Data Mining & Machine Learning

CS37300
Purdue University

November 1, 2017

Model Search Tricks

Midsemester Course Evaluation (+1% points)

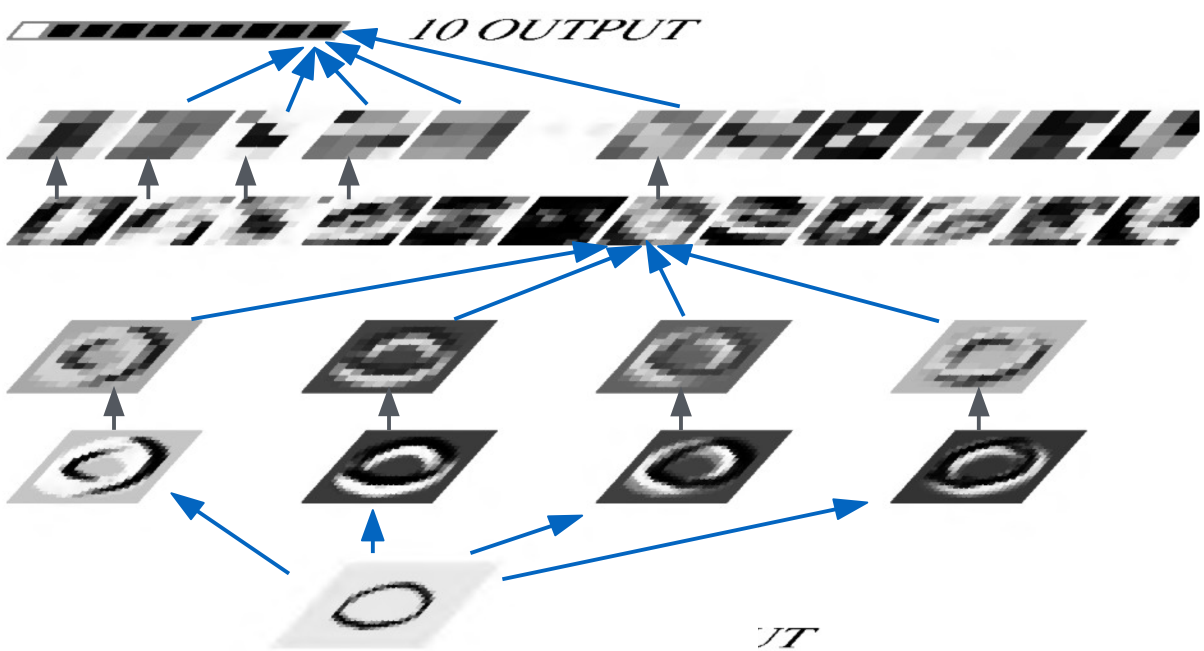
Instructions on piazza

Outline

- Regularization
  - L2 regularization
  - Dropout
- Other Tricks

Neural Networks: Parameters x Training data sizes

- Consider a simple convolutional neural network for handwritten black/white digit classification:
  - The training data has only 50,000 images (and 50,000 labels).
  - Network has 1.2M parameters
  - The network likely will overfit the training data (not necessarily a problem, but could be)

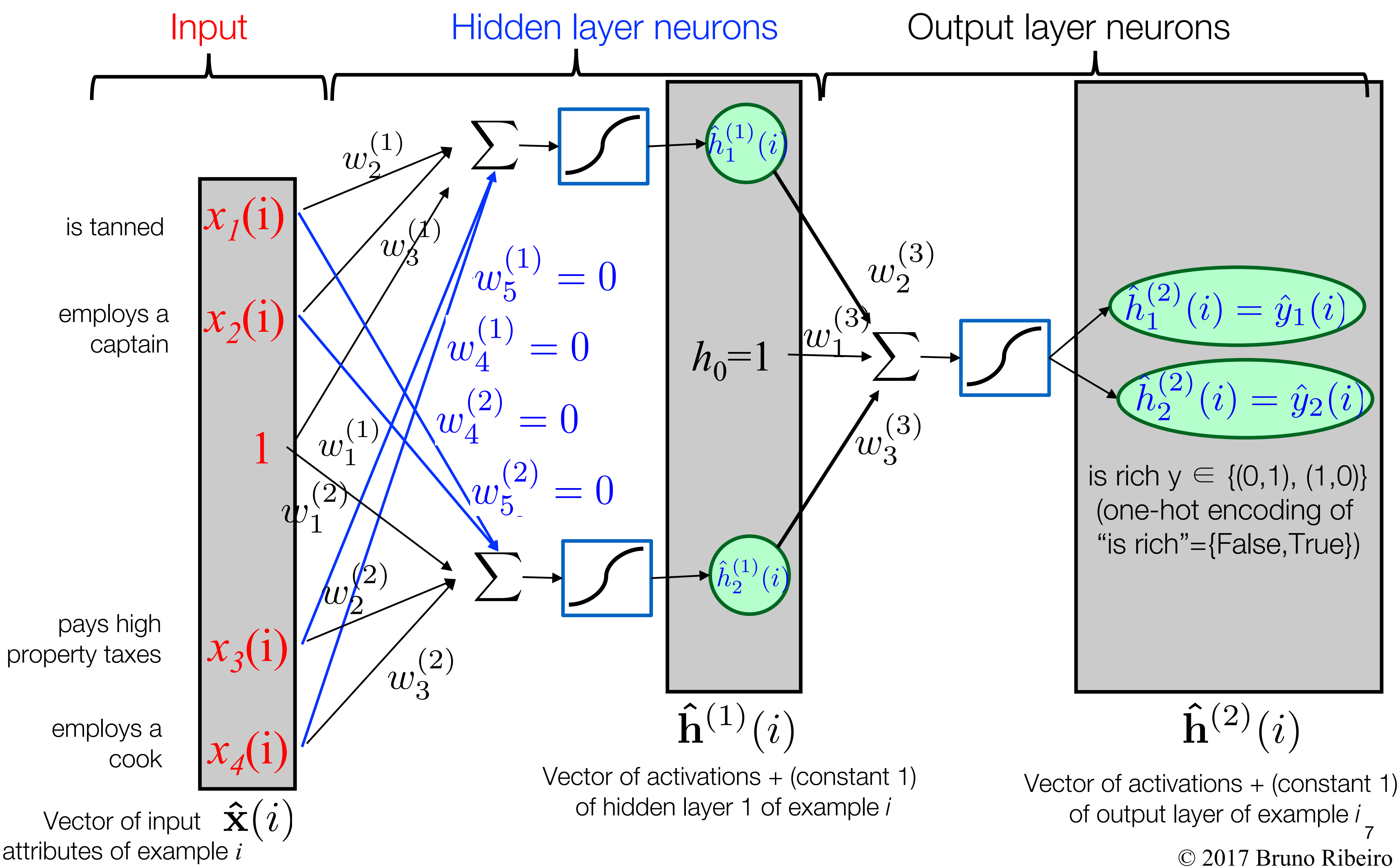


Regularization

- There are many ways to avoid overfitting *too much*
    - Deep neural networks generally will overfit

- The idea behind regularization is to constraint either in the **model space** or the **model search**
    - In model space: E.g., forcing some parameters to be zero constrains the model
    - In model search: E.g., penalizing weights with large values constrains the search, and thus, the model

Constraining the **Model Space** (Illustration)



Constraining the **Model Search** (illustration)

- Constraining the model search:
  - In the previous example we pre-defined specific weights to be zero (constraining the model)
  - We could instead force 4 weights to be zero, but letting search algorithm decide which weights will be zero
  - This can improve the training accuracy of the model. Why?
    - If the the 4 pre-defined weights give better models, then the a good search algorithm will decide that these 4 weights give a better model
  - Asking for exactly 4 weights = zero is undesirable
    - Model should have some extra flexibility in deciding what is best
    - Answer: Penalties, penalize large weights (or the number of non-zero weights)

Constraining Model Search in Practice

Let $f(x; \mathbf{w})$ be the model the classifier that outputs the label of x given parameters $\mathbf{w}$. The training examples are $\{x(i), y(i)\}_{i=1}^{n}$, where $x(i)$ are the attributes of observation i and $y(i)$ its label. Suppose that $S(y, y')$ is a score function between label y and predicted probability y': better models have larger values (e.g., likelihood function). The model learning can be described as

$$\arg\max_{\mathbf{w}} \frac{1}{n} \sum_i S(y(i), f(x(i); \mathbf{w})) - \lambda \sum_j \sum_k (\mathbf{w}_j^{(k)})^2,$$

a.k.a. L_2 norm
penalizes large weights

The parameter λ is known as the regularization strength.

Other alternative penalties include:

$$L_1 \text{ norm } = -\sum_j \sum_k |\mathbf{w}_j^{(k)}|$$ Penalizes the absolute weight values

$$L_0 \text{ norm } = -\sum_j \sum_k \mathbf{1}\{\mathbf{w}_j^{(k)}| > 0\}$$ Penalizes the number of non-zero weights

Regularization Strength

- Parameter λ is the regularization strength
  - How much we are penalizing the model?

$$\arg\max_{\mathbf{w}} \frac{1}{n} \sum_i S(y(i), f(x(i); \mathbf{w})) - \lambda \sum_j \sum_k (\mathbf{w}_j^{(k)})^2$$

Q: What happens if $\lambda \to \infty$?

What happens if $\lambda \to 0$?

A:
For $\lambda \to \infty$, the penalty for non-zero weights it too large, weights will be zero. The model is too constrained.

For $\lambda \to 0$, there is no penalty. The model is unconstrained.

Dropout (Model Space constraint)

Averaging Many Models

- To win a machine learning competition (e.g. Netflix) you often need to combine many different types of strong models (complex models) and combine them to make predictions

- For instance, decision trees are not very powerful models, but

    - Averaging many decision trees works very well with some random model constraints. This is called random forest.

    - Random model constraints include only being able to split each decision tree node using a random subset of the attributes

        - This is a random constraint in the model space

Two ways to average models

- Mixture: We combine the models by taking the arithmetic mean of their output probabilities (probability of a class).

Model A:	.3	.2	.5
Model B:	.1	.8	.1
Combined:	.2	.5	.3

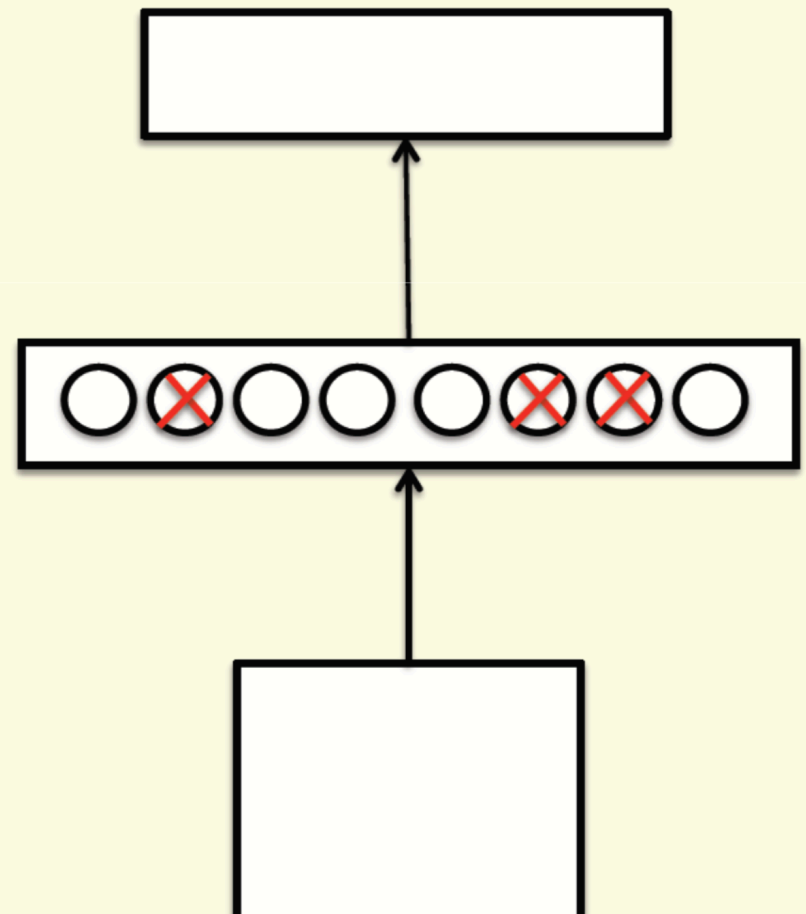
- Product: Combine the models by taking the geometric mean of their probability.

Model A:	.3	.2	.5
Model B:	.1	.8	.1
Combined:	$\sqrt{.3}$	$\sqrt{1.6}$	$\sqrt{.5}$

(need to renormalize to get a probability again)

Dropout: An efficient way to average many large neural nets.

- Consider a neural net with one hidden layer.
- Each time we present a training example, we randomly omit each hidden unit with probability 0.5.
- So we are randomly sampling from 2^H different architectures.
  - All architectures share weights.



Performed in training, validation, and testing

Hinton

Dropout as a form of model averaging

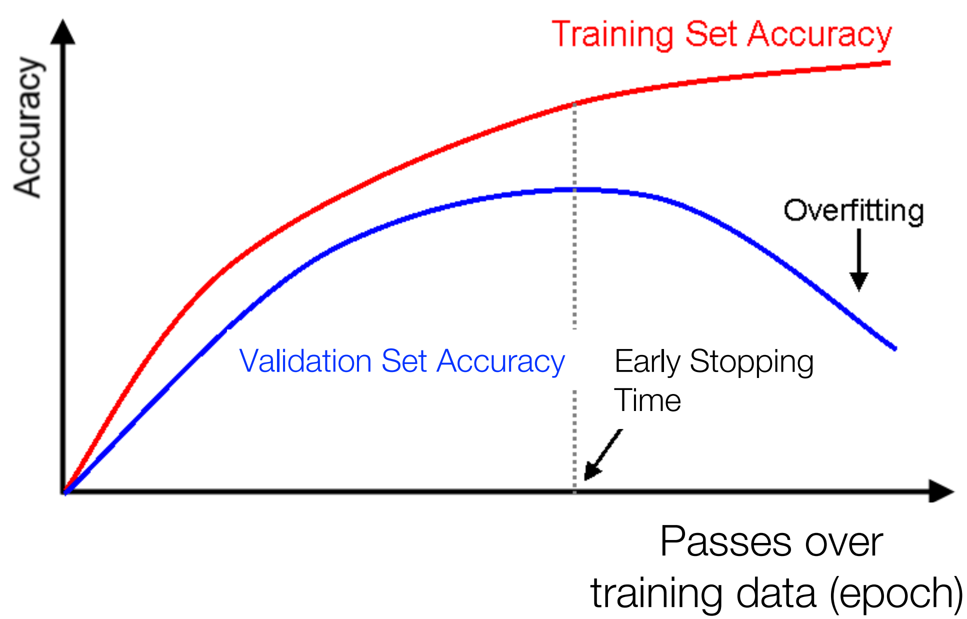
- We sample from 2^H models. So only a few of the models ever get trained, and they only get one training example.
  - This is as extreme as bagging can get.
- The sharing of the weights means that every model is very strongly regularized.
  - It's a much better regularizer than L2 or L1 penalties that pull the weights towards zero.

- The neural network classifier output is the average over multiple different random dropouts

Hinton

Other tricks 1

- Early stopping:
  - Monitor the model accuracy in a separate validation dataset
  - As we are performing gradient ascent, if the model accuracy first increases but then starts to drop in the validation data, we stop training the model…
    - The output is the model parameter with the best accuracy in the validation data



Other tricks 2

- Momentum
    - We will see it in HW 5