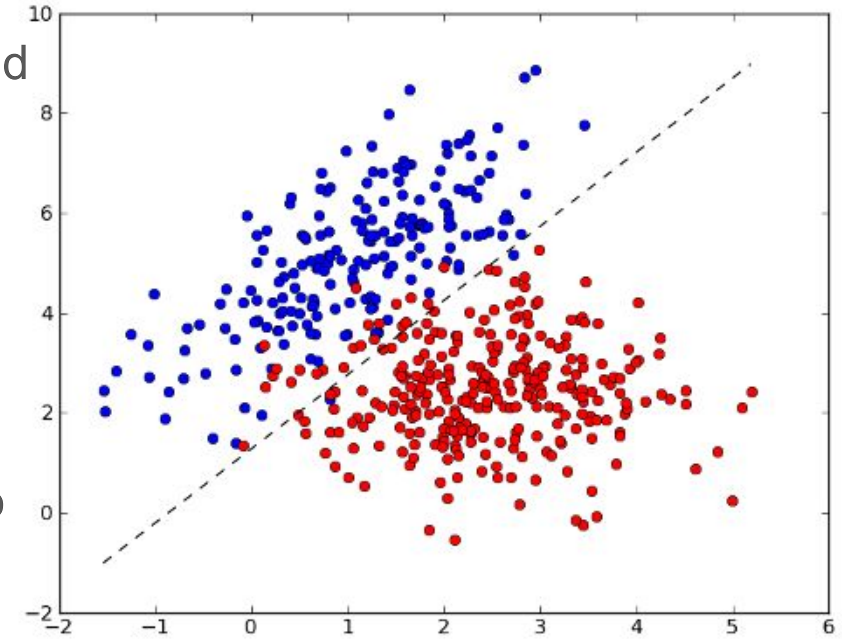
CIS 419/519 Recitation

23 September 2020

Linear Classifiers

When should you use linear classifiers, and when should you use a more expressive class of functions?

The bigger your data set, the more parameters (of a function) you can learn from it. Linear classifiers have fewer parameters than most function classes, so you can learn a linear classifier with relatively little data.



source: http://mlpy.sourceforge.net/docs/3.5/lin_class.html

Loss Functions

Used to evaluate performance of models and guide training

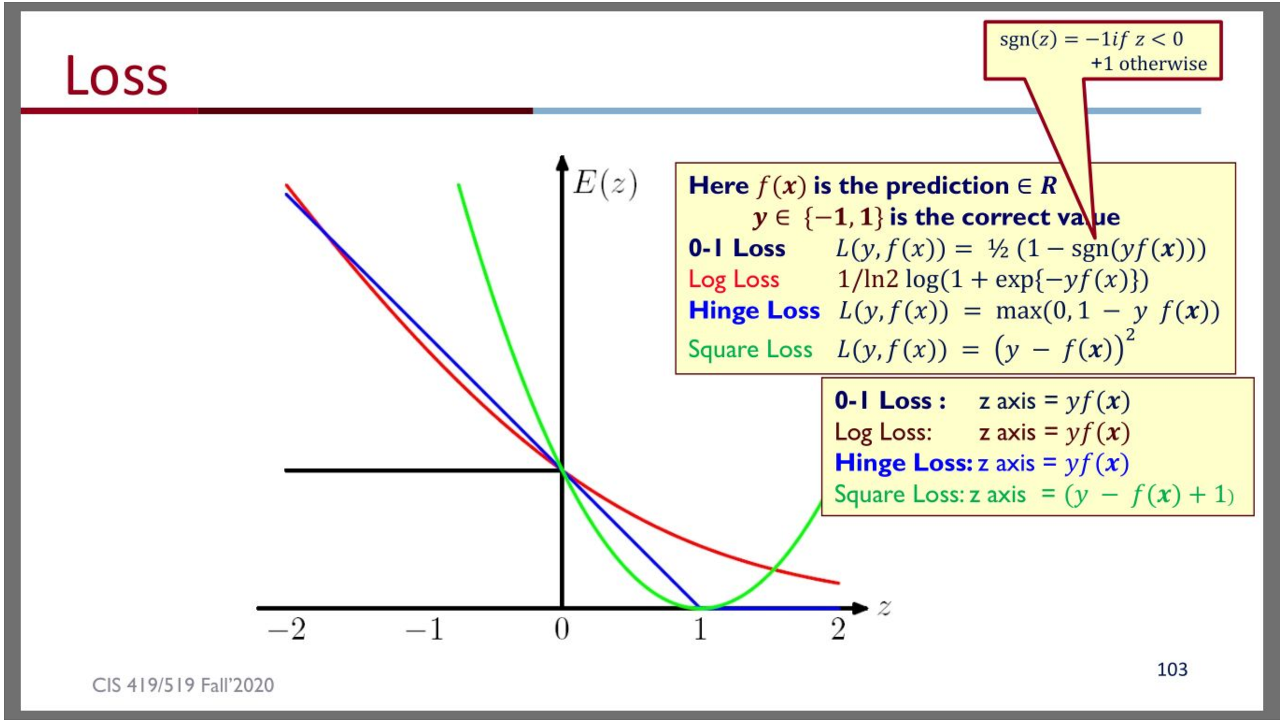
0-1 loss: 0 if model gives correct label, 1 if model gives incorrect label

We usually care about the 0-1 loss, but minimizing 0-1 loss is computationally hard, so we use surrogate losses instead.

Loss functions: examples

Label space is {-1, +1}

Functions in hypothesis space are of the form sgn(f(x)), where f is a real-valued function



From lecture slides

Gradient Descent

Procedure that finds a local minimum of a differentiable function: at each iteration, computing the gradient at the current guess and update the guess accordingly

In our case, we want to minimize average loss over a training set (our “values” are weight vectors taken from a parameter space).

Stochastic gradient descent: instead of computing the loss over the entire training set, we instead use a small subset of the training set. Each iteration of stochastic gradient descent makes less progress (or possibly no progress) toward a local minimum, but we can do more iterations with the same training time.

Example: Gradient descent using LMS loss

Recall from lecture that LMS loss is defined as $\frac{1}{2} \Sigma_d(t_d - o_d)^2$

The update rule is $\mathbf{w}' = \mathbf{w} + R\Sigma_d(t_d - o_d)\mathbf{x}_d$

In the degenerate case where there is one training example, it is easy to see that this update rule reduces the square loss

$o' = \mathbf{w}'^T\mathbf{x} = (\mathbf{w} + R(t - o)\mathbf{x})^T\mathbf{x} = \mathbf{w}^T\mathbf{x} + R(t - o)\|\mathbf{x}\|^2$

If the original prediction is too small, then the update rule makes the new prediction larger, and vice versa

Decision Trees

Sunny	Snowy	**Play Outside?**
Y	Y	T
N	Y	F
Y	Y	T
N	N	F
Y	N	T
N	Y	F
N	N	T
Y	N	F

Entropy(S) = 1

$$\text{Entropy}(S_{\text{Snowy}}) = \frac{4}{8}\left(-\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right)\right) + \frac{4}{8}\left(-\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right)\right)$$

$$= 1$$

$$\text{Entropy}(S_{\text{Sunny}}) = \frac{4}{8}\left(-\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right) + \frac{4}{8}\left(-\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right)$$

$$= 0.811$$

Decision Trees

Sunny	Snowy	**Play Outside?**
Y	Y	T
[redacted]	[redacted]	[redacted]
Y	Y	T
[redacted]	[redacted]	[redacted]
Y	N	T
[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]
Y	N	F

$$\text{Entropy}(S_{\text{Sunny = Yes,}}\ S_{\text{Snowy}}) = \frac{2}{4}\left(-\frac{2}{2}\log_2\left(\frac{2}{2}\right) - 0\right) + \frac{2}{4}\left(-\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right)$$

$$= 0.5$$

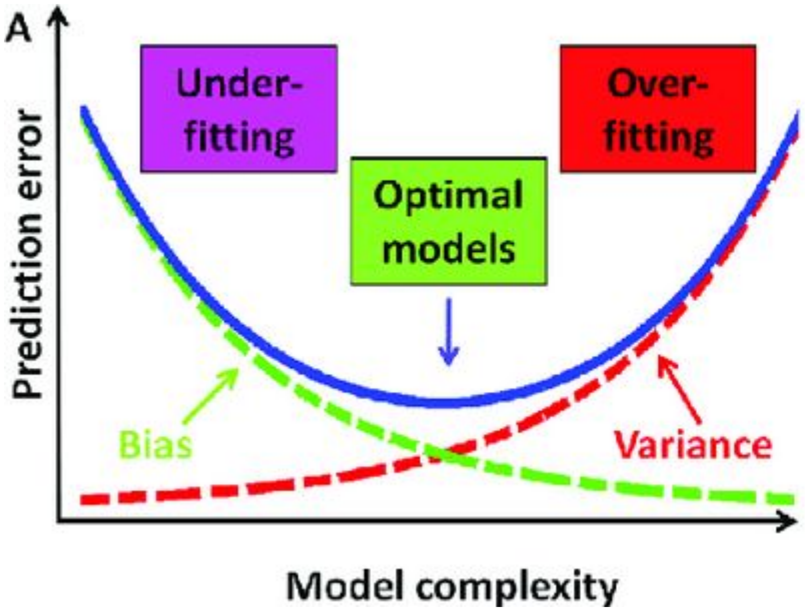
Bias/Variance Tradeoff

- Bias: How likely is the model to learn the target function?
    - High bias: The model is able to fully approximate target function
    - Low bias: The model is not able to express the function

- Variance: How affected is the model by changes in training data?
    - Low variance: Slight changes in training data does not affect model
    - High variance: Slight changes in training data affects model heavily

- Bias and variance have an inverse relationship, we want to balance them

Bias/Variance Tradeoff



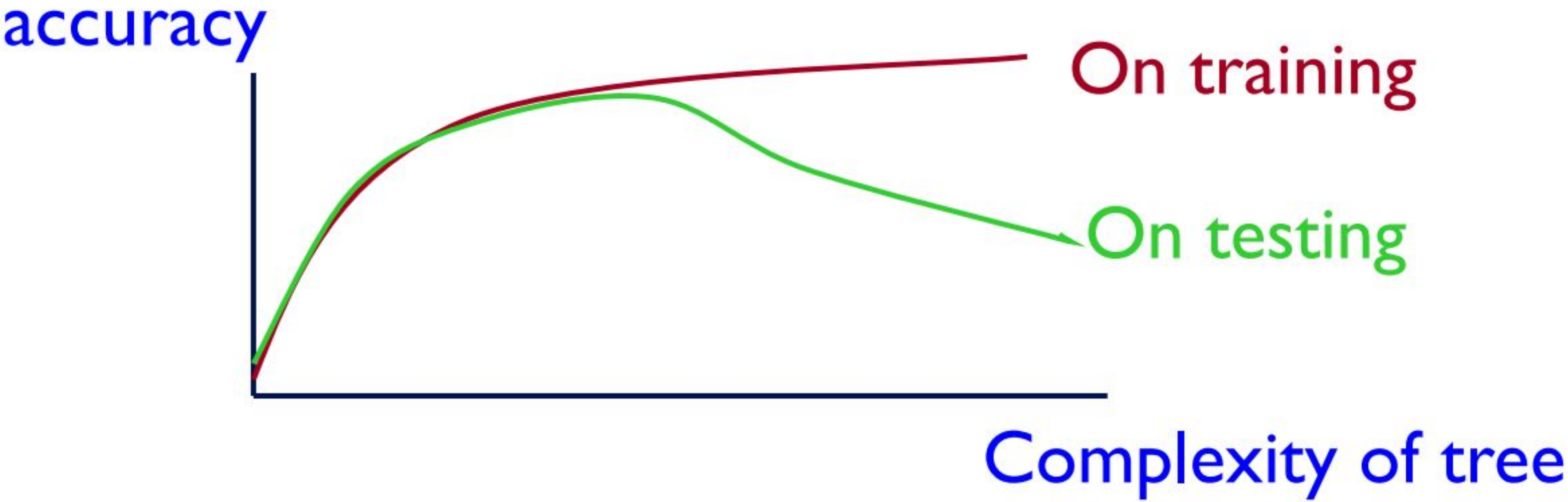
Overfitting

- The causes of overfitting are twofold
    - Overly complex model
    - Not enough data
    - These lead you to fitting noise in the training data

- Bad news: overfitting is something you will encounter often

- Good news: It is easy to detect/fix

Detecting Overfitting



Fixing Overfitting

- Reduce complexity of model
  - Complexity = #parameters
  - Eg. Depth of decision tree

- Don’t train too long
  - Going over training data multiple times can increase chance of overfitting

- Try obtain more training data
  - Obtain new data from similar sources
  - Use more data for training and less for validation/testing

Overfitting Illustration

- Black line represents true boundary between classes

- Data is noisy, certain points with incorrect/unexpected labels

- Green line represents overfitting model

- Model fits the noise in the training data, learning incorrect decision boundary

- This leads to errors when predicting test labels

- Model does not generalize well