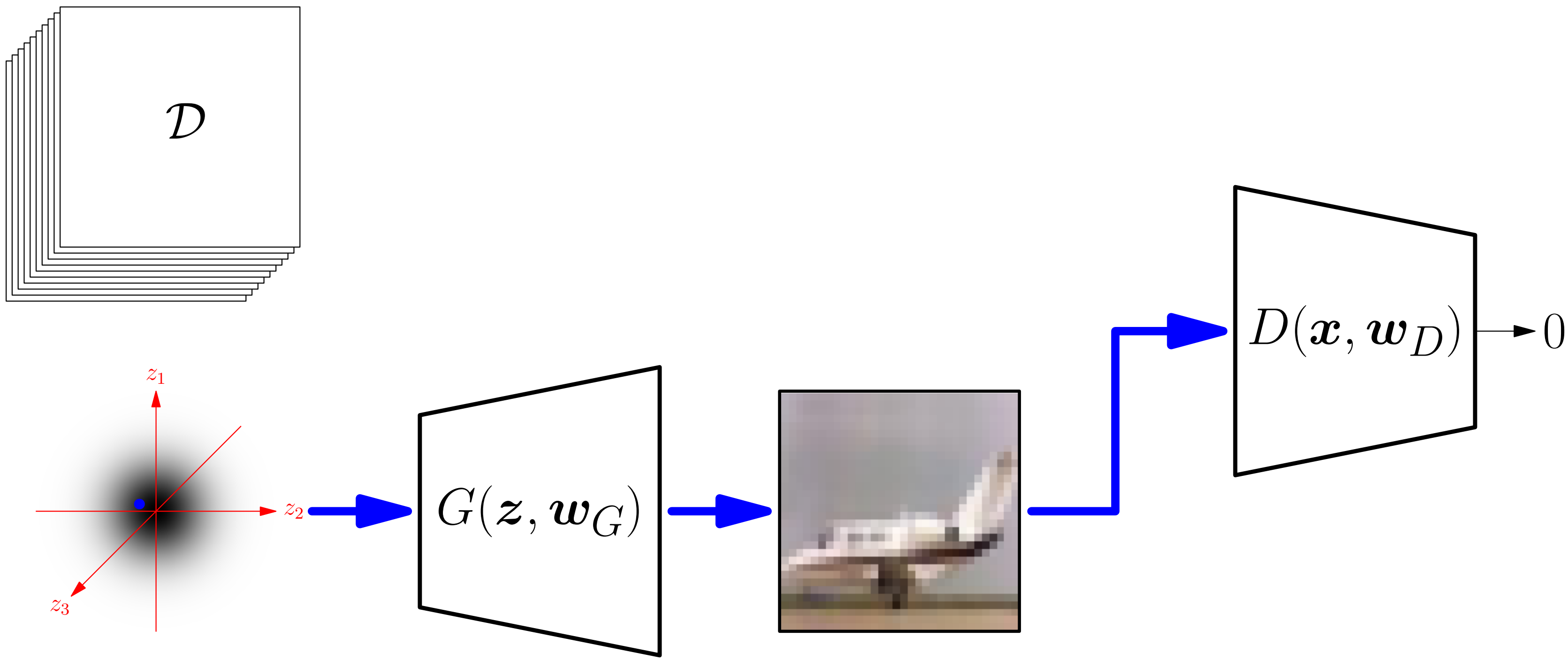
Advanced Machine Learning

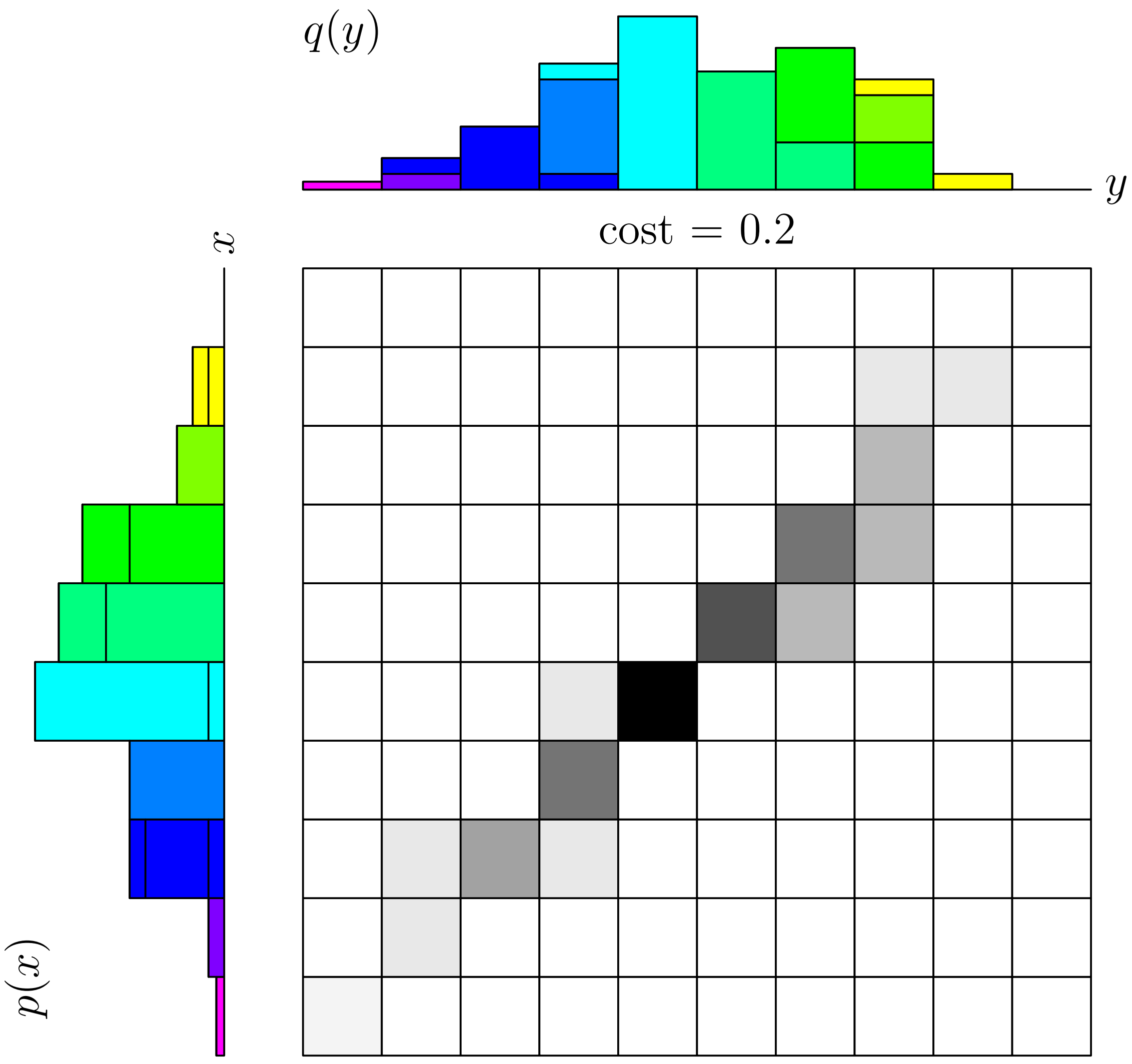
Wasserstein GANs



GANs, Wasserstein distance, Duality, WGANs

Outline

1. **GANs**
2. Wasserstein Distance
3. Wasserstein GANs



Generative Adversarial Networks

- One of the applications of Deep Learning that has most excited the public are **Generative Adversarial Networks** or GANs
- Their aim is to generate new random samples from the same distribution as some training set, $\mathcal{D}$
- Their number of real world applications are

Generative Adversarial Networks

- One of the applications of Deep Learning that has most excited the public are **Generative Adversarial Networks** or GANs
- Their aim is to generate new random samples from the same distribution as some training set, $\mathcal{D}$
- Their number of real world applications are

Generative Adversarial Networks

- One of the applications of Deep Learning that has most excited the public are **Generative Adversarial Networks** or GANs
- Their aim is to generate new random samples from the same distribution as some training set, $\mathcal{D}$
- Their number of real world applications are

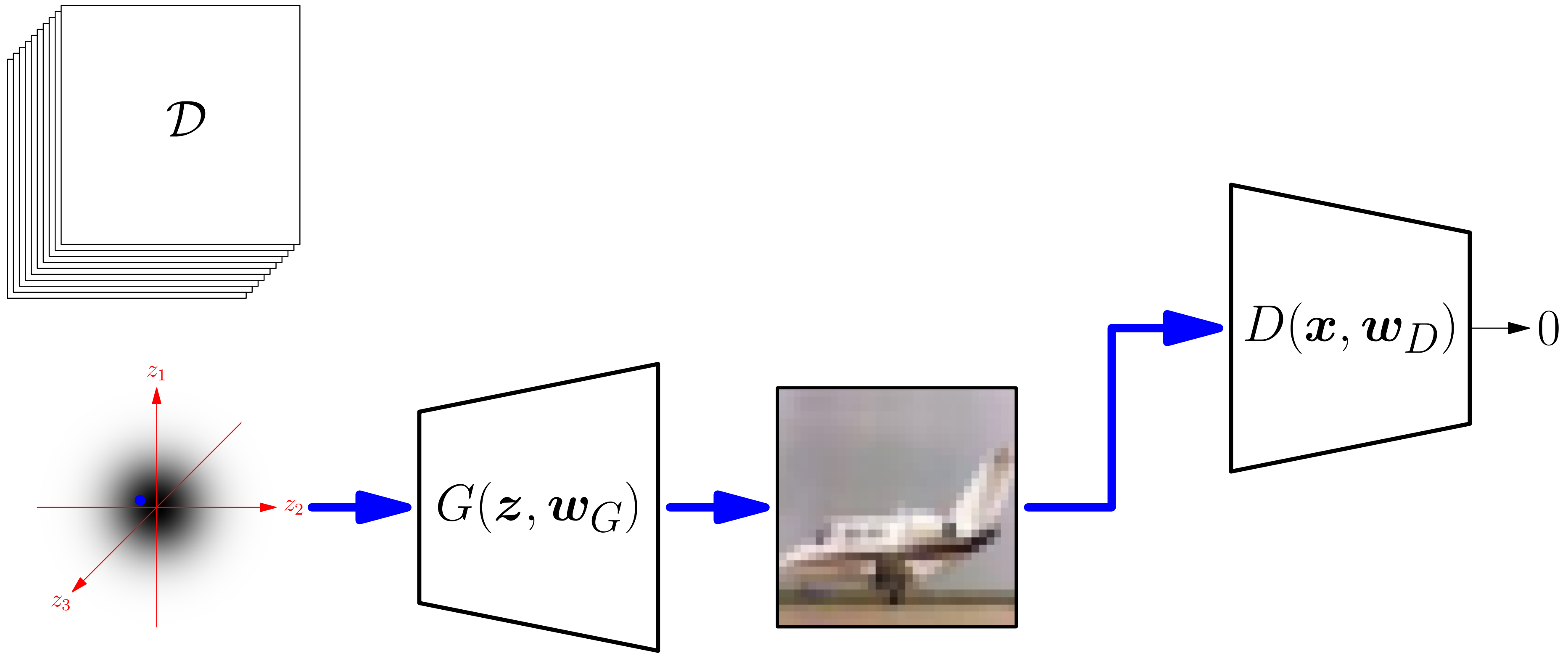
Generative Adversarial Networks

- One of the applications of Deep Learning that has most excited the public are **Generative Adversarial Networks** or GANs
- Their aim is to generate new random samples from the same distribution as some training set, $\mathcal{D}$
- Their number of real world applications are questionable

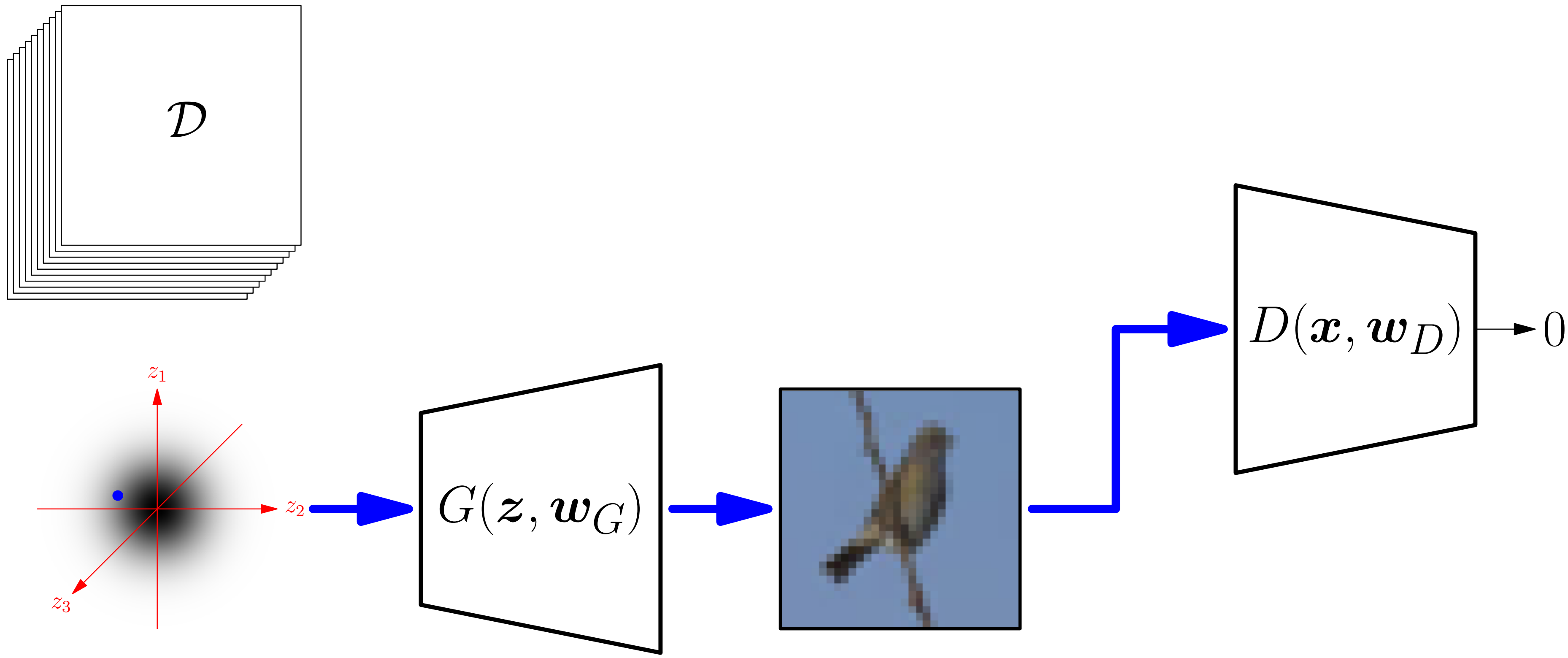
Generative Adversarial Networks

- One of the applications of Deep Learning that has most excited the public are **Generative Adversarial Networks** or GANs
- Their aim is to generate new random samples from the same distribution as some training set, $\mathcal{D}$
- Their number of real world applications are questionable
- But nobody cares because they are cool!

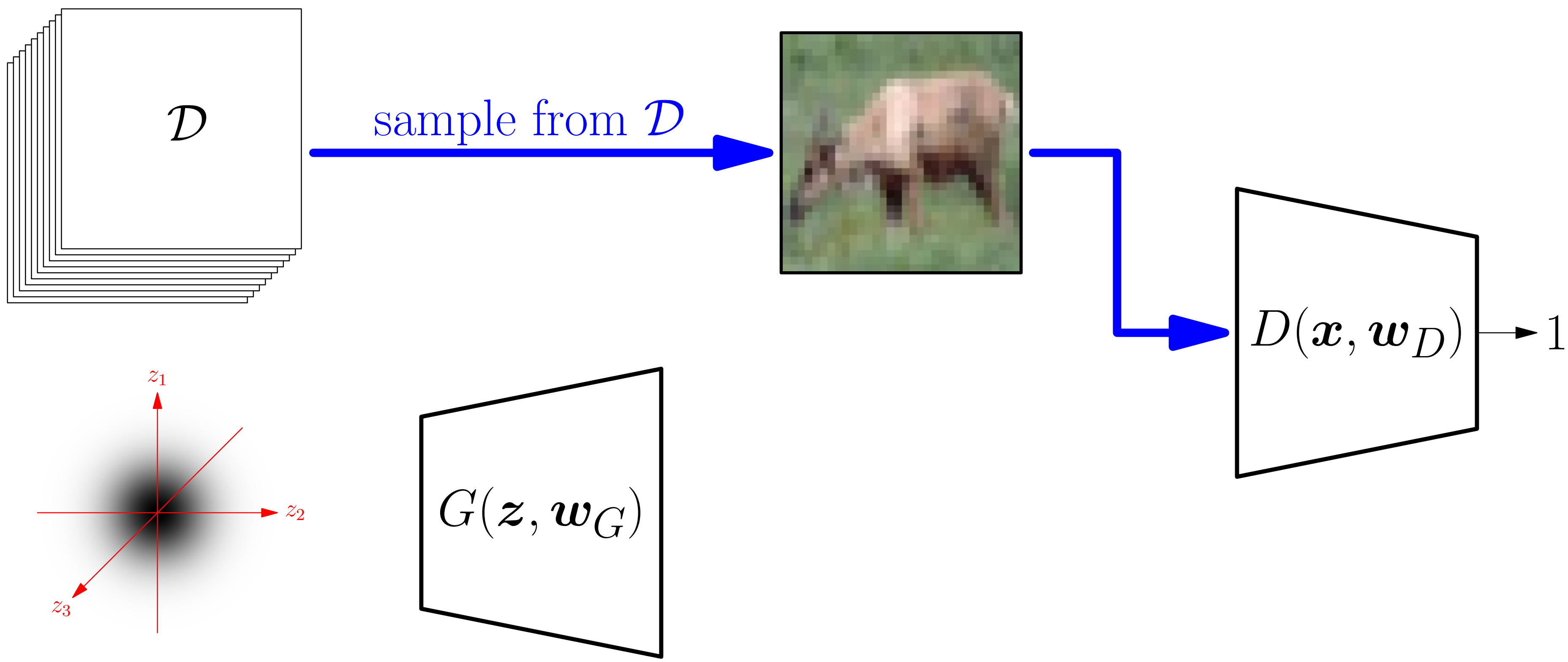
How GANs Work



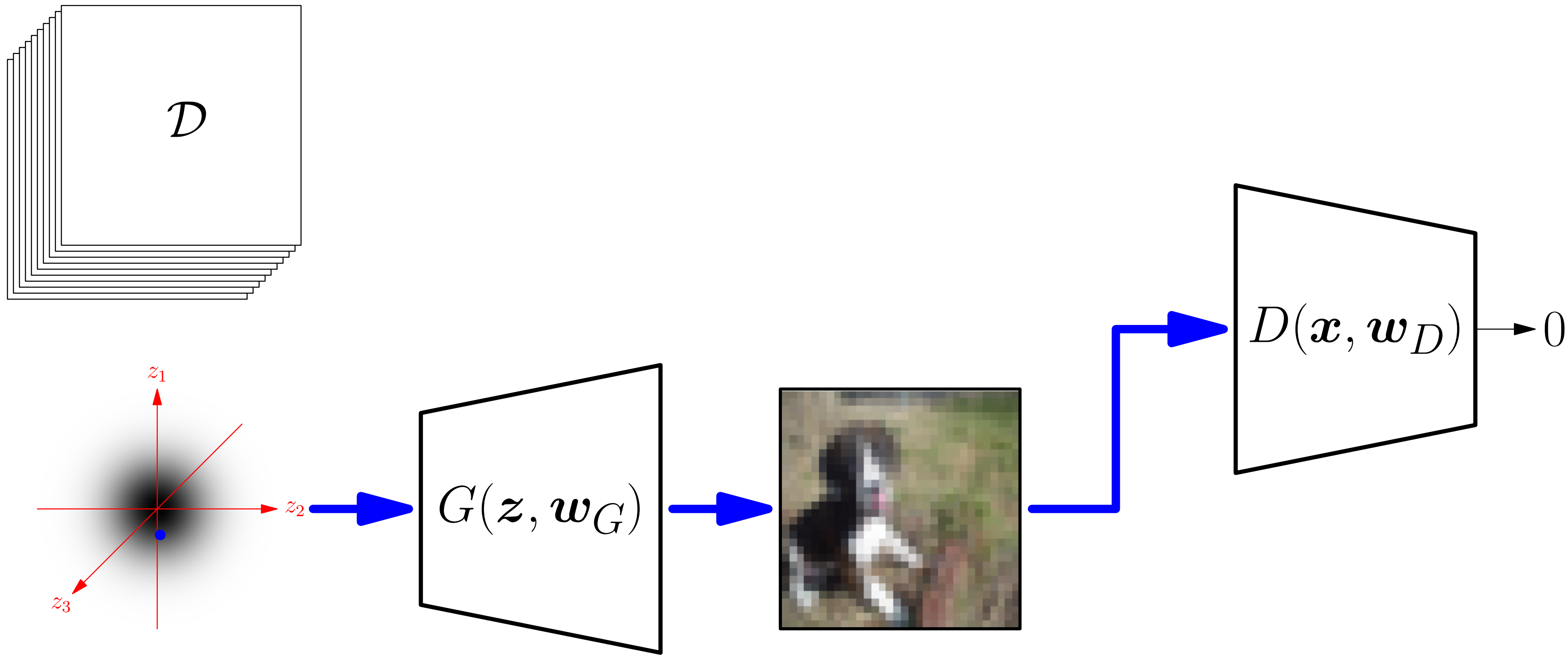
How GANs Work



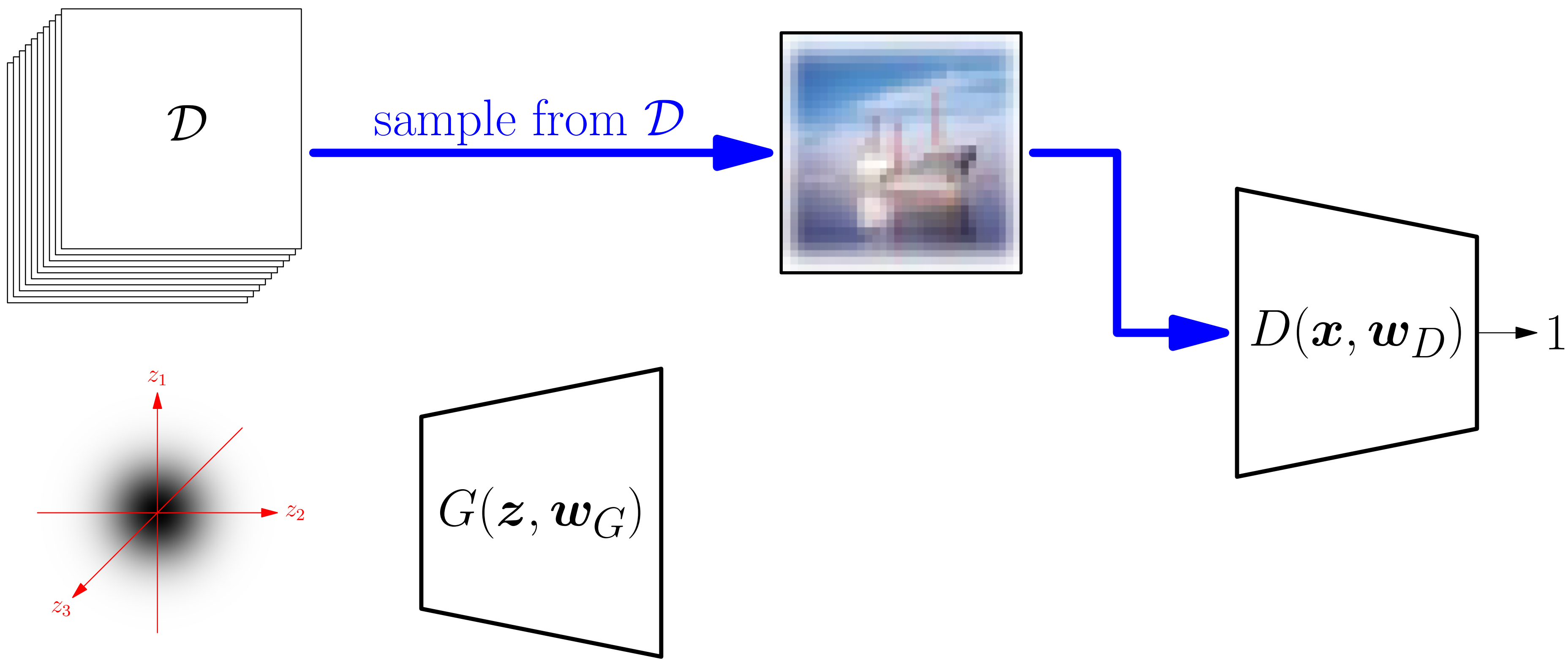
How GANs Work



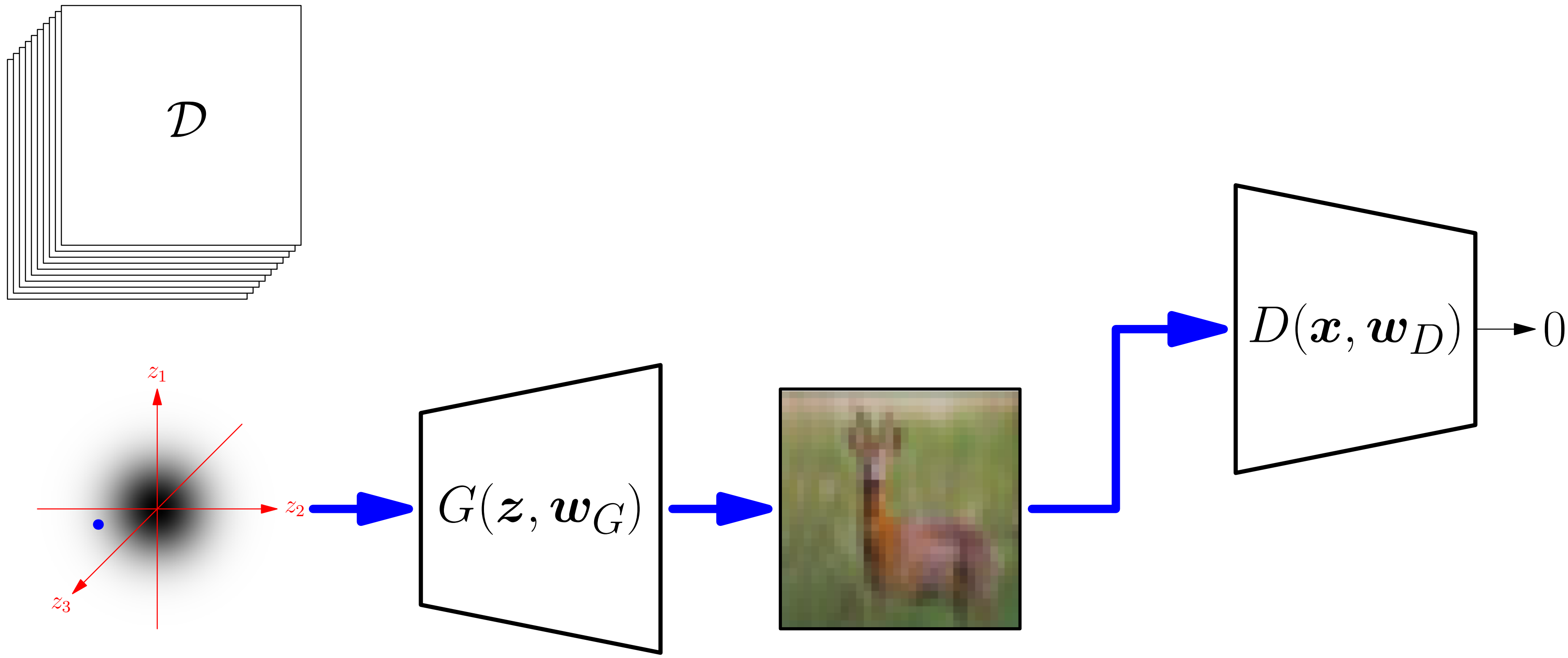
How GANs Work



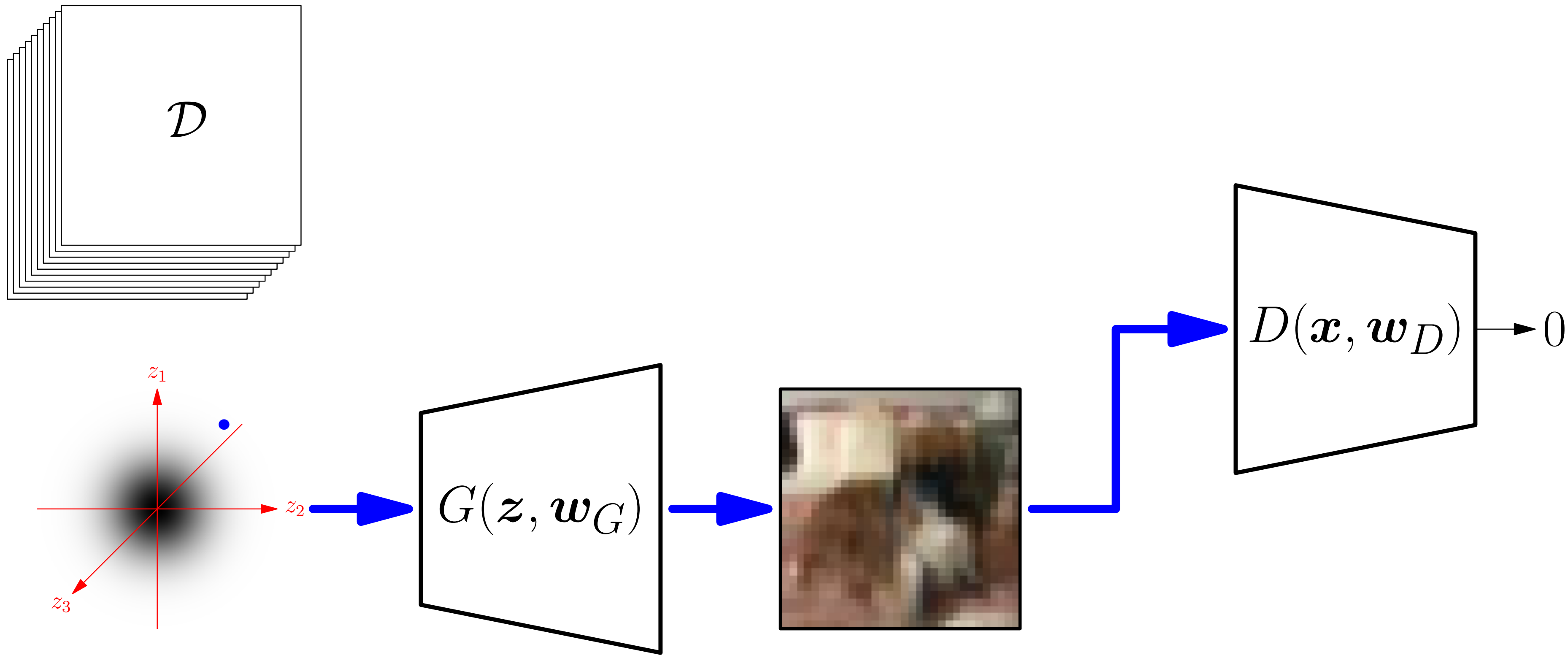
How GANs Work



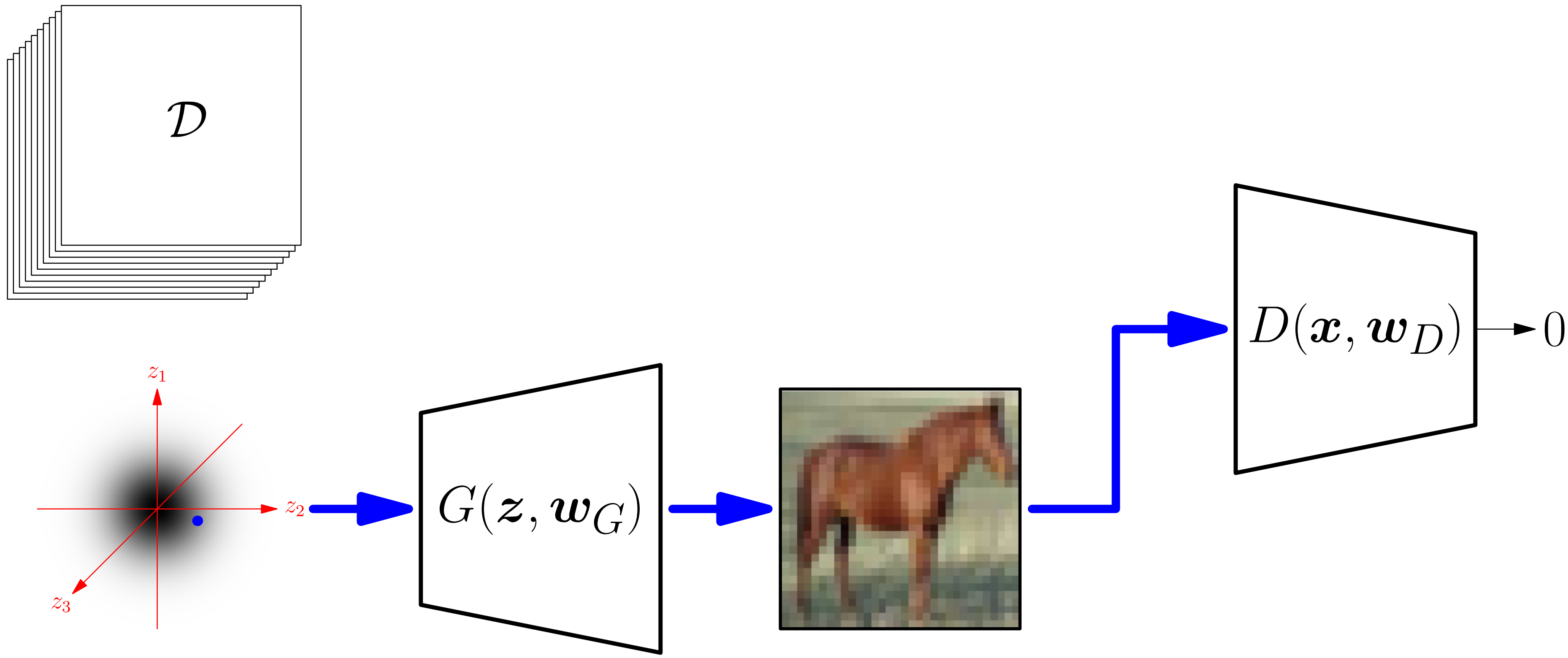
How GANs Work



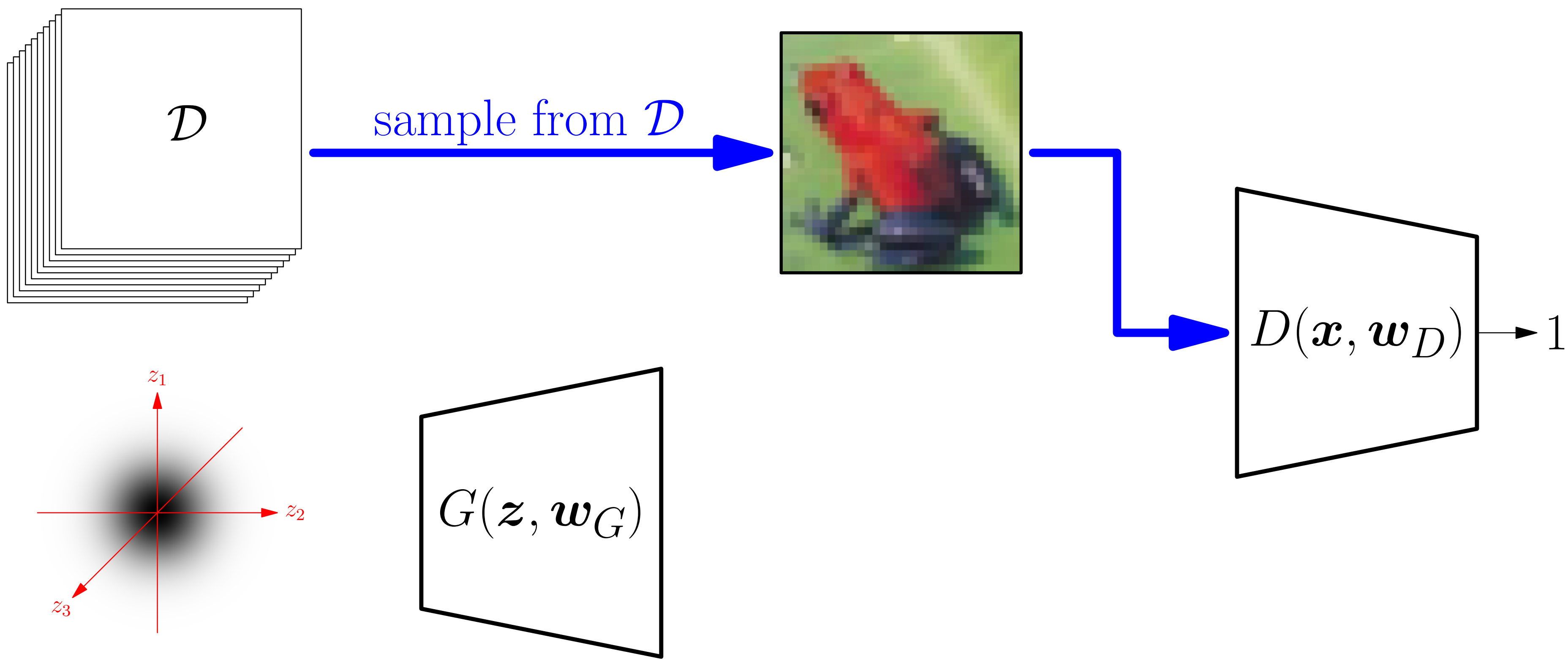
How GANs Work



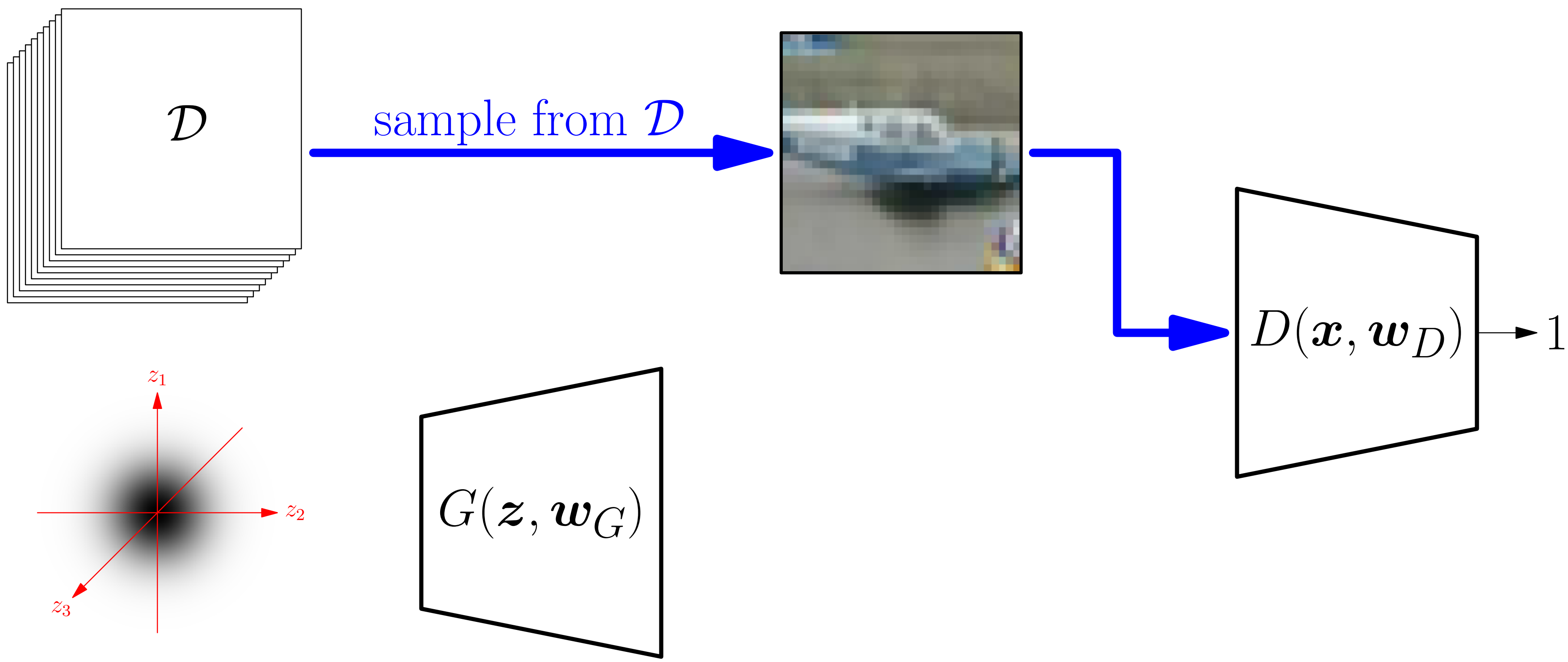
How GANs Work



How GANs Work



How GANs Work



Training GANs

- The loss of the generator depends on its ability to trick the discriminator
- The loss of the discriminator depends on its ability not to be tricked
- We try to train the two networks simultaneously
- We hope that over time the generator produces better and better fakes

Training GANs

- The loss of the generator depends on its ability to trick the discriminator
- The loss of the discriminator depends on its ability not to be tricked
- We try to train the two networks simultaneously
- We hope that over time the generator produces better and better fakes

Training GANs

- The loss of the generator depends on its ability to trick the discriminator
- The loss of the discriminator depends on its ability not to be tricked
- We try to train the two networks simultaneously
- We hope that over time the generator produces better and better fakes

Training GANs

- The loss of the generator depends on its ability to trick the discriminator
- The loss of the discriminator depends on its ability not to be tricked
- We try to train the two networks simultaneously
- We hope that over time the generator produces better and better fakes

Problems of GANs

- GANs are notoriously difficult to train
- The generator and discriminator training can decouple
- Often the discriminator becomes too good at correctly identifying the generated images
- Then there can be little gradient information to help the generator as every small change in parameters doesn't significantly change the discriminator decision
- To try to solve this problem we first make a seemly unconnected diversion

Problems of GANs

- GANs are notoriously difficult to train
- The generator and discriminator training can decouple
- Often the discriminator becomes too good at correctly identifying the generated images
- Then there can be little gradient information to help the generator as every small change in parameters doesn't significantly change the discriminator decision
- To try to solve this problem we first make a seemly unconnected diversion

Problems of GANs

- GANs are notoriously difficult to train
- The generator and discriminator training can decouple
- Often the discriminator becomes too good at correctly identifying the generated images
- Then there can be little gradient information to help the generator as every small change in parameters doesn't significantly change the discriminator decision
- To try to solve this problem we first make a seemly unconnected diversion

Problems of GANs

- GANs are notoriously difficult to train
- The generator and discriminator training can decouple
- Often the discriminator becomes too good at correctly identifying the generated images
- Then there can be little gradient information to help the generator as every small change in parameters doesn't significantly change the discriminator decision
- To try to solve this problem we first make a seemly unconnected diversion

Problems of GANs

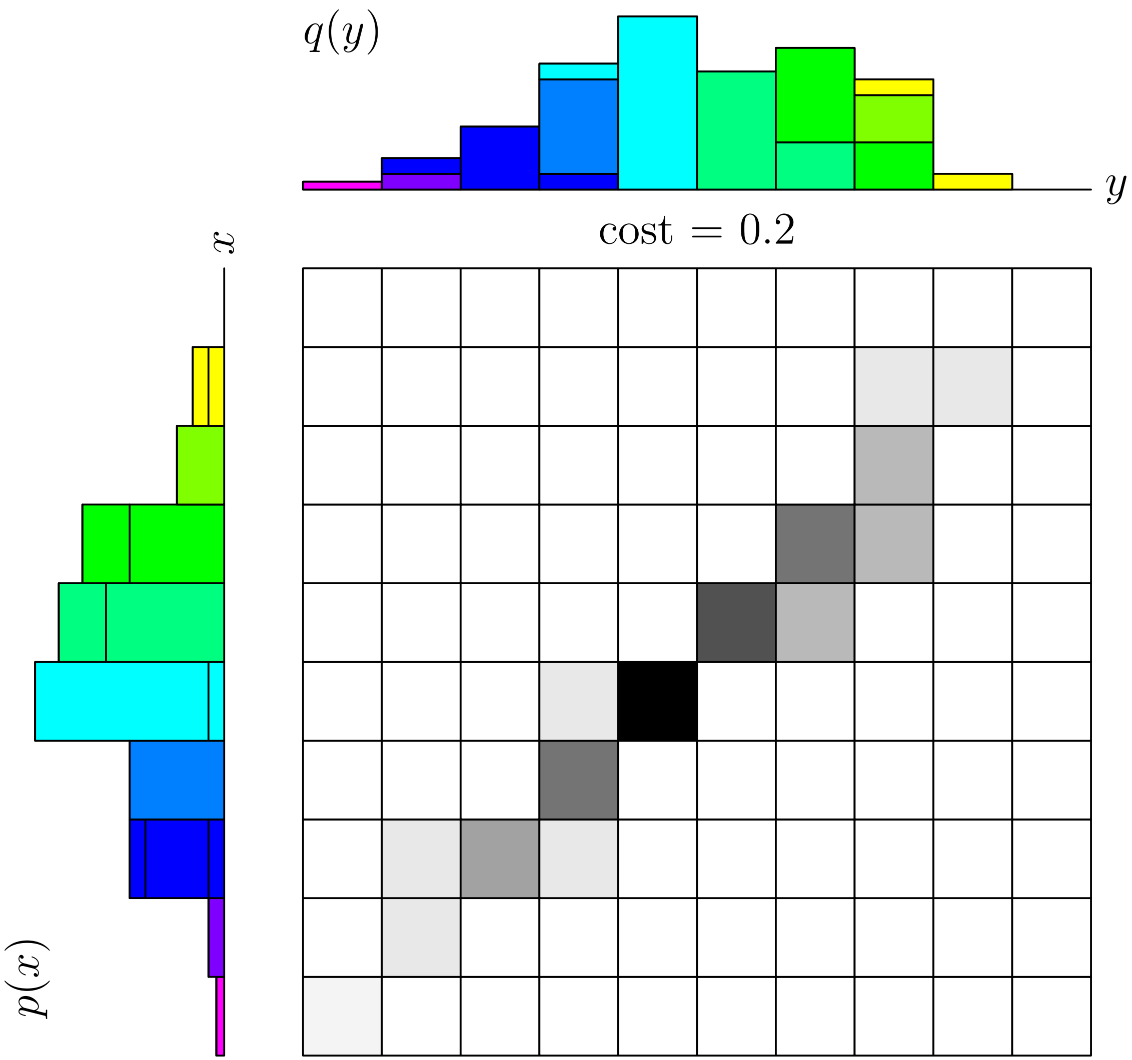
- GANs are notoriously difficult to train
- The generator and discriminator training can decouple
- Often the discriminator becomes too good at correctly identifying the generated images
- Then there can be little gradient information to help the generator as every small change in parameters doesn't significantly change the discriminator decision
- To try to solve this problem we first make a seemly unconnected diversion

Outline

1. GANs

2. **Wasserstein Distance**

3. Wasserstein GANs



Measuring Distances Between Distributions

- In many machine learning tasks we want to minimise the distance between two probability distributions
- This requires that we can measure distances between probability distributions
- One prominent measure is the Kullback-Leibler or KL divergence

$$\mathrm{KL}(p\|q) = \int p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{y})}\right) \mathrm{d}\boldsymbol{x}$$

- This is very commonly used in ML (e.g. VAEs, Variational Approximation)

Measuring Distances Between Distributions

- In many machine learning tasks we want to minimise the distance between two probability distributions
- This requires that we can measure distances between probability distributions
- One prominent measure is the Kullback-Leibler or KL divergence

$$\mathrm{KL}(p\|q) = \int p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{y})}\right) \mathrm{d}\boldsymbol{x}$$

- This is very commonly used in ML (e.g. VAEs, Variational Approximation)

Measuring Distances Between Distributions

- In many machine learning tasks we want to minimise the distance between two probability distributions
- This requires that we can measure distances between probability distributions
- One prominent measure is the Kullback-Leibler or KL divergence

$$\mathrm{KL}(p\|q) = \int p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{y})}\right) \mathrm{d}\boldsymbol{x}$$

- This is very commonly used in ML (e.g. VAEs, Variational Approximation)

Measuring Distances Between Distributions

- In many machine learning tasks we want to minimise the distance between two probability distributions
- This requires that we can measure distances between probability distributions
- One prominent measure is the Kullback-Leibler or KL divergence

$$\mathrm{KL}(p\|q) = \int p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{y})}\right) \mathrm{d}\boldsymbol{x}$$

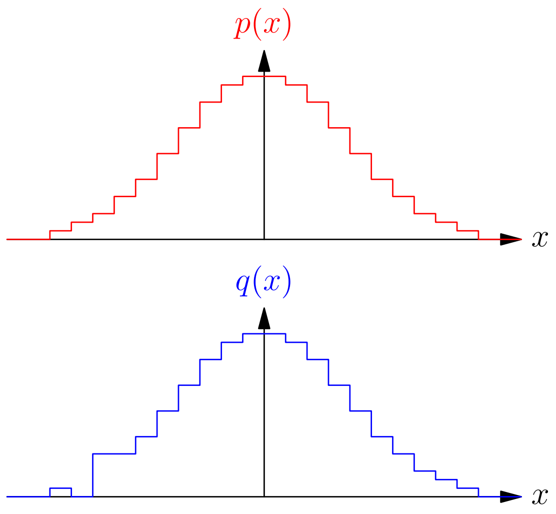
- This is very commonly used in ML (e.g. VAEs, Variational Approximation)

Trouble with KL

- KL-divergences are non-negative quantities that are minimised when the two probability distributions are the same

- They are not distances (they aren't symmetric and they don't satisfy the triangular inequality)

- We don't really care about this, but what we do care about is that if $q(\boldsymbol{x}) = 0$ when $p(\boldsymbol{x}) \neq 0$ then $\log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{y})}\right)$ diverges



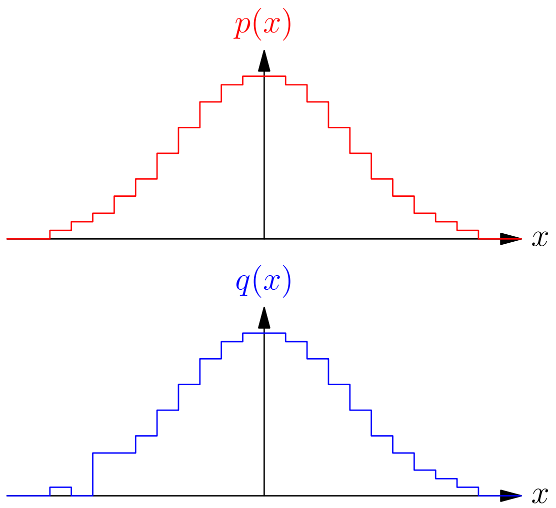
- We can therefore have distributes that seem very similar but their KL-divergence is huge (or infinite)

Trouble with KL

- KL-divergences are non-negative quantities that are minimised when the two probability distributions are the same

- They are not distances (they aren't symmetric and they don't satisfy the triangular inequality)

- We don't really care about this, but what we do care about is that if $q(\boldsymbol{x}) = 0$ when $p(\boldsymbol{x}) \neq 0$ then $\log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{y})}\right)$ diverges



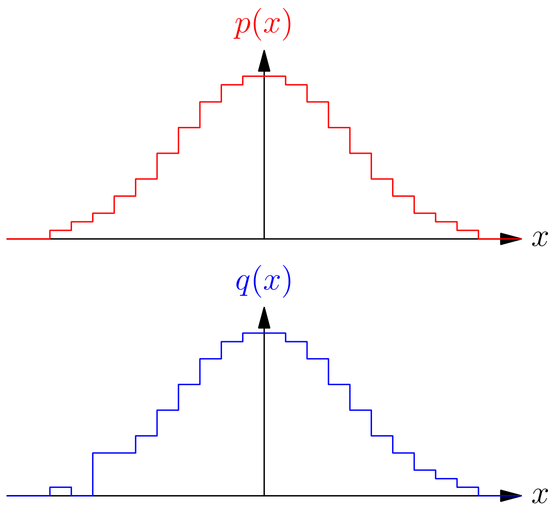
- We can therefore have distributes that seem very similar but their KL-divergence is huge (or infinite)

Trouble with KL

- KL-divergences are non-negative quantities that are minimised when the two probability distributions are the same

- They are not distances (they aren't symmetric and they don't satisfy the triangular inequality)

- We don't really care about this, but what we do care about is that if $q(\boldsymbol{x}) = 0$ when $p(\boldsymbol{x}) \neq 0$ then $\log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{y})}\right)$ diverges



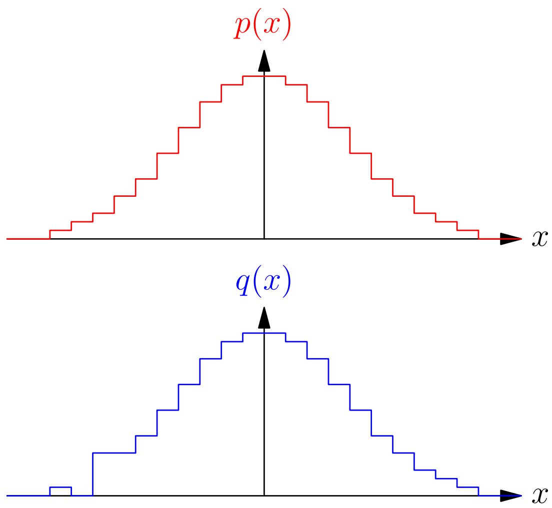
- We can therefore have distributes that seem very similar but their KL-divergence is huge (or infinite)

Trouble with KL

- KL-divergences are non-negative quantities that are minimised when the two probability distributions are the same

- They are not distances (they aren't symmetric and they don't satisfy the triangular inequality)

- We don't really care about this, but what we do care about is that if $q(\boldsymbol{x}) = 0$ when $p(\boldsymbol{x}) \neq 0$ then $\log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{y})}\right)$ diverges

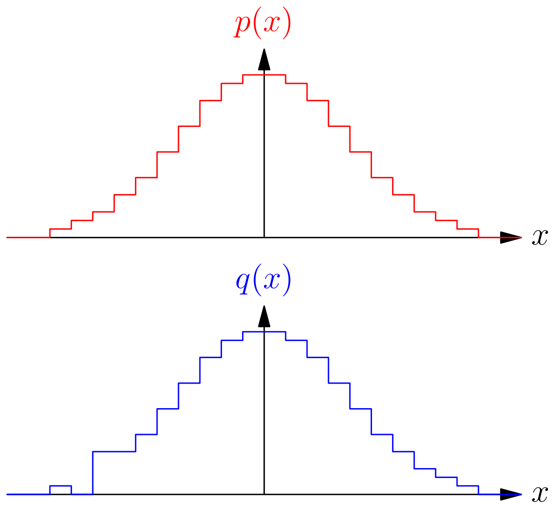


- We can therefore have distributes that seem very similar but their KL-divergence is huge (or infinite)

Wasserstein Distance

- A more benign measure of the differences between two probability functions is the **Wasserstein** or **Earth Moving** distance

- This is a true distance, but more importantly for us it measure distance in a very natural way so that distributions that are close has a small Wasserstein distance

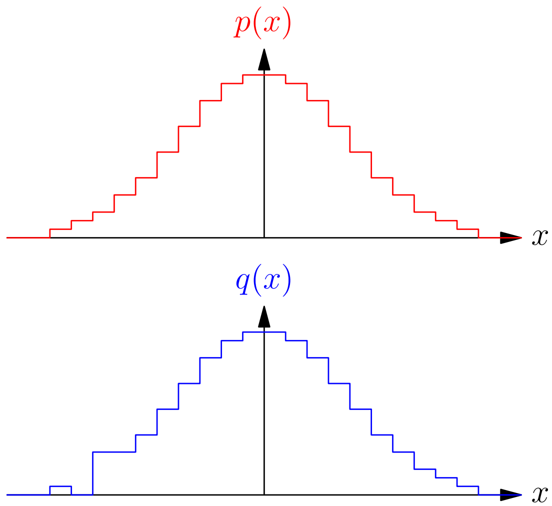


- Although this seems contrived if our probability distribution represents the probability of a 128×128 matrix of real valued triples represents an image of dog, then it is easy to imagine that the Wasserstein distance may be more benign than the KL-divergence

Wasserstein Distance

- A more benign measure of the differences between two probability functions is the **Wasserstein** or **Earth Moving** distance

- This is a true distance, but more importantly for us it measure distance in a very natural way so that distributions that are close has a small Wasserstein distance

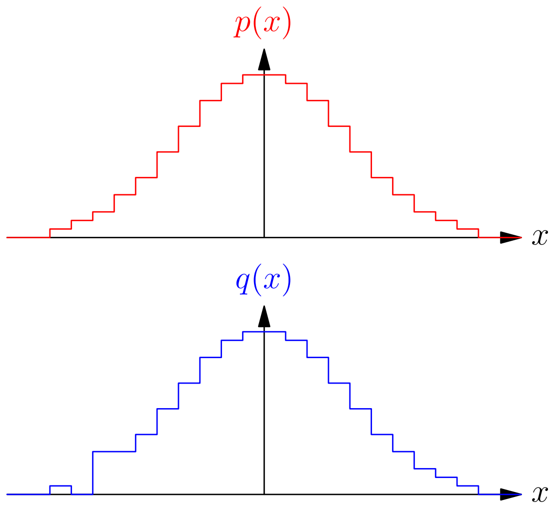


- Although this seems contrived if our probability distribution represents the probability of a 128×128 matrix of real valued triples represents an image of dog, then it is easy to imagine that the Wasserstein distance may be more benign than the KL-divergence

Wasserstein Distance

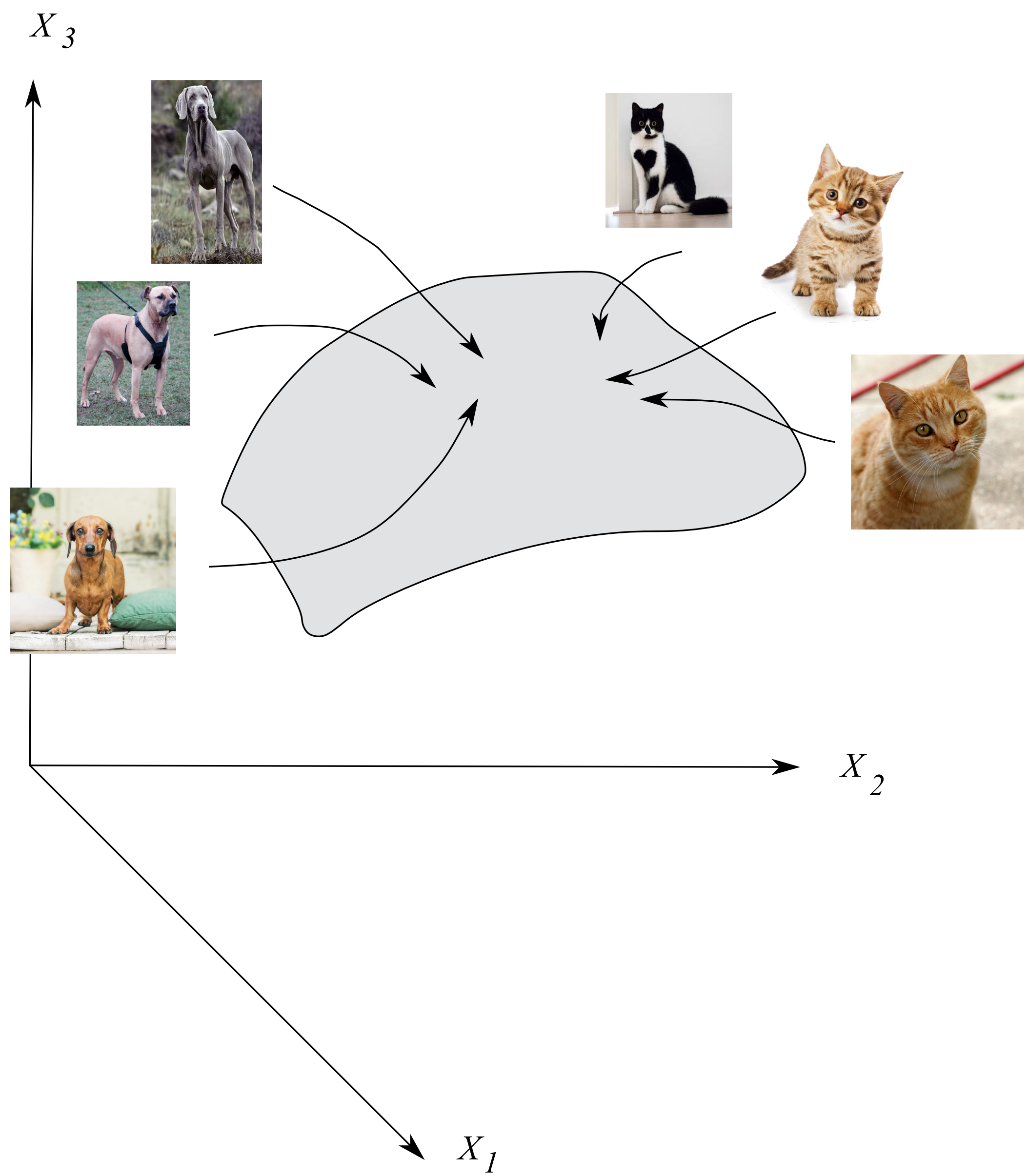
- A more benign measure of the differences between two probability functions is the **Wasserstein** or **Earth Moving** distance

- This is a true distance, but more importantly for us it measure distance in a very natural way so that distributions that are close has a small Wasserstein distance



- Although this seems contrived if our probability distribution represents the probability of a 128×128 matrix of real valued triples represents an image of dog, then it is easy to imagine that the Wasserstein distance may be more benign than the KL-divergence

High Probability Manifold



Transportion Policy

- But how do we formalise the Wasserstein distance?
- A good place to start is to define a transportation policy $\gamma(\boldsymbol{x},\boldsymbol{y})$ with

$$\int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} = p(\boldsymbol{x}) \qquad \int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x} = q(\boldsymbol{y})$$

- This looks like a joint probability distribution, but we interpret $\gamma(\boldsymbol{x},\boldsymbol{y})$ as the amount of probability mass/density that we transfer from $p(\boldsymbol{x})$ to $q(\boldsymbol{y})$

Transportion Policy

- But how do we formalise the Wasserstein distance?
- A good place to start is to define a transportation policy $\gamma(\boldsymbol{x},\boldsymbol{y})$ with

$$\int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} = p(\boldsymbol{x}) \qquad \int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x} = q(\boldsymbol{y})$$

- This looks like a joint probability distribution, but we interpret $\gamma(\boldsymbol{x},\boldsymbol{y})$ as the amount of probability mass/density that we transfer from $p(\boldsymbol{x})$ to $q(\boldsymbol{y})$

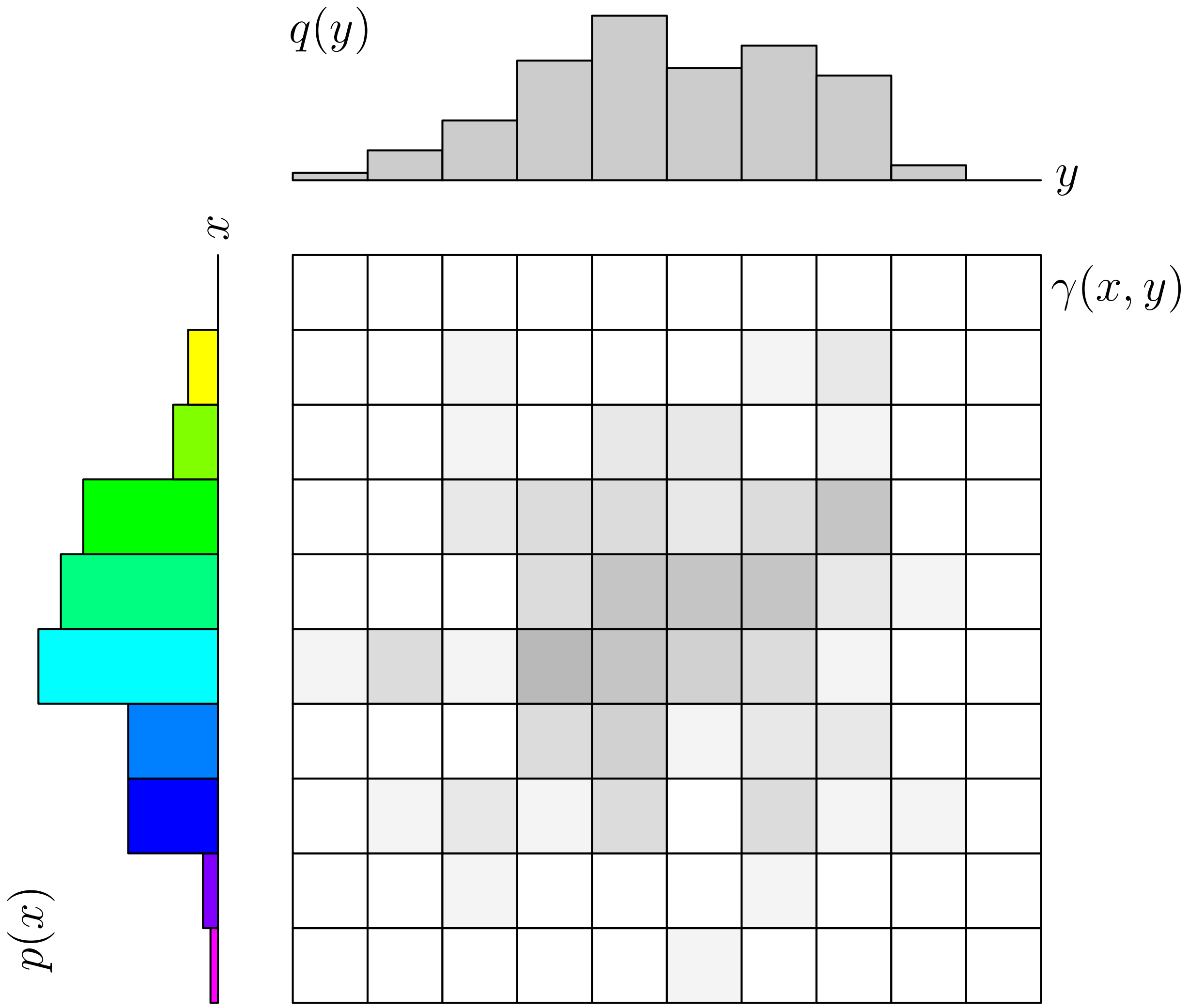
Transportion Policy

- But how do we formalise the Wasserstein distance?
- A good place to start is to define a transportation policy $\gamma(\boldsymbol{x}, \boldsymbol{y})$ with

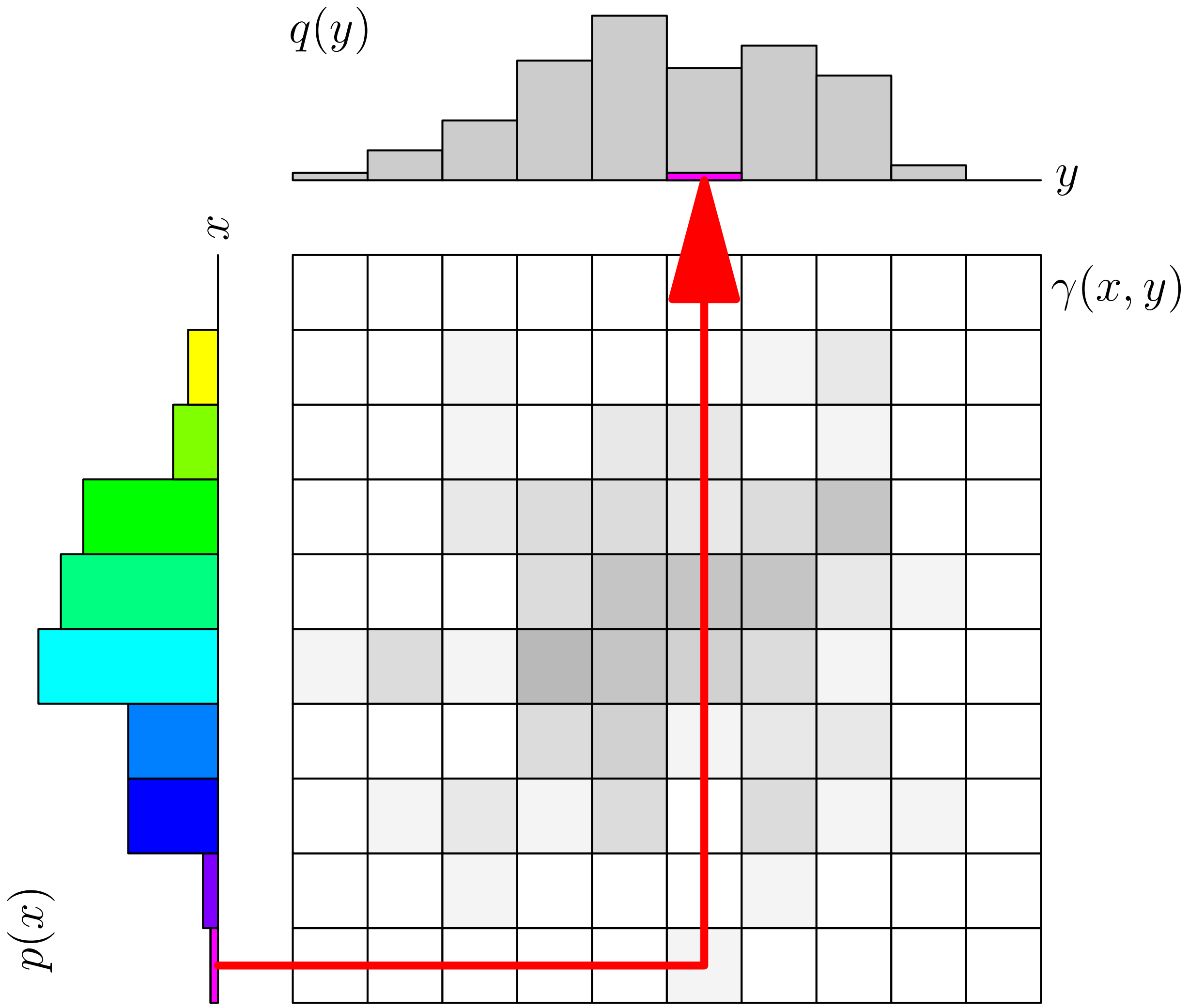
$$\int \gamma(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{y} = p(\boldsymbol{x}) \qquad \int \gamma(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x} = q(\boldsymbol{y})$$

- This looks like a joint probability distribution, but we interpret $\gamma(\boldsymbol{x}, \boldsymbol{y})$ as the amount of probability mass/density that we transfer from $p(\boldsymbol{x})$ to $q(\boldsymbol{y})$

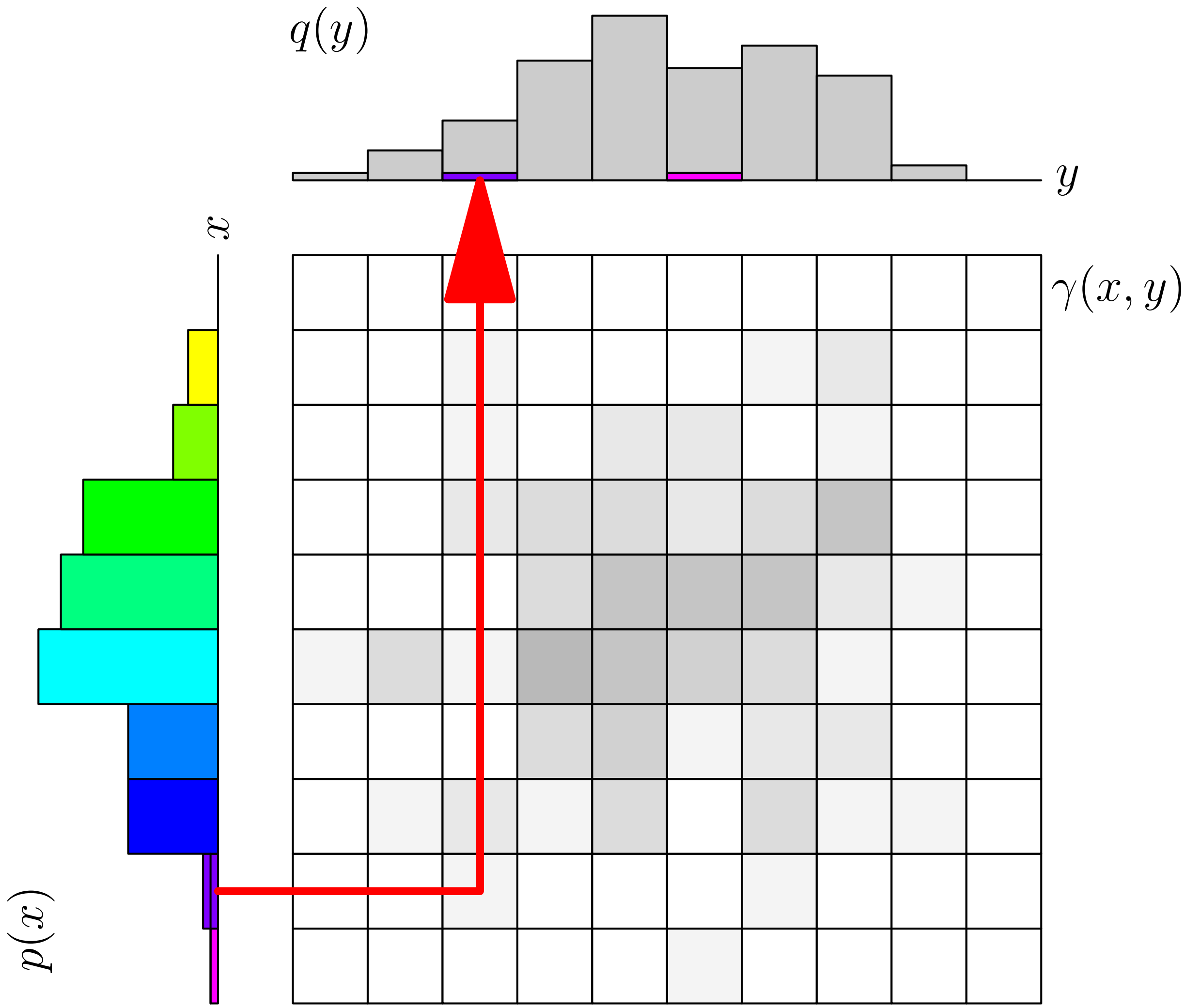
Transportation Policy



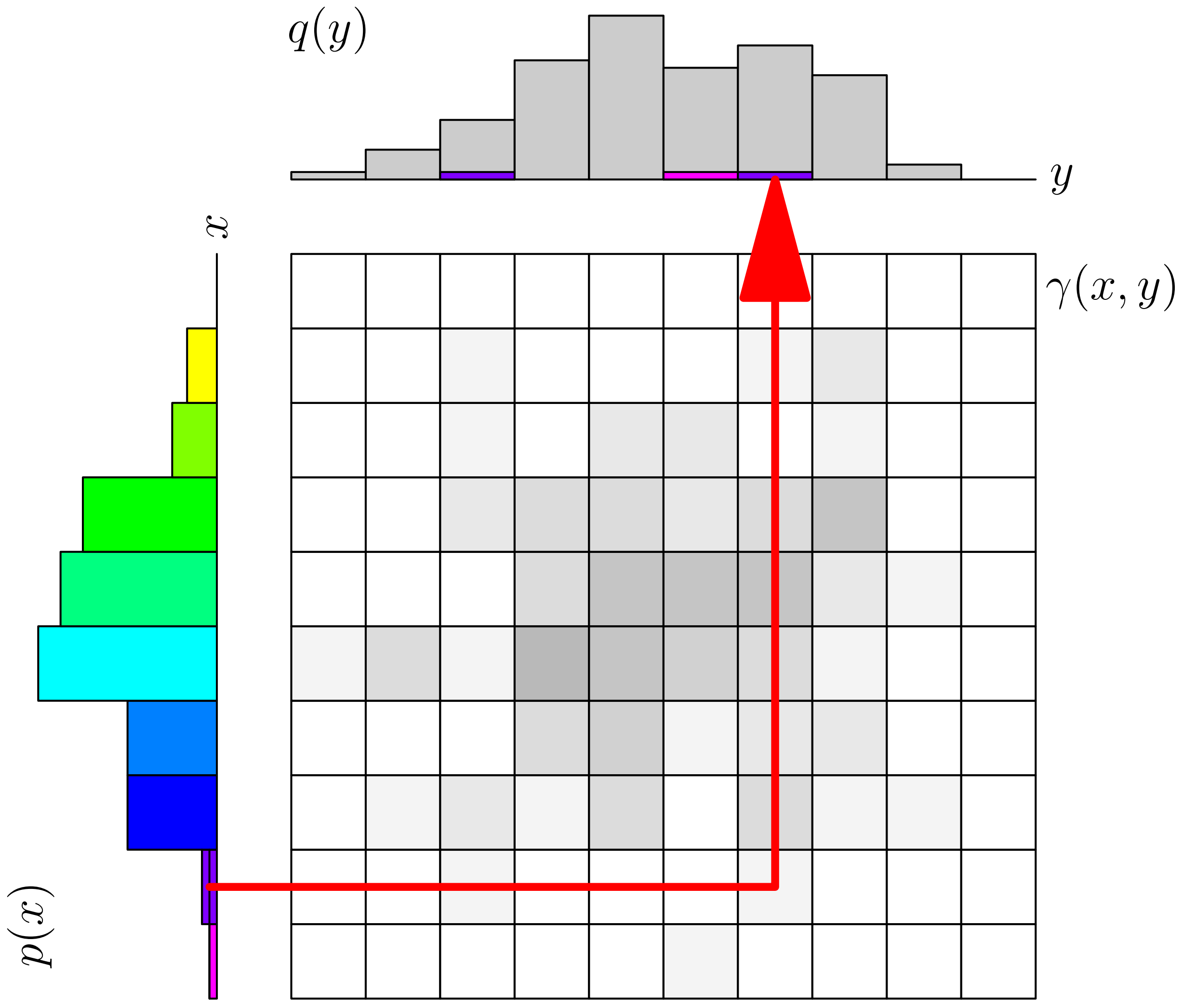
Transportation Policy



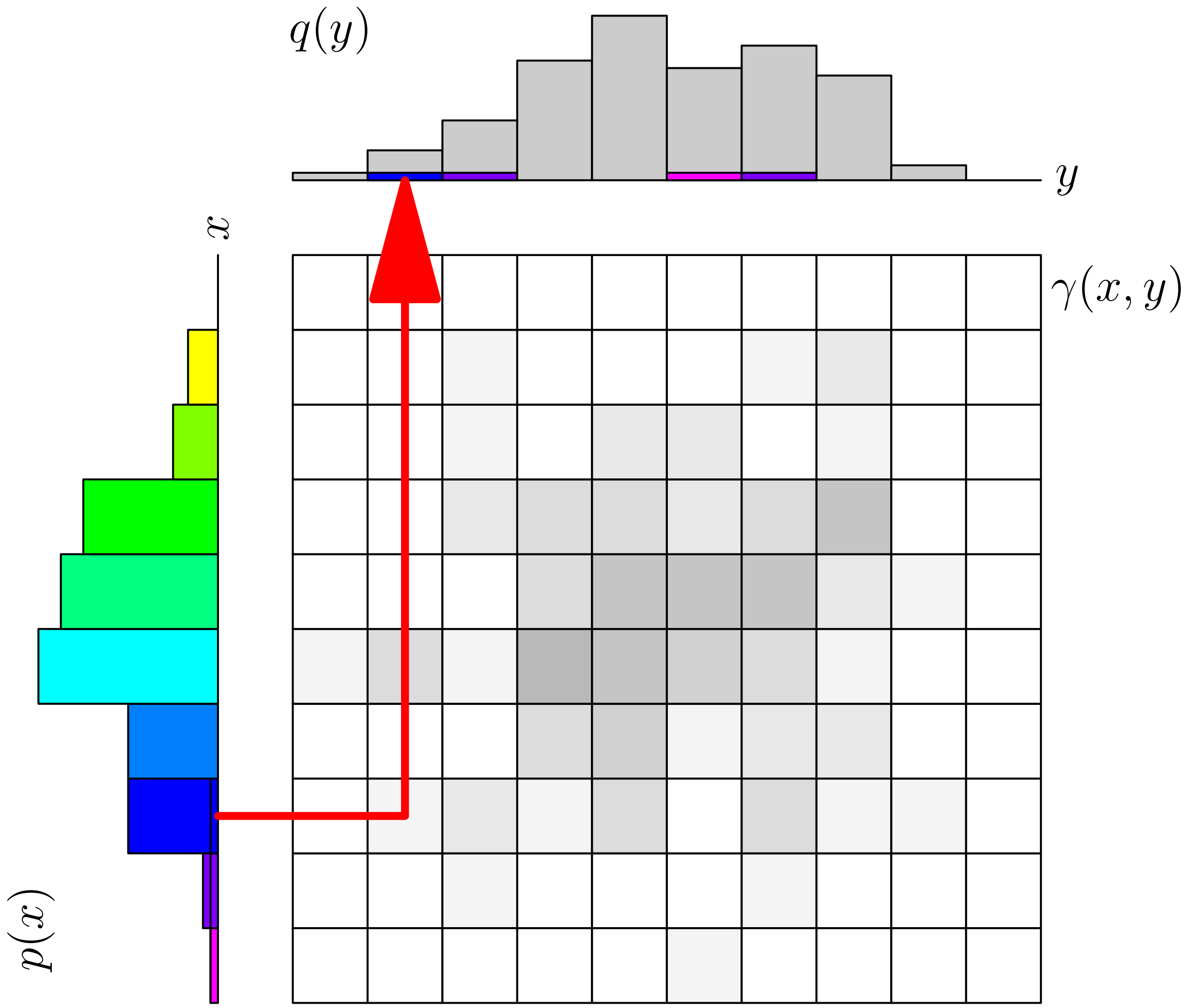
Transportation Policy



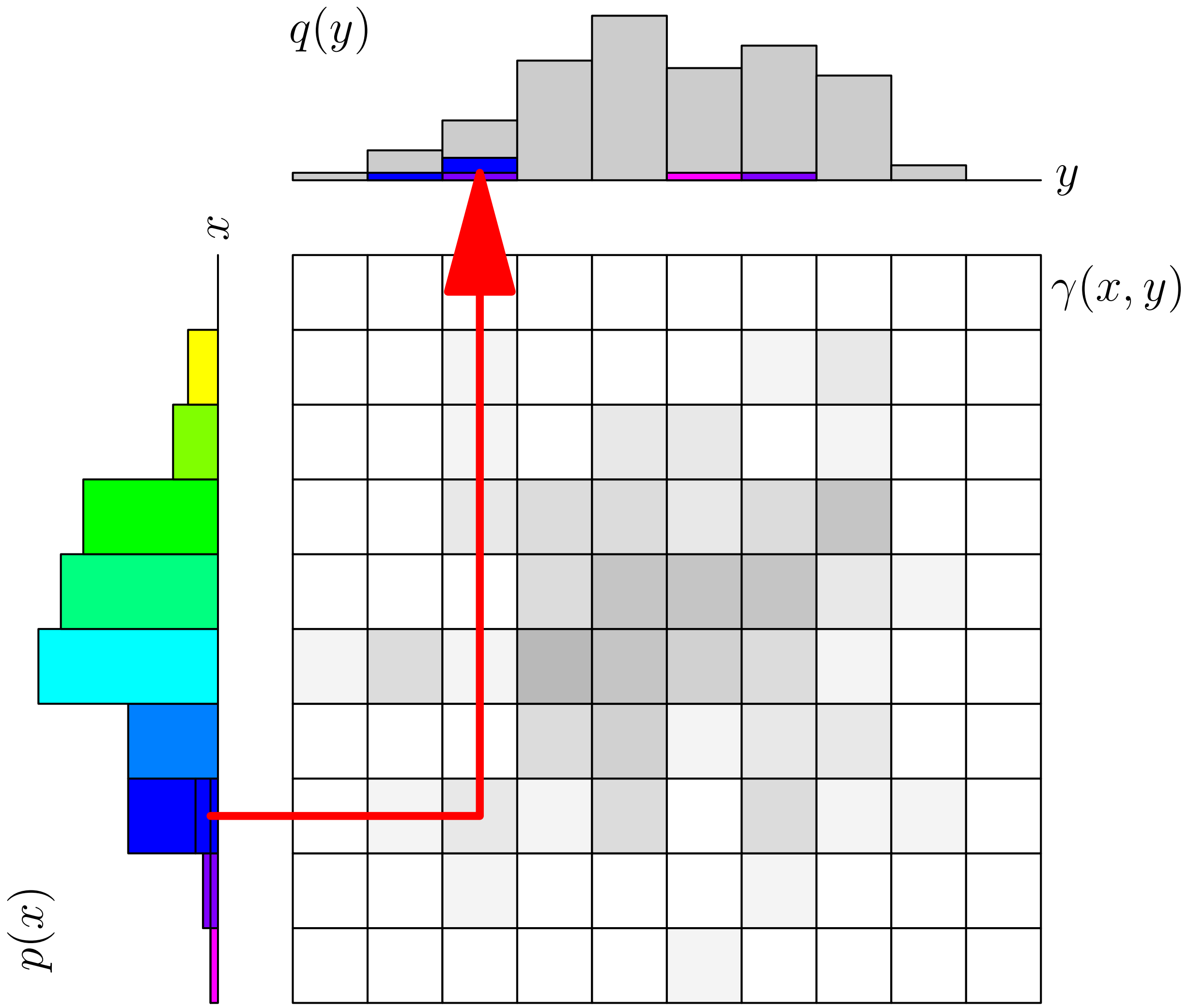
Transportation Policy



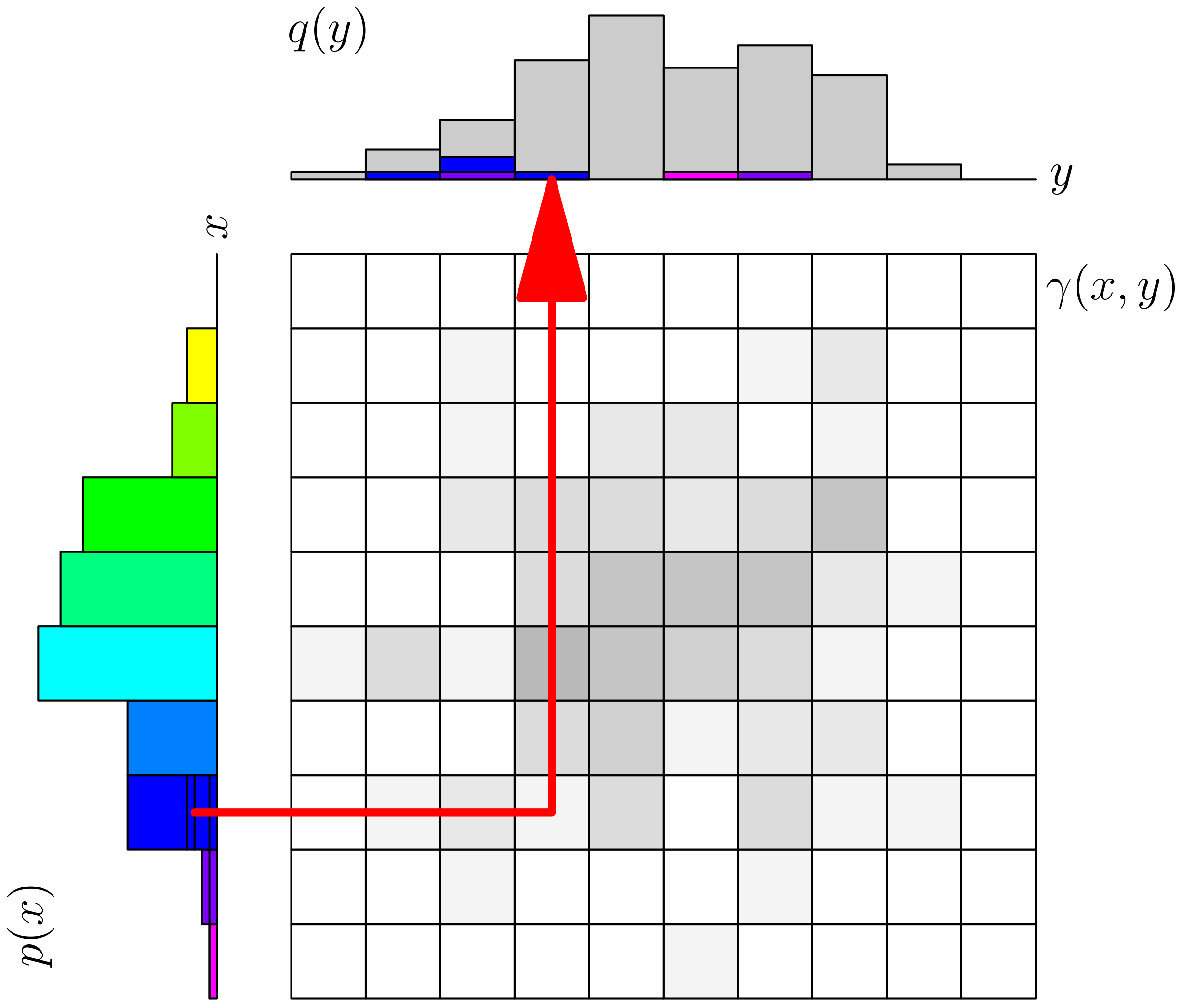
Transportation Policy



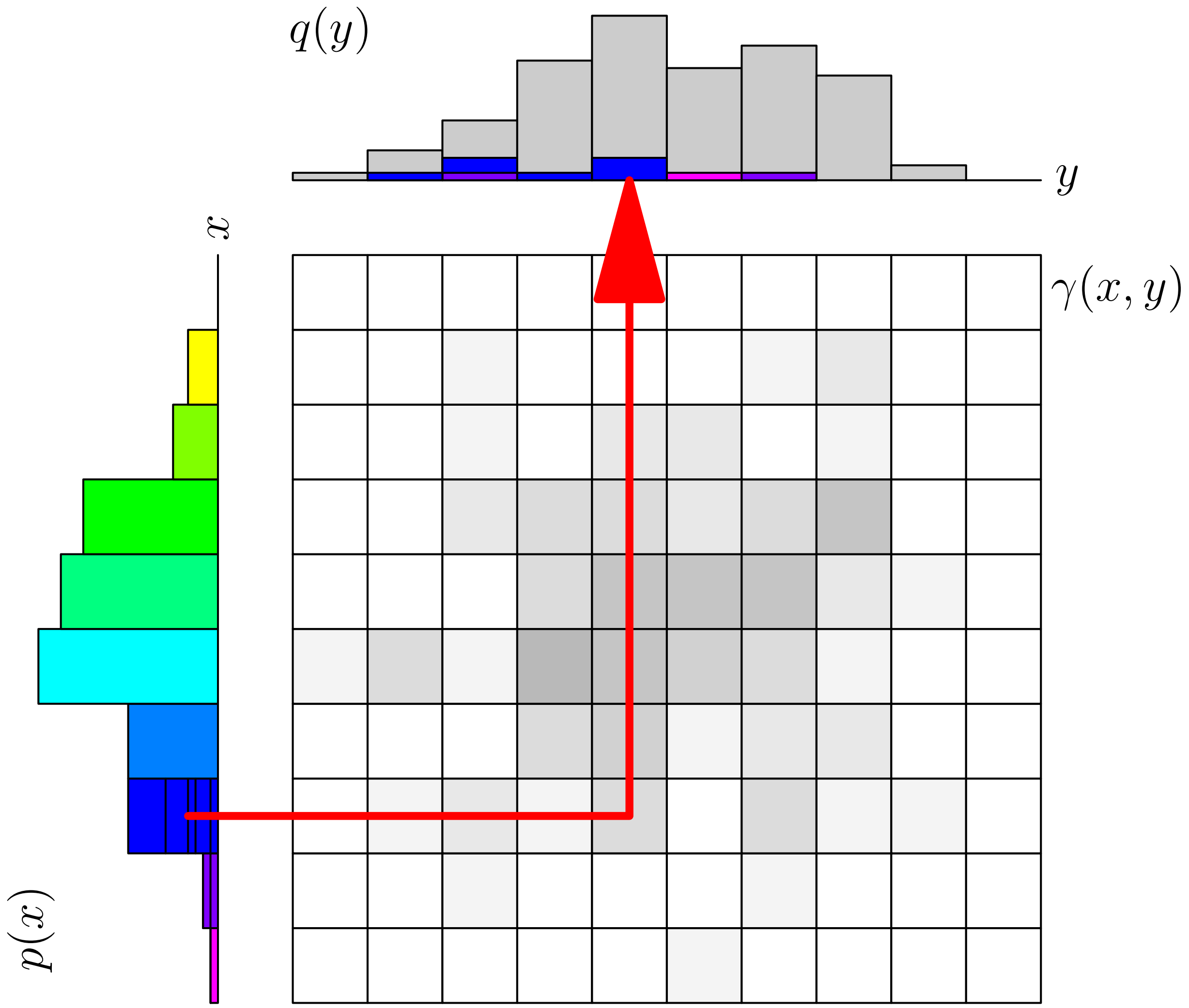
Transportation Policy



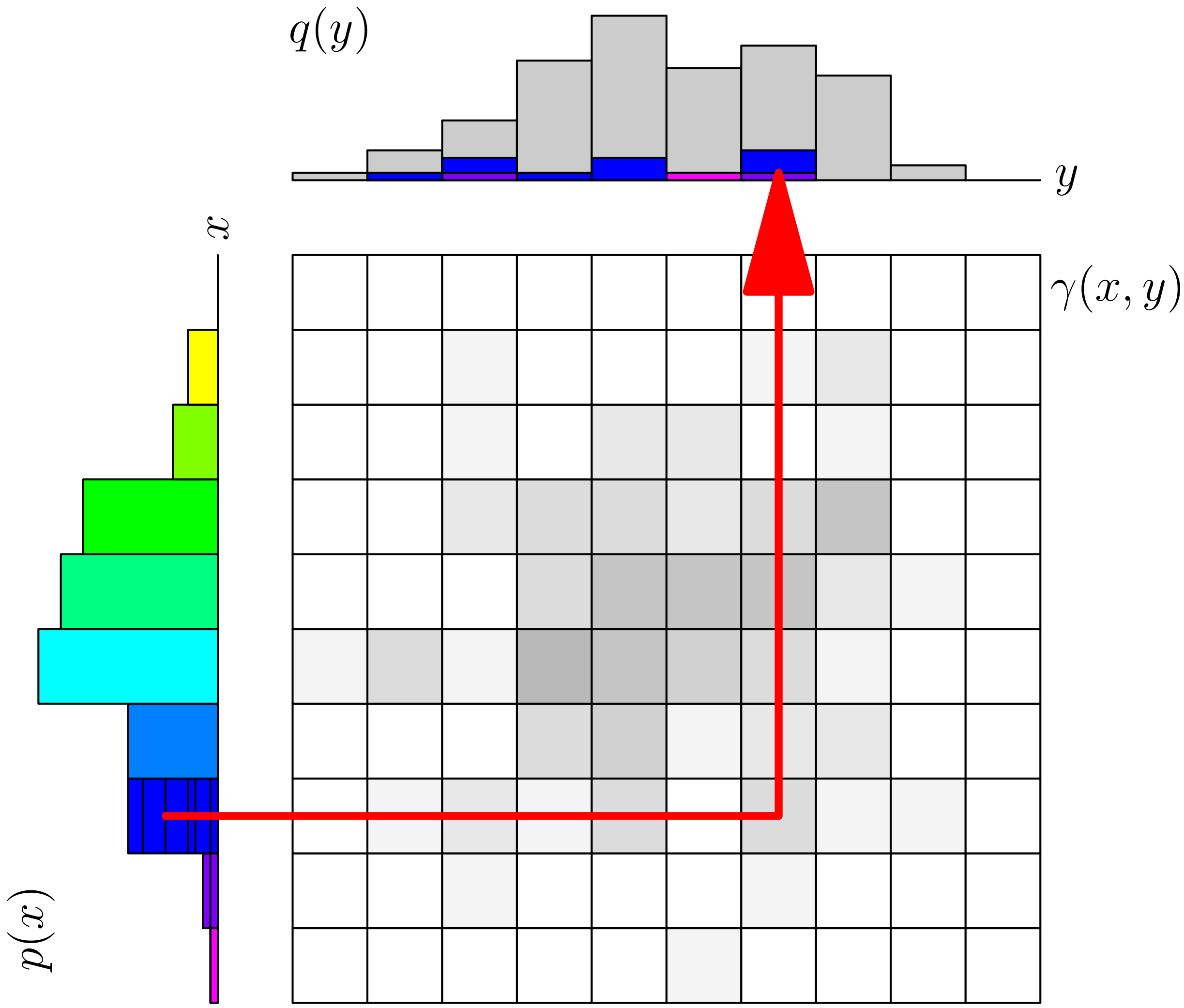
Transportation Policy



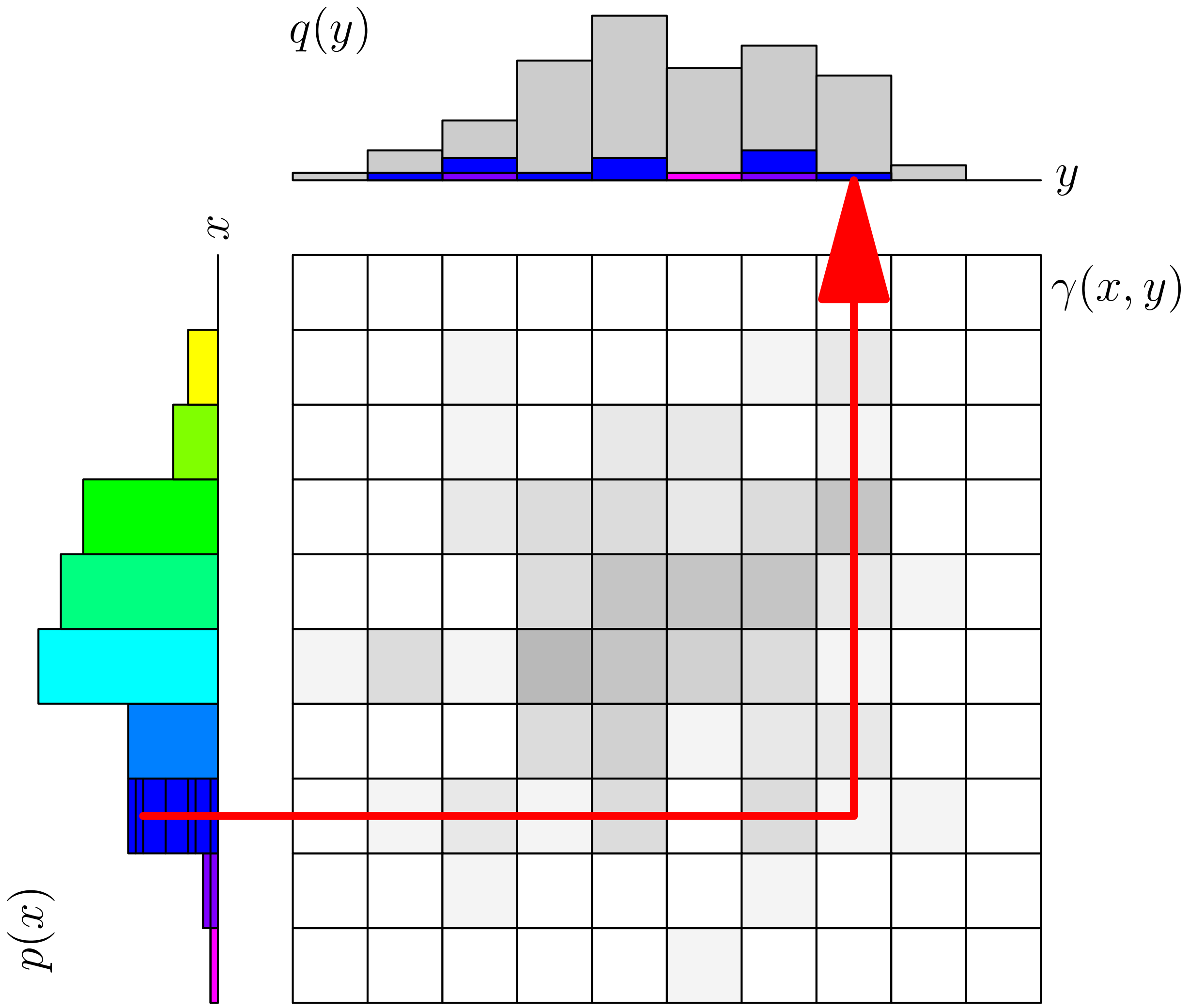
Transportation Policy



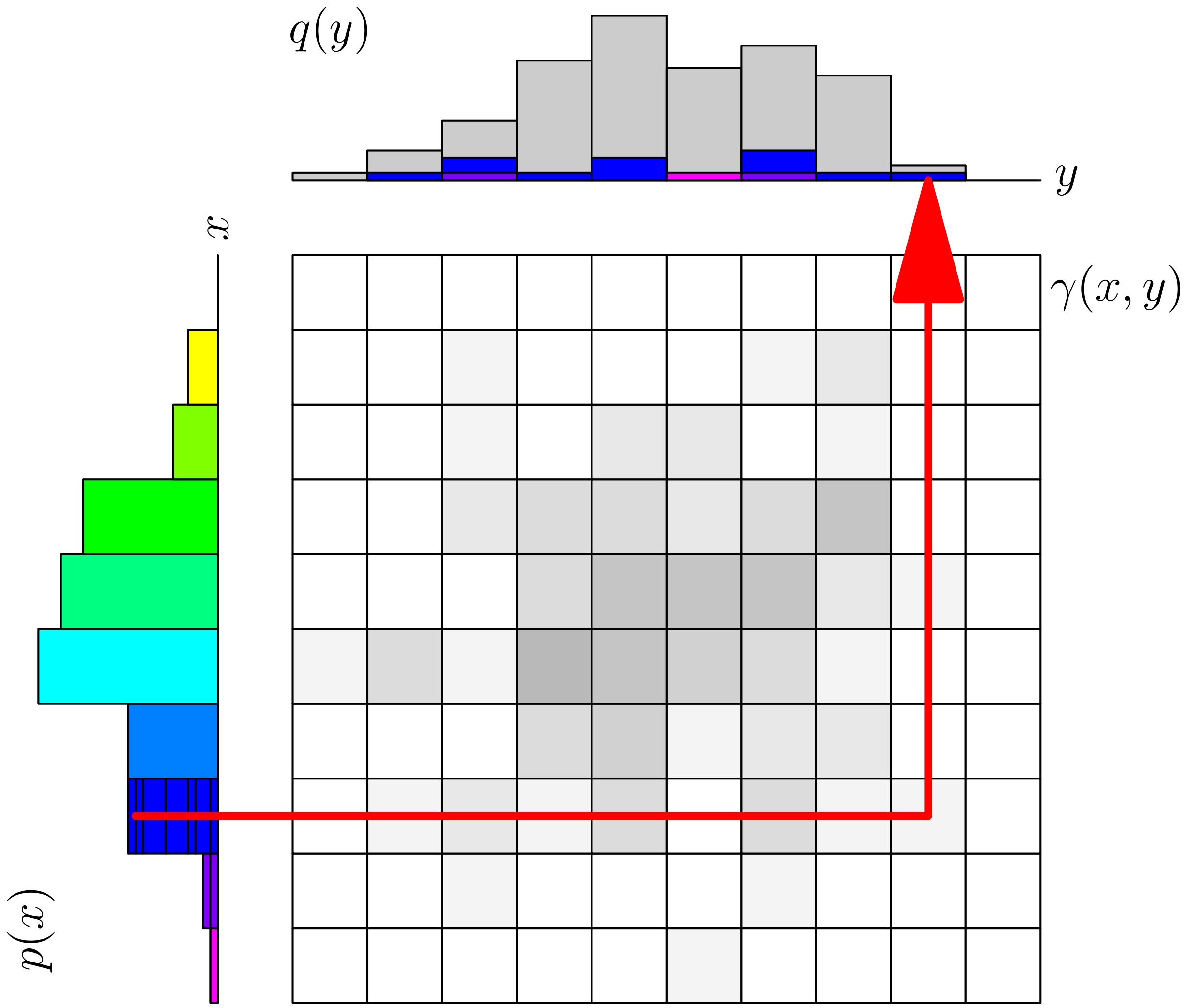
Transportation Policy



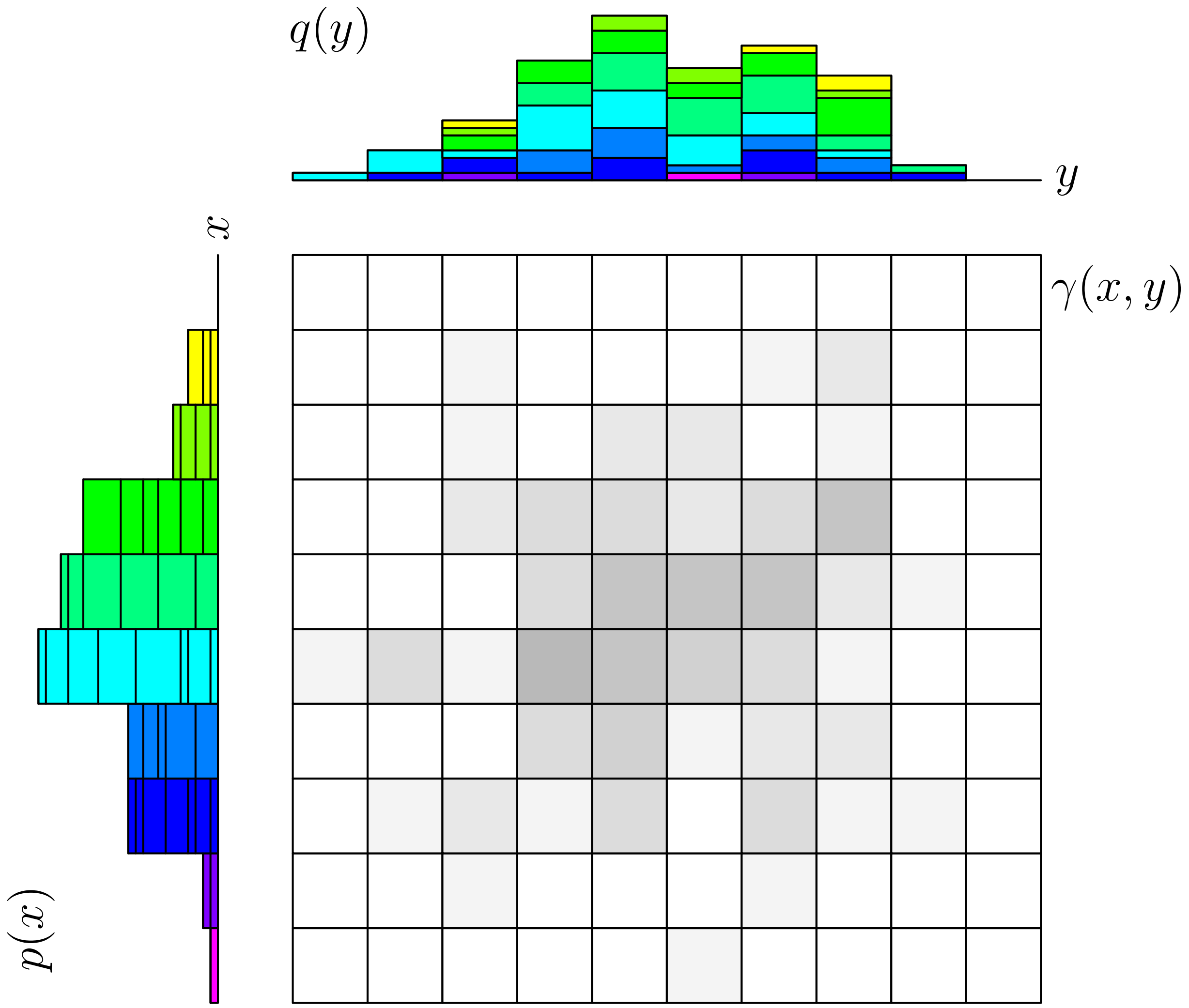
Transportation Policy



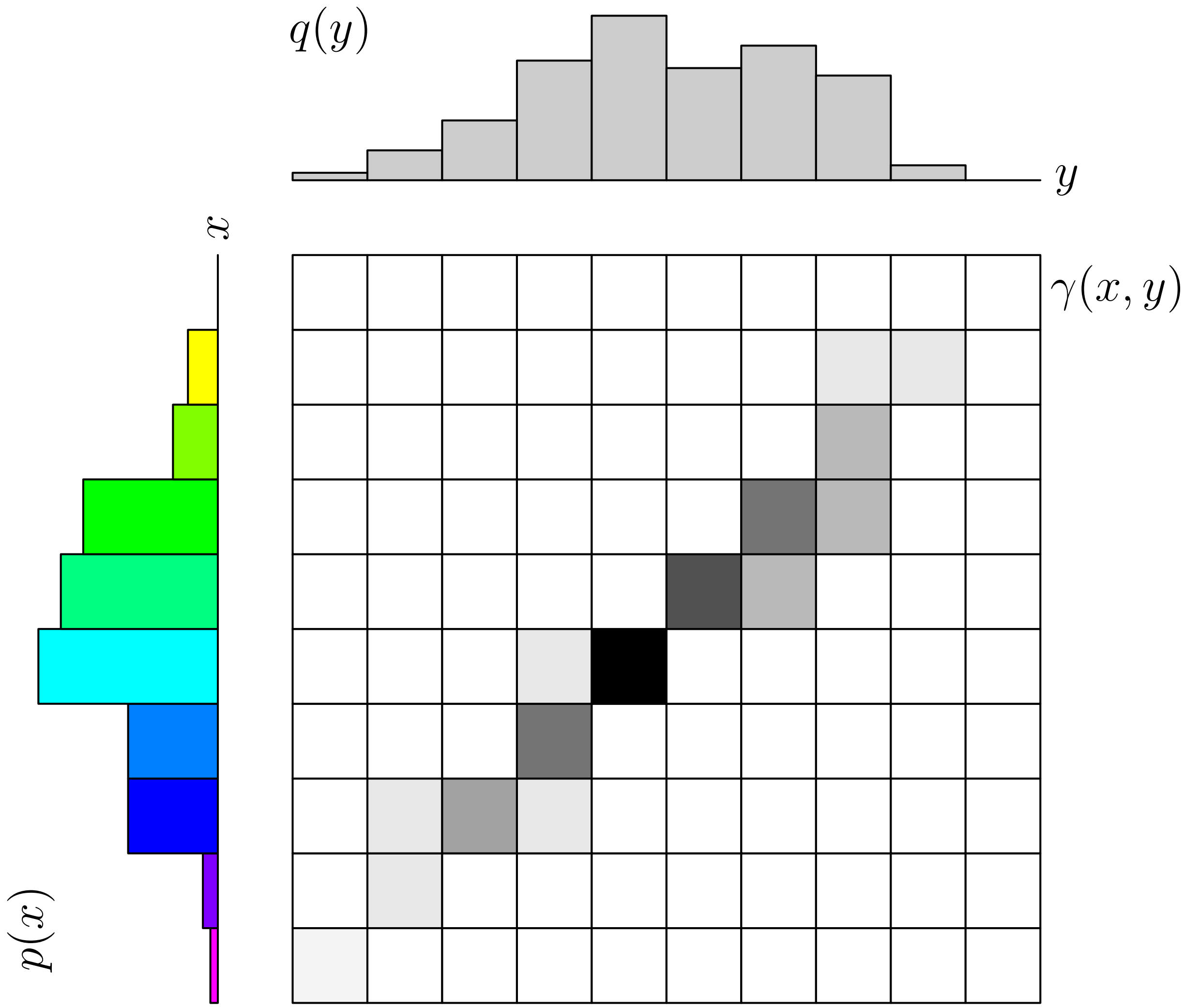
Transportation Policy



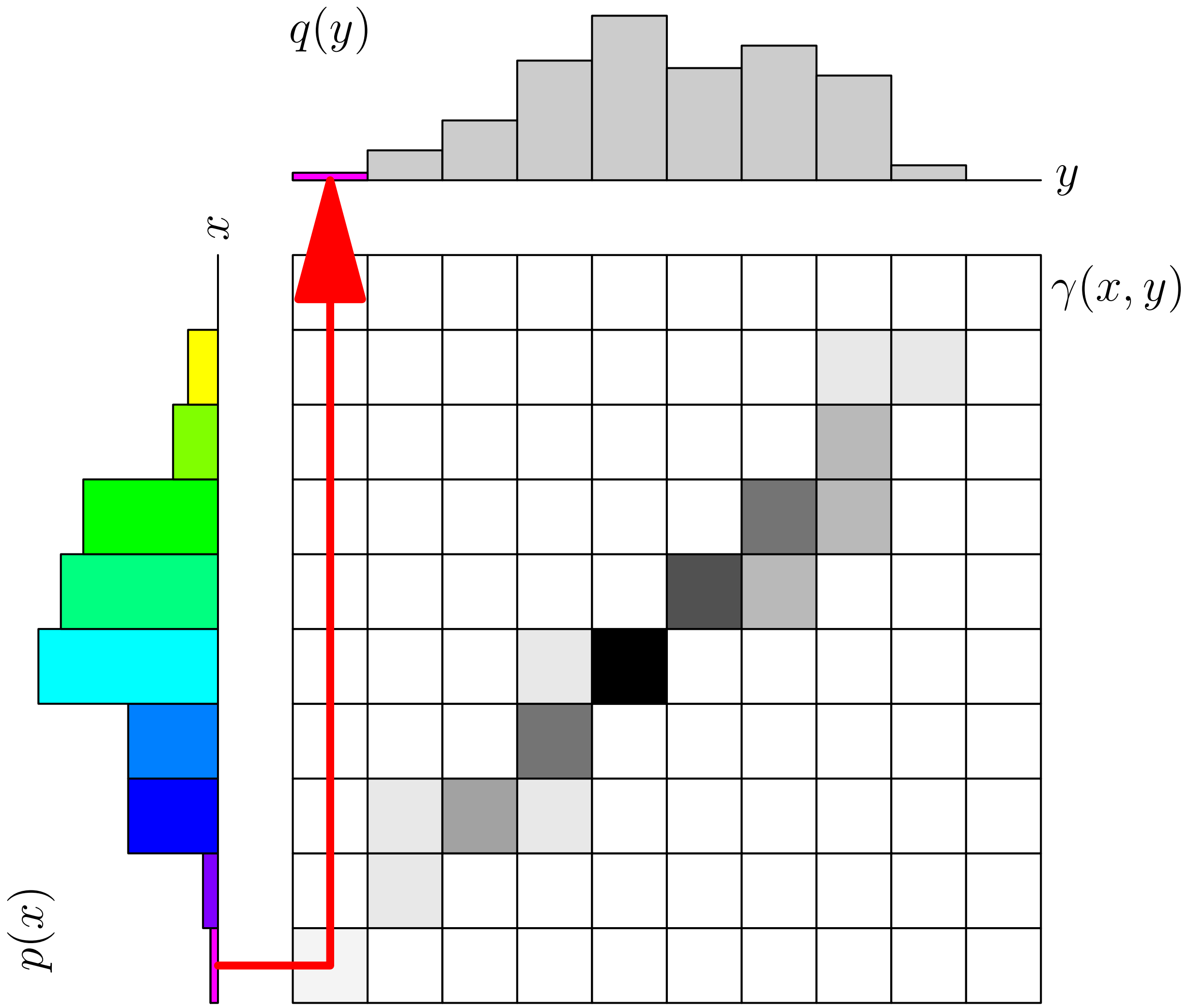
Transportation Policy



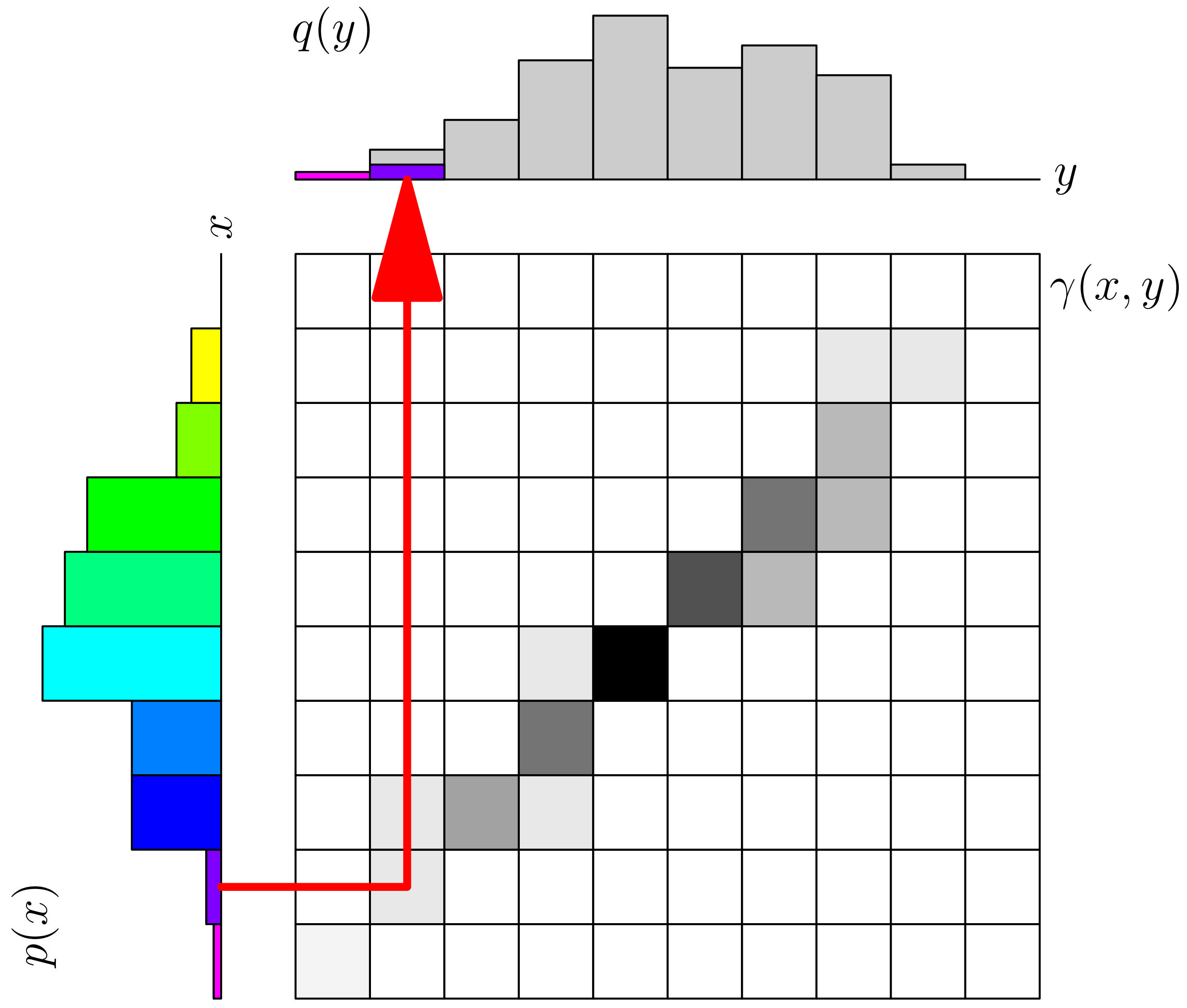
Transportation Policy



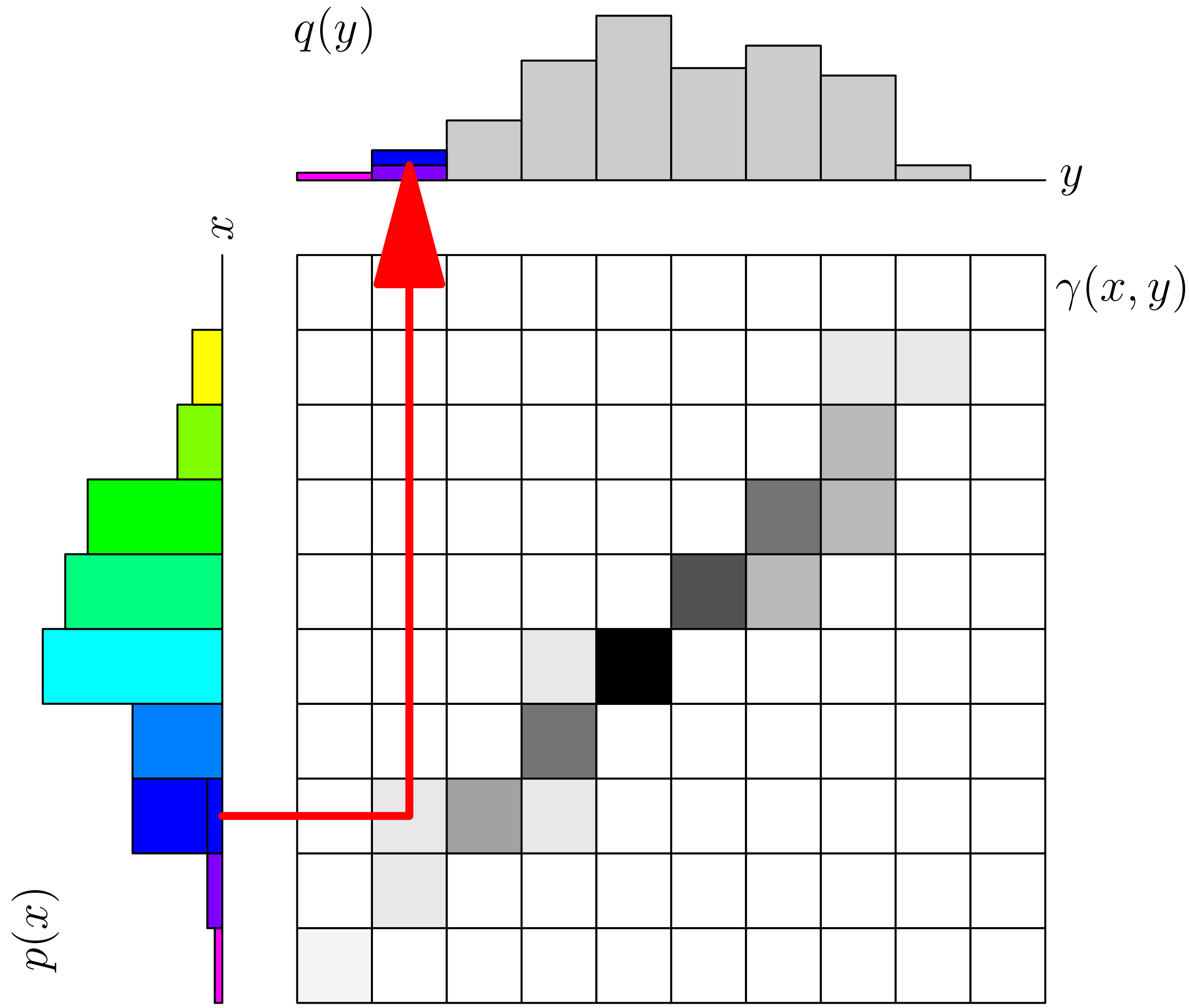
Transportation Policy



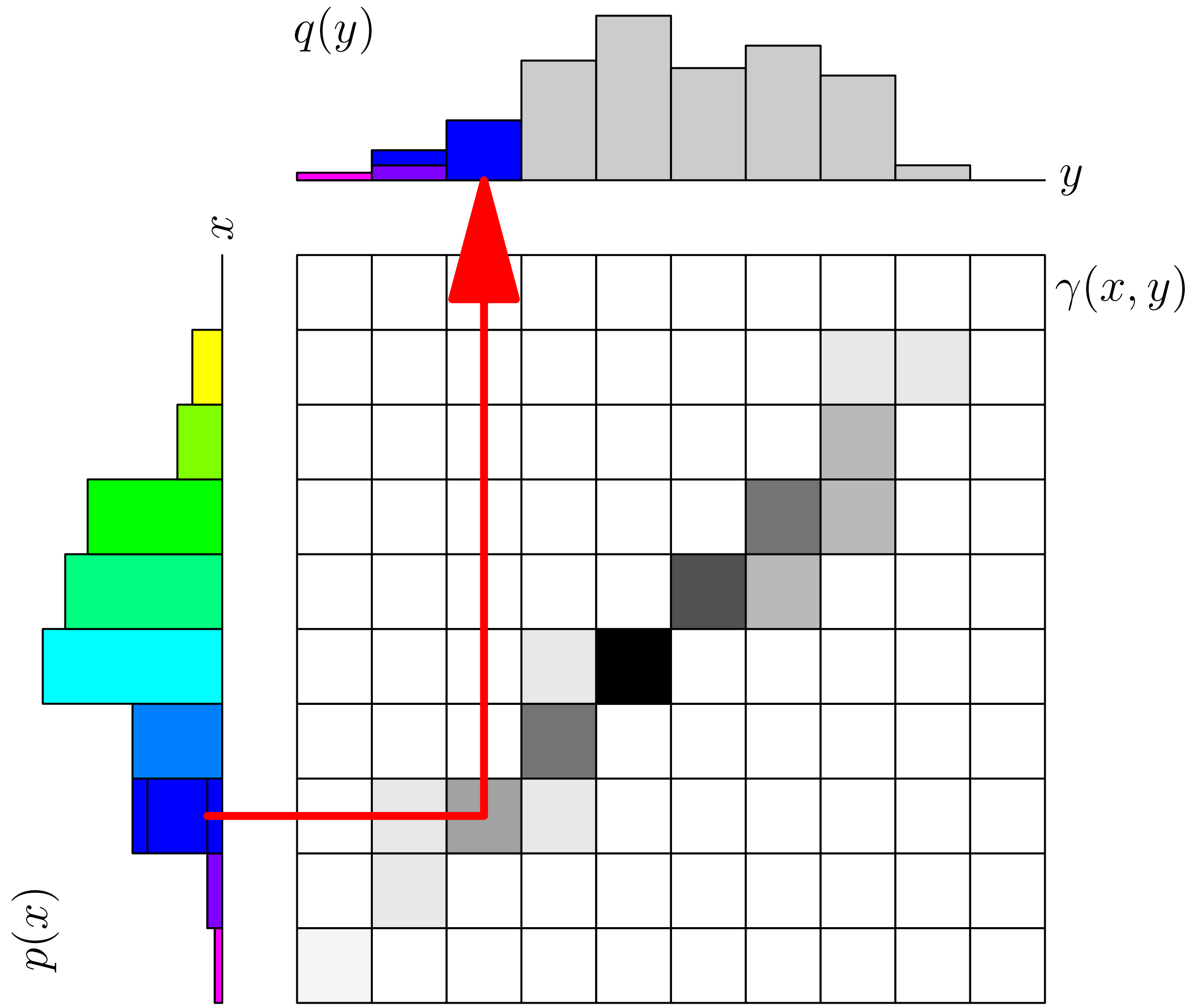
Transportation Policy



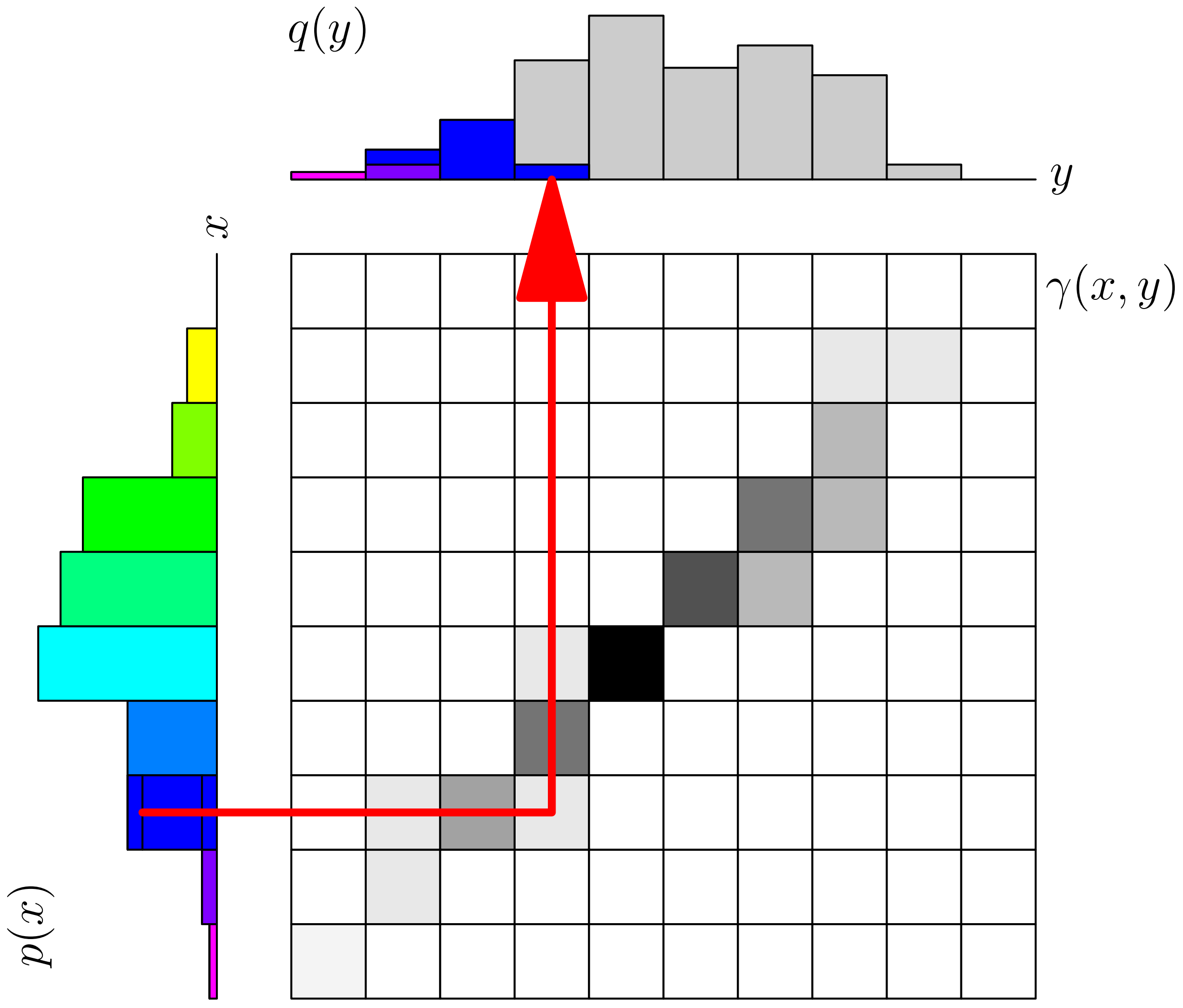
Transportation Policy



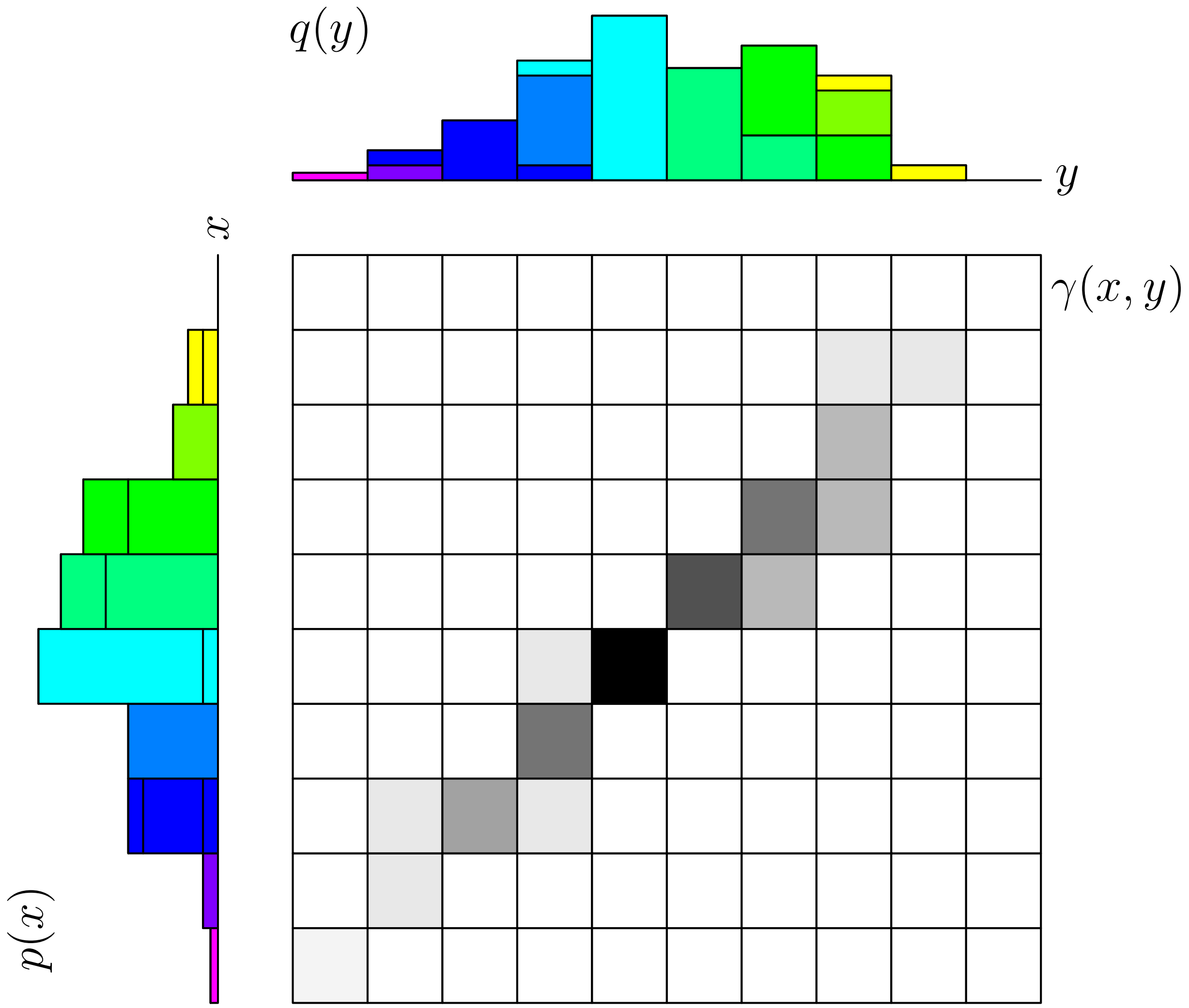
Transportation Policy



Transportation Policy



Transportation Policy



The Cost of Transport

- We want to choose the transportation policy that minimises the amount of probability mass we need to move
- Let $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$ be a distance measure then the cost of a transportation policy is

$$C(\gamma) = \int\int d(\boldsymbol{x}, \boldsymbol{y}) \gamma(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y}$$

- Usually we take $d(\boldsymbol{x}, \boldsymbol{y})$ to be the Euclidean distance, but we can choose any distance

The Cost of Transport

- We want to choose the transportation policy that minimises the amount of probability mass we need to move
- Let $d(\boldsymbol{x},\boldsymbol{y}) = \|\boldsymbol{x}-\boldsymbol{y}\|$ be a distance measure then the cost of a transportation policy is

$$C(\gamma) = \int\int d(\boldsymbol{x},\boldsymbol{y})\gamma(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y}$$

- Usually we take $d(\boldsymbol{x},\boldsymbol{y})$ to be the Euclidean distance, but we can choose any distance

The Cost of Transport

- We want to choose the transportation policy that minimises the amount of probability mass we need to move
- Let $d(\boldsymbol{x},\boldsymbol{y}) = \|\boldsymbol{x}-\boldsymbol{y}\|$ be a distance measure then the cost of a transportation policy is

$$C(\gamma) = \int\int d(\boldsymbol{x},\boldsymbol{y})\gamma(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y} = \mathbb{E}_\gamma[d(\boldsymbol{x},\boldsymbol{y})]$$

  where we interpret $\gamma(\boldsymbol{x},\boldsymbol{y})$ as a probability distribution

- Usually we take $d(\boldsymbol{x},\boldsymbol{y})$ to be the Euclidean distance, but we can choose any distance

The Cost of Transport

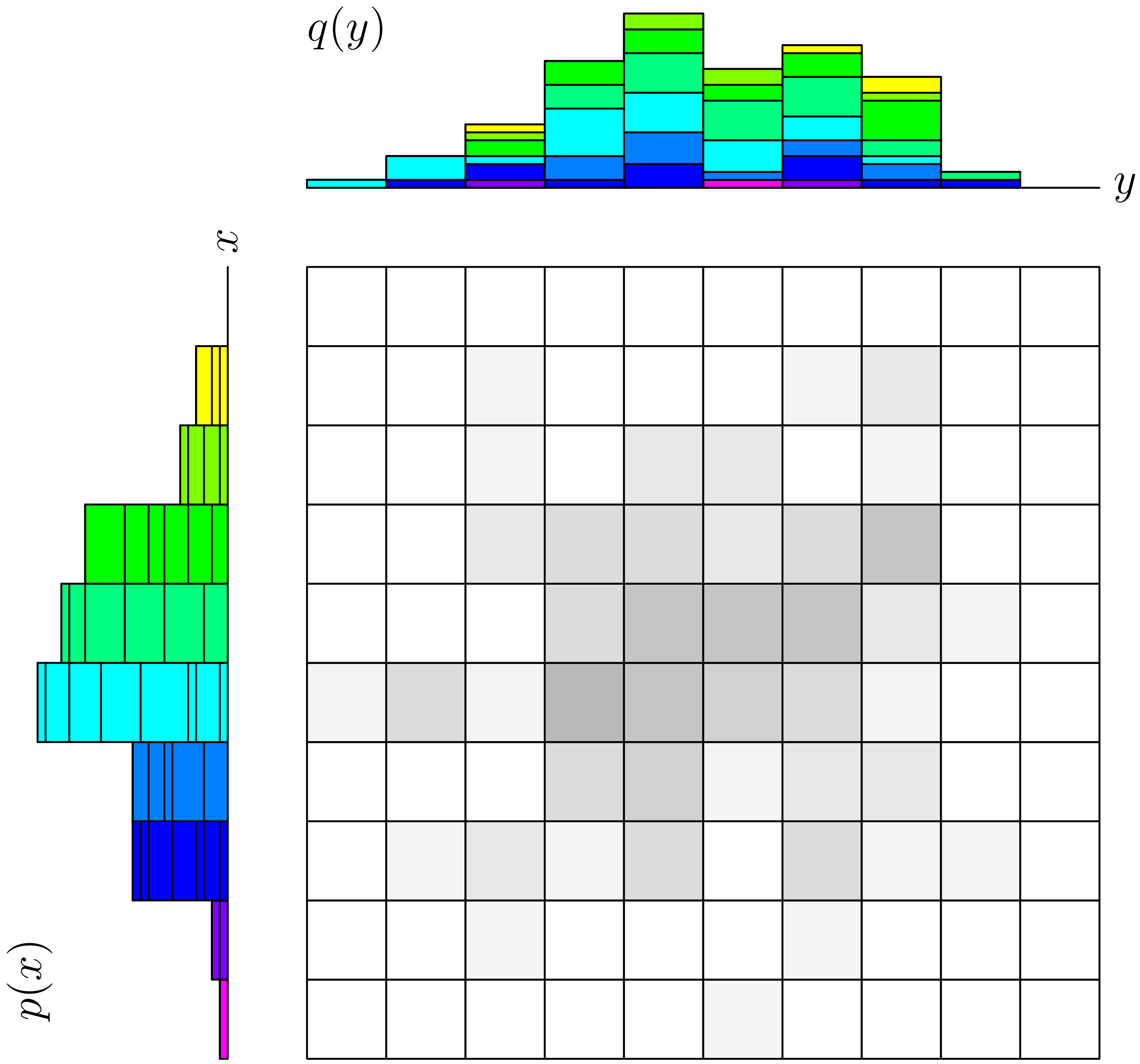
- We want to choose the transportation policy that minimises the amount of probability mass we need to move
- Let $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$ be a distance measure then the cost of a transportation policy is

$$C(\gamma) = \int \int d(\boldsymbol{x}, \boldsymbol{y}) \gamma(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} = \mathbb{E}_{\gamma}[d(\boldsymbol{x}, \boldsymbol{y})]$$

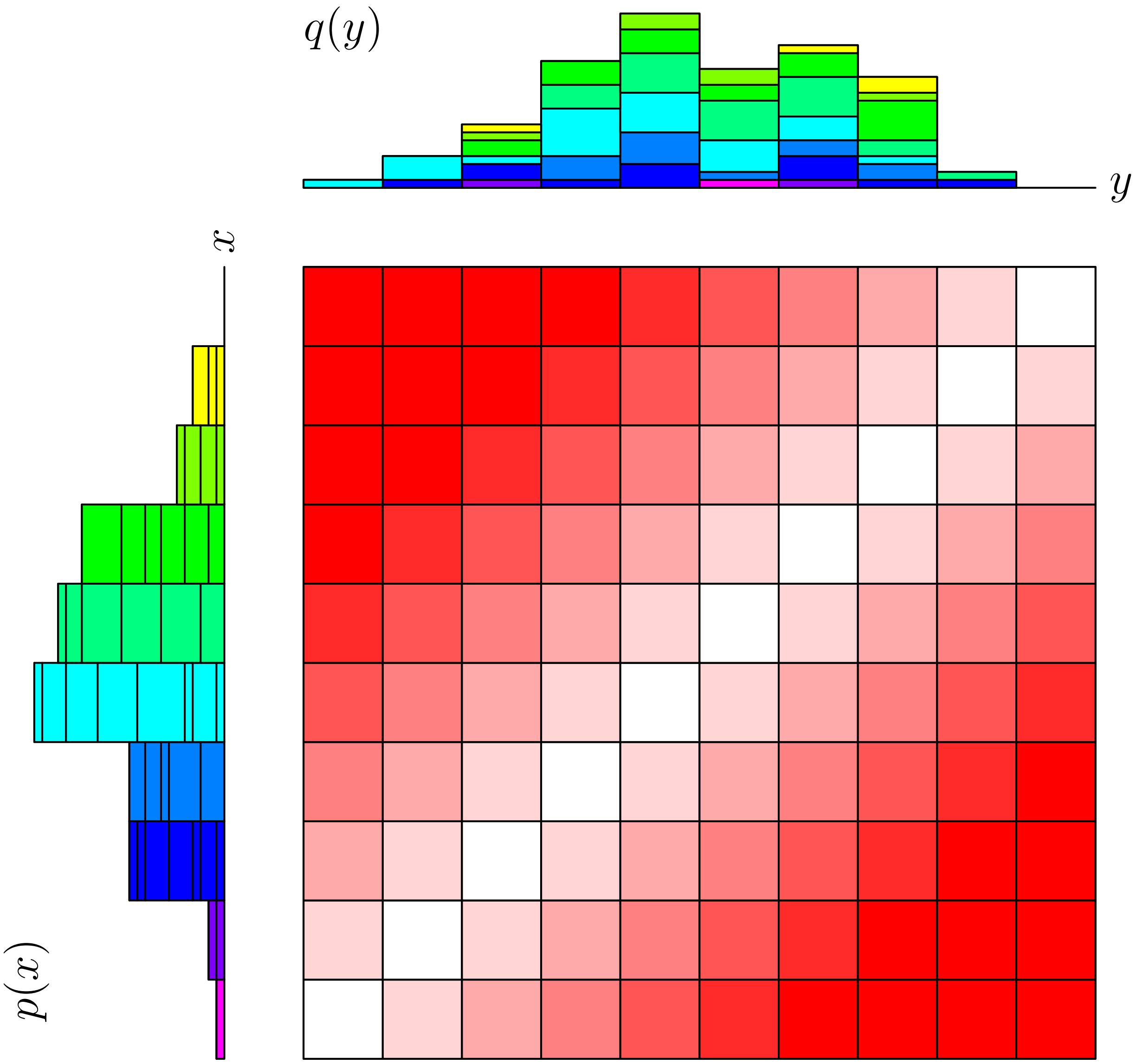
  where we interpret $\gamma(\boldsymbol{x}, \boldsymbol{y})$ as a probability distribution

- Usually we take $d(\boldsymbol{x}, \boldsymbol{y})$ to be the Euclidean distance, but we can choose any distance

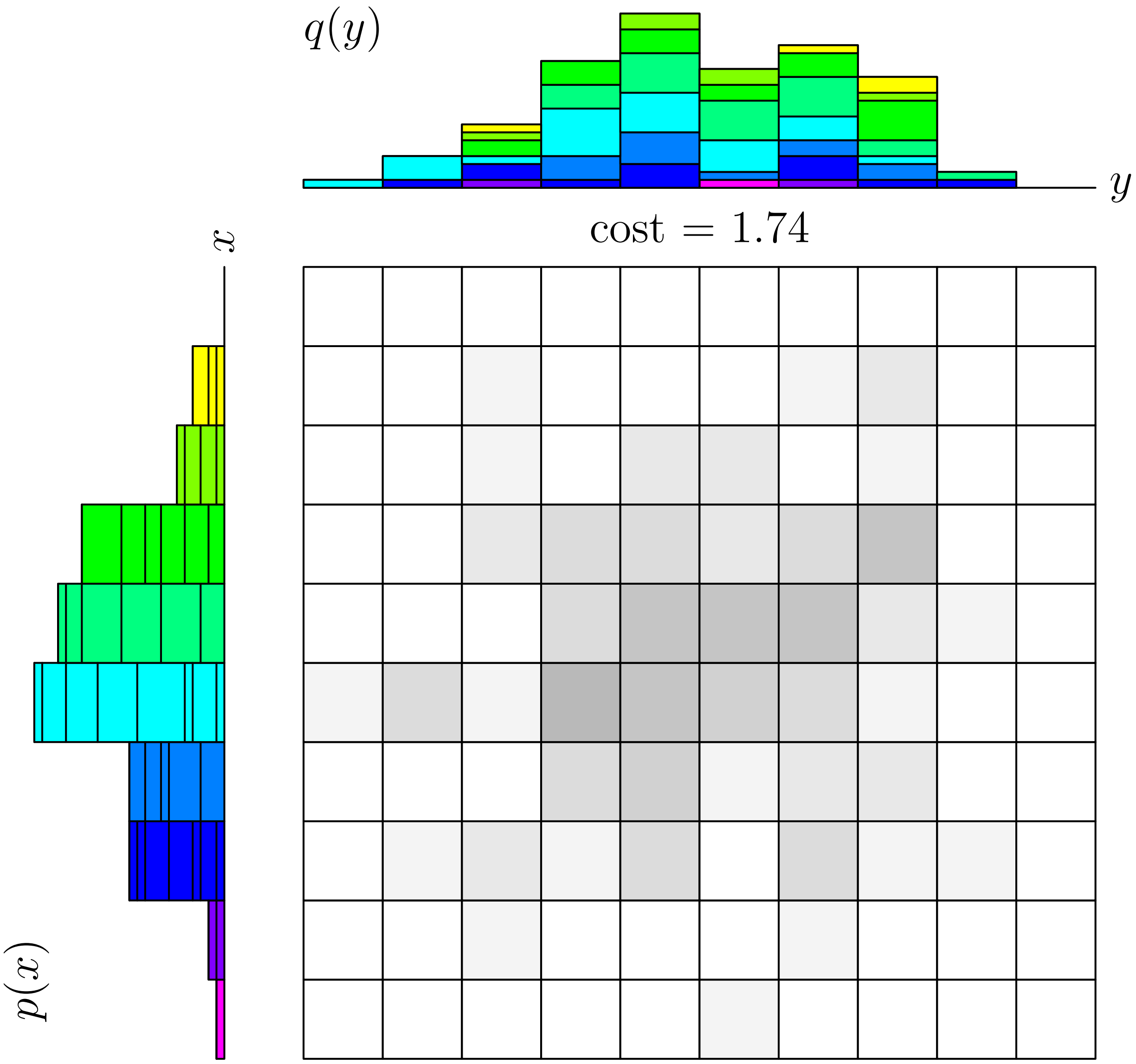
Transportation Cost



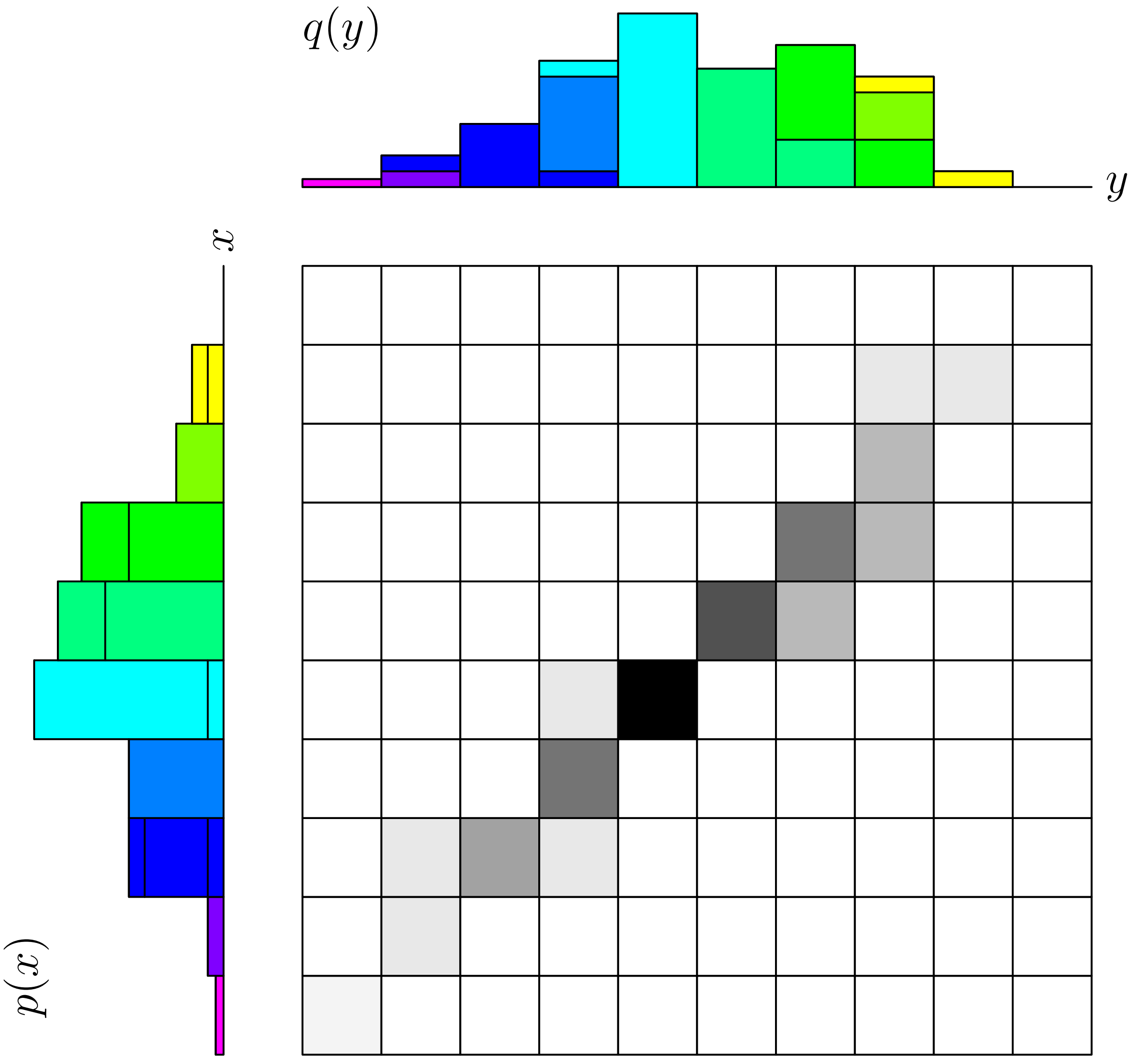
Transportation Cost



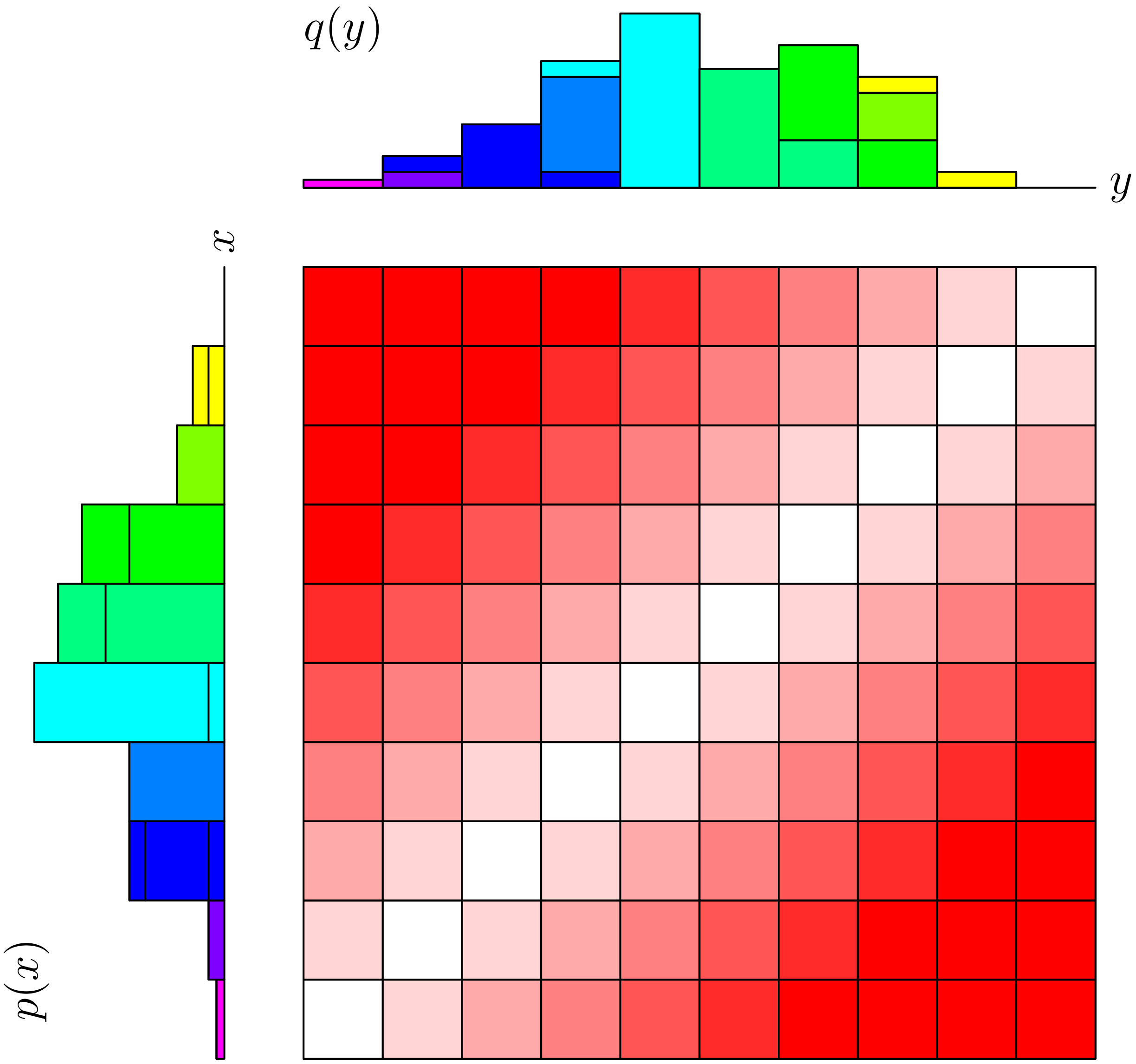
Transportation Cost



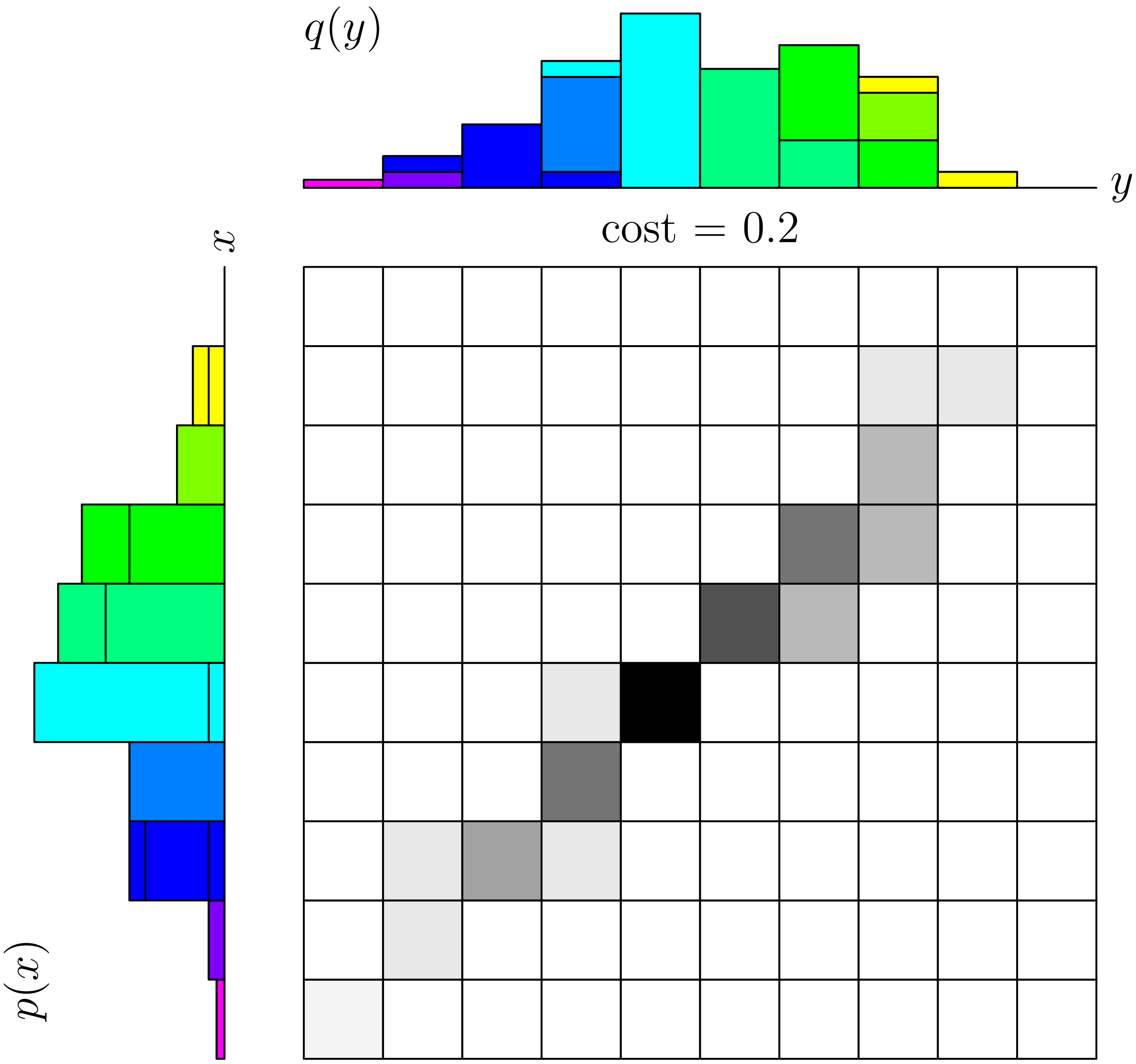
Transportation Cost



Transportation Cost



Transportation Cost



The Wasserstein Distance

- The Wasserstein distance $W(p,q)$ between two probability distributions is defined as

$$W(p,q) = \min_{\gamma \in \Lambda(p,q)} \mathbb{E}_\gamma[d(\boldsymbol{x},\boldsymbol{y})]$$

- Where $\Lambda(p,q)$ is the set of joint distributions $\gamma(\boldsymbol{x},\boldsymbol{y})$ such that

$$\int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} = p(\boldsymbol{x}) \qquad\qquad \int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x} = q(\boldsymbol{y})$$

The Wasserstein Distance

- The Wasserstein distance $W(p,q)$ between two probability distributions is defined as

$$W(p,q) = \min_{\gamma \in \Lambda(p,q)} \mathbb{E}_\gamma\left[d(\boldsymbol{x},\boldsymbol{y})\right]$$

- Where $\Lambda(p,q)$ is the set of joint distributions $\gamma(\boldsymbol{x},\boldsymbol{y})$ such that

$$\int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} = p(\boldsymbol{x}) \qquad \int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x} = q(\boldsymbol{y})$$

Computing the Wasserstein Distance

- To compute the Wasserstein distance we have to solve a minimisation task!

- This looks nasty, but it is a (continuous) linear programmming problem

- Suppose p and q were discrete distribution (i.e. $\boldsymbol{x}$ and $\boldsymbol{y}$ only take discrete points)

- Then we could treat each value of $\gamma(\boldsymbol{x}, \boldsymbol{y})$ as an element of a vector $\boldsymbol{\gamma}$ and each value of $d(\boldsymbol{x}, \boldsymbol{y})$ as an element of a vector $\boldsymbol{D}$

- Our objective is to choose $\boldsymbol{\gamma}$ to minimise $\boldsymbol{D}^{\mathsf{T}}\boldsymbol{\gamma}$

Computing the Wasserstein Distance

- To compute the Wasserstein distance we have to solve a minimisation task!
- This looks nasty, but it is a (continuous) linear programmming problem
- Suppose p and q were discrete distribution (i.e. $\boldsymbol{x}$ and $\boldsymbol{y}$ only take discrete points)
- Then we could treat each value of $\gamma(\boldsymbol{x},\boldsymbol{y})$ as an element of a vector $\boldsymbol{\gamma}$ and each value of $d(\boldsymbol{x},\boldsymbol{y})$ as an element of a vector $\boldsymbol{D}$
- Our objective is to choose $\boldsymbol{\gamma}$ to minimise $\boldsymbol{D}^{\mathsf{T}}\boldsymbol{\gamma}$

Computing the Wasserstein Distance

- To compute the Wasserstein distance we have to solve a minimisation task!
- This looks nasty, but it is a (continuous) linear programmming problem
- Suppose p and q were discrete distribution (i.e. $\boldsymbol{x}$ and $\boldsymbol{y}$ only take discrete points)
- Then we could treat each value of $\gamma(\boldsymbol{x}, \boldsymbol{y})$ as an element of a vector $\boldsymbol{\gamma}$ and each value of $d(\boldsymbol{x}, \boldsymbol{y})$ as an element of a vector $\boldsymbol{D}$
- Our objective is to choose $\boldsymbol{\gamma}$ to minimise $\boldsymbol{D}^{\mathsf{T}}\boldsymbol{\gamma}$

Computing the Wasserstein Distance

- To compute the Wasserstein distance we have to solve a minimisation task!
- This looks nasty, but it is a (continuous) linear programmming problem
- Suppose p and q were discrete distribution (i.e. $\boldsymbol{x}$ and $\boldsymbol{y}$ only take discrete points)
- Then we could treat each value of $\gamma(\boldsymbol{x}, \boldsymbol{y})$ as an element of a vector $\boldsymbol{\gamma}$ and each value of $d(\boldsymbol{x}, \boldsymbol{y})$ as an element of a vector $\boldsymbol{D}$
- Our objective is to choose $\boldsymbol{\gamma}$ to minimise $\boldsymbol{D}^{\mathsf{T}}\boldsymbol{\gamma}$

Computing the Wasserstein Distance

- To compute the Wasserstein distance we have to solve a minimisation task!
- This looks nasty, but it is a (continuous) linear programmming problem
- Suppose p and q were discrete distribution (i.e. $\boldsymbol{x}$ and $\boldsymbol{y}$ only take discrete points)
- Then we could treat each value of $\gamma(\boldsymbol{x}, \boldsymbol{y})$ as an element of a vector $\boldsymbol{\gamma}$ and each value of $d(\boldsymbol{x}, \boldsymbol{y})$ as an element of a vector $\boldsymbol{D}$
- Our objective is to choose $\boldsymbol{\gamma}$ to minimise $\boldsymbol{D}^{\mathsf{T}}\boldsymbol{\gamma}$

Constraints

$$\sum_j \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) = p(\boldsymbol{x}_i) \qquad\qquad \sum_i \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) = q(\boldsymbol{y}_j)$$

$$\mathbf{A}\,\boldsymbol{\gamma} = \boldsymbol{P}$$

$$\left(\begin{array}{cccc:cccc:c:cccc}
1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & \cdots\cdots\cdots & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & \cdots\cdots\cdots & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \;\; \vdots \;\cdots\; \vdots & \vdots & \vdots & \cdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots\cdots\cdots & 1 & 1 & \cdots & 1 \\
\hdashline
1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & \cdots\cdots\cdots & 1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 & \cdots\cdots\cdots & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \;\; \vdots \;\cdots\; \vdots & \vdots & \vdots & \cdots & \vdots \\
0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 & \cdots\cdots\cdots & 0 & 0 & \cdots & 1
\end{array}\right)
\left(\begin{array}{c}
\gamma(x_1,\, y_1) \\ \gamma(x_2,\, y_1) \\ \vdots \\ \gamma(x_n,\, y_1) \\ \hdashline
\gamma(x_1,\, y_2) \\ \gamma(x_2,\, y_2) \\ \vdots \\ \gamma(x_n,\, y_2) \\ \hdashline
\vdots \\ \vdots \\ \vdots \\ \vdots \\ \hdashline
\gamma(x_1,\, y_n) \\ \gamma(x_2,\, y_n) \\ \vdots \\ \gamma(x_n,\, y_n)
\end{array}\right)
=
\left(\begin{array}{c}
q(y_1) \\ q(y_2) \\ \vdots \\ q(y_n) \\ \hdashline
p(x_1) \\ p(x_2) \\ \vdots \\ p(x_n)
\end{array}\right)$$

Lagrange Formulation

- For discrete distributions

$$\min_{\boldsymbol{\gamma}} \boldsymbol{D}^{\mathsf{T}}\gamma$$

$$\text{subject to} \quad \mathbf{A}\boldsymbol{\gamma} = \boldsymbol{P}, \quad \boldsymbol{\gamma} \geq 0$$

- Writing the Lagrangian

$$\mathcal{L}(\boldsymbol{\gamma},\boldsymbol{\alpha}) = \boldsymbol{D}^{\mathsf{T}}\boldsymbol{\gamma} - \boldsymbol{\alpha}^{\mathsf{T}}\left(\mathbf{A}^{\mathsf{T}}\boldsymbol{\gamma} - \boldsymbol{P}\right)$$

where $\boldsymbol{\alpha}$ is a vector of Lagrange multipliers

- The solution to the discrete optimisation problem is given by

$$\min_{\boldsymbol{\gamma}} \max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\gamma},\boldsymbol{\alpha})$$

Lagrange Formulation

- For discrete distributions

$$\min_{\boldsymbol{\gamma}} \boldsymbol{D}^{\mathsf{T}} \gamma$$

$$\text{subject to} \quad \mathbf{A}\boldsymbol{\gamma} = \boldsymbol{P}, \quad \boldsymbol{\gamma} \geq 0$$

- Writing the Lagrangian

$$\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\alpha}) = \boldsymbol{D}^{\mathsf{T}} \boldsymbol{\gamma} - \boldsymbol{\alpha}^{\mathsf{T}} \left(\mathbf{A}^{\mathsf{T}} \boldsymbol{\gamma} - \boldsymbol{P} \right)$$

where $\boldsymbol{\alpha}$ is a vector of Lagrange multipliers

- The solution to the discrete optimisation problem is given by

$$\min_{\boldsymbol{\gamma}} \max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\alpha})$$

Lagrange Formulation

- For discrete distributions

$$\min_{\boldsymbol{\gamma}} \boldsymbol{D}^{\mathsf{T}} \gamma$$

$$\text{subject to} \quad \mathbf{A}\boldsymbol{\gamma} = \boldsymbol{P}, \quad \boldsymbol{\gamma} \geq 0$$

- Writing the Lagrangian

$$\mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\alpha}) = \boldsymbol{D}^{\mathsf{T}} \boldsymbol{\gamma} - \boldsymbol{\alpha}^{\mathsf{T}} \left(\mathbf{A}^{\mathsf{T}} \boldsymbol{\gamma} - \boldsymbol{P} \right)$$

  where $\boldsymbol{\alpha}$ is a vector of Lagrange multipliers

- The solution to the discrete optimisation problem is given by

$$\min_{\boldsymbol{\gamma}} \max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\gamma}, \boldsymbol{\alpha})$$

Dual Form

- We can rearrange

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\gamma},\boldsymbol{\alpha}) &= \boldsymbol{D}^{\mathsf{T}}\boldsymbol{\gamma} - \boldsymbol{\alpha}^{\mathsf{T}}(\mathbf{A}\boldsymbol{\gamma} - \boldsymbol{P}) \\
&= \boldsymbol{P}^{\mathsf{T}}\boldsymbol{\alpha} - \boldsymbol{\gamma}^{\mathsf{T}}\left(\mathbf{A}^{\mathsf{T}}\boldsymbol{\alpha} - \boldsymbol{D}\right)
\end{aligned}
$$

- We note that $\boldsymbol{\gamma} \geq 0$ so the dual problem is to find a vector $\boldsymbol{\alpha}$ that maximises $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{\alpha}$ subject to the constraints $\mathbf{A}^{\mathsf{T}}\boldsymbol{\alpha} \leq \boldsymbol{D}$

- Although the vector form allows us to make connections with our earlier discussion of linear programming, it is a little difficult to interpret

Dual Form

- We can rearrange

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\gamma},\boldsymbol{\alpha}) &= \boldsymbol{D}^{\mathsf{T}}\boldsymbol{\gamma} - \boldsymbol{\alpha}^{\mathsf{T}}(\mathbf{A}\boldsymbol{\gamma} - \boldsymbol{P}) \\
&= \boldsymbol{P}^{\mathsf{T}}\boldsymbol{\alpha} - \boldsymbol{\gamma}^{\mathsf{T}}\left(\mathbf{A}^{\mathsf{T}}\boldsymbol{\alpha} - \boldsymbol{D}\right)
\end{aligned}
$$

- We note that $\boldsymbol{\gamma} \geq 0$ so the dual problem is to find a vector $\boldsymbol{\alpha}$ that maximises $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{\alpha}$ subject to the constraints $\mathbf{A}^{\mathsf{T}}\boldsymbol{\alpha} \leq \boldsymbol{D}$

- Although the vector form allows us to make connections with our earlier discussion of linear programming, it is a little difficult to interpret

Dual Form

- We can rearrange

$$\begin{aligned}\mathcal{L}(\gamma,\boldsymbol{\alpha}) &= \boldsymbol{D}^{\mathsf{T}}\gamma - \boldsymbol{\alpha}^{\mathsf{T}}(\mathbf{A}\gamma - \boldsymbol{P}) \\ &= \boldsymbol{P}^{\mathsf{T}}\boldsymbol{\alpha} - \gamma^{\mathsf{T}}\left(\mathbf{A}^{\mathsf{T}}\boldsymbol{\alpha} - \boldsymbol{D}\right)\end{aligned}$$

- We note that $\gamma \geq 0$ so the dual problem is to find a vector $\boldsymbol{\alpha}$ that maximises $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{\alpha}$ subject to the constraints $\mathbf{A}^{\mathsf{T}}\boldsymbol{\alpha} \leq \boldsymbol{D}$

- Although the vector form allows us to make connections with our earlier discussion of linear programming, it is a little difficult to interpret

Dual Form

- We can rearrange

$$\begin{aligned}\mathcal{L}(\boldsymbol{\gamma},\boldsymbol{\alpha}) &= \boldsymbol{D}^{\mathsf{T}}\boldsymbol{\gamma} - \boldsymbol{\alpha}^{\mathsf{T}}(\mathbf{A}\boldsymbol{\gamma} - \boldsymbol{P}) \\ &= \boldsymbol{P}^{\mathsf{T}}\boldsymbol{\alpha} - \boldsymbol{\gamma}^{\mathsf{T}}\big(\mathbf{A}^{\mathsf{T}}\boldsymbol{\alpha} - \boldsymbol{D}\big)\end{aligned}$$

- We note that $\boldsymbol{\gamma} \geq 0$ so the dual problem is to find a vector $\boldsymbol{\alpha}$ that maximises $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{\alpha}$ subject to the constraints $\mathbf{A}^{\mathsf{T}}\boldsymbol{\alpha} \leq \boldsymbol{D}$

- Although the vector form allows us to make connections with our earlier discussion of linear programming, it is a little difficult to interpret

Explicit Form

- We can write a Lagrangian for the original problem

$$\mathcal{L} = \sum_{i,j} d(\boldsymbol{x}_i, \boldsymbol{y}_i)\gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - \sum_i \alpha(\boldsymbol{x}_i)\left(\sum_j \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - p(\boldsymbol{x}_i)\right) - \sum_j \beta(\boldsymbol{y}_j)\left(\sum_i \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - q(\boldsymbol{y}_j)\right)$$

subject to $\gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) \geq 0$

- Rearranging

$$\mathcal{L} = \sum_i \alpha(\boldsymbol{x}_i)p(\boldsymbol{x}_i) + \sum_j \beta(\boldsymbol{y}_j)q(\boldsymbol{y}_j) - \sum_{i,j} \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j)(\alpha(\boldsymbol{x}_i) + \beta(\boldsymbol{y}_j) - d(\boldsymbol{x}_i, \boldsymbol{y_i}))$$

- This is eqivalent to maximising $\sum_i \alpha(\boldsymbol{x}_i)p(\boldsymbol{x}_i) + \sum_j \beta(\boldsymbol{y}_j)q(\boldsymbol{y}_j)$, subject to

$$\forall i,j \quad \alpha(\boldsymbol{x}_i) + \beta(\boldsymbol{y}_j) \leq d(\boldsymbol{x}_i, \boldsymbol{y_j})$$

Explicit Form

- We can write a Lagrangian for the original problem

$$\mathcal{L} = \sum_{i,j} d(\boldsymbol{x}_i, \boldsymbol{y}_i)\gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - \sum_i \alpha(\boldsymbol{x}_i)\left(\sum_j \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - p(\boldsymbol{x}_i)\right)$$
$$- \sum_j \beta(\boldsymbol{y}_j)\left(\sum_i \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - q(\boldsymbol{y}_j)\right)$$

subject to $\gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) \geq 0$ where $\alpha(\boldsymbol{x}_i)$ and $\beta(\boldsymbol{y}_j)$ are Lagrange multipliers (they are components of $\boldsymbol{\alpha}$)

- Rearranging

$$\mathcal{L} = \sum_i \alpha(\boldsymbol{x}_i)p(\boldsymbol{x}_i) + \sum_j \beta(\boldsymbol{y}_j)q(\boldsymbol{y}_j) - \sum_{i,j} \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j)\big(\alpha(\boldsymbol{x}_i) + \beta(\boldsymbol{y}_j) - d(\boldsymbol{x}_i, \boldsymbol{y_i})\big)$$

- This is eqivalent to maximising $\sum_i \alpha(\boldsymbol{x}_i)p(\boldsymbol{x}_i) + \sum_j \beta(\boldsymbol{y}_j)q(\boldsymbol{y}_j)$, subject to

$$\forall i, j \quad \alpha(\boldsymbol{x}_i) + \beta(\boldsymbol{y}_j) \leq d(\boldsymbol{x}_i, \boldsymbol{y_j})$$

Explicit Form

- We can write a Lagrangian for the original problem

$$\mathcal{L} = \sum_{i,j} d(\boldsymbol{x}_i, \boldsymbol{y}_i)\gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - \sum_i \alpha(\boldsymbol{x}_i)\left(\sum_j \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - p(\boldsymbol{x}_i)\right)$$
$$- \sum_j \beta(\boldsymbol{y}_j)\left(\sum_i \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - q(\boldsymbol{y}_j)\right)$$

subject to $\gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) \geq 0$ where $\alpha(\boldsymbol{x}_i)$ and $\beta(\boldsymbol{y}_j)$ are Lagrange multipliers (they are components of $\boldsymbol{\alpha}$)

- Rearranging

$$\mathcal{L} = \sum_i \alpha(\boldsymbol{x}_i)p(\boldsymbol{x}_i) + \sum_j \beta(\boldsymbol{y}_j)q(\boldsymbol{y}_j) - \sum_{i,j} \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j)\left(\alpha(\boldsymbol{x}_i) + \beta(\boldsymbol{y}_j) - d(\boldsymbol{x}_i, \boldsymbol{y_i})\right)$$

- This is eqivalent to maximising $\sum_i \alpha(\boldsymbol{x}_i)p(\boldsymbol{x}_i) + \sum_j \beta(\boldsymbol{y}_j)q(\boldsymbol{y}_j)$, subject to

$$\forall i,j \quad \alpha(\boldsymbol{x}_i) + \beta(\boldsymbol{y}_j) \leq d(\boldsymbol{x}_i, \boldsymbol{y_j})$$

Explicit Form

- We can write a Lagrangian for the original problem

$$\mathcal{L} = \sum_{i,j} d(\boldsymbol{x}_i, \boldsymbol{y}_i)\gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - \sum_i \alpha(\boldsymbol{x}_i) \left(\sum_j \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - p(\boldsymbol{x}_i) \right) - \sum_j \beta(\boldsymbol{y}_j) \left(\sum_i \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) - q(\boldsymbol{y}_j) \right)$$

subject to $\gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) \geq 0$ where $\alpha(\boldsymbol{x}_i)$ and $\beta(\boldsymbol{y}_j)$ are Lagrange multipliers (they are components of $\boldsymbol{\alpha}$)

- Rearranging

$$\mathcal{L} = \sum_i \alpha(\boldsymbol{x}_i) p(\boldsymbol{x}_i) + \sum_j \beta(\boldsymbol{y}_j) q(\boldsymbol{y}_j) - \sum_{i,j} \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j) (\alpha(\boldsymbol{x}_i) + \beta(\boldsymbol{y}_j) - d(\boldsymbol{x}_i, \boldsymbol{y}_{\boldsymbol{i}}))$$

- This is eqivalent to maximising $\sum_i \alpha(\boldsymbol{x}_i) p(\boldsymbol{x}_i) + \sum_j \beta(\boldsymbol{y}_j) q(\boldsymbol{y}_j)$, subject to

$$\forall i, j \quad \alpha(\boldsymbol{x}_i) + \beta(\boldsymbol{y}_j) \leq d(\boldsymbol{x}_i, \boldsymbol{y}_{\boldsymbol{j}})$$

Continuous Form

- We can write a Lagrangian for the continuous problem

$$\begin{aligned}\mathcal{L} = \iint d(\boldsymbol{x},\boldsymbol{y})\gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y} &- \int \alpha(\boldsymbol{x})\left(\int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} - p(\boldsymbol{x})\right)\mathrm{d}\boldsymbol{x} \\ &- \int \beta(\boldsymbol{y})\left(\int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x} - q(\boldsymbol{y})\right)\mathrm{d}\boldsymbol{y}\end{aligned}$$

  subject to $\gamma(\boldsymbol{x},\boldsymbol{y}) \geq 0$

- Rearranging

$$\mathcal{L} = \int \alpha(\boldsymbol{x})p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y})q(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} - \iint \gamma(\boldsymbol{x},\boldsymbol{y})\left(\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) - d(\boldsymbol{x},\boldsymbol{y})\right)\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y}$$

- This is eqivalent to maximising $\int \alpha(\boldsymbol{x})p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y})q(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}$, subject to

$$\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) \leq d(\boldsymbol{x},\boldsymbol{y})$$

Continuous Form

- We can write a Lagrangian for the continuous problem

$$\mathcal{L} = \iint d(\boldsymbol{x}, \boldsymbol{y}) \gamma(\boldsymbol{x}, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{y} - \int \alpha(\boldsymbol{x}) \left(\int \gamma(\boldsymbol{x}, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} - p(\boldsymbol{x}) \right) \mathrm{d}\boldsymbol{x} \\ - \int \beta(\boldsymbol{y}) \left(\int \gamma(\boldsymbol{x}, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} - q(\boldsymbol{y}) \right) \mathrm{d}\boldsymbol{y}$$

subject to $\gamma(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ where $\alpha(\boldsymbol{x})$ and $\beta(\boldsymbol{y})$ are Lagrange multiplier functions

- Rearranging

$$\mathcal{L} = \int \alpha(\boldsymbol{x}) p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y}) q(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} - \iint \gamma(\boldsymbol{x}, \boldsymbol{y}) \left(\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) - d(\boldsymbol{x}, \boldsymbol{y}) \right) \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{y}$$

- This is eqivalent to maximising $\int \alpha(\boldsymbol{x}) p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y}) q(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y}$, subject to

$$\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) \leq d(\boldsymbol{x}, \boldsymbol{y})$$

Continuous Form

- We can write a Lagrangian for the continuous problem

$$\mathcal{L} = \iint d(\boldsymbol{x},\boldsymbol{y})\gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y} - \int \alpha(\boldsymbol{x})\left(\int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} - p(\boldsymbol{x})\right)\mathrm{d}\boldsymbol{x}$$
$$- \int \beta(\boldsymbol{y})\left(\int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x} - q(\boldsymbol{y})\right)\mathrm{d}\boldsymbol{y}$$

subject to $\gamma(\boldsymbol{x},\boldsymbol{y}) \geq 0$ where $\alpha(\boldsymbol{x})$ and $\beta(\boldsymbol{y})$ are Lagrange multiplier functions

- Rearranging

$$\mathcal{L} = \int \alpha(\boldsymbol{x})p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y})q(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} - \iint \gamma(\boldsymbol{x},\boldsymbol{y})\left(\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) - d(\boldsymbol{x},\boldsymbol{y})\right)\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y}$$

- This is eqivalent to maximising $\int \alpha(\boldsymbol{x})p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y})q(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}$, subject to

$$\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) \leq d(\boldsymbol{x},\boldsymbol{y})$$

Continuous Form

- We can write a Lagrangian for the continuous problem

$$\begin{aligned}\mathcal{L} = \iint d(\boldsymbol{x},\boldsymbol{y})\gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y} &- \int \alpha(\boldsymbol{x})\left(\int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} - p(\boldsymbol{x})\right)\mathrm{d}\boldsymbol{x} \\ &- \int \beta(\boldsymbol{y})\left(\int \gamma(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}\boldsymbol{x} - q(\boldsymbol{y})\right)\mathrm{d}\boldsymbol{y}\end{aligned}$$

  subject to $\gamma(\boldsymbol{x},\boldsymbol{y}) \geq 0$ where $\alpha(\boldsymbol{x})$ and $\beta(\boldsymbol{y})$ are Lagrange multiplier functions

- Rearranging

$$\mathcal{L} = \int \alpha(\boldsymbol{x})p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y})q(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} - \iint \gamma(\boldsymbol{x},\boldsymbol{y})\left(\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) - d(\boldsymbol{x},\boldsymbol{y})\right)\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y}$$

- This is eqivalent to maximising $\int \alpha(\boldsymbol{x})p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y})q(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}$, subject to

$$\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) \leq d(\boldsymbol{x},\boldsymbol{y})$$

Dual Form Constraint

- We note that $\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) \leq d(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{x}$ and $\boldsymbol{y}$
- This has to be true when $\boldsymbol{x} = \boldsymbol{y}$ so that

$$\alpha(\boldsymbol{x}) + \beta(\boldsymbol{x}) \leq d(\boldsymbol{x},\boldsymbol{x}) = 0$$

- So $\beta(\boldsymbol{x}) = -\alpha(\boldsymbol{x}) - \epsilon(\boldsymbol{x})$ where $\epsilon(\boldsymbol{x}) \geq 0$
- But want to maximise

$$\int \alpha(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y}) q(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \int \alpha(\boldsymbol{x}) \left(p(\boldsymbol{x}) - q(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} - \int q(\boldsymbol{x}) \epsilon(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

- This is maximised when $\epsilon(\boldsymbol{x}) = 0$

Dual Form Constraint

- We note that $\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) \leq d(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{x}$ and $\boldsymbol{y}$

- This has to be true when $\boldsymbol{x} = \boldsymbol{y}$ so that

$$\alpha(\boldsymbol{x}) + \beta(\boldsymbol{x}) \leq d(\boldsymbol{x},\boldsymbol{x}) = 0$$

- So $\beta(\boldsymbol{x}) = -\alpha(\boldsymbol{x}) - \epsilon(\boldsymbol{x})$ where $\epsilon(\boldsymbol{x}) \geq 0$

- But want to maximise

$$\int \alpha(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y}) q(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \int \alpha(\boldsymbol{x}) \left(p(\boldsymbol{x}) - q(\boldsymbol{x}) \right) \mathrm{d}\boldsymbol{x} - \int q(\boldsymbol{x}) \epsilon(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

- This is maximised when $\epsilon(\boldsymbol{x}) = 0$

Dual Form Constraint

- We note that $\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) \leq d(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{x}$ and $\boldsymbol{y}$
- This has to be true when $\boldsymbol{x} = \boldsymbol{y}$ so that

$$\alpha(\boldsymbol{x}) + \beta(\boldsymbol{x}) \leq d(\boldsymbol{x},\boldsymbol{x}) = 0$$

- So $\beta(\boldsymbol{x}) = -\alpha(\boldsymbol{x}) - \epsilon(\boldsymbol{x})$ where $\epsilon(\boldsymbol{x}) \geq 0$
- But want to maximise

$$\int \alpha(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y}) q(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \int \alpha(\boldsymbol{x}) \left(p(\boldsymbol{x}) - q(\boldsymbol{x}) \right) \mathrm{d}\boldsymbol{x} - \int q(\boldsymbol{x}) \epsilon(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

- This is maximised when $\epsilon(\boldsymbol{x}) = 0$

Dual Form Constraint

- We note that $\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) \leq d(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{x}$ and $\boldsymbol{y}$
- This has to be true when $\boldsymbol{x} = \boldsymbol{y}$ so that

$$\alpha(\boldsymbol{x}) + \beta(\boldsymbol{x}) \leq d(\boldsymbol{x},\boldsymbol{x}) = 0$$

- So $\beta(\boldsymbol{x}) = -\alpha(\boldsymbol{x}) - \epsilon(\boldsymbol{x})$ where $\epsilon(\boldsymbol{x}) \geq 0$
- But want to maximise

$$\int \alpha(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y}) q(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \int \alpha(\boldsymbol{x}) \left(p(\boldsymbol{x}) - q(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} - \int q(\boldsymbol{x}) \epsilon(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

- This is maximised when $\epsilon(\boldsymbol{x}) = 0$

Dual Form Constraint

- We note that $\alpha(\boldsymbol{x}) + \beta(\boldsymbol{y}) \le d(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{x}$ and $\boldsymbol{y}$
- This has to be true when $\boldsymbol{x} = \boldsymbol{y}$ so that

$$\alpha(\boldsymbol{x}) + \beta(\boldsymbol{x}) \le d(\boldsymbol{x},\boldsymbol{x}) = 0$$

- So $\beta(\boldsymbol{x}) = -\alpha(\boldsymbol{x}) - \epsilon(\boldsymbol{x})$ where $\epsilon(\boldsymbol{x}) \ge 0$
- But want to maximise

$$\int \alpha(\boldsymbol{x}) p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int \beta(\boldsymbol{y}) q(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \int \alpha(\boldsymbol{x}) \left(p(\boldsymbol{x}) - q(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} - \int q(\boldsymbol{x}) \epsilon(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

- This is maximised when $\epsilon(\boldsymbol{x}) = 0$

Dual Form

- Thus the dual problem is to find a function $\alpha(\boldsymbol{x})$—or a vector of functions $(\alpha(\boldsymbol{x}_i)|i)$—that maximises

$$\int \alpha(\boldsymbol{x}) \left(p(\boldsymbol{x}) - q(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}$$

- Subject to the constraint

$$\alpha(\boldsymbol{x}) - \alpha(\boldsymbol{y}) \leq d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$$

- This is a continuity constraint on the Lagrange multiplier function $\alpha(\boldsymbol{x})$ known as Lipschitz-1

Dual Form

- Thus the dual problem is to find a function $\alpha(\boldsymbol{x})$—or a vector of functions $(\alpha(\boldsymbol{x}_i)|i)$—that maximises

$$\int \alpha(\boldsymbol{x})\,(p(\boldsymbol{x}) - q(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x}$$

- Subject to the constraint

$$\alpha(\boldsymbol{x}) - \alpha(\boldsymbol{y}) \leq d(\boldsymbol{x},\boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$$

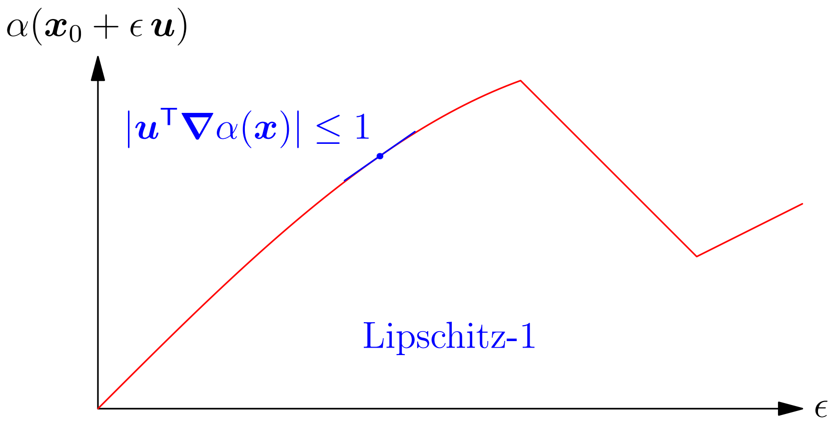
- This is a continuity constraint on the Lagrange multiplier function $\alpha(\boldsymbol{x})$ known as Lipschitz-1

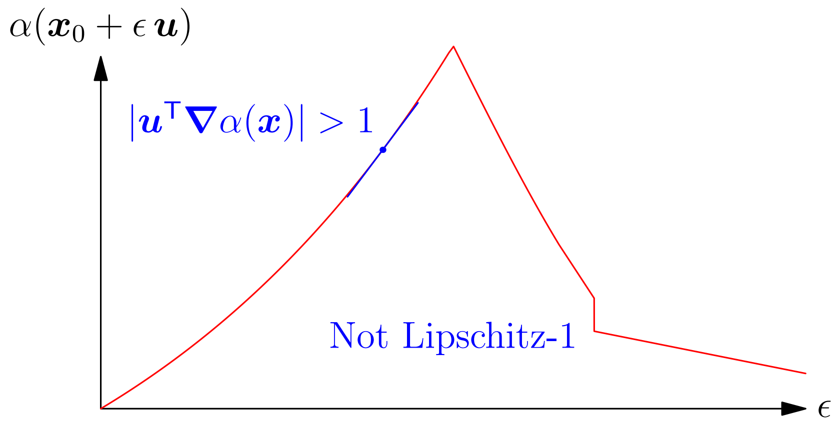
Lipschitz-1 Functions

- We note for a Lipschitz-1 function and any unit vector $\boldsymbol{u}$

$$\boldsymbol{u}^{\mathsf{T}} \boldsymbol{\nabla} \alpha(\boldsymbol{x}) = \lim_{\epsilon \to 0} \frac{\alpha(\boldsymbol{x}) - \alpha(\boldsymbol{x} + \epsilon \boldsymbol{u})}{\epsilon} \leq 1$$

- That is, at every point the gradient in all directions must be less than 1



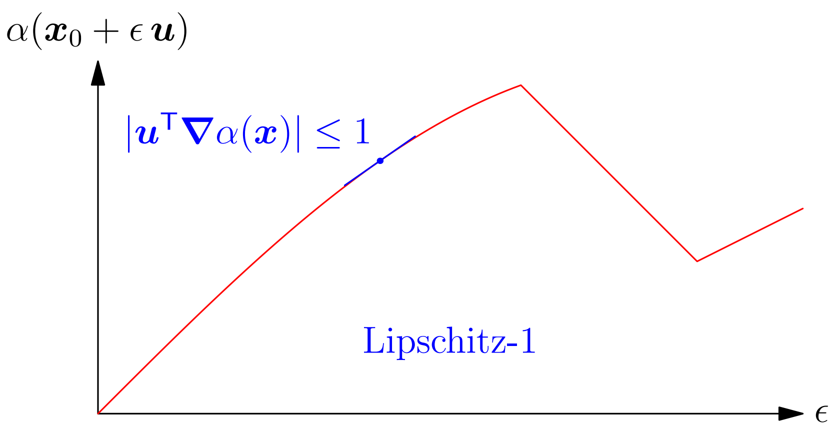


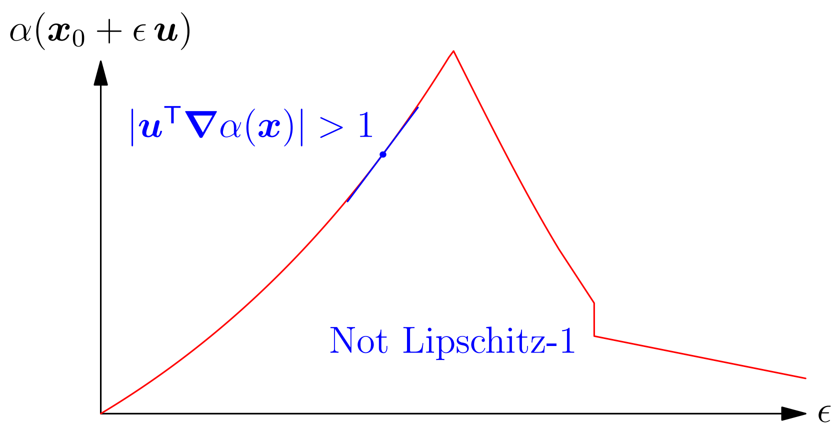
Lipschitz-1 Functions

- We note for a Lipschitz-1 function and any unit vector $\boldsymbol{u}$

$$\boldsymbol{u}^{\mathsf{T}} \boldsymbol{\nabla} \alpha(\boldsymbol{x}) = \lim_{\epsilon \to 0} \frac{\alpha(\boldsymbol{x}) - \alpha(\boldsymbol{x} + \epsilon \boldsymbol{u})}{\epsilon} \leq 1$$

- That is, at every point the gradient in all directions must be less than 1



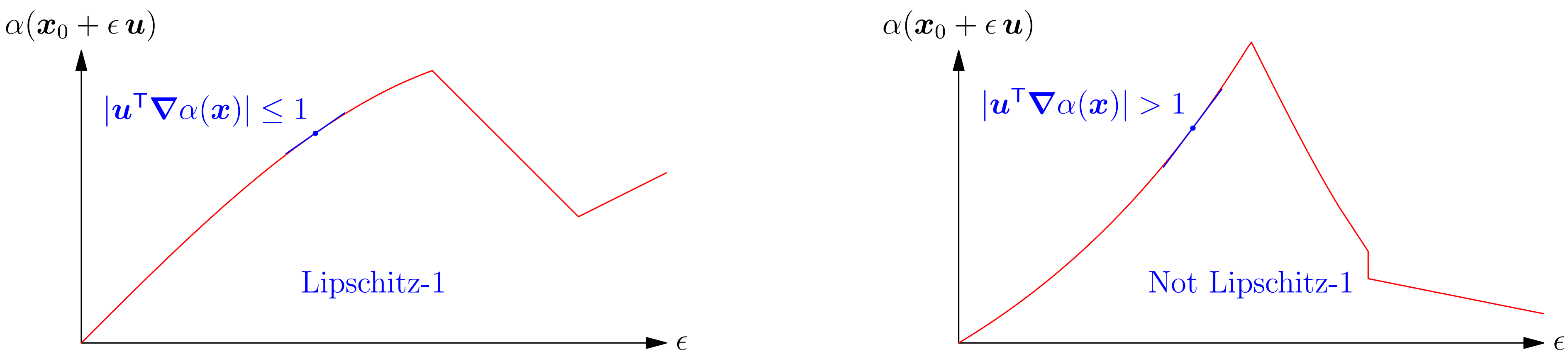


Lipschitz-1 Functions

- We note for a Lipschitz-1 function and any unit vector $\boldsymbol{u}$

$$\boldsymbol{u}^\mathsf{T}\boldsymbol{\nabla}\alpha(\boldsymbol{x}) = \lim_{\epsilon\to 0}\frac{\alpha(\boldsymbol{x}) - \alpha(\boldsymbol{x}+\epsilon\,\boldsymbol{u})}{\epsilon} \leq 1$$

- That is, at every point the gradient in all directions must be less than 1 (since the gradient defines the direction of greatest increase it is both necessary and sufficient for $\|\boldsymbol{\nabla}\alpha(\boldsymbol{x})\| \leq 1$ everywhere)

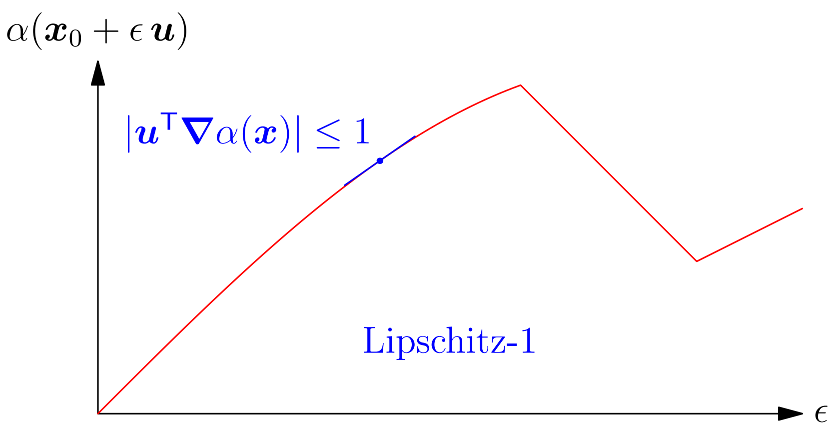


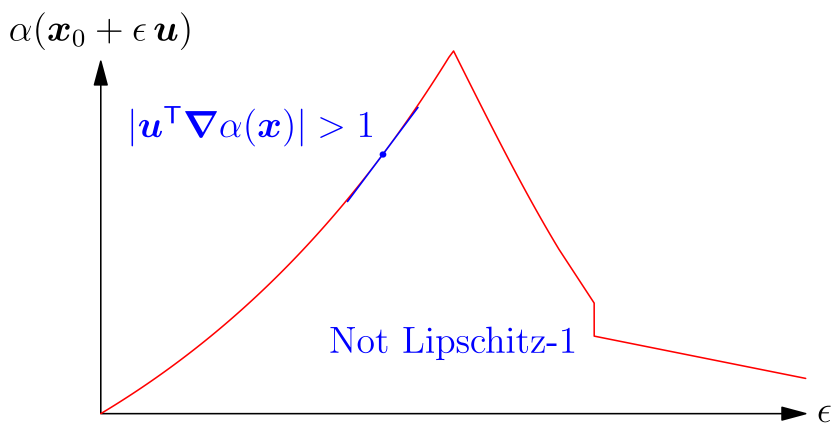
Lipschitz-1 Functions

- We note for a Lipschitz-1 function and any unit vector $\boldsymbol{u}$

$$\boldsymbol{u}^{\mathsf{T}} \boldsymbol{\nabla} \alpha(\boldsymbol{x}) = \lim_{\epsilon \to 0} \frac{\alpha(\boldsymbol{x}) - \alpha(\boldsymbol{x} + \epsilon \boldsymbol{u})}{\epsilon} \leq 1$$

- That is, at every point the gradient in all directions must be less than 1 (since the gradient defines the direction of greatest increase it is both necessary and sufficient for $\|\boldsymbol{\nabla} \alpha(\boldsymbol{x})\| \leq 1$ everywhere)





Calculating the Wasserstein Distance

- To recall the big picture we want to compute the Wasserstein distance

$$W(p,q) = \min_{\gamma \in \Lambda(p,q)} \mathbb{E}_\gamma[d(\boldsymbol{x},\boldsymbol{y})]$$

- For high dimensional objects $\gamma(\boldsymbol{x},\boldsymbol{y})$ would be a huge object to approximate

- Instead we can compute the Wasserstein distance in the dual formulation

$$W(p,q) = \max_{\alpha(\boldsymbol{x})} \int \alpha(\boldsymbol{x}) \left(p(\boldsymbol{x}) - q(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}$$

subject to the constraint that $\alpha(\boldsymbol{x})$ is a Lipschitz-1 function

Calculating the Wasserstein Distance

- To recall the big picture we want to compute the Wasserstein distance

$$W(p,q) = \min_{\gamma \in \Lambda(p,q)} \mathbb{E}_\gamma[d(\boldsymbol{x},\boldsymbol{y})]$$

- For high dimensional objects $\gamma(\boldsymbol{x},\boldsymbol{y})$ would be a huge object to approximate

- Instead we can compute the Wasserstein distance in the dual formulation

$$W(p,q) = \max_{\alpha(\boldsymbol{x})} \int \alpha(\boldsymbol{x}) \left(p(\boldsymbol{x}) - q(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}$$

subject to the constraint that $\alpha(\boldsymbol{x})$ is a Lipschitz-1 function

Calculating the Wasserstein Distance

- To recall the big picture we want to compute the Wasserstein distance

$$W(p,q) = \min_{\gamma \in \Lambda(p,q)} \mathbb{E}_\gamma[d(\boldsymbol{x},\boldsymbol{y})]$$

- For high dimensional objects $\gamma(\boldsymbol{x},\boldsymbol{y})$ would be a huge object to approximate

- Instead we can compute the Wasserstein distance in the dual formulation

$$W(p,q) = \max_{\alpha(\boldsymbol{x})} \int \alpha(\boldsymbol{x})\,(p(\boldsymbol{x}) - q(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x}$$

subject to the constraint that $\alpha(\boldsymbol{x})$ is a Lipschitz-1 function

Calculating the Wasserstein Distance

- To recall the big picture we want to compute the Wasserstein distance

$$W(p,q) = \min_{\gamma \in \Lambda(p,q)} \mathbb{E}_\gamma[d(\boldsymbol{x},\boldsymbol{y})]$$

- For high dimensional objects $\gamma(\boldsymbol{x},\boldsymbol{y})$ would be a huge object to approximate

- Instead we can compute the Wasserstein distance in the dual formulation

$$W(p,q) = \max_{\alpha(\boldsymbol{x})} \int \alpha(\boldsymbol{x}) \left(p(\boldsymbol{x}) - q(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} = \max_{\alpha} \mathbb{E}_p[\alpha(\boldsymbol{X})] - \mathbb{E}_q[\alpha(\boldsymbol{X})]$$

subject to the constraint that $\alpha(\boldsymbol{x})$ is a Lipschitz-1 function

Calculating the Wasserstein Distance

- To recall the big picture we want to compute the Wasserstein distance

$$W(p,q) = \min_{\gamma \in \Lambda(p,q)} \mathbb{E}_\gamma[d(\boldsymbol{x},\boldsymbol{y})]$$

- For high dimensional objects $\gamma(\boldsymbol{x},\boldsymbol{y})$ would be a huge object to approximate

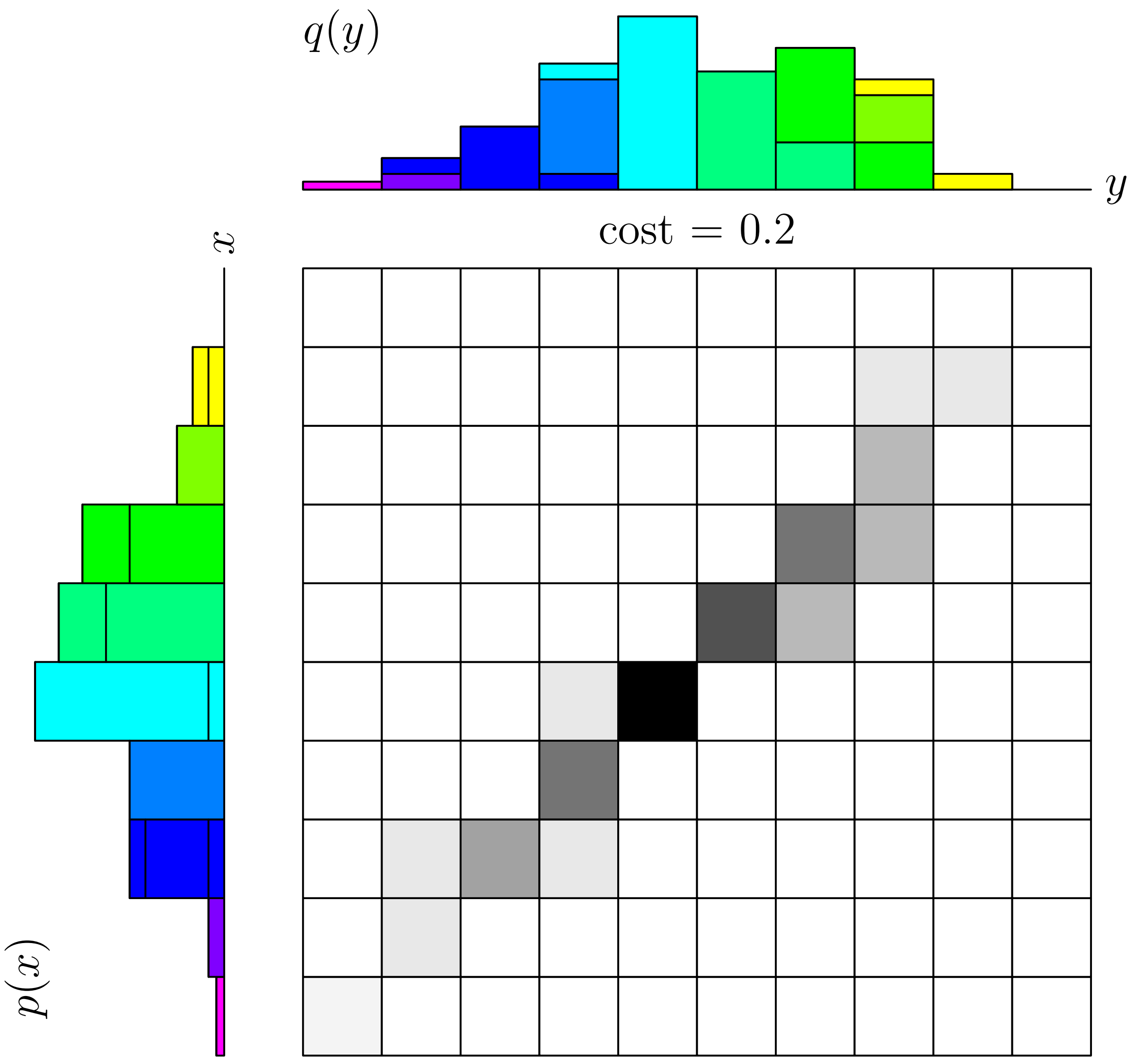
- Instead we can compute the Wasserstein distance in the dual formulation

$$W(p,q) = \max_{\alpha(\boldsymbol{x})} \int \alpha(\boldsymbol{x}) \left(p(\boldsymbol{x}) - q(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} = \max_{\alpha} \mathbb{E}_p[\alpha(\boldsymbol{X})] - \mathbb{E}_q[\alpha(\boldsymbol{X})]$$

subject to the constraint that $\alpha(\boldsymbol{x})$ is a Lipschitz-1 function

Outline

1. GANs
2. Wasserstein Distance
3. **Wasserstein GANs**



Back to GANs

- What has this got do with GANs?
- Suppose we want to minimise the distance between the distribution $p(\boldsymbol{x})$ of real images (of which $\mathcal{D}$ are samples) and the distribution $q(\boldsymbol{x})$ of images drawn from a generator
- We can use a normal GAN generator, $G(\boldsymbol{z}, \boldsymbol{w}_G)$, that generates an image when given a random variable $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- To do this we choose the weights, $\boldsymbol{w}_G$ of the generator to minimise

$$W(p, q) = \max_{\alpha(\boldsymbol{x})} \left(\mathbb{E}_{\boldsymbol{x} \sim p}[\alpha(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim q}[\alpha(\boldsymbol{x})] \right)$$

Back to GANs

- What has this got do with GANs?
- Suppose we want to minimise the distance between the distribution $p(\boldsymbol{x})$ of real images (of which $\mathcal{D}$ are samples) and the distribution $q(\boldsymbol{x})$ of images drawn from a generator
- We can use a normal GAN generator, $G(\boldsymbol{z}, \boldsymbol{w}_G)$, that generates an image when given a random variable $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- To do this we choose the weights, $\boldsymbol{w}_G$ of the generator to minimise

$$W(p,q) = \max_{\alpha(\boldsymbol{x})} \left(\mathbb{E}_{\boldsymbol{x}\sim p}[\alpha(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim q}[\alpha(\boldsymbol{x})] \right)$$

Back to GANs

- What has this got do with GANs?
- Suppose we want to minimise the distance between the distribution $p(\boldsymbol{x})$ of real images (of which $\mathcal{D}$ are samples) and the distribution $q(\boldsymbol{x})$ of images drawn from a generator
- We can use a normal GAN generator, $G(\boldsymbol{z}, \boldsymbol{w}_G)$, that generates an image when given a random variable $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- To do this we choose the weights, $\boldsymbol{w}_G$ of the generator to minimise

$$W(p, q) = \max_{\alpha(\boldsymbol{x})} \left(\mathbb{E}_{\boldsymbol{x} \sim p}[\alpha(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim q}[\alpha(\boldsymbol{x})] \right)$$

Back to GANs

- What has this got do with GANs?
- Suppose we want to minimise the distance between the distribution $p(\boldsymbol{x})$ of real images (of which $\mathcal{D}$ are samples) and the distribution $q(\boldsymbol{x})$ of images drawn from a generator
- We can use a normal GAN generator, $G(\boldsymbol{z}, \boldsymbol{w}_G)$, that generates an image when given a random variable $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- To do this we choose the weights, $\boldsymbol{w}_G$ of the generator to minimise

$$W(p, q) = \max_{\alpha(\boldsymbol{x})} \left(\mathbb{E}_{\boldsymbol{x}\sim p}[\alpha(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim q}[\alpha(\boldsymbol{x})] \right)$$

Estimating Expectations

- Although we can't compute $\mathbb{E}_p[\alpha(\boldsymbol{x})]$ and $\mathbb{E}_q[\alpha(\boldsymbol{x})]$ exactly, we can estimate them from samples

$$\mathbb{E}_p[\alpha(\boldsymbol{x})] \approx \frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x}\in\mathcal{B}} \alpha(\boldsymbol{x}), \qquad \mathbb{E}_q[\alpha(\boldsymbol{x})] \approx \frac{1}{n}\sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G))$$

- where $\mathcal{B} \subset \mathcal{D}$ is a minibatch of true images and $\boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

- From this we can choose $\boldsymbol{w}_G$ to minimise

$$C = \frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x}\in\mathcal{B}} \alpha(\boldsymbol{x}) - \frac{1}{n}\sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G))$$

Estimating Expectations

- Although we can't compute $\mathbb{E}_p[\alpha(\boldsymbol{x})]$ and $\mathbb{E}_q[\alpha(\boldsymbol{x})]$ exactly, we can estimate them from samples

$$\mathbb{E}_p[\alpha(\boldsymbol{x})] \approx \frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x}\in\mathcal{B}} \alpha(\boldsymbol{x}), \qquad \mathbb{E}_q[\alpha(\boldsymbol{x})] \approx \frac{1}{n}\sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G))$$

- where $\mathcal{B} \subset \mathcal{D}$ is a minibatch of true images and $\boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

- From this we can choose $\boldsymbol{w}_G$ to minimise

$$C = \frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x}\in\mathcal{B}} \alpha(\boldsymbol{x}) - \frac{1}{n}\sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G))$$

Estimating Expectations

- Although we can't compute $\mathbb{E}_p[\alpha(\boldsymbol{x})]$ and $\mathbb{E}_q[\alpha(\boldsymbol{x})]$ exactly, we can estimate them from samples

$$\mathbb{E}_p[\alpha(\boldsymbol{x})] \approx \frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x}\in\mathcal{B}}\alpha(\boldsymbol{x}), \qquad \mathbb{E}_q[\alpha(\boldsymbol{x})] \approx \frac{1}{n}\sum_{i=1}^{n}\alpha(G(\boldsymbol{z}_i,\boldsymbol{w}_G))$$

- where $\mathcal{B} \subset \mathcal{D}$ is a minibatch of true images and $\boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0},\mathbf{I})$

- From this we can choose $\boldsymbol{w}_G$ to minimise

$$C = \frac{1}{|\mathcal{B}|}\sum_{\boldsymbol{x}\in\mathcal{B}}\alpha(\boldsymbol{x}) - \frac{1}{n}\sum_{i=1}^{n}\alpha(G(\boldsymbol{z}_i,\boldsymbol{w}_G))$$

The Critic

- For this quantity to approximate the Wasserstein distance we need to find a function $\alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha)$ that maximises C
- To do this we learn a second network, the critic or discriminator whose job it is to maximise

$$C = \sum_{\boldsymbol{x} \in \mathcal{B}} \alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha) - \frac{1}{n} \sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G), \boldsymbol{w}_\alpha)$$

- The network $\alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha)$ should be Lipschitz-1 (which we usually botched by, for example, putting in a punishment term to ensure the gradient of the output for a randomly chosen input $\boldsymbol{x}$ never exceeds 1)

The Critic

- For this quantity to approximate the Wasserstein distance we need to find a function $\alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha)$ that maximises C
- To do this we learn a second network, the critic or discriminator whose job it is to maximise

$$C = \sum_{\boldsymbol{x} \in \mathcal{B}} \alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha) - \frac{1}{n} \sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G), \boldsymbol{w}_\alpha)$$

- The network $\alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha)$ should be Lipschitz-1 (which we usually botched by, for example, putting in a punishment term to ensure the gradient of the output for a randomly chosen input $\boldsymbol{x}$ never exceeds 1)

The Critic

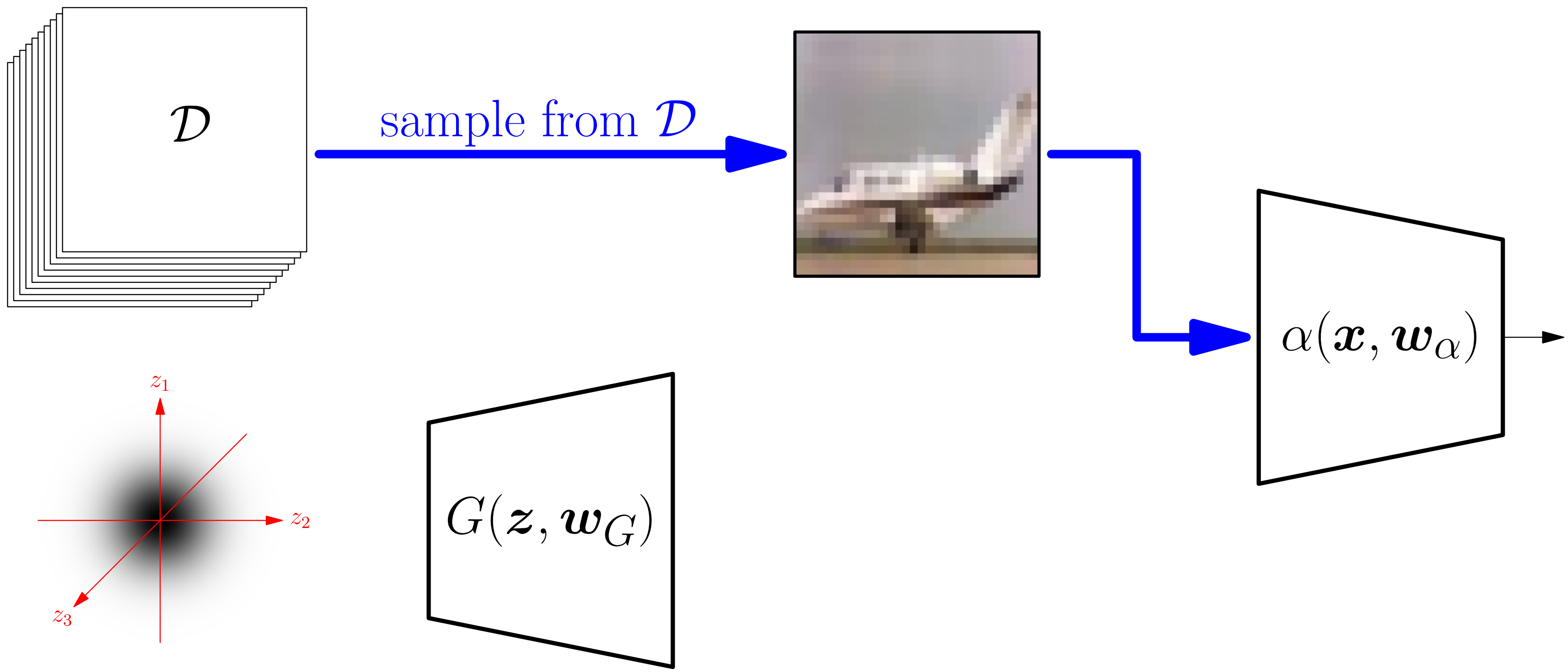
- For this quantity to approximate the Wasserstein distance we need to find a function $\alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha)$ that maximises C
- To do this we learn a second network, the critic or discriminator whose job it is to maximise

$$C = \sum_{\boldsymbol{x} \in \mathcal{B}} \alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha) - \frac{1}{n} \sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G), \boldsymbol{w}_\alpha)$$

- The network $\alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha)$ should be Lipschitz-1 (which we usually botched by, for example, putting in a punishment term to ensure the gradient of the output for a randomly chosen input $\boldsymbol{x}$ never exceeds 1)

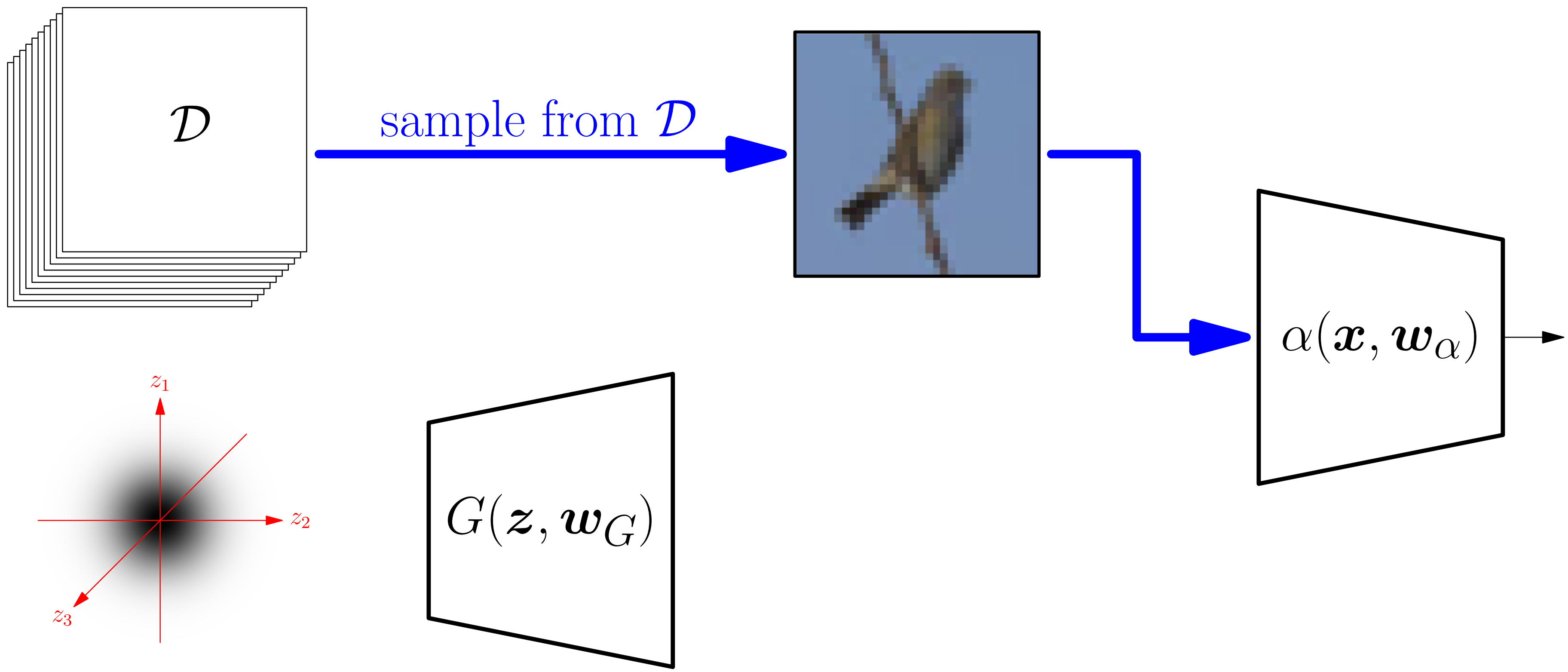
Wasserstein GANs

$$\max_{\boldsymbol{w}_\alpha} \min_{\boldsymbol{w}_G} \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{x} \in \mathcal{B}} \alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha) - \frac{1}{n} \sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G), \boldsymbol{w}_\alpha)$$



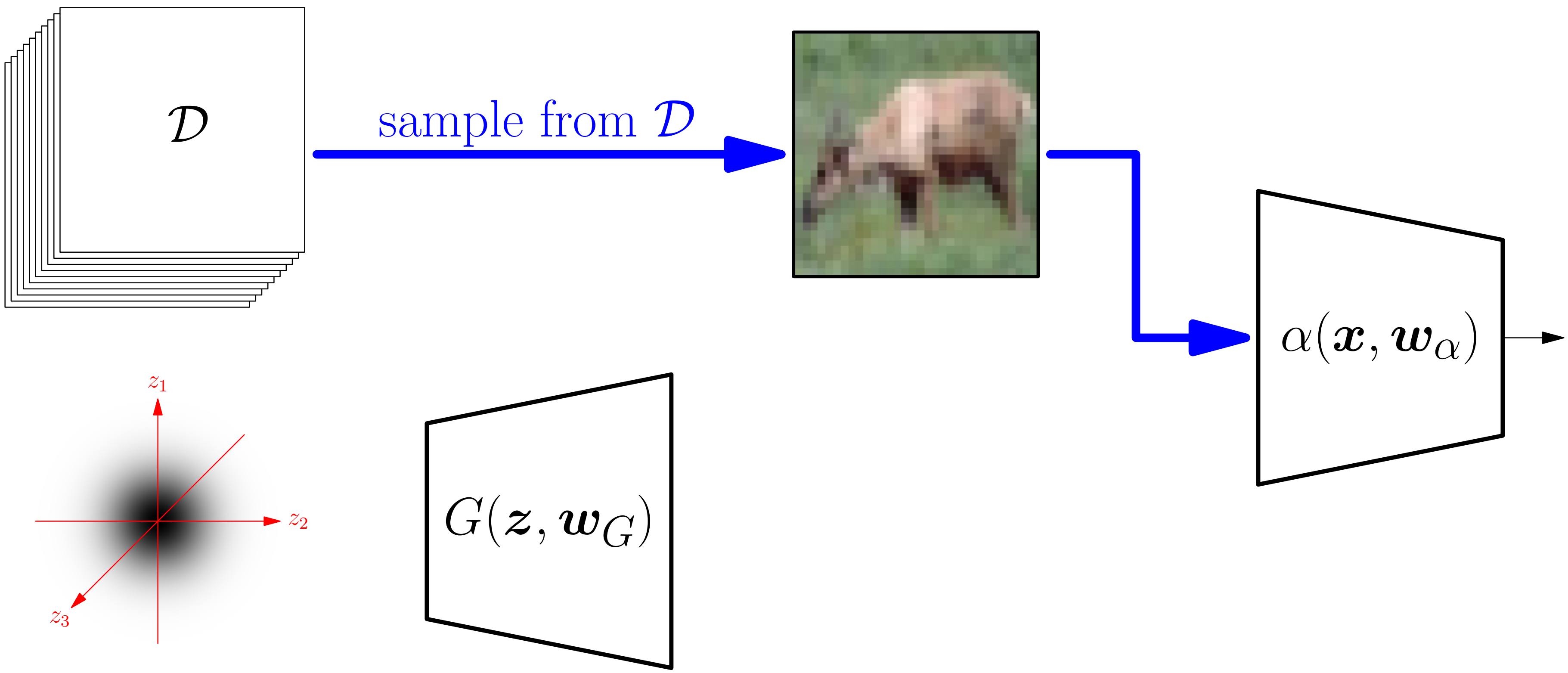
Wasserstein GANs

$$\max_{\boldsymbol{w}_\alpha} \min_{\boldsymbol{w}_G} \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{x}\in\mathcal{B}} \alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha) - \frac{1}{n}\sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G), \boldsymbol{w}_\alpha)$$



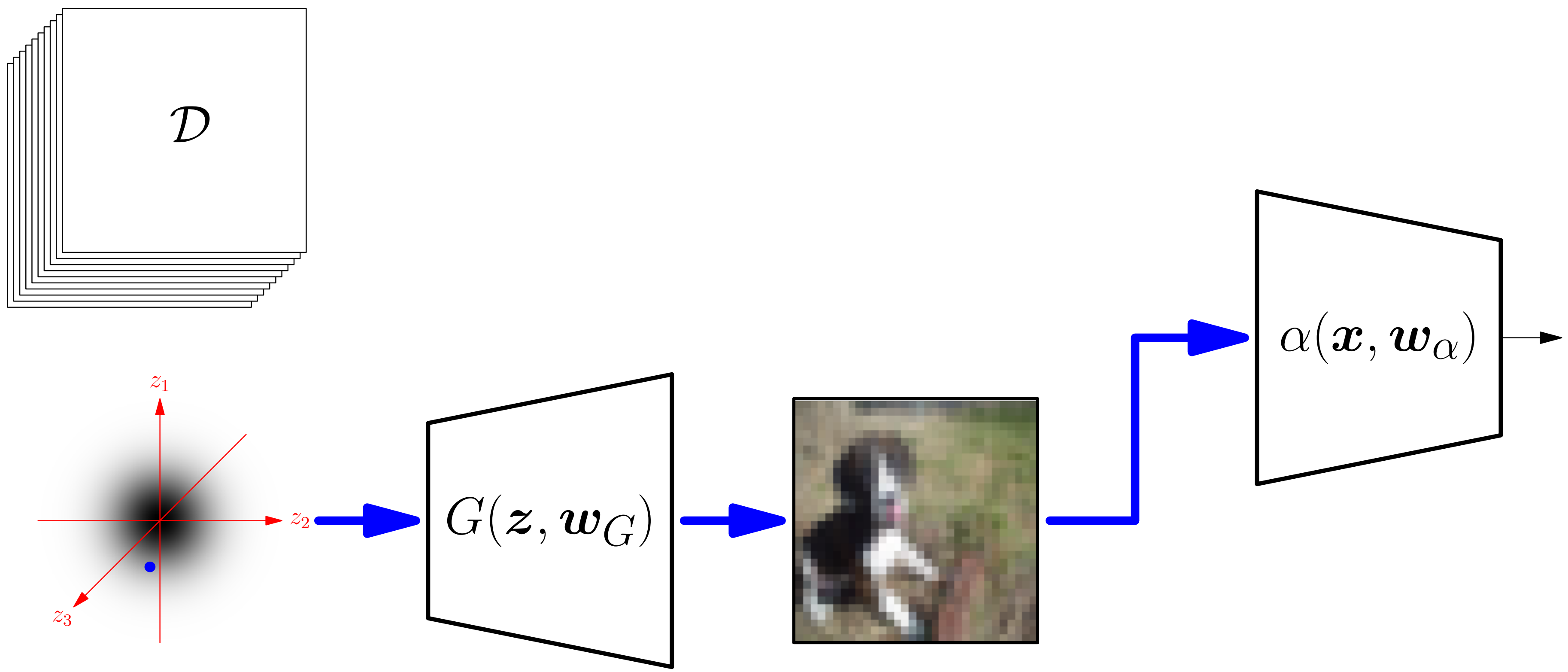
Wasserstein GANs

$$\max_{\boldsymbol{w}_\alpha} \min_{\boldsymbol{w}_G} \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{x} \in \mathcal{B}} \alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha) - \frac{1}{n} \sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G), \boldsymbol{w}_\alpha)$$



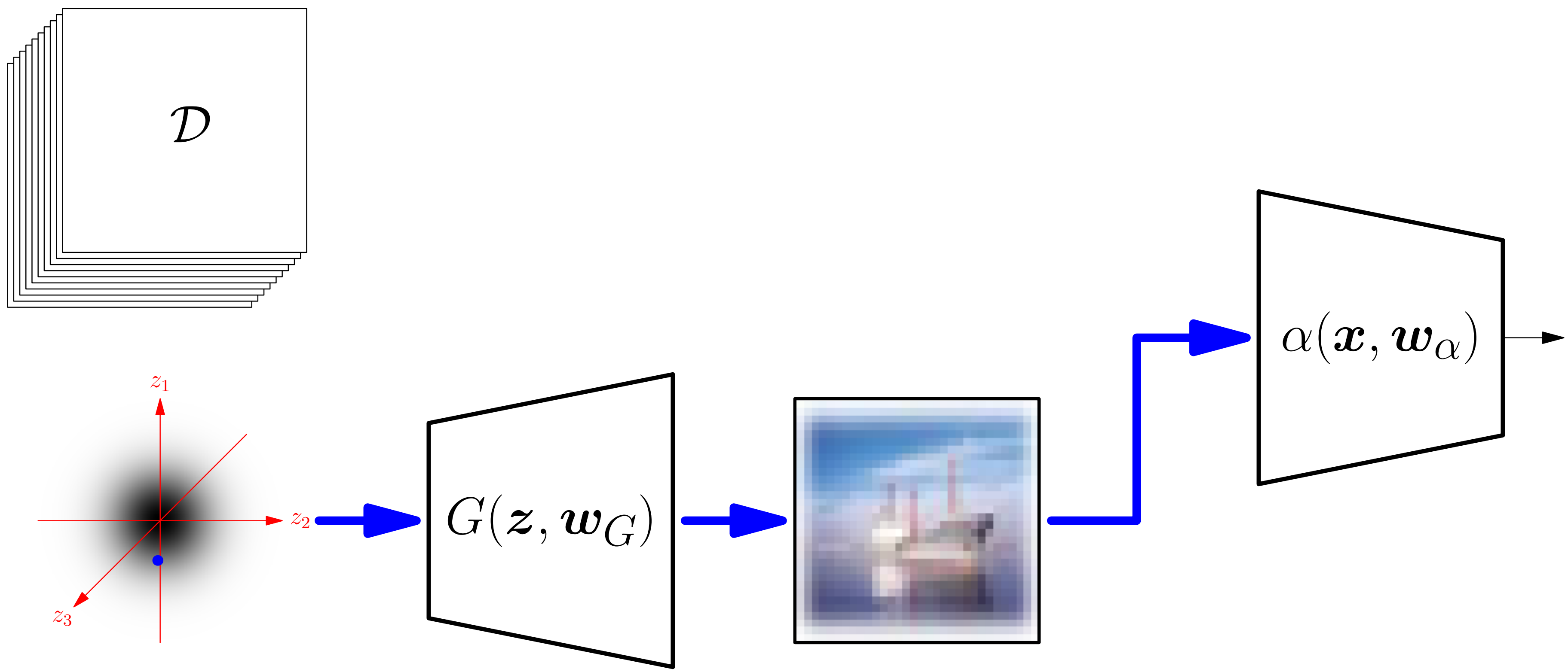
Wasserstein GANs

$$\max_{\boldsymbol{w}_\alpha} \min_{\boldsymbol{w}_G} \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{x} \in \mathcal{B}} \alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha) - \frac{1}{n} \sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G), \boldsymbol{w}_\alpha)$$



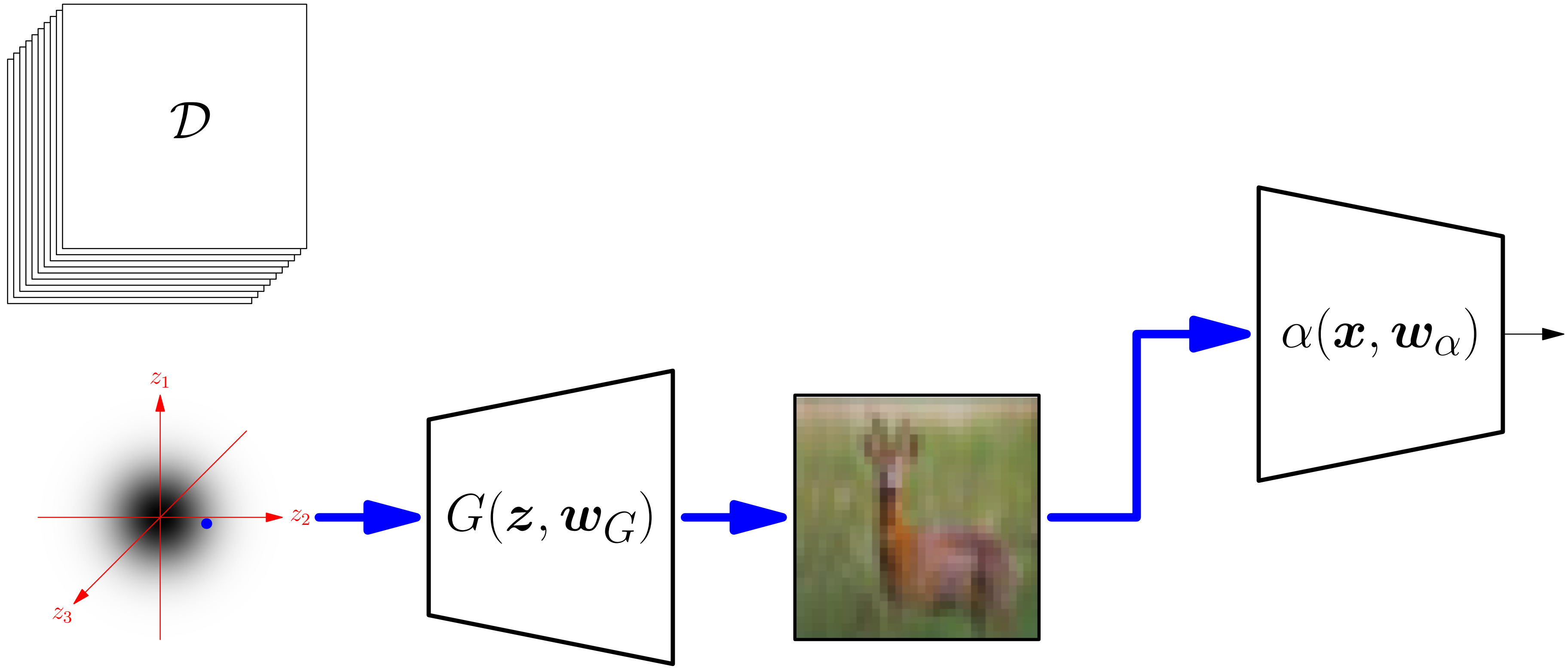
Wasserstein GANs

$$\max_{\boldsymbol{w}_\alpha} \min_{\boldsymbol{w}_G} \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{x}\in\mathcal{B}} \alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha) - \frac{1}{n}\sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G), \boldsymbol{w}_\alpha)$$



Wasserstein GANs

$$\max_{\boldsymbol{w}_\alpha} \min_{\boldsymbol{w}_G} \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{x} \in \mathcal{B}} \alpha(\boldsymbol{x}, \boldsymbol{w}_\alpha) - \frac{1}{n} \sum_{i=1}^{n} \alpha(G(\boldsymbol{z}_i, \boldsymbol{w}_G), \boldsymbol{w}_\alpha)$$



Lesson

- Wasserstein GANs are, at least for me, one of the most elegant pieces of theory in recent years
- By trying to minimise the Wasserstein distance between the distribution of a generator and a true distribution we arrive at optimising two adversarial networks just like a GAN
- This uses a rather beautiful dual formulation
- It is claimed that W-GANs solve many of the problems of traditional GANs

Lesson

- Wasserstein GANs are, at least for me, one of the most elegant pieces of theory in recent years
- By trying to minimise the Wasserstein distance between the distribution of a generator and a true distribution we arrive at optimising two adversarial networks just like a GAN
- This uses a rather beautiful dual formulation
- It is claimed that W-GANs solve many of the problems of traditional GANs

Lesson

- Wasserstein GANs are, at least for me, one of the most elegant pieces of theory in recent years
- By trying to minimise the Wasserstein distance between the distribution of a generator and a true distribution we arrive at optimising two adversarial networks just like a GAN
- This uses a rather beautiful dual formulation
- It is claimed that W-GANs solve many of the problems of traditional GANs

Lesson

- Wasserstein GANs are, at least for me, one of the most elegant pieces of theory in recent years
- By trying to minimise the Wasserstein distance between the distribution of a generator and a true distribution we arrive at optimising two adversarial networks just like a GAN
- This uses a rather beautiful dual formulation
- It is claimed that W-GANs solve many of the problems of traditional GANs