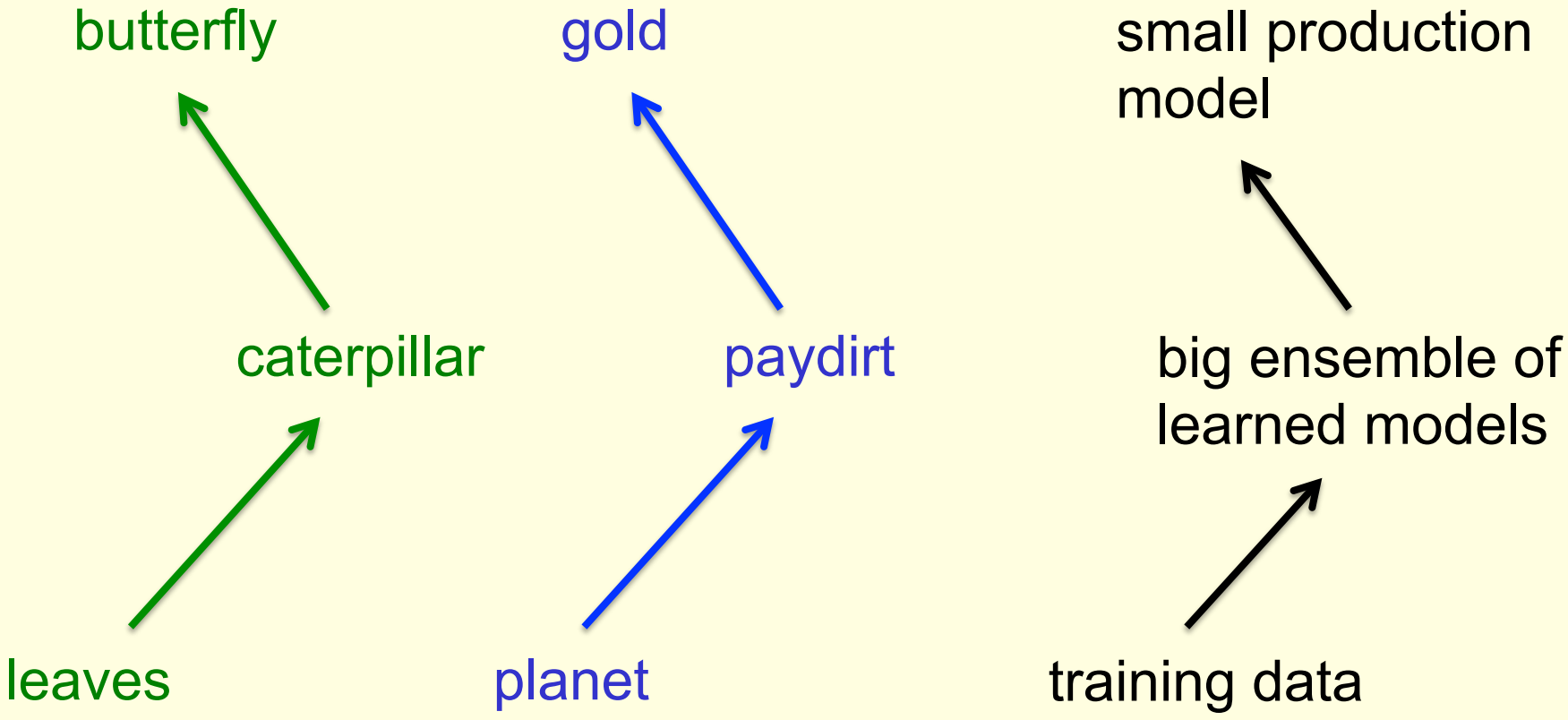
Dark knowledge

“Geoffrey Hinton, Oriol Vinyals & Jeff Dean)

Google

An analogy


butterfly
gold
small production model
caterpillar
paydirt
big ensemble of learned models
leaves
planet
training data

The conflicting constraints of learning and using

- The easiest way to extract a lot of knowledge from the training data is to learn many different models in parallel.
  - We want to make the models as different as possible to minimize the correlations between their errors.
  - We can use different initializations or different architectures or different subsets of the training data.
  - It is helpful to over-fit the individual models.
- A test time we average the predictions of all the models or of a selected subset of good models that make different errors.
  - That's how almost all ML competitions are won (e.g. Netflix)

Why ensembles are bad at test time

- A big ensemble is highly redundant. It has a very very little knowledge per parameter.

- At test time we want to minimize the amount of computation and the memory footprint.
  - These constraints are much more severe at test time than during training.

The main idea (Caruana et. al. ☹)

- The ensemble implements a function from input to output. Forget the models in the ensemble and the way they are parameterized and focus on the function.
  - After learning the ensemble, we have our hands on the function.
  - Can we transfer the knowledge in the function into a single smaller model?

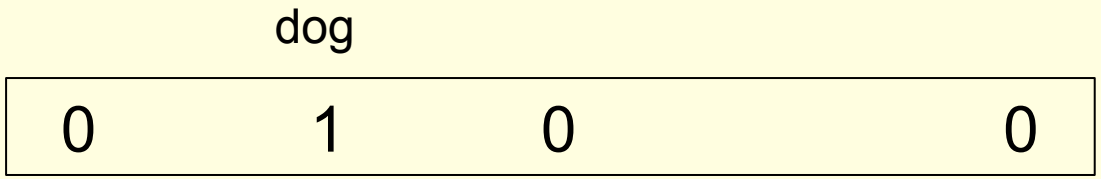
A way to transfer the function

- If the output is a big N-way softmax, the targets are usually a single 1 and a whole lot of 0's.
  - On average each target puts at most log N bits of constraint on the function.
- If we have the ensemble, we can divide the averaged logits from the ensemble by a "temperature" to get a much softer distribution.

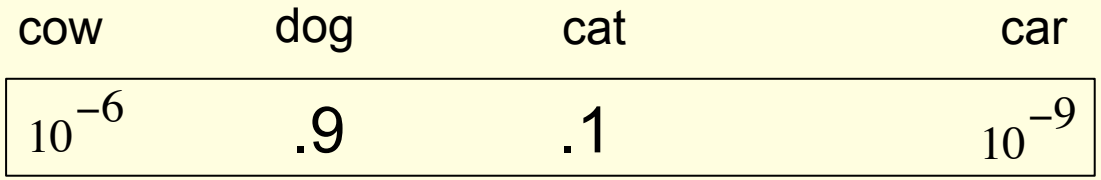
$$p_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

This conveys much more information about the function per training case.

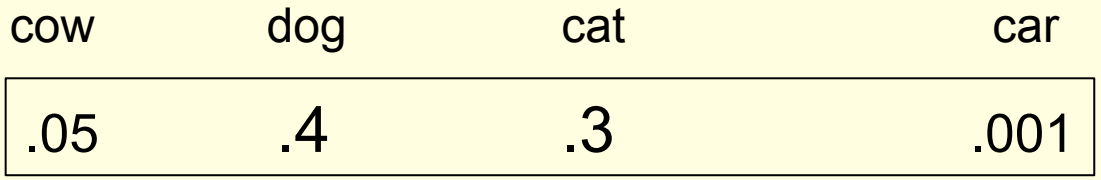
An example of hard and soft targets



original targets



output of ensemble



softened output of ensemble

Softened outputs reveal the dark knowledge in the ensemble.

Adding in the true targets

- If we just train the final model on the soft targets from the ensemble, we do quite well.
- We learn fast because each training case conveys much more information than with a single hard target.
- But its better to fit both the hard targets and the soft targets from the ensemble.

Adding hard targets

- We try to learn logits that minimize the sum of two different cross entropies.

- Using a high temperature in the softmax, we minimize the cross entropy with the soft targets derived from the ensemble.

- Using the very same logits at a temperature of 1, we minimize the cross entropy with the hard targets from the original training set.

Experiment on MNIST

- Vanilla backprop in a 784->500->300->10 net gives about 160 test errors

- If we train a 784->1200->1200->10 net using dropout and weight constraints and jittering the input, we eventually get 67 errors.

- Using both hard and soft targets (no dropout or jitter) we can get 74 errors in a 784->500->300->10 net.

A surprising result on MNIST

- Train the 784->500->300->10 net on a transfer set that does not contain any examples of a 3. After this training, raise the bias of the 3 by the right amount.
  - The distilled net then gets 98.6% of the test threes correct even though it never saw any.
- If we train the distilled net with only images of 7 and 8, and then lower the biases of 7 and 8 by the right amount, it gets 87% correct over all classes.

Some results on Speech

- Start with a trained model that classifies 58.9% of the test frames correctly.
- Use that model to provide soft targets for a new model (that also sees hard targets).
  - The new model gets 57.0% correct even when it is only trained on 3% of the data.
  - Without the soft targets it only gets to 44.5% correct and then gets much worse.
- Soft targets area a VERY good regularizer.
  - They prevent the model from being too sure.

Improving a speech model

- Train 10 models separately. They average 58.9% correct.
  - The ensemble gets 61.1%

- Now distill the ensemble into a single model of the same size using both hard and soft targets.
  - The distilled model gets 60.8% correct

ImageNet results (work in progress)

- A single “Alex-net” model gets 60.5% top-1 correct.
- An ensemble of 10 models gets 65.1%
- Distilling the ensemble into a single model gets 62%
  - We currently dont know why its not better.

The high temperature limit

$$e^{\varepsilon} \approx 1+\varepsilon \quad \text{if } \varepsilon \text{ is small}$$

logits for soft targets

$$T\frac{\partial C}{\partial z_i} = p_i - t_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)} - \frac{\exp\left(\frac{v_i}{T}\right)}{\sum_j \exp\left(\frac{v_j}{T}\right)}$$

$$T\frac{\partial C}{\partial z_i} \approx \frac{1+\frac{z_i}{T}}{N+\sum_j \frac{z_j}{T}} - \frac{1+\frac{v_i}{T}}{N+\sum_j \frac{v_j}{T}} = \frac{1}{NT}\left(z_i - v_i\right)$$

assume we have zero-meaned both sets of logits for every case

How to make the ensemble mine knowledge efficiently

- We can encourage different members of the ensemble to focus on resolving different confusions.
  - In ImageNet, one “specialist” net could see examples that are enriched in mushrooms.
  - Another specialist net could see examples enriched in sports cars.
- We can choose the confusable classes in several ways.
- The problem is that the specialists tend to overfit.

Two ways to prevent specialists over-fitting

- Each specialist gets data that is very enriched in its particular subset of classes but its softmax covers all classes.
  - Matching the community logits on general data will prevent overfitting.
- Each specialist has a reduced softmax that has one dustbin class for all the classes it does not specialize in.
  - Initalize the specialist with the weights of a full model and use early stopping to prevent over-fitting.

Early stopping specialists on JFT

- Start from JFT model that gets 25% top-1 correct.

#spec	#cases	#win	relative error
0	350037	0	0.0%
1	141993	+1421	+3.4%
2	67161	+1572	+7.4%
3	38801	+1124	+8.8%
4	26298	+835	+10.5%
5	16474	+561	+11.1%
6	10682	+362	+11.3%
7	7376	+232	+12.8%
8	4703	+182	+13.6%
9	4706	+208	+16.6%
10+	9082	+324	+14.1%

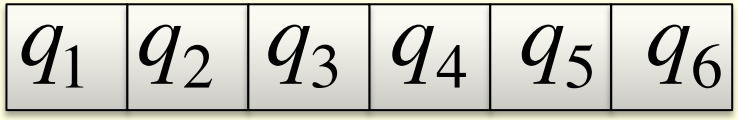
Combining models that have dustbin classes

- Its not trivial. A specialist is NOT claiming that everything in its dustbin class is equally probable. Its making a claim about the sum of those probabilities.
- Basic idea: For each test case, we iteratively revise the logits for the detailed classes to try to agree with all of the specialists.
  - i.e. We try to make the sum of the relevant detailed probabilities match the dustbin probability.

A picture of how to combine models that each have a dustbin class

- For each test or transfer case we run a fast iterative loop to find the set of logits that fit best with the partial distributions produced by the specialists.

target probs from specialist



actual probs of combination