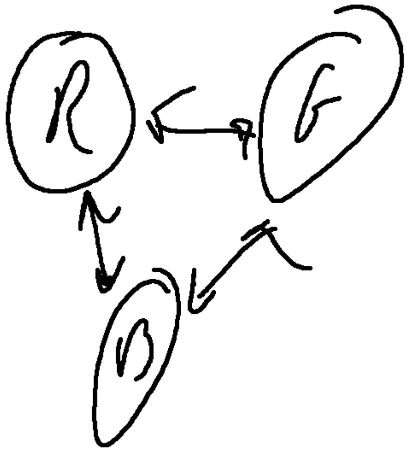


| | R | G | B |
|---|---|---|---|
| R | 0 | 1 | 1 |
| G | 1 | 0 | 1 |
| B | 1 | 1 | 0 |

1. calculate cartesian distances.
2. we ignore the hinge of self distance.
3. all distances are normalized

note: all x-components should be normalized as well, and ideally a greedy feature selection should be done to identify useless features of which there may be many. Or at least pull out the homogenous ones.

4. choose a falloff point where weights tend to zero. since the distances are normalized this value is ideally between 0 and 1 although in some cases a value higher than

1 may be chosen to allow further points to have a more significant impact on diffussion.

5. Select a falloff function of either


---

→ we also need a diffussion rate that works in a similar way to a residuals learning rate. whereby a lower rate is a slower more ...?. diffusion and a higher rate takes less steps to fully diffuse. Detailed perhaps?. Its a little like ray tracing, or physics updates you want small enough steps to be realistic, but there is diminishing returns at a certain point.

So far I think the best analogue is skittles on a plate of hot water as the colors slowly diffuse out and mix with each other. Though there is more of an exchange going on as in a graph diffusion.

---

We have significant advantage in the sense that we precompute all distances in the graph as well as the weights. This means that in the diffusion update we simply iterate over, and back buffer flip essentially. Making it quite parallel as well.

---

At prediction-time I think we do the same distance based falloff but we may need to fallback to knn with normalised distance weighting.

I actually think the residual add is subpar here. Each part is like a fountain of value production say we have:

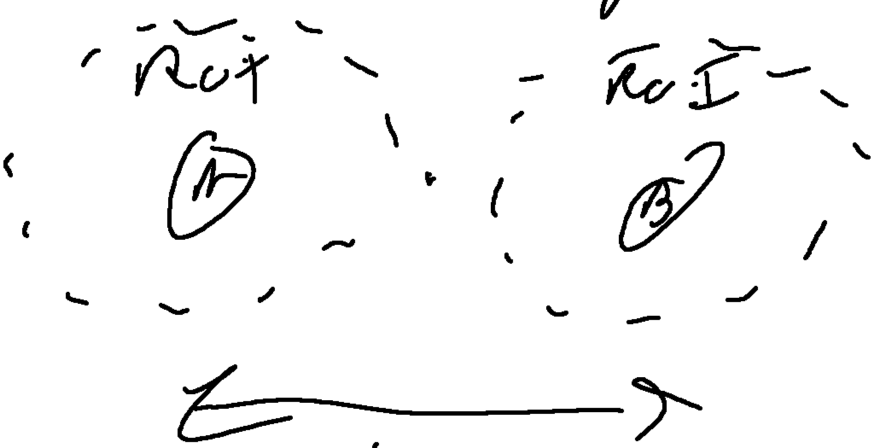
[0, 0, 0, 1, 0, 0]

As the basis for a points value this is averaged on based on a distance of '0' since it is directly on the part itself. And it competes with its previous value of say:

[0.2, 0.2, 0.2, 0.5, 0.1] etc.

Along with all the other parts in its region of influence.

in this way it may actually be beneficial to add a-productive points to the space to act as go betweens or diffusers. These points dont produce their own local values, but they allow value to diffuse through them, artificially increasing the density of the graph. In this way even if two points are too far to directly influence each other, A provides a bridge, almost like reconnecting a vanishing gradient.

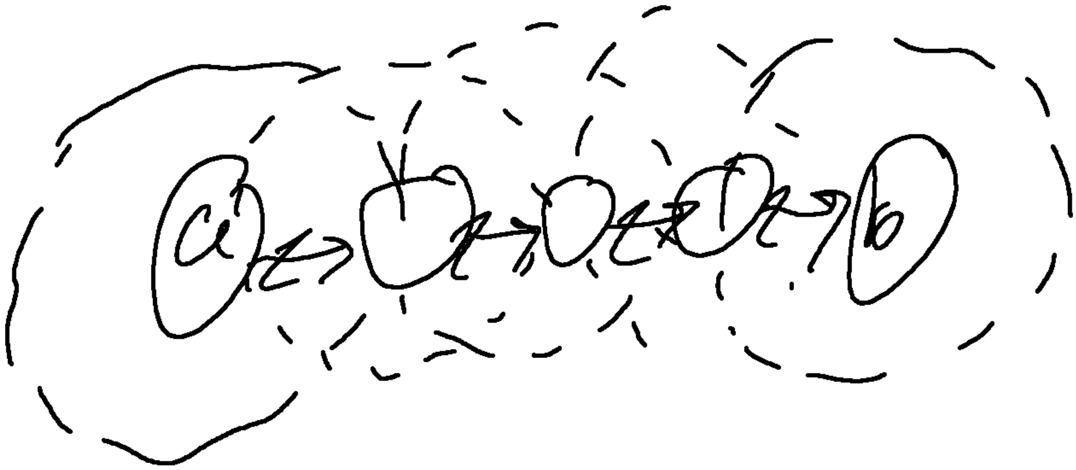


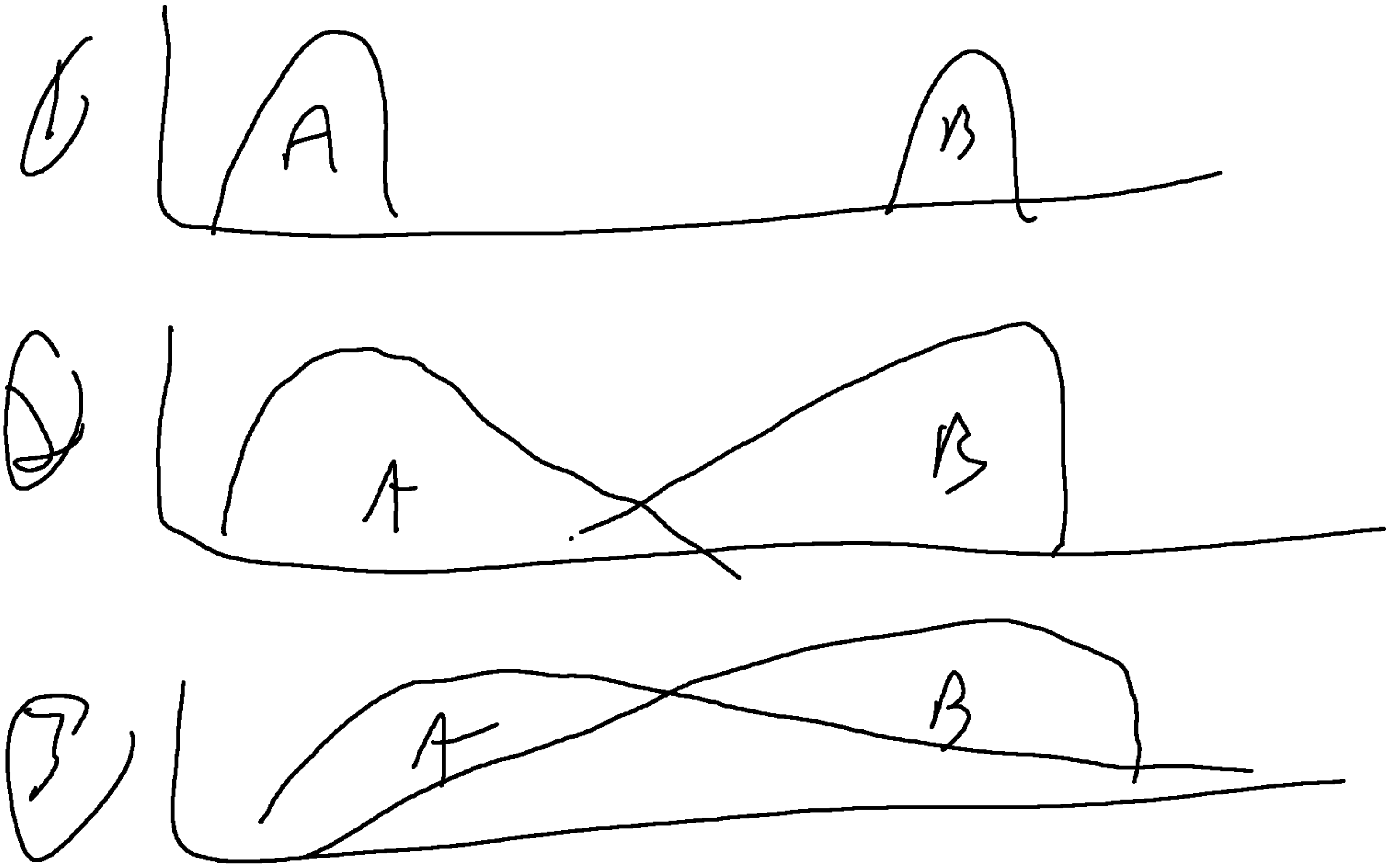

non influencing points

next page


These are non producing bridge parts that stop the diffusion gradient from vanishing into the void.
Allowing A ↔ B to affect one another over multiple steps.

next page →




without bridge parts we end at 1 but with them we diffuse properly.