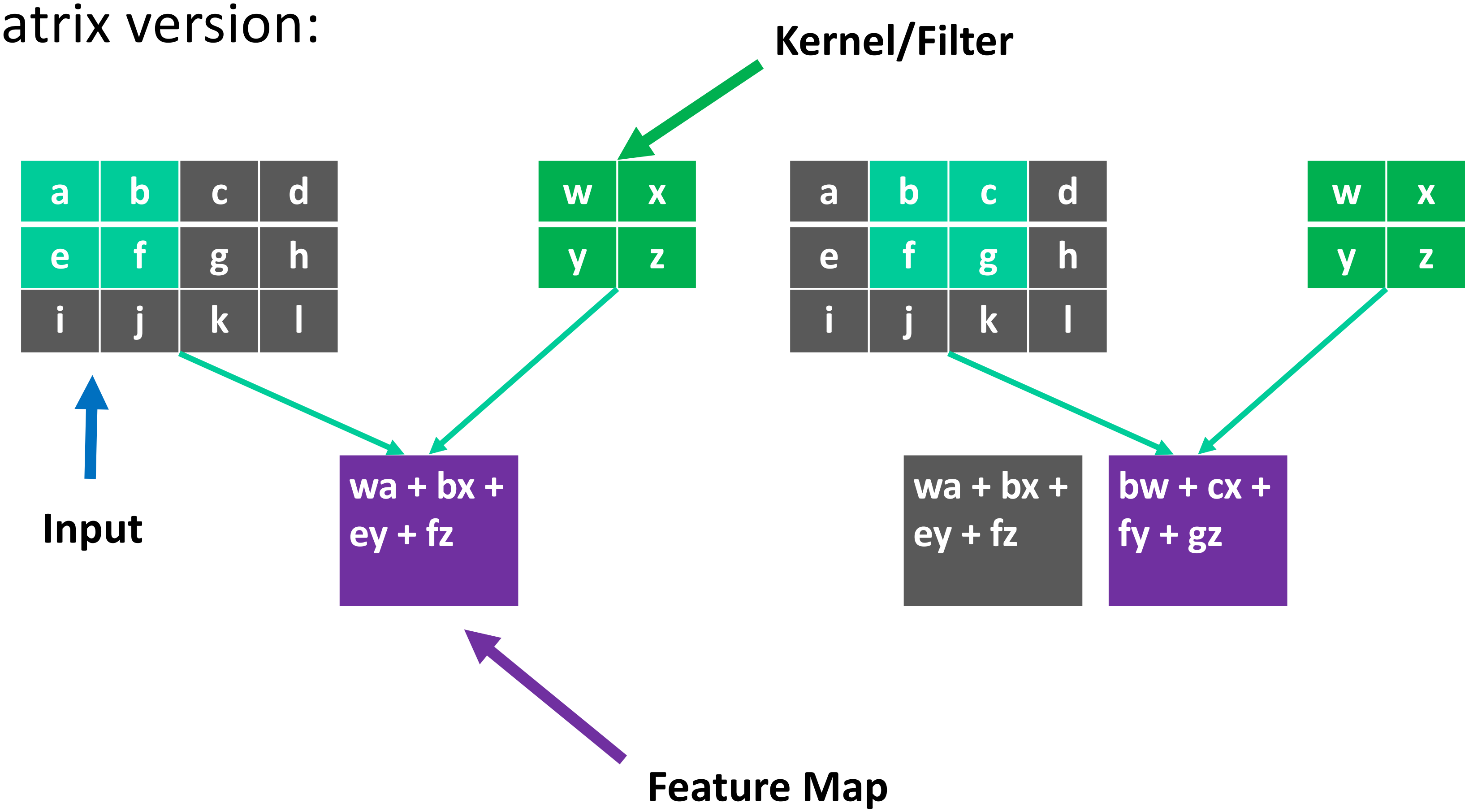
Convolution Operation

- Stride: # of positions we move per step



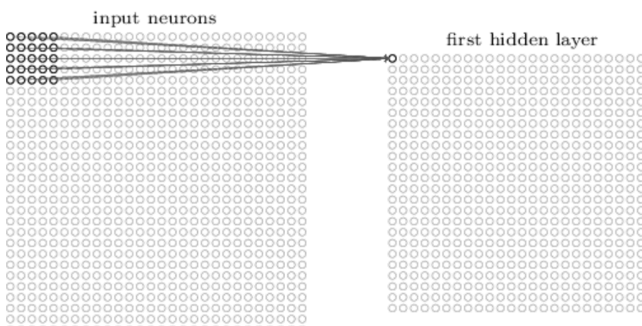
Convolution Operation

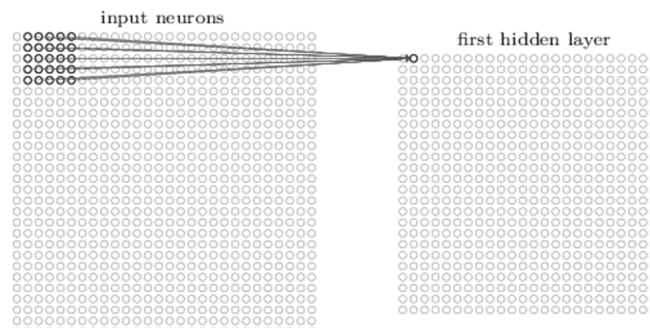
- Matrix version:



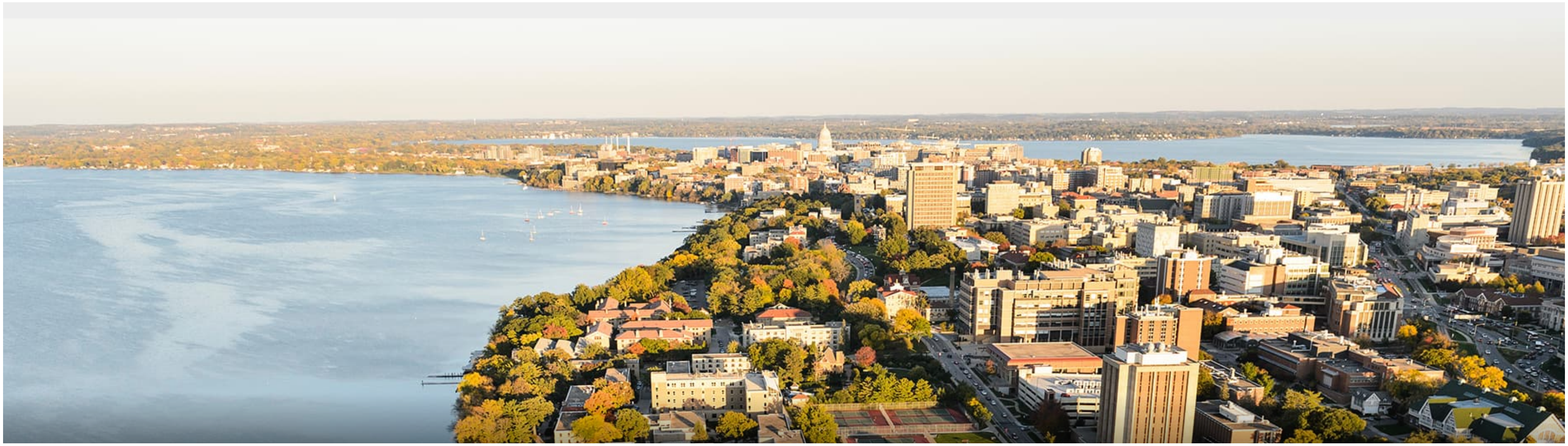
Convolution Operation

- All the units used the same set of weights (kernel)
- The units detect the same “feature” but at different locations





[Figure from neuralnetworksanddeeplearning.com]



Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, Sharon Li, Fred Sala