Neural Networks for Learning Counterfactual G-Invariances from Single Environments

a.k.a. "Fixing the Image Rotation Problem"

J. Setpal

February 24, 2023

Outline

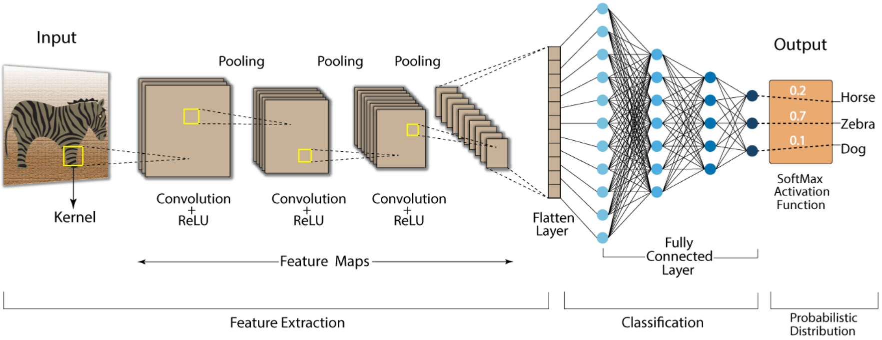
1. Task Description
2. A Lot of Linear Algebra
3. Fun Part

Outline

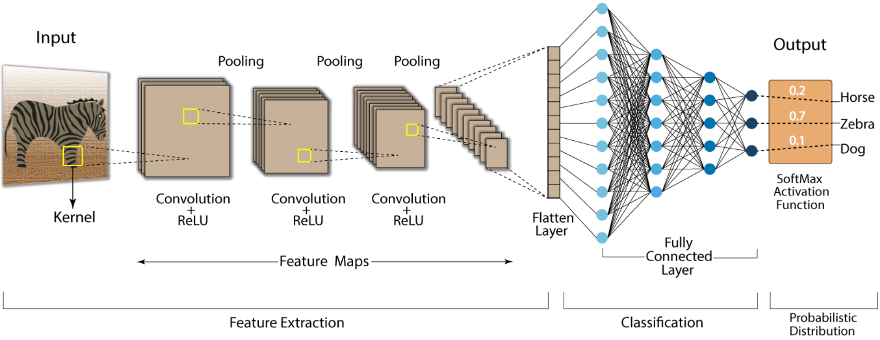
1. Task Description
2. A Lot of Linear Algebra
3. Fun Part

Introduction

Convolutional Neural Networks are *fantastic*. They efficiently extract a vast range of relevant contextual information and are resilient to pixel shift.



Convolutional Neural Networks are *fantastic*. They efficiently extract a vast range of relevant contextual information and are resilient to pixel shift.



However, they have a <u>critical flaw</u>.

Why aren't CNNs rotationally robust?

Q_1: Do you think that a CNN trained on a distribution of the left image *should* correctly classify the right image?

Why aren't CNNs rotationally robust?

Q_1: Do you think that a CNN trained on a distribution of the left image *should* correctly classify the right image?

A_1: Definitely!

Why aren't CNNs rotationally robust?

Q_1: Do you think that a CNN trained on a distribution of the left image *should* correctly classify the right image?

A_1: Definitely!

Q_2: Does that actually happen?

Why aren't CNNs rotationally robust?

Q_1: Do you think that a CNN trained on a distribution of the left image *should* correctly classify the right image?

A_1: Definitely!
Q_2: Does that actually happen?
A_2: Nope – all these images were misclassified :(

Why aren't CNNs rotationally robust?

Q_1: Do you think that a CNN trained on a distribution of the left image *should* correctly classify the right image?

A_1: Definitely!
Q_2: Does that actually happen?
A_2: Nope – all these images were misclassified :(
Q_3: How can we fix this?
A_3: Data Augmentation (boring)

Why aren't CNNs rotationally robust?

Q_1: Do you think that a CNN trained on a distribution of the left image *should* correctly classify the right image?

A_1: Definitely!
Q_2: Does that actually happen?
A_2: Nope – all these images were misclassified :(
Q_3: How can we fix this?
A_3: Data Augmentation (boring), **G-invariance** (fun)!

Why aren't CNNs rotationally robust?

Q_1: Do you think that a CNN trained on a distribution of the left image *should* correctly classify the right image?

A_1: Definitely!
Q_2: Does that actually happen?
A_2: Nope – all these images were misclassified :(
Q_3: How can we fix this?
A_3: Data Augmentation (boring), **G-invariance** (fun)!

Today, we'll explore a rotation invariant solution for an MLP. CNN's need **G-equivariance**, which we'll discuss some other time.

Outline

Understanding Groups

Groups: A set of elements G containing the following properties:

a. It has an operator that combines two elements. The operator is abstract. $g_1 \odot g_2 = g_3$

Understanding Groups

Groups: A set of elements G containing the following properties:

a. It has an operator that combines two elements. The operator is abstract. $g_1 \odot g_2 = g_3$

b. It is closed under operation. $g_1, g_2, g_3 \in G$

Understanding Groups

Groups: A set of elements G containing the following properties:

a. It has an operator that combines two elements. The operator is abstract. $g_1 \odot g_2 = g_3$
b. It is closed under operation. $g_1, g_2, g_3 \in G$
c. Inverses always exist. $g_n \odot g_n^{-1} = \mathcal{I}$

Understanding Groups

Groups: A set of elements G containing the following properties:

a. It has an operator that combines two elements. The operator is abstract. $g_1 \odot g_2 = g_3$

b. It is closed under operation. $g_1, g_2, g_3 \in G$

c. Inverses always exist. $g_n \odot g_n^{-1} = \mathcal{I}$

d. There exists an identity. $g_n \odot \mathcal{I} = g_n$

Understanding Groups

Groups: A set of elements G containing the following properties:

a. It has an operator that combines two elements. The operator is abstract. $g_1 \odot g_2 = g_3$

b. It is closed under operation. $g_1, g_2, g_3 \in G$

c. Inverses always exist. $g_n \odot g_n^{-1} = \mathcal{I}$

d. There exists an identity. $g_n \odot \mathcal{I} = g_n$

e. The operation is associative. $g_1 \odot (g_2 \odot g_3) = (g_1 \odot g_2) \odot g_3$

Understanding Groups

Groups: A set of elements G containing the following properties:

a. It has an operator that combines two elements. The operator is abstract. $g_1 \odot g_2 = g_3$

b. It is closed under operation. $g_1, g_2, g_3 \in G$

c. Inverses always exist. $g_n \odot g_n^{-1} = \mathcal{I}$

d. There exists an identity. $g_n \odot \mathcal{I} = g_n$

e. The operation is associative. $g_1 \odot (g_2 \odot g_3) = (g_1 \odot g_2) \odot g_3$

Optionally, it is commutative. $g_1 \odot g_2 = g_2 \odot g_1$.

Understanding Groups

Groups: A set of elements G containing the following properties:

a. It has an operator that combines two elements. The operator is abstract. $g_1 \odot g_2 = g_3$
b. It is closed under operation. $g_1, g_2, g_3 \in G$
c. Inverses always exist. $g_n \odot g_n^{-1} = \mathcal{I}$
d. There exists an identity. $g_n \odot \mathcal{I} = g_n$
e. The operation is associative. $g_1 \odot (g_2 \odot g_3) = (g_1 \odot g_2) \odot g_3$

Optionally, it is commutative. $g_1 \odot g_2 = g_2 \odot g_1$. If it is commutative, the group is called **abelian**.

Understanding Groups

Groups: A set of elements G containing the following properties:

a. It has an operator that combines two elements. The operator is abstract. $g_1 \odot g_2 = g_3$
b. It is closed under operation. $g_1, g_2, g_3 \in G$
c. Inverses always exist. $g_n \odot g_n^{-1} = \mathcal{I}$
d. There exists an identity. $g_n \odot \mathcal{I} = g_n$
e. The operation is associative. $g_1 \odot (g_2 \odot g_3) = (g_1 \odot g_2) \odot g_3$

Optionally, it is commutative. $g_1 \odot g_2 = g_2 \odot g_1$. If it is commutative, the group is called **abelian**.

Q: Why is it important?

Understanding Groups

Groups: A set of elements G containing the following properties:

a. It has an operator that combines two elements. The operator is abstract. $g_1 \odot g_2 = g_3$
b. It is closed under operation. $g_1, g_2, g_3 \in G$
c. Inverses always exist. $g_n \odot g_n^{-1} = \mathcal{I}$
d. There exists an identity. $g_n \odot \mathcal{I} = g_n$
e. The operation is associative. $g_1 \odot (g_2 \odot g_3) = (g_1 \odot g_2) \odot g_3$

Optionally, it is commutative. $g_1 \odot g_2 = g_2 \odot g_1$. If it is commutative, the group is called **abelian**.

Q: Why is it important?
A: These are the axioms on which we define our solution to the rotation problem. Only if these axioms hold true can our solution exist.

The General Linear Group

It's a special group G consisting of $n \times n$ matrices with matrix product as the defined operation. Formally,

$$GL_n : (M_{n \times n}(\mathbb{R}), \cdot)$$

The General Linear Group

It's a special group G consisting of $n \times n$ matrices with matrix product as the defined operation. Formally,

$$GL_n : (M_{n\times n}(\mathbb{R}), \cdot)$$

Our training dataset contains inputs $x \in \mathbb{R}^{3n^2}$.

The General Linear Group

It's a special group G consisting of $n \times n$ matrices with matrix product as the defined operation. Formally,

$$GL_n : (M_{n \times n}(\mathbb{R}), \cdot)$$

Our training dataset contains inputs $x \in \mathbb{R}^{3n^2}$.
Therefore $GL_{3n^2} : (M_{3n^2}(\mathbb{R}), \cdot)$ represents images within our dataset.

The General Linear Group

It's a special group G consisting of $n \times n$ matrices with matrix product as the defined operation. Formally,

$$GL_n : (M_{n \times n}(\mathbb{R}), \cdot)$$

Our training dataset contains inputs $x \in \mathbb{R}^{3n^2}$.
Therefore $GL_{3n^2} : (M_{3n^2}(\mathbb{R}), \cdot)$ represents images within our dataset.

Now, we can define *transformations* that can be performed on this group – the only restriction being the axioms of a group.

Setting Up Transformations

To set up the transformations we can run, we define another group, called a **Transformation Group**. We assume the same prior input $x \in \mathbb{R}^{3n^2}$

Setting Up Transformations

To set up the transformations we can run, we define another group, called a **Transformation Group**. We assume the same prior input $x \in \mathbb{R}^{3n^2}$

$$G_{rot} \equiv \left\{ T^{\theta} \right\}_{\theta \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}} \tag{1}$$

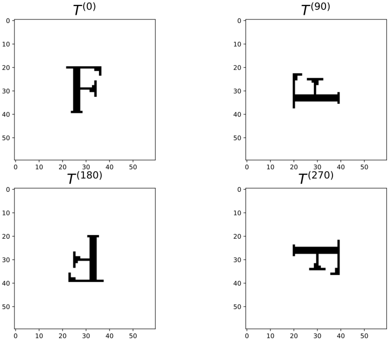
$$G_{flip} \equiv \left\{ T^{v}, T^{h}, T^{180^\circ}, T^{0^\circ} \right\} \tag{2}$$

Setting Up Transformations

To set up the transformations we can run, we define another group, called a **Transformation Group**. We assume the same prior input $x \in \mathbb{R}^{3n^2}$

$$G_{rot} \equiv \left\{ T^{\theta} \right\}_{\theta \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}} \tag{1}$$

$$G_{flip} \equiv \left\{ T^{v}, T^{h}, T^{180^\circ}, T^{0^\circ} \right\} \tag{2}$$

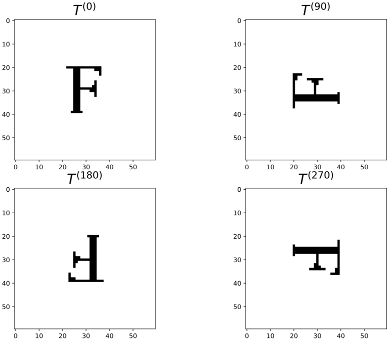


Setting Up Transformations

To set up the transformations we can run, we define another group, called a **Transformation Group**. We assume the same prior input $x \in \mathbb{R}^{3n^2}$

$$G_{rot} \equiv \left\{ T^{\theta} \right\}_{\theta \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}} \quad (1)$$

$$G_{flip} \equiv \left\{ T^{v}, T^{h}, T^{180^\circ}, T^{0^\circ} \right\} \quad (2)$$



Both groups are defined $G : \mathbb{R}^{3n^2} \to \mathbb{R}^{3n^2}$

Outline

The G-Invariant Neuron

Our objective is to learn the optimal weight on a layer such that

$$\sigma(w^T x + b) = \sigma(w^T T_G x + b)$$

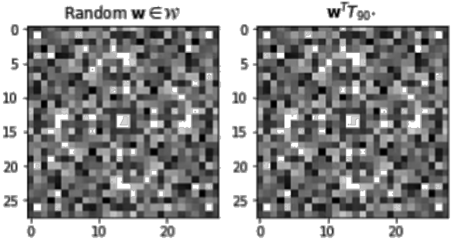
where T_G is the transformation group.

The G-Invariant Neuron

Our objective is to learn the optimal weight on a layer such that

$$\sigma(w^T x + b) = \sigma(w^T T_G x + b)$$

where T_G is the transformation group.



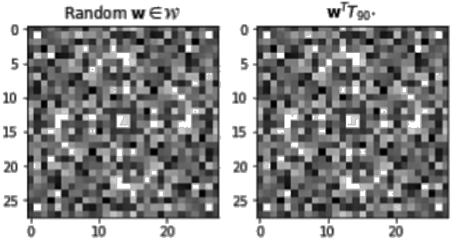
This latent subspace is transformation invariant!

The G-Invariant Neuron

Our objective is to learn the optimal weight on a layer such that

$$\sigma(w^T x + b) = \sigma(w^T T_G x + b)$$

where T_G is the transformation group.



This latent subspace is transformation invariant!

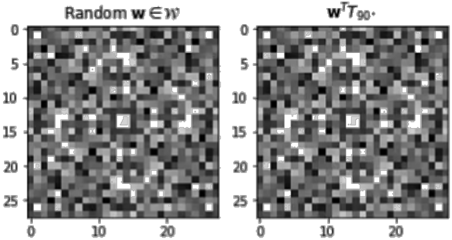
If the above equation holds, that means that our layer is G-invariant.

The G-Invariant Neuron

Our objective is to learn the optimal weight on a layer such that

$$\sigma(w^T x + b) = \sigma(w^T T_G x + b)$$

where T_G is the transformation group.



This latent subspace is transformation invariant!

If the above equation holds, that means that our layer is G-invariant.
Q: How can we go about finding this?

Reynolds Operator

A given transformation is G-invariant if,

$$T_1(T_2x) = T_1x;\, T_2 \in G, x \in \mathbb{R}^{3n^2}$$

So the objective is to find T_2, formally called the **Reynolds Operator**.

Reynolds Operator

A given transformation is G-invariant if,

$$T_1(T_2 x) = T_1 x; T_2 \in G, x \in \mathbb{R}^{3n^2}$$

So the objective is to find T_2, formally called the **Reynolds Operator**. Incidentally, we can find this using mean of the group:

$$T_2 = \frac{1}{|G|} \sum_{g \in G}^{G} g$$

Reynolds Operator

A given transformation is G-invariant if,

$$T_1(T_2x) = T_1x;\, T_2 \in G, x \in \mathbb{R}^{3n^2}$$

So the objective is to find T_2, formally called the **Reynolds Operator**. Incidentally, we can find this using mean of the group:

$$T_2 = \frac{1}{|G|}\sum_{g \in G}^{G} g$$

Extracting the left eigenvectors of T_2, we obtain v_n. By definition of the eigenvectors:

$$\begin{aligned} v_i T_2 &= \lambda_i v_i \\ v_i T_2 &= v_i \end{aligned}$$

T_2 is a projection operator

Defining the Output Latent Space

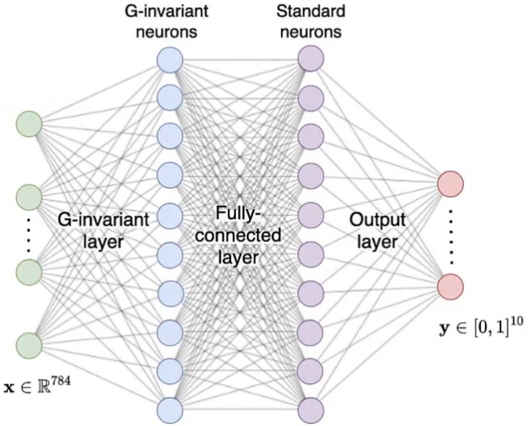
We can now define our weights such that,

$$w^T = \sum_{i=1}^{k} \alpha_i v_i^T$$

where T stands for transpose, & α_i is arbitrary

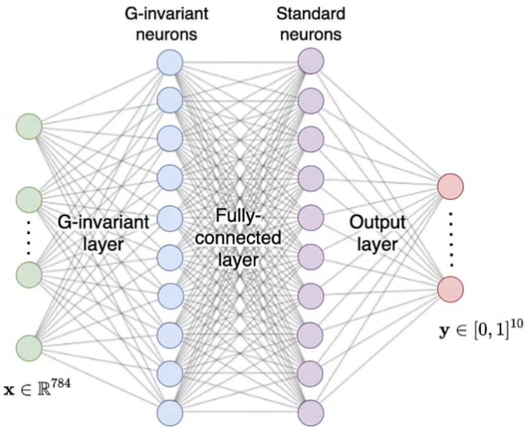
Bringing it all Together

In order to set up our complete neural network, all we have to do is prepend the G-invariant layer we just built to our standard model.



Bringing it all Together

In order to set up our complete neural network, all we have to do is prepend the G-invariant layer we just built to our standard model.



This will ensure that we build a feature space that's invariant to rotation!

Executing Code

Let's classify!!

If you can view this screen, I am making a mistake.

Thank you!

Have an awesome rest of your day!

Slides: `https://cs.purdue.edu/homes/jsetpal/g-invariance.pdf`