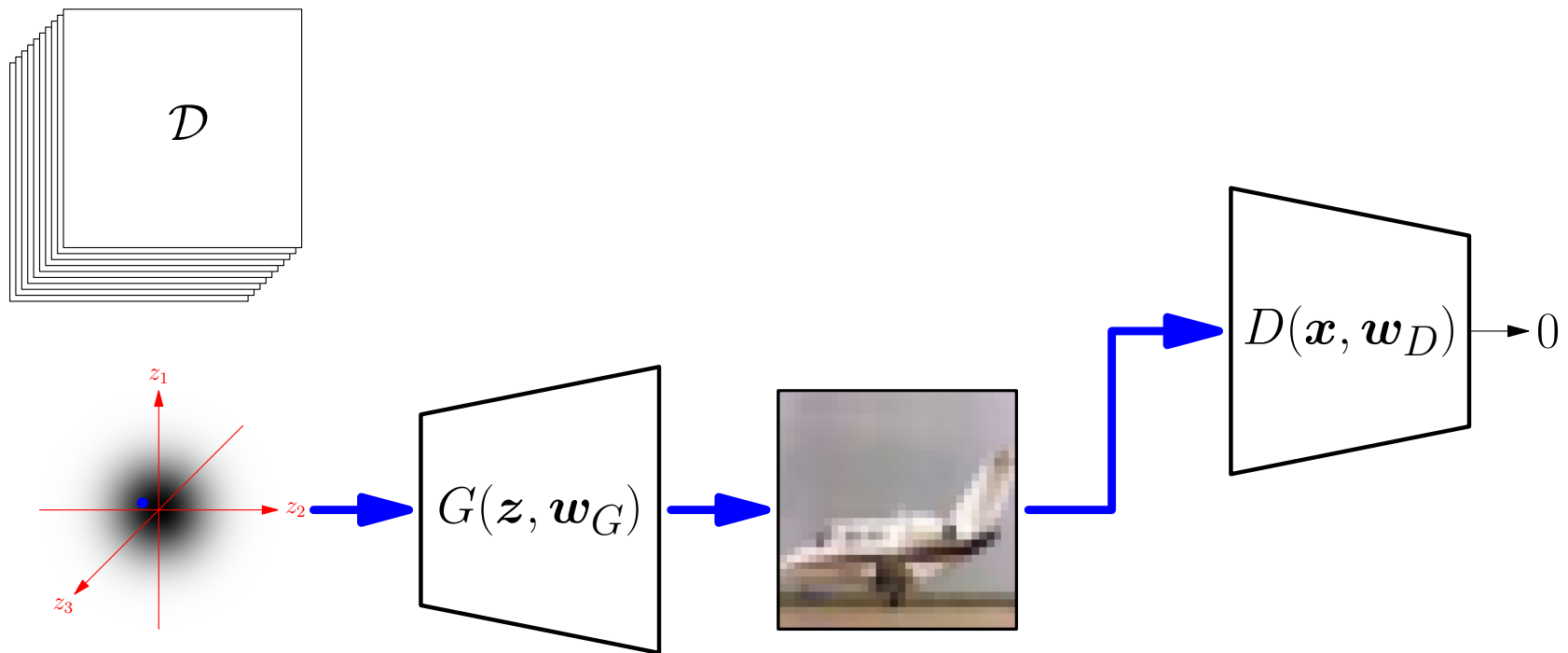


Advanced Machine Learning

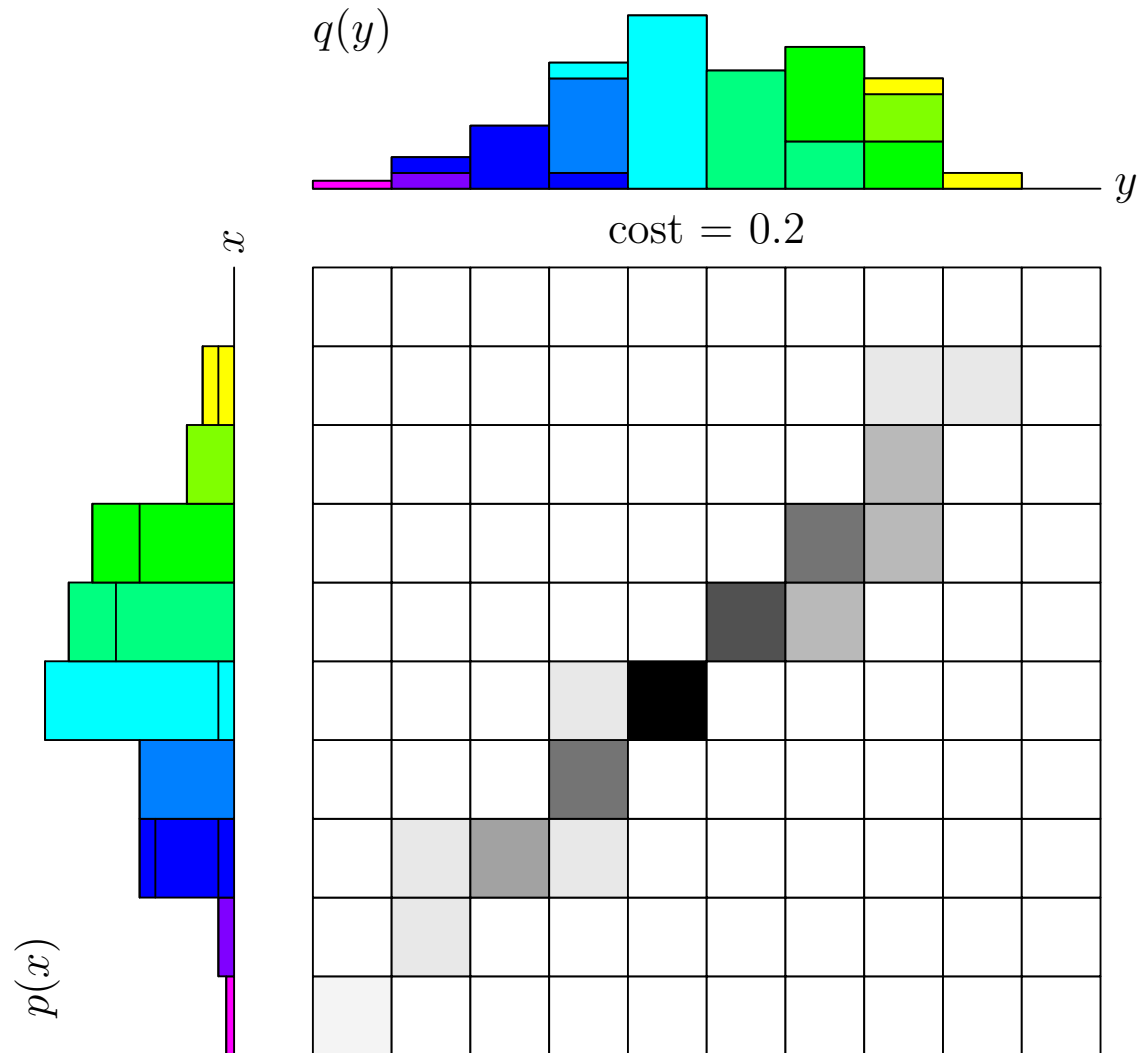
Wasserstein GANs



GANs, Wasserstein distance, Duality, WGANs

Outline

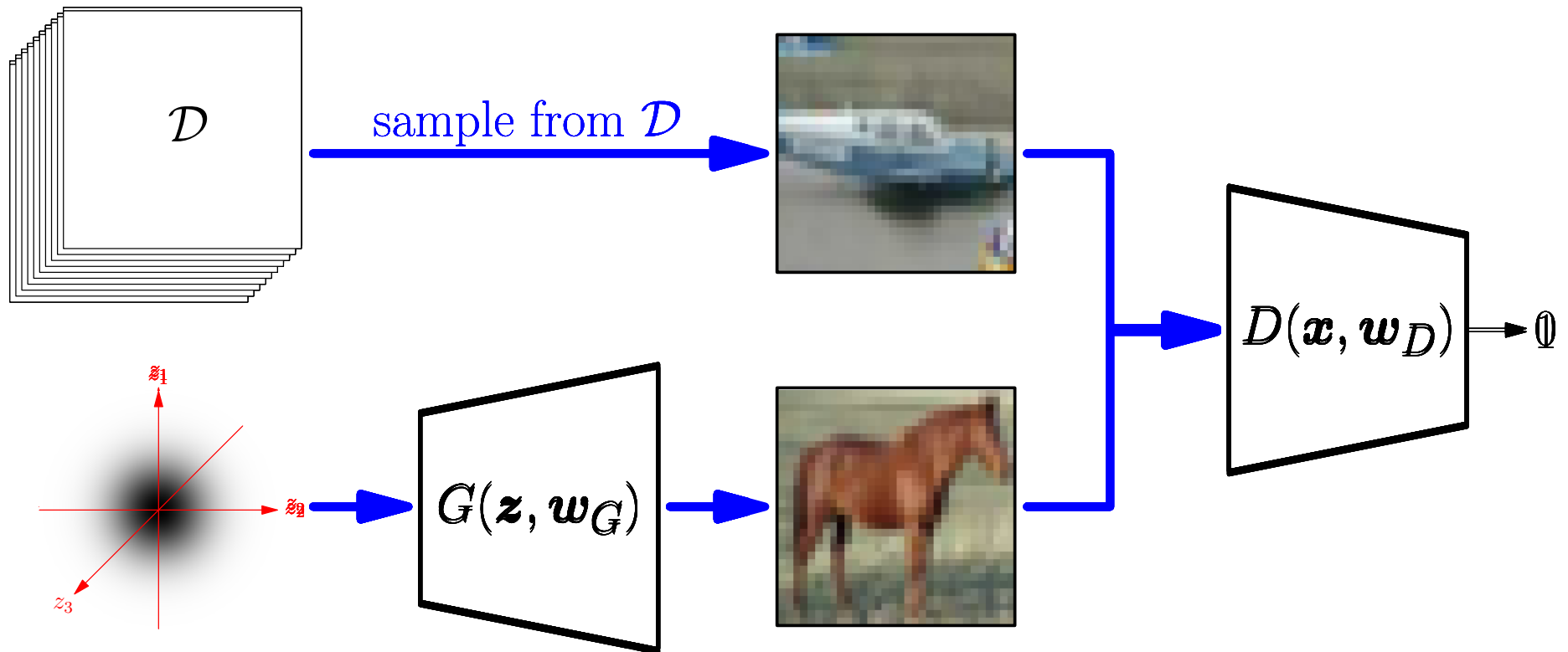
1. **GANs**
2. Wasserstein Distance
3. Wasserstein GANs



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 questionable
- But nobody cares because they are cool!
- *Out of date warning:* someone invented diffusion models

How GANs Work



Training GANs

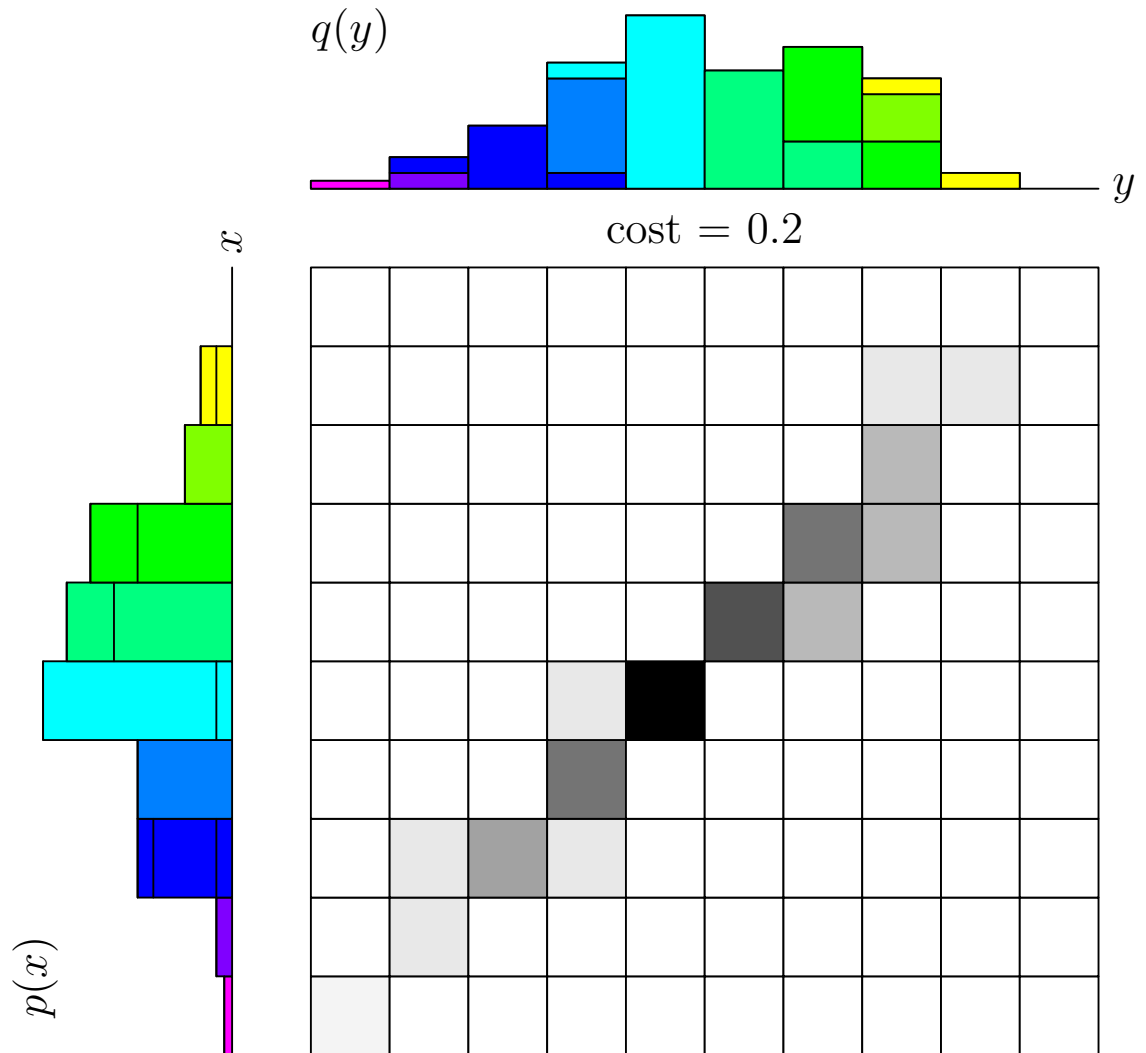
- The loss of the generator depends on its ability to trick the discriminator■
- The loss of the discriminator depends on its ability not to be tricked■
- We try to train the two networks simultaneously■
- We hope that over time the generator produces better and better fakes■

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 seemingly 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

$$\text{KL}(p\|q) = \int p(\mathbf{x}) \log\left(\frac{p(\mathbf{x})}{q(\mathbf{y})}\right) d\mathbf{x} \blacksquare$$

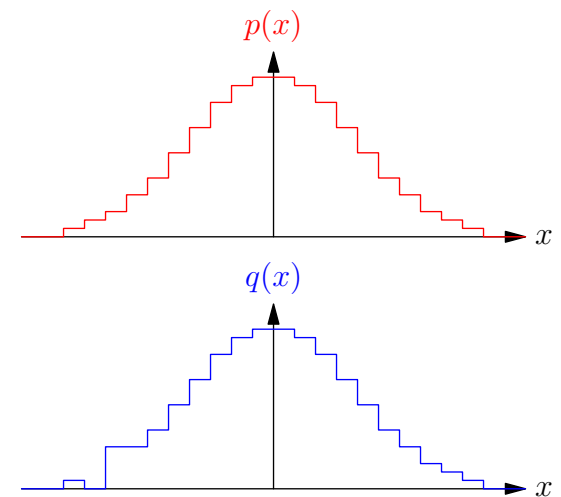
- This is very commonly used in ML (e.g. VAEs, Variational Approximation)■

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 $q(\mathbf{x}) = 0$ when $p(\mathbf{x}) \neq 0$ then $\log\left(\frac{p(\mathbf{x})}{q(\mathbf{y})}\right)$ diverges■

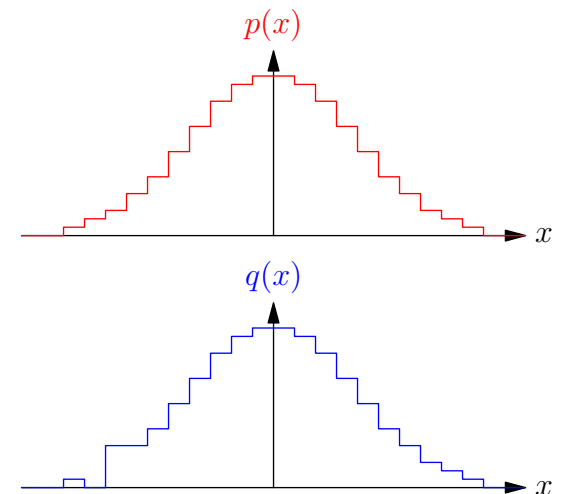


- We can therefore have distributions that seem very similar but their KL-divergence is huge (or infinite)■

Wasserstein Distance

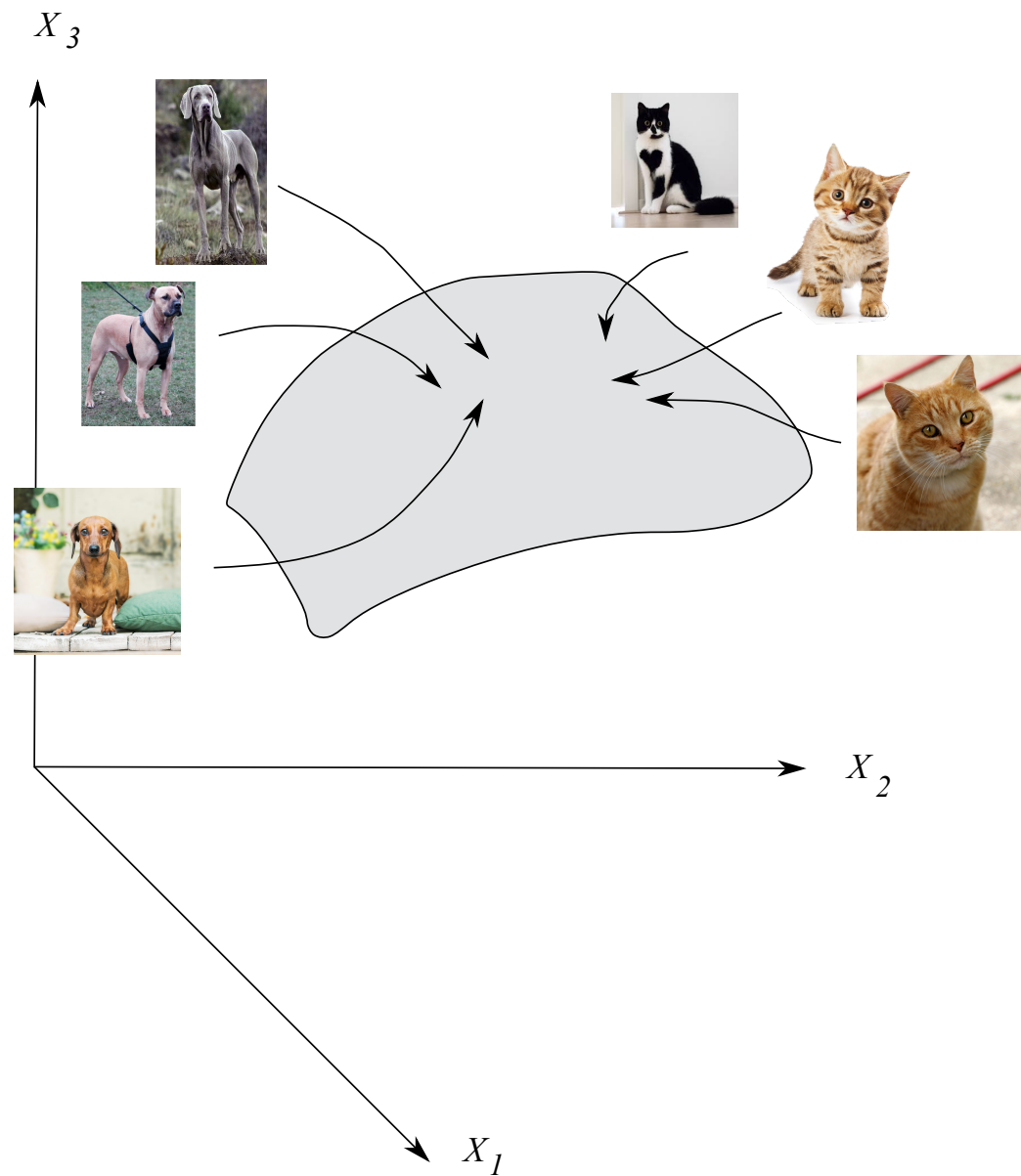
- A more benign measure of the differences between two probability functions is the **Wasserstein** or **Earth Moving** distance■

- 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■

High Probability Manifold



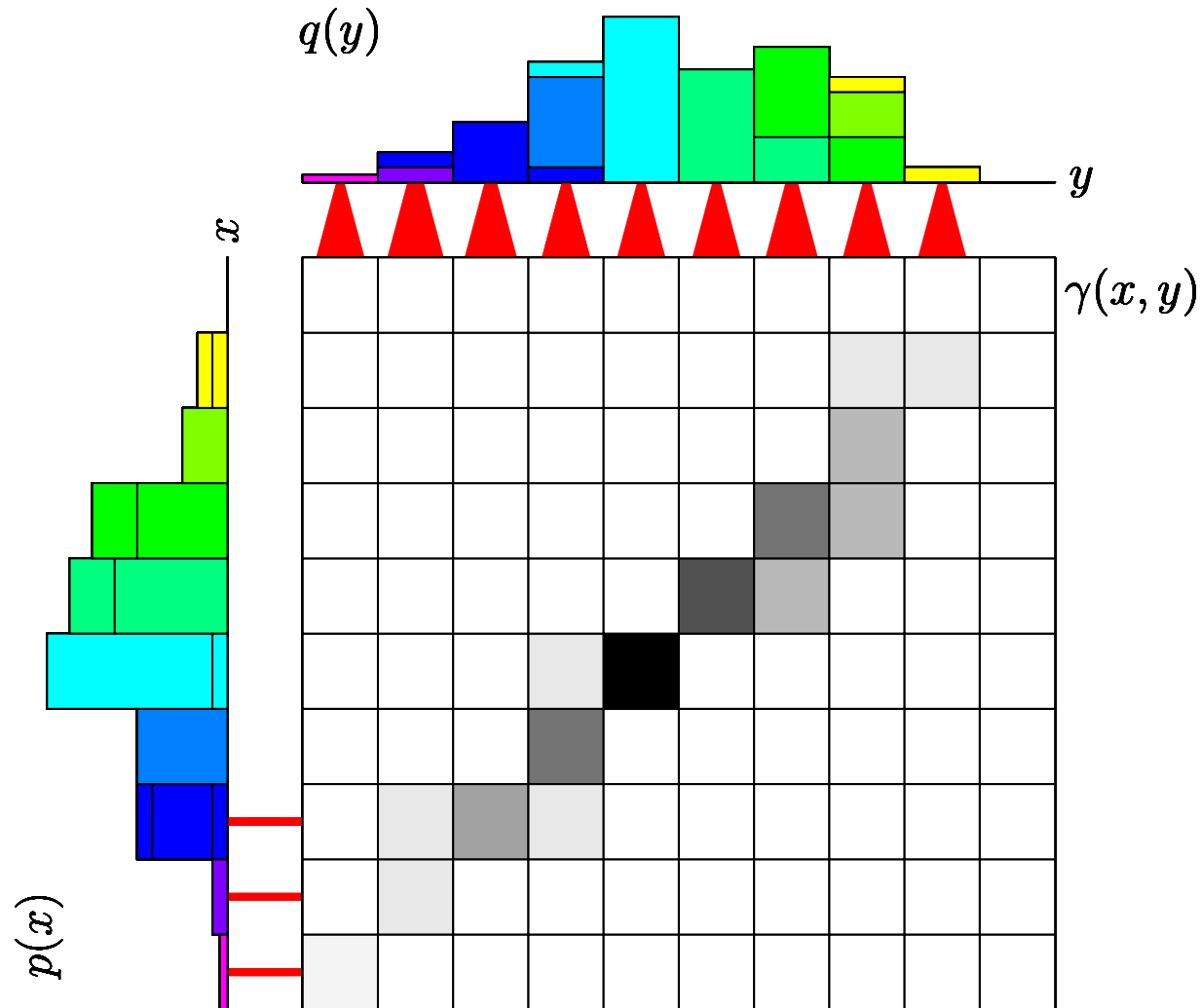
Transportation Policy

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

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

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

Transportation Policy



The Cost of Transport

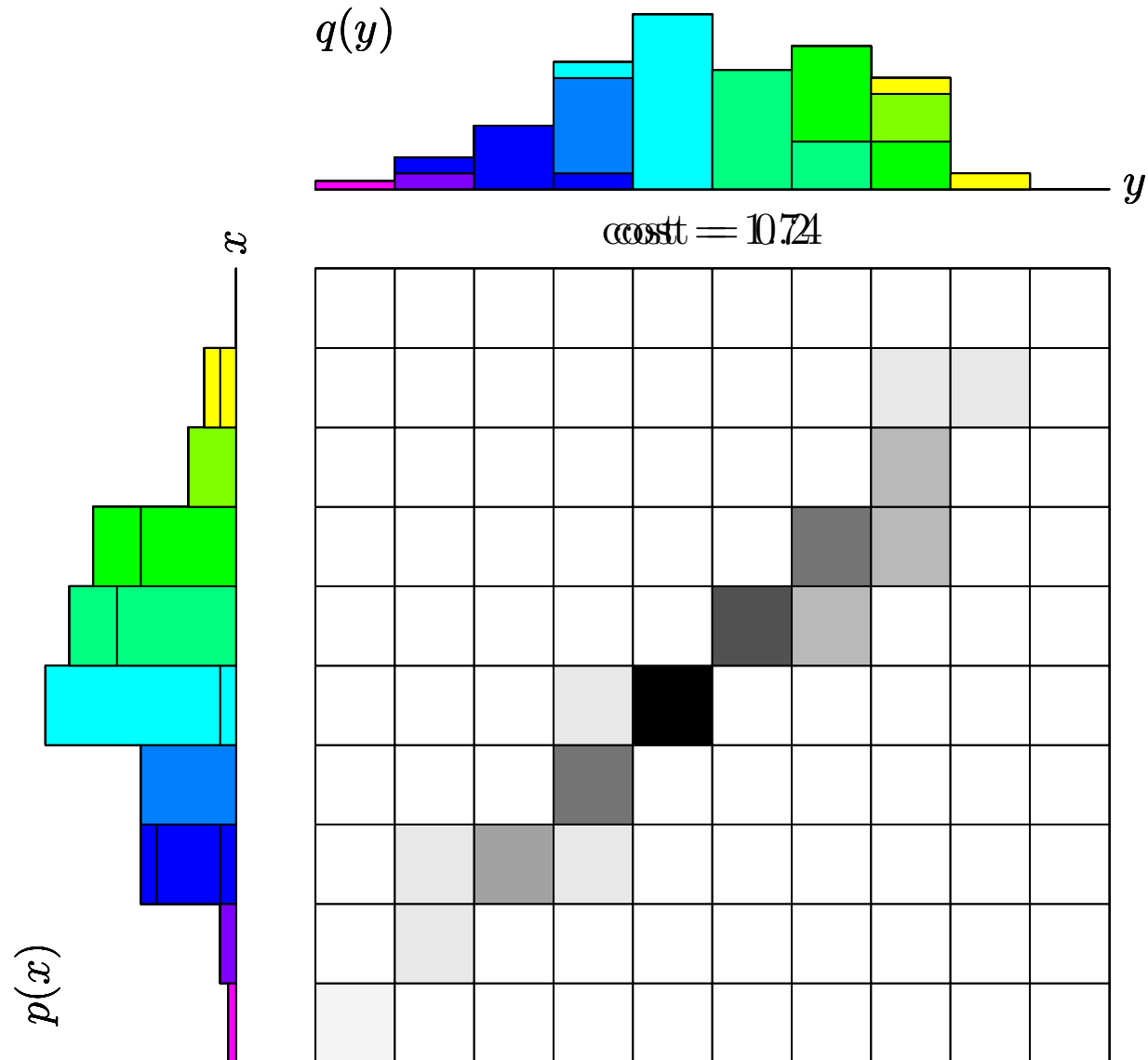
- We want to choose the transportation policy that minimises the amount of probability mass we need to move■
- Let $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$ be a distance measure then the cost of a transportation policy is

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

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

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

Transportation Cost



The Wasserstein Distance

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

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

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

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

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 programming problem■
- Suppose p and q were discrete distribution (i.e. x and y only take discrete points)■
- Then we could treat each value of $\gamma(x, y)$ as an element of a vector γ and each value of $d(x, y)$ as an element of a vector D ■
- Our objective is to choose γ to minimise $D^T \gamma$ ■

Constraints

$$\sum_j \gamma(\mathbf{x}_i, \mathbf{y}_j) = p(\mathbf{x}_i)$$

$$\sum_i \gamma(\mathbf{x}_i, \mathbf{y}_j) = q(\mathbf{y}_j)$$

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

$$\begin{pmatrix} 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & \cdots & 1 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & \cdots & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 & \cdots & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 & \cdots & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} \gamma(x_1, y_1) \\ \gamma(x_2, y_1) \\ \vdots \\ \gamma(x_n, y_1) \\ \gamma(x_1, y_2) \\ \gamma(x_2, y_2) \\ \vdots \\ \gamma(x_n, y_2) \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \gamma(x_1, y_n) \\ \gamma(x_2, y_n) \\ \vdots \\ \gamma(x_n, y_n) \end{pmatrix} = \begin{pmatrix} q(y_1) \\ q(y_2) \\ \vdots \\ q(y_n) \\ p(x_1) \\ p(x_2) \\ \vdots \\ p(x_n) \end{pmatrix}$$

Lagrange Formulation

- For discrete distributions

$$\min_{\gamma} \mathbf{D}^T \gamma$$

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

- Writing the Lagrangian

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

where α is a vector of Lagrange multipliers

- The solution to the discrete optimisation problem is given by

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

Dual Form

- We can rearrange

$$\begin{aligned}\mathcal{L}(\gamma, \alpha) &= D^\top \gamma - \alpha^\top (A\gamma - P) \\ &= P^\top \alpha - \gamma^\top (A^\top \alpha - D)\end{aligned}$$

- We note that $\gamma \geq 0$ so the dual problem is to find a vector α that maximises $P^\top \alpha$ subject to the constraints $A^\top \alpha \leq D$
- 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

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

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

- Rearranging

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

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

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

Continuous Form

- We can write a Lagrangian for the continuous problem

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

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

- Rearranging

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

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

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

Dual Form Constraint

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

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

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

$$\int \alpha(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} + \int \beta(\mathbf{y}) q(\mathbf{y}) d\mathbf{y} = \int \alpha(\mathbf{x}) (p(\mathbf{x}) - q(\mathbf{x})) d\mathbf{x} - \int q(\mathbf{x}) \epsilon(\mathbf{x}) d\mathbf{x} \blacksquare$$

- This is maximised when $\epsilon(\mathbf{x}) = 0$ ■ i.e. $\beta(\mathbf{x}) = -\alpha(\mathbf{x})$ ■

Dual Form

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

$$\int \alpha(\mathbf{x}) (p(\mathbf{x}) - q(\mathbf{x})) d\mathbf{x}$$

- Subject to the constraint

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

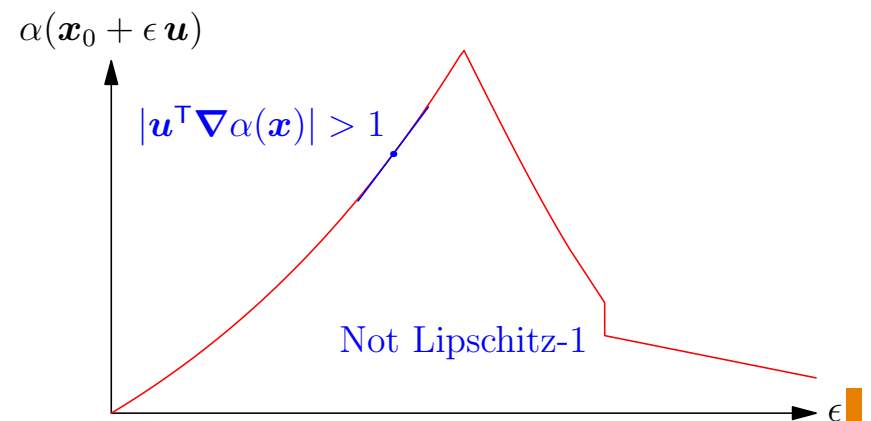
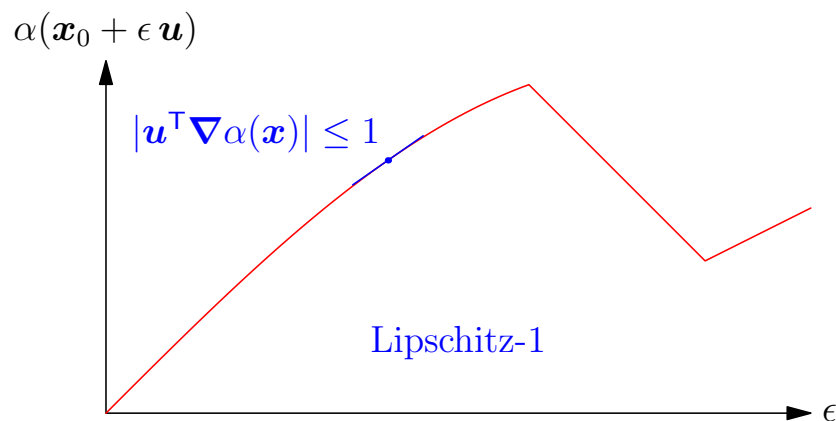
- This is a continuity constraint on the Lagrange multiplier function $\alpha(\mathbf{x})$ known as Lipschitz-1

Lipschitz-1 Functions

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

$$\mathbf{u}^\top \nabla \alpha(\mathbf{x}) = \lim_{\epsilon \rightarrow 0} \frac{\alpha(\mathbf{x}) - \alpha(\mathbf{x} + \epsilon \mathbf{u})}{\epsilon} \leq \frac{\epsilon}{\epsilon} = 1$$

- 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 $\|\nabla \alpha(\mathbf{x})\| \leq 1$ everywhere)



Calculating the Wasserstein Distance

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

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

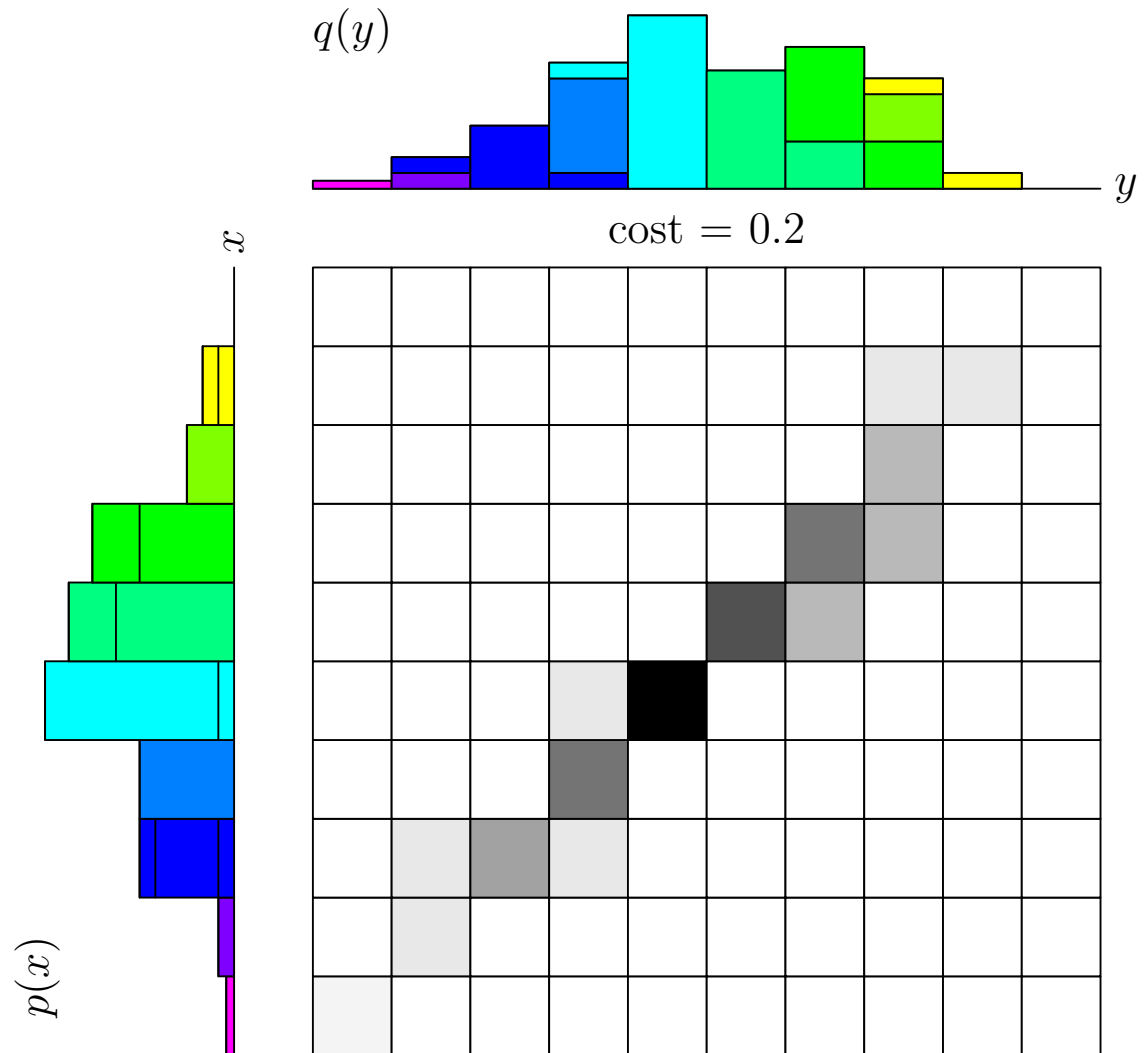
- For high dimensional objects $\gamma(\mathbf{x}, \mathbf{y})$ would be a huge object to approximate
- Instead we can compute the Wasserstein distance in the dual formulation

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

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

Outline

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



Back to GANs

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

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

Estimating Expectations

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

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

- where $\mathcal{B} \subset \mathcal{D}$ is a minibatch of true images and $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ \blacksquare
- From this we can choose \mathbf{w}_G to minimise

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

The Critic

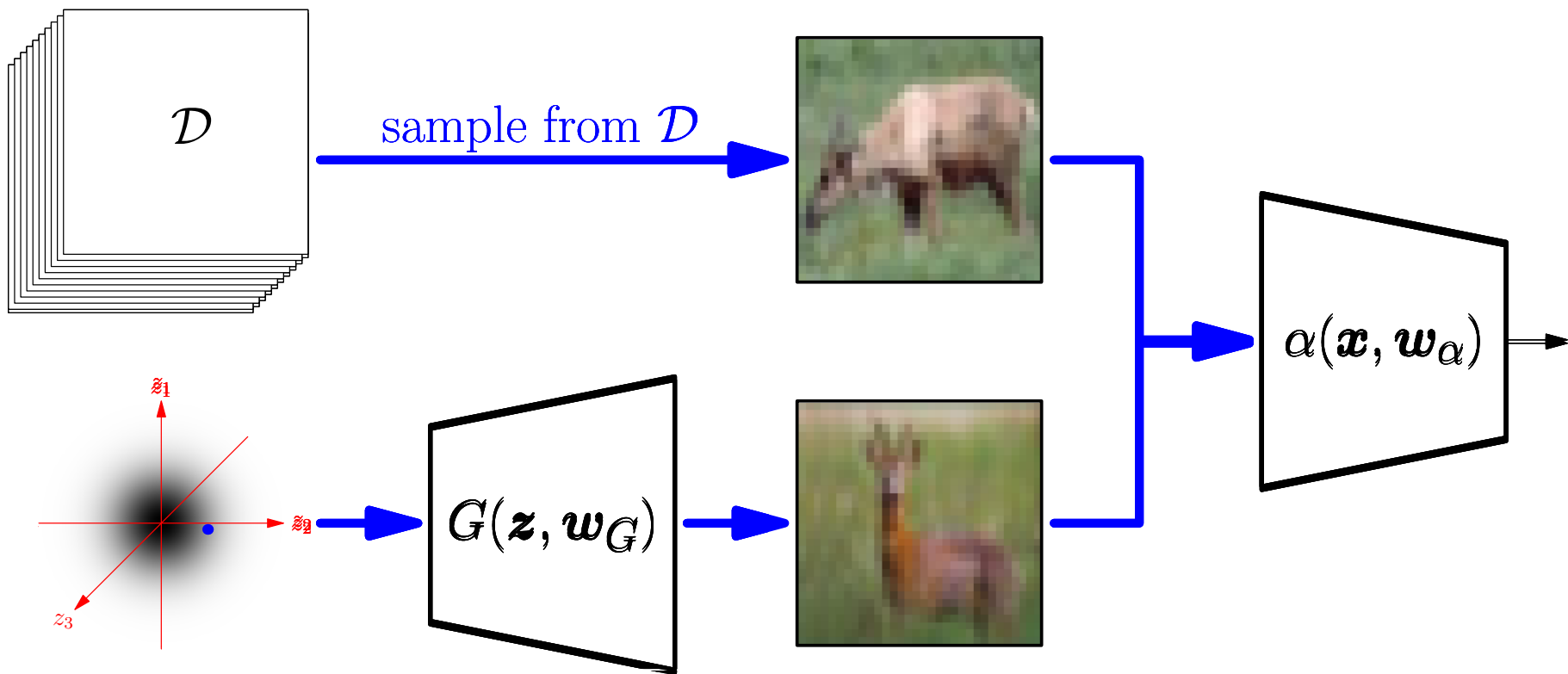
- For this quantity to approximate the Wasserstein distance we need to find a function $\alpha(\mathbf{x}, \mathbf{w}_\alpha)$ that maximises C ■
- To do this we learn a second network, the critic or discriminator whose job it is to maximise

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

- The network $\alpha(\mathbf{x}, \mathbf{w}_\alpha)$ should be Lipschitz-1 (which we usually botched by, for example, by setting the spectral norm of the convolutional weight matrix to 1)■

Wasserstein GANs

$$\max_{w_\alpha} \min_{w_G} \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \alpha(x, w_\alpha) - \frac{1}{n} \sum_{i=1}^n \alpha(G(z_i, w_G), w_\alpha)$$



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■