Regularization strategies for Deep Learning

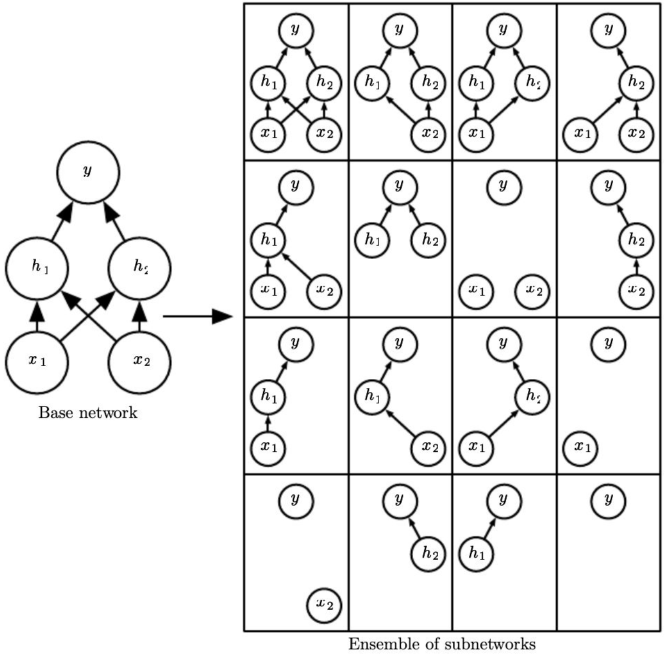
Dropout and Tangent Prop

N. Rich Nguyen, PhD
SYS 6016

6. Dropout

Dropout

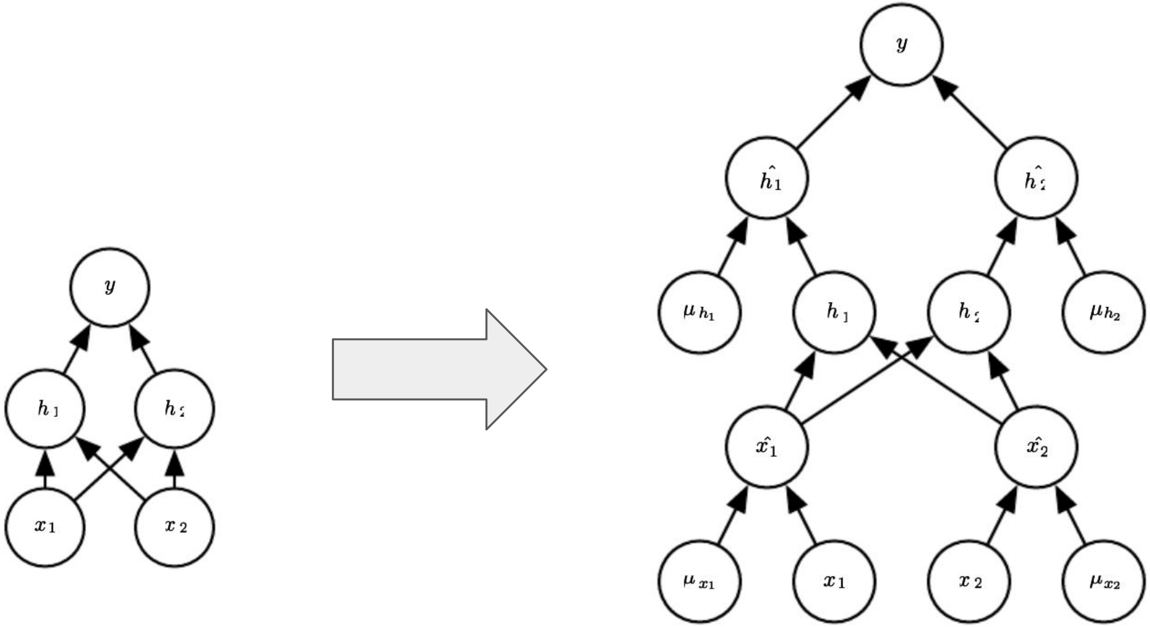
- Bagging involves training multiple models and evaluating them on each test examples → **impractical** when each model is a large neural network since it's costly in terms of runtime and memory.
- **Dropout** provides an inexpensive approximation to training and evaluating bagged ensemble of **exponentially many** neural networks.
- Dropout trains the ensemble consisting of all **sub-networks** that can be formed by removing non-output units from an underlying **base network.**



Justification for dropout

- At every training step, every neuron has a probability p of being temporarily “dropped out” (entirely ignore until the next step). The hyperparameter p is called the dropout rate (typically set at 50%) $(1-p)$ is the keep probability
- **Thought Experiment:** *Would a company perform better if its employees were told to **toss a coin every day** to decide **whether or not to go to work**?*
    - Company needs to adapt its organization: cannot rely on any single person
    - Employees would have to learn to cooperate with many coworkers
    - If one person quit, it wouldn’t make much a difference
- It’s unclear whether it works for **real** company, but it **works** for neural net
- It imposes no restriction on the architecture of the model or training methods
- Research has shown that dropout is **more effective** than other standard computationally inexpensive regularizers (weight decay or sparse constraints)

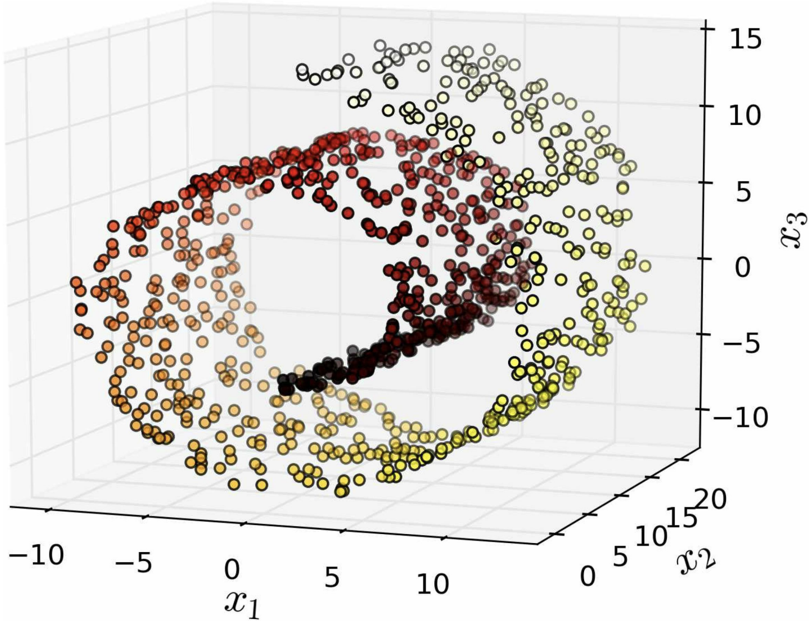
Forward propagation using dropout



- Randomly sample a vector μ with one entry for each input or hidden unit
- The entries of μ are binary and are sample independently from each other
- This is equivalent to **randomly selecting one** of the subnetworks

7. Tangent Prop

Review: What is a Manifold?



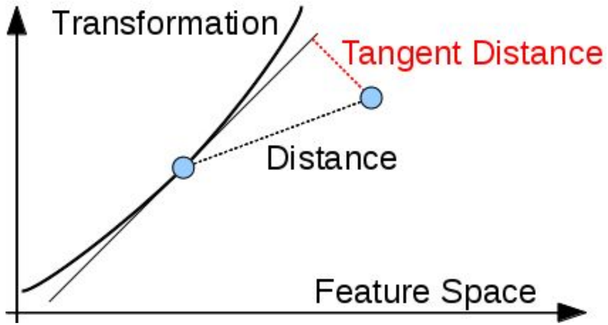
Many ML algorithms aim to overcome the **curse of dimensionality** by assuming that the data lies near a low-dimensional manifold.

A d-dimensional manifold is a part of an n-dimensional space (where `d < n`) that locally resembles a d-dimensional hyperplane.

The Swiss roll is an example of a 2D manifold where `d=2` and `n=3`: it locally resembles a 2D plane, but it is rolled in the 3rd dimension.

Real world examples of a manifold

Tangent Distance



- **Tangent distance** algorithm is a non-parametric *nearest neighbor algorithm* in which the *distance metric* is derived from knowledge of the *manifolds*.
- **Assumption:** Examples on the same manifold are from the same class.
- Since the classifier should be **invariant** to *local factors of variation* that correspond to movement of the manifold, it makes sense to use nearest neighbor distance between $\mathbf{x}1$ and $\mathbf{x}2$ the distance between manifolds $\mathbf{M}1$ and $\mathbf{M}2$ to which they respectively belong → this is *computationally difficult!*
- A cheaper alternative is to approximate manifold $\mathbf{M}i$ by **its tangent plane** at $\mathbf{x}i$ and measure the distance between the tangent plane and a point, which can be achieved by solving linear systems.

Tangent Prop

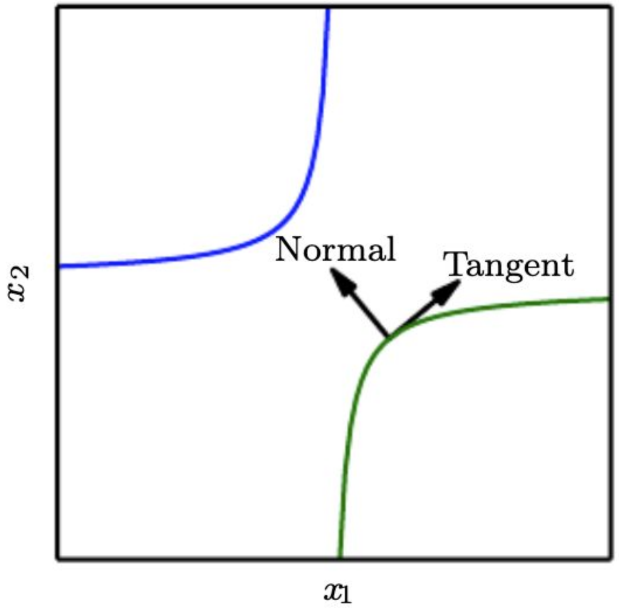
Tangent Prop algorithm trains a neural net classifier with an extra penalty to make each output $f(\mathbf{x})$ of the neural net **locally invariant** to known factors of variation.

Locally invariance is achieved by requiring $\nabla_{\mathbf{x}} f(\mathbf{x})$ to be **orthogonal** to the known manifold tangent vectors $\mathbf{v}^{(i)}$ at $\mathbf{x}$ by adding a regularization penalty:

$$\Omega(f) = \sum_i \left(\left(\nabla_{\mathbf{x}} f(\mathbf{x}) \right)^\top \mathbf{v}^{(i)} \right)^2$$

- Equivalently, directional derivative of f at $\mathbf{x}$ in direction of $\mathbf{v}^{(i)}$ should be small
- For most neural nets, it will need to be sum over many outputs
- The tangent vector $\mathbf{v}^{(i)}$ is derived as **a priori** from formal knowledge of the effect of transformations (ie. translation, rotations, scaling in images)
- Tangent prop has been used not just for *supervised learning*, but also *reinforcement learning*.

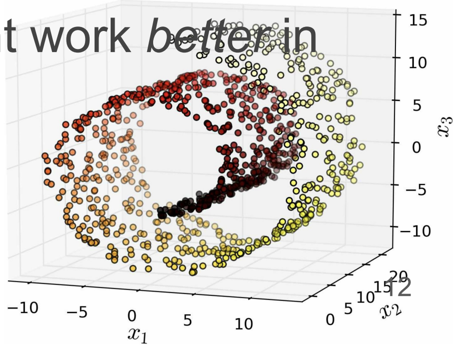
Illustration of Tangent Prop algorithm



- Each blue and green curve represents a manifold of different class (1-D manifold in 2-D space)
- We expect the classification function to change rapidly as it moves in the direction of the **Normal** vector, but not to change as much as it moves in the direction of the **Tangent** vector (along the manifold).
- Tangent prop regularizes `f(x)` to not change much as **x** moves along the manifold.

Relation to dataset augmentation

- Closely related as the user encodes their knowledge of the task by **specifying a set of transformations** that should not alter the network output
- The difference is that in dataset augmentation, the network is trained to correctly classify **distinct inputs** created by applying some transformations.
- Tangent prop does not require explicitly visiting of new input point. Instead, it regularizes the model to **resist perturbation** in the directions corresponding to the specified transformation.
- Augmentation has resistance to larger perturbation than tangent prop.
- Tangent prop is *intellectually elegant*, but augmentation might work *better* in practice



Summary: Regularization strategies

We have covered most of the strategies used to regularize neural networks:

1. Parameter Norms
2. Dataset Augmentation
3. Multi-task Learning
4. Early Stopping
5. Bagging
6. Dropout
7. Tangent Prop

Neural nets with regularization achieve **better** performance on **unobserved** data.

Regularization is a **central theme** of machine learning, and we will **revisit** them periodically in the course.