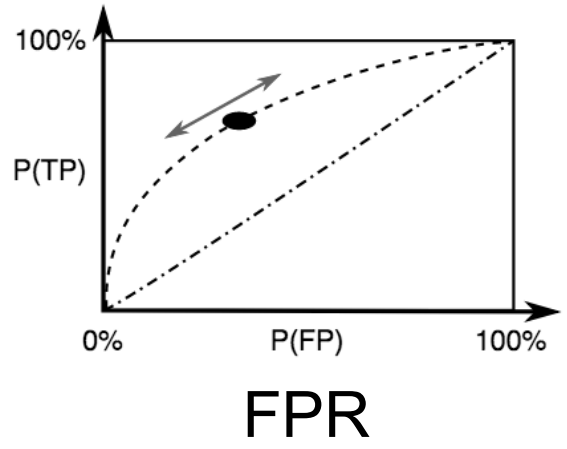
# Tufts COMP 135: Introduction to Machine Learning

https://www.cs.tufts.edu/comp/135/2020f/

# Evaluating Binary Classifiers

| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | TN | FP |
| | Y=1 | FN | TP |





Prof. Mike Hughes

*Many slides attributable to:*
*Erik Sudderth (UCI)*
*Finale Doshi-Velez (Harvard)*
*James, Witten, Hastie, Tibshirani (ISL/ESL books)*

# Today's objectives (day 08)
# Evaluating Binary Classifiers

1) Evaluate binary decisions at specific threshold

    accuracy, TPR, TNR, PPV, NPV, ...

2) Evaluate across range of thresholds

    ROC curve, Precision-Recall curve

3) Evaluate probabilities / scores directly

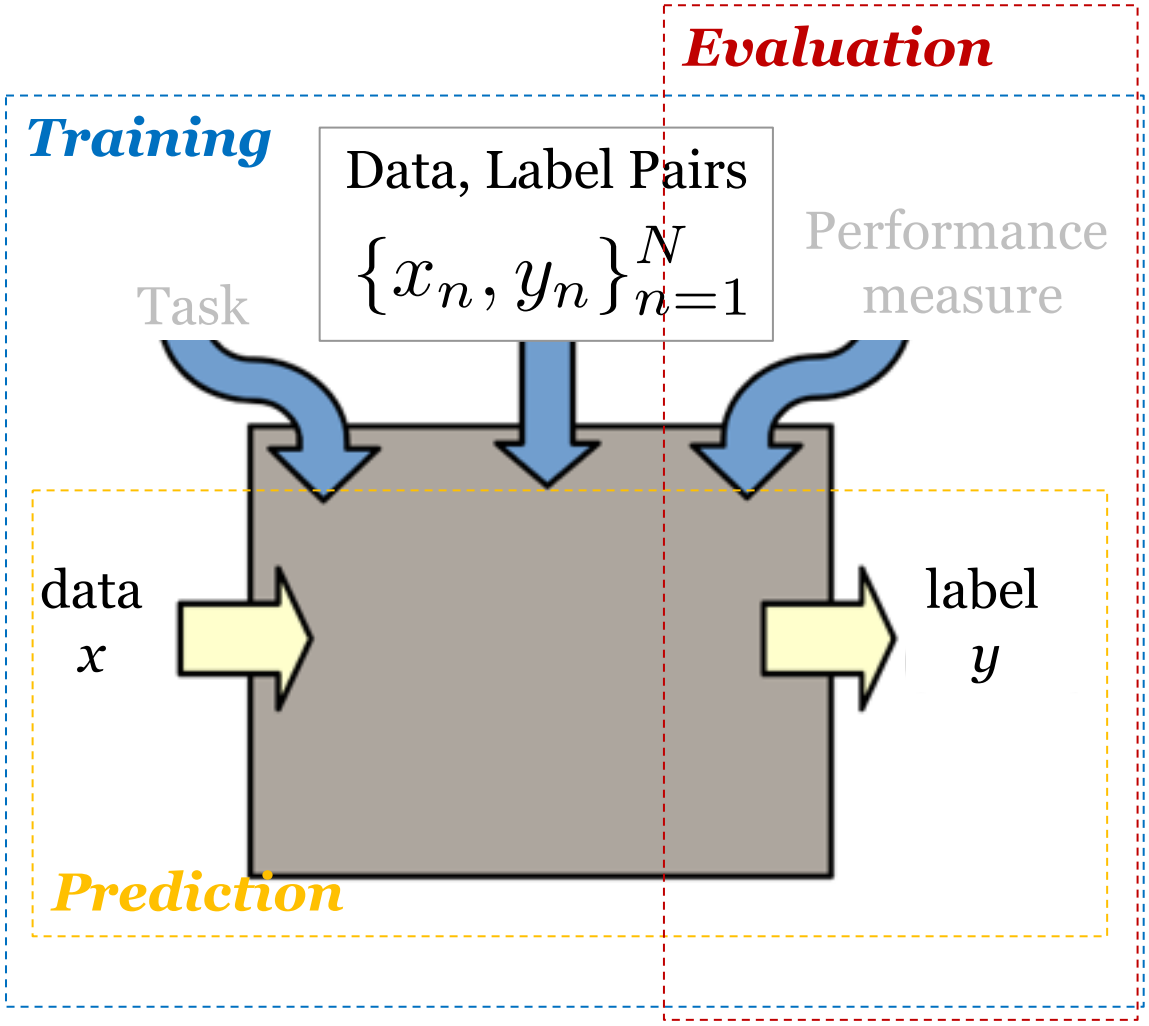
    cross entropy loss (aka log loss)

# What will we learn?

Supervised Learning

Unsupervised Learning

Reinforcement Learning



# Task: Binary Classification

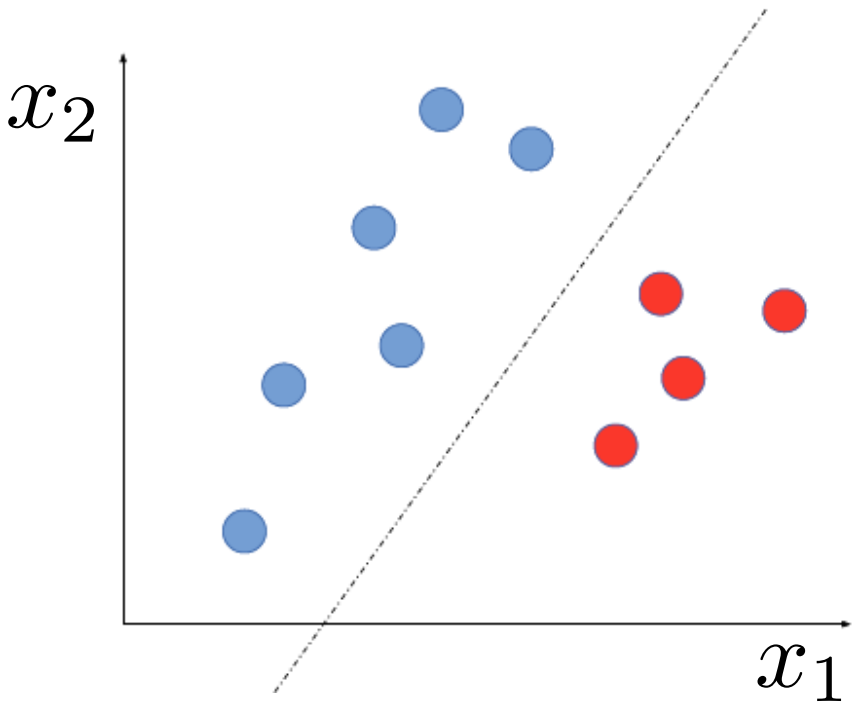
Supervised Learning

**binary classification**

Unsupervised Learning

Reinforcement Learning

$y$ is a binary variable (red or blue)



# Example: Hotdog or Not

https://www.theverge.com/tldr/2017/5/14/15639784/hbo-silicon-valley-not-hotdog-app-download

# From Features to Predictions

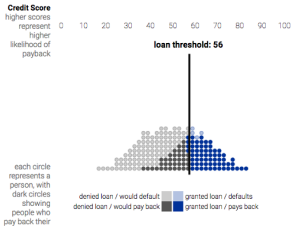
Goal: Predict label (0 or 1) given features x

$$x_i \triangleq [x_{i1}, x_{i2}, \ldots x_{if} \ldots x_{iF}]$$

Input features

$$s_i = h(x_i, \theta)$$

Score
(a real number)




**Chosen threshold**

$$y_i \in \{0, 1\}$$

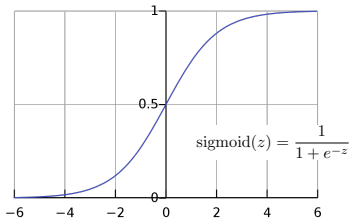
Binary label
(0 or 1)

# From Features to Predictions via Probabilities

Goal: Predict label (0 or 1) given features x

$$x_i \triangleq [x_{i1}, x_{i2}, \ldots x_{if} \ldots x_{iF}]$$

Input features




$$s_i = h(x_i, \theta)$$

Score
(a real number)

$$p_i = \sigma(s_i)$$

Probability of positive class
(between 0.0 and 1.0)

**Chosen threshold between 0.0 and 1.0**

$$y_i \in \{0, 1\}$$

Binary label
(0 or 1)

# Classifier: **Evaluation Step**

Goal: Assess quality of predictions

Many ways in practice:

1) Evaluate binary decisions at specific threshold

    accuracy, TPR, TNR, PPV, NPV, …

2) Evaluate across range of thresholds

    ROC curve, Precision-Recall curve

3) Evaluate probabilities / scores directly

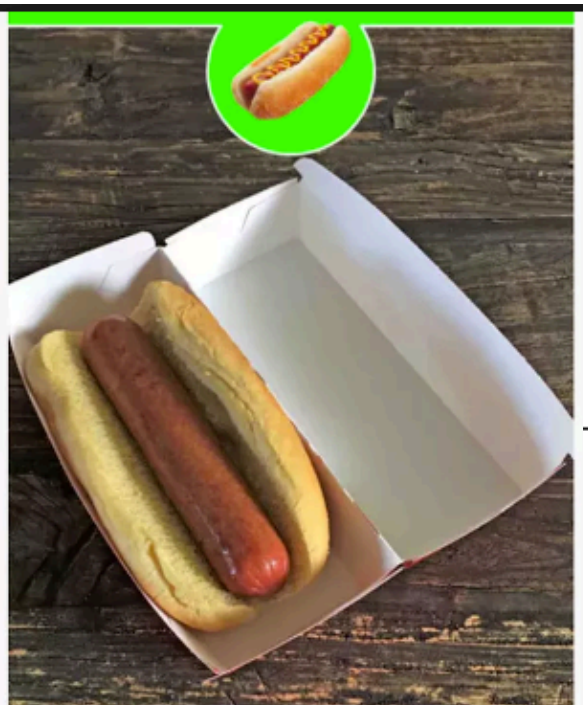
    cross entropy loss (aka log loss), hinge loss, …

# Types of binary predictions

TN : true negative
FN : false negative
FP : false positive
TP : true positive

| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | TN | FP |
| | Y=1 | FN | TP |

# Example:



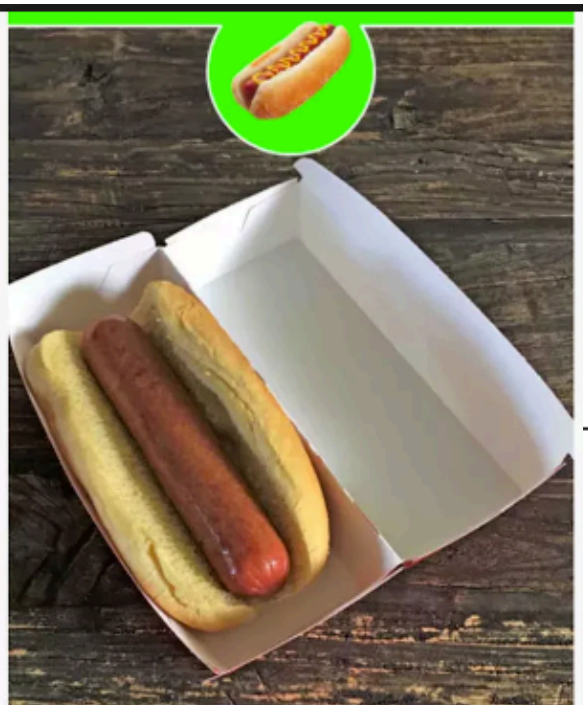
Which outcome is this?

| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | TN | FP |
| | Y=1 | FN | TP |

TN : true negative
FN : false negative

FP : false positive
TP : true positive

# Example:



Which outcome is this?

Answer:
True Positive

| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | TN | FP |
| | Y=1 | FN | TP |

TN : true negative
FN : false negative

FP : false positive
TP : true positive

# Example:

Which outcome is this?

| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | TN | FP |
| | Y=1 | FN | TP |

TN : true negative
FN : false negative

FP : false positive
TP : true positive

# Example:

Which outcome is this?

Answer:
True Negative (TN)

| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | TN | FP |
| | Y=1 | FN | TP |

TN : true negative
FN : false negative

FP : false positive
TP : true positive

# Example:

Which outcome is this?

| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | TN | FP |
| | Y=1 | FN | TP |

TN : true negative
FN : false negative

FP : false positive
TP : true positive

# Example:

Which outcome is this?

Answer:
False Negative (FN)

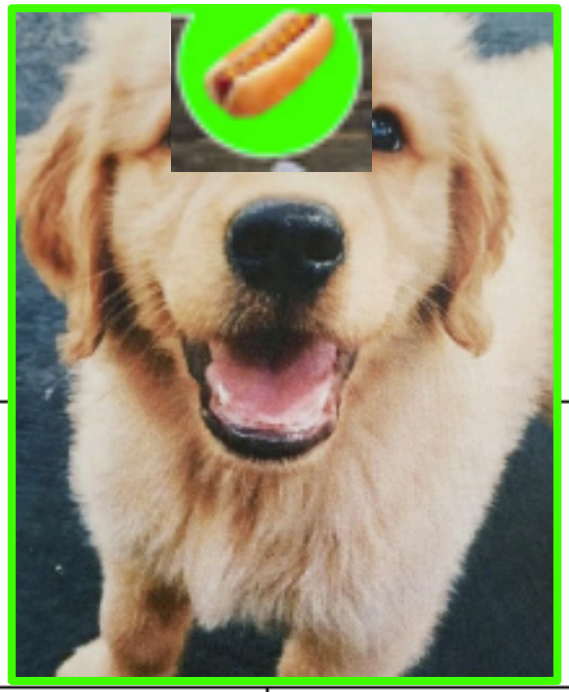
| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | TN | FP |
| | Y=1 | FN | TP |

TN : true negative
FN : false negative

FP : false positive
TP : true positive

# Example:

Which outcome is this?

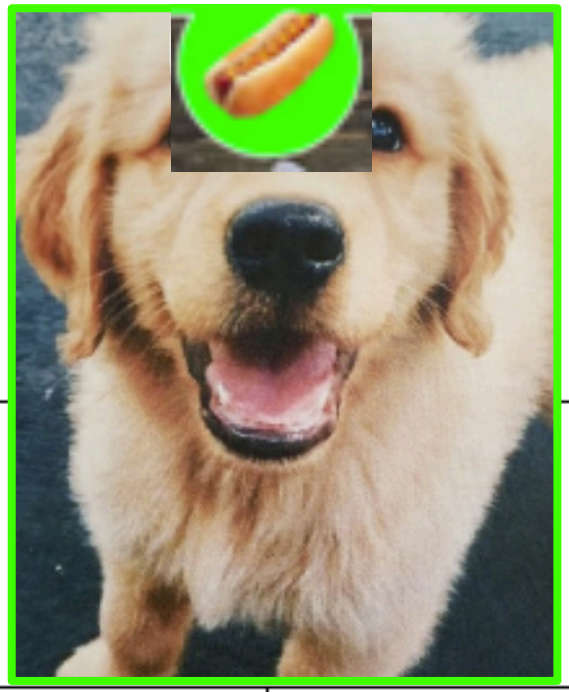


| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | TN | FP |
| | Y=1 | FN | TP |

TN : true negative
FN : false negative

FP : false positive
TP : true positive

# Example:



Which outcome is this?

Answer:
False Positive (FP)

| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | TN | FP |
| | Y=1 | FN | TP |

TN : true negative
FN : false negative

FP : false positive
TP : true positive

# Metric: Confusion Matrix

## Counting mistakes in binary predictions

#TN : num. true negative
#FN : num. false negative

#TP : num. true positive
#FP : num. false positive

| | | classifier calls | |
|---|---|---|---|
| | | "negative" C=0 | "positive" C=1 |
| true outcome | Y=0 | #TN | #FP |
| | Y=1 | #FN | #TP |

# Metric: Accuracy

accuracy = fraction of correct predictions

$$= \frac{TP + TN}{TP + TN + FN + FP}$$

*Potential problem:*

*Suppose your dataset has 1 positive example and 99 negative examples*

*What is the accuracy of the classifier that always predicts "negative"?*

# Metric: Accuracy

accuracy = fraction of correct predictions

$$= \frac{TP + TN}{TP + TN + FN + FP}$$

*Potential problem:*

*Suppose your dataset has 1 positive example and 99 negative examples*

*What is the accuracy of the classifier that always predicts "negative"?*

99%!

# Metrics for Binary Decisions

| METRIC | FORMULA | IN WORDS<br><br>"Probability that …"<br>Or<br>"How often the …" | EXPRESSION |
|---|---|---|---|
| True Positive Rate (TPR)<br>*"sensitivity", "recall"* | $\frac{TP}{TP + FN}$ | subject who is positive will be called positive | $Pr(C = 1 \mid Y = 1)$ |
| True Negative Rate (TNR)<br>*"specificity", 1 - FPR* | $\frac{TN}{FP + TN}$ | subject who is negative will be called negative | $Pr(C = 0 \mid Y = 0)$ |
| Positive Predictive Value (PPV)<br>*"precision"* | $\frac{TP}{TP + FP}$ | subject called positive will actually be positive | $Pr(Y = 1 \mid C = 1)$ |
| Negative Predictive Value (NPV) | $\frac{TN}{TN + FN}$ | subject called negative will actually be negative | $Pr(Y = 0 \mid C = 0)$ |

In practice, you need to emphasize the metrics **appropriate** for your application.

Goal: Classifier to find relevant tweets to list on Tufts website
- If in top 10 by predicted probability, put on website
- If not, discard that tweet

Which metric might be most important? Could we just use accuracy?

| METRIC | FORMULA | IN WORDS<br><br>"Probability that …"<br>Or<br>"How often the …" | EXPRESSION |
|---|---|---|---|
| True Positive Rate (TPR) | $\frac{TP}{TP + FN}$ | subject who is positive will be called positive | $Pr(C = 1 \mid Y = 1)$ |
| True Negative Rate (TNR) | $\frac{TN}{FP + TN}$ | subject who is negative will be called negative | $Pr(C = 0 \mid Y = 0)$ |
| Positive Predictive Value (PPV) | $\frac{TP}{TP + FP}$ | subject called positive will actually be positive | $Pr(Y = 1 \mid C = 1)$ |
| Negative Predictive Value (NPV) | $\frac{TN}{TN + FN}$ | subject called negative will actually be negative | $Pr(Y = 0 \mid C = 0)$ |

Goal: Detector for cancer based on medical image
- If called positive, patient gets further screening
- If called negative, no further attention until 5+ years later

Which metric might be most important? Could we just use accuracy?

| METRIC | FORMULA | IN WORDS<br><br>"Probability that …"<br>Or<br>"How often the …" | EXPRESSION |
|---|---|---|---|
| True Positive Rate (TPR) | $\frac{TP}{TP + FN}$ | subject who is positive will be called positive | $Pr(C = 1 \mid Y = 1)$ |
| True Negative Rate (TNR) | $\frac{TN}{FP + TN}$ | subject who is negative will be called negative | $Pr(C = 0 \mid Y = 0)$ |
| Positive Predictive Value (PPV) | $\frac{TP}{TP + FP}$ | subject called positive will actually be positive | $Pr(Y = 1 \mid C = 1)$ |
| Negative Predictive Value (NPV) | $\frac{TN}{TN + FN}$ | subject called negative will actually be negative | $Pr(Y = 0 \mid C = 0)$ |

# Classifier: **Evaluation Step**

Goal: Assess quality of predictions

Many ways in practice:

1) Evaluate binary decisions at specific threshold

    accuracy, TPR, TNR, PPV, NPV, …

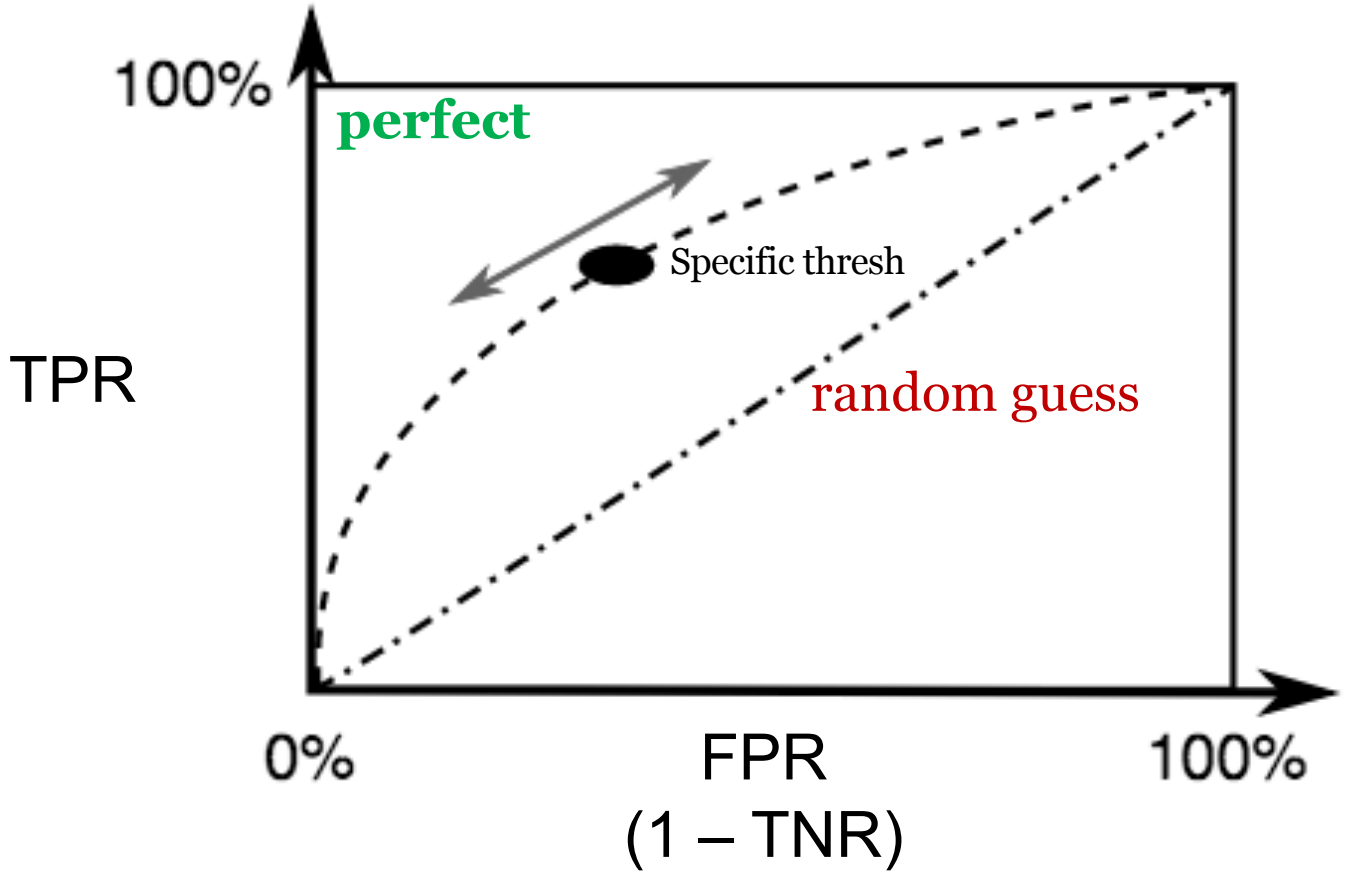
**2) Evaluate across range of thresholds**

    **ROC curve, Precision-Recall curve**

3) Evaluate probabilities / scores directly

    cross entropy loss (aka log loss), hinge loss, …

# ROC curve



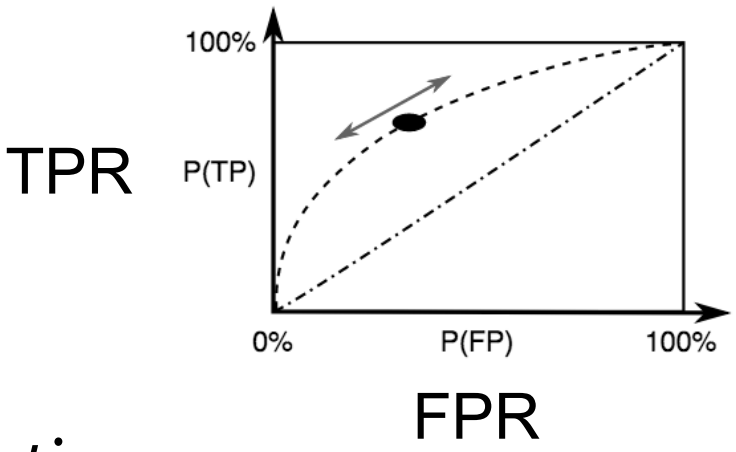

Each point represents TPR and FPR of one specific threshold

Connecting all points (all thresholds) produces the curve

# Area under ROC curve
(aka AUROC or AUC or "C statistic")

Area varies from 0.0 – 1.0.
0.5 is random guess.
1.0 is perfect.

*Graphical view:*




*Probabilistic interpretation:*

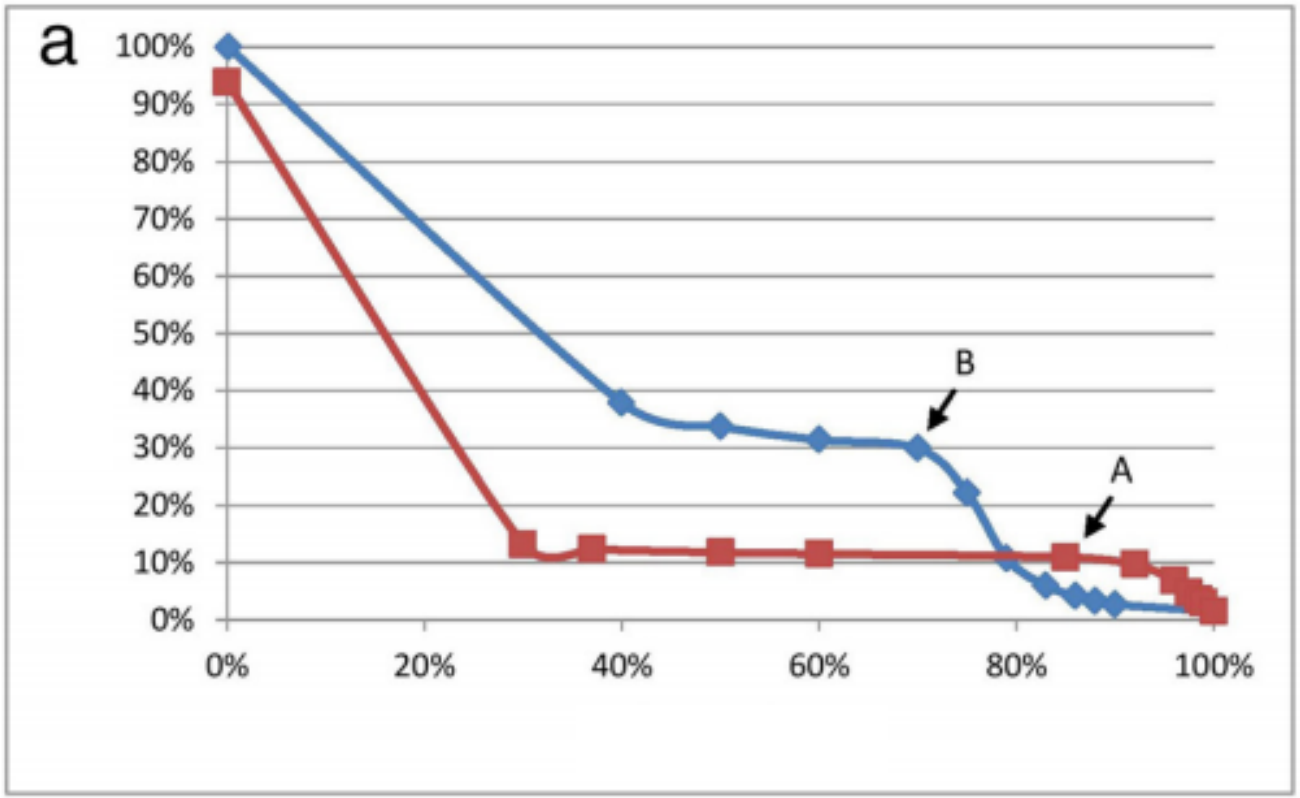
$$\text{AUROC} \triangleq \Pr(\hat{y}(x_i) > \hat{y}(x_j) | y_i = 1, y_j = 0)$$

For random pair of examples, one positive and one negative,
What is probability classifier will rank positive one higher?

# Precision-Recall Curve

PPV

precision

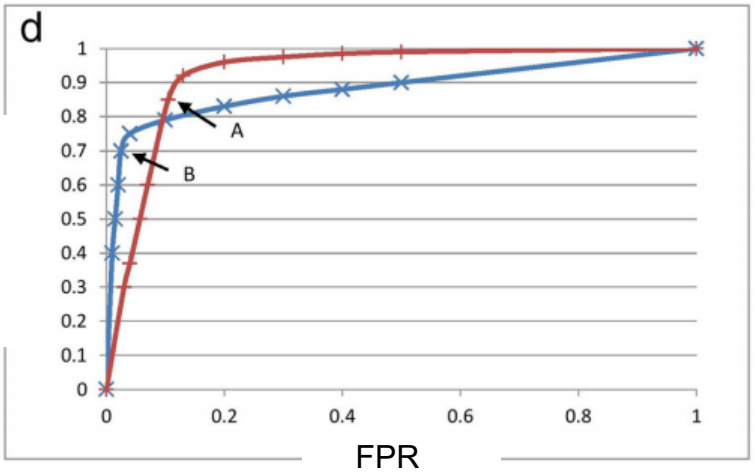



recall (aka TPR)

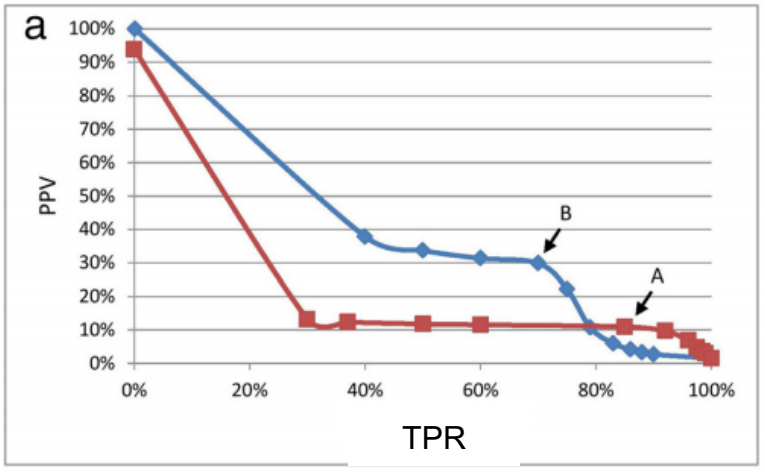
# AUROC not always best choice

## Why the *C*-statistic is not informative to evaluate early warning scores and what metrics to use

Santiago Romero-Brufau[1,2*], Jeanne M. Huddleston[1,2,3], Gabriel J. Escobar[4] and Mark Liebow[5]




AUROC says red is better




Blue much better for avoiding false alarms

# Classifier: **Evaluation Step**

Goal: Assess quality of predictions

Many ways in practice:

1) Evaluate binary decisions at specific threshold

    accuracy, TPR, TNR, PPV, NPV, …

2) Evaluate across range of thresholds

    ROC curve, Precision-Recall curve

**3) Evaluate probabilities / scores directly**

    **cross entropy loss (aka log loss)**

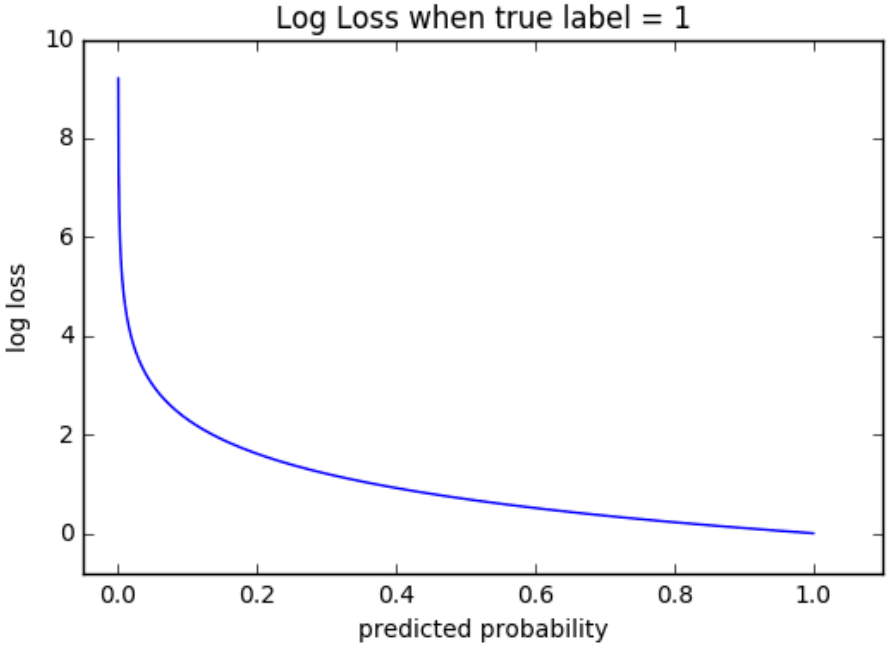
    Not covered yet: hinge loss, many others

# Measuring quality of predicted probabilities

Use the log loss (aka "binary cross entropy")

**from sklearn.metrics import** log_loss

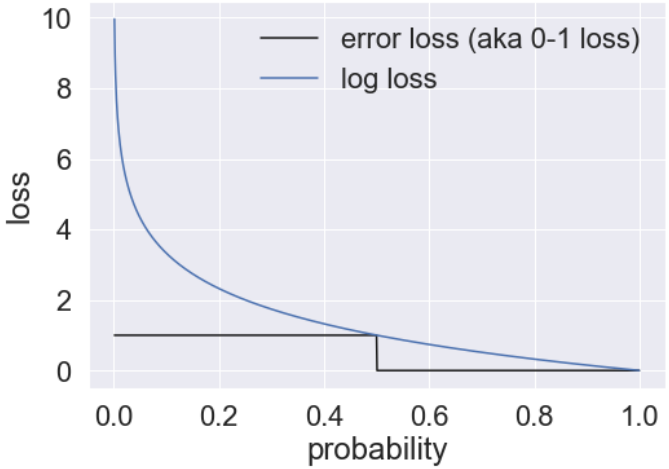
$$\text{log\_loss}(y, \hat{p}) = -y \log \hat{p} - (1 - y) \log(1 - \hat{p})$$




Advantages:

- smooth
- easy to take derivatives!

# Why minimize log loss?
# The upper bound justification




Log loss (if implemented in correct base) is a **smooth** **upper bound** of the error rate.

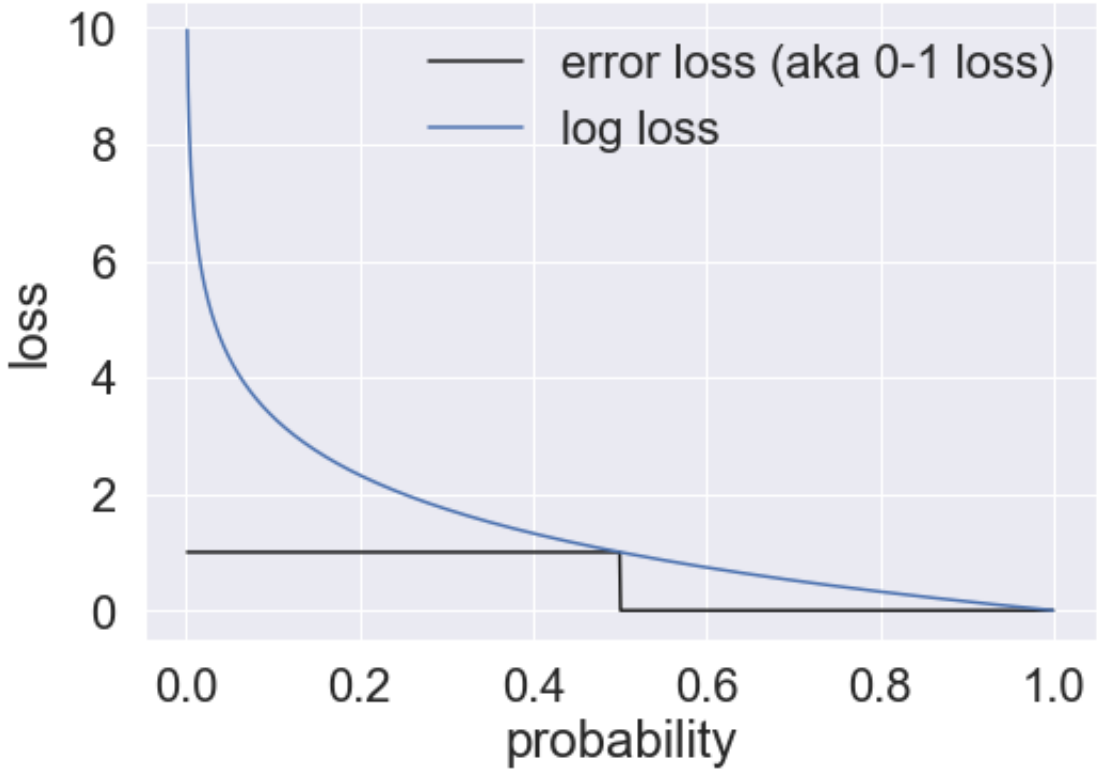
Why **smooth** matters: easy to do gradient descent

Why **upper bound** matters: achieving a log loss of 0.1 (averaged over dataset) guarantees us that error rate is no worse than 0.1 (10%)

# Log loss upper bounds 0-1 error

$$\text{error}(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{if } y = \hat{y} \end{cases}$$

$$\text{log\_loss}(y, \hat{p}) = -y \log \hat{p} - (1 - y) \log(1 - \hat{p})$$




Plot assumes:

- True label is 1

- Threshold is 0.5

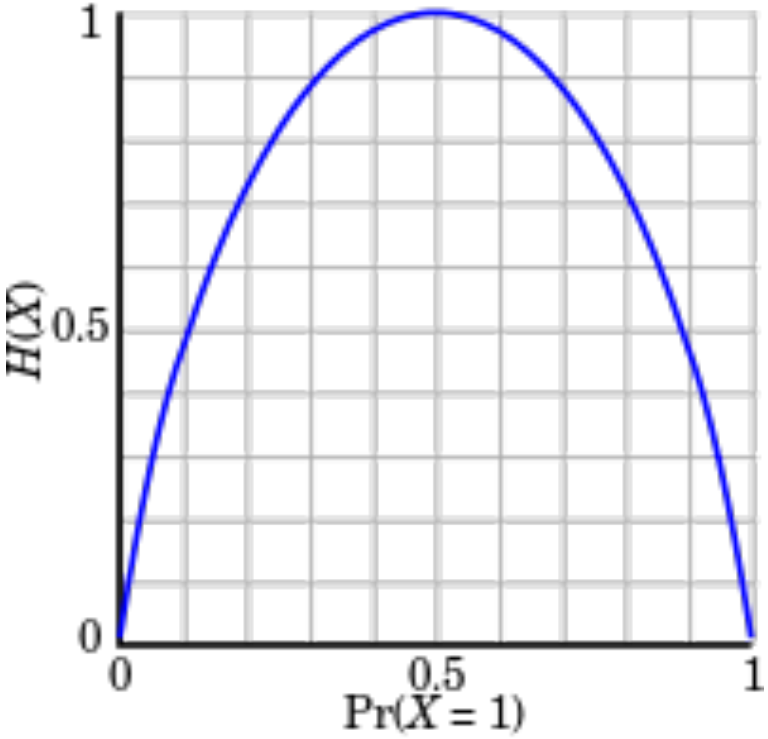
- Log base 2

# Why minimize log loss?
## An information-theory justification

# Entropy of Binary Random Var.

Goal: Entropy of a distribution captures the amount of uncertainty

$$\text{entropy}(X) = -p(X=1)\log_2 p(X=1) - p(X=0)\log_2 p(X=0)$$




Log base 2: Units are “bits”
Log base e: Units are “nats”

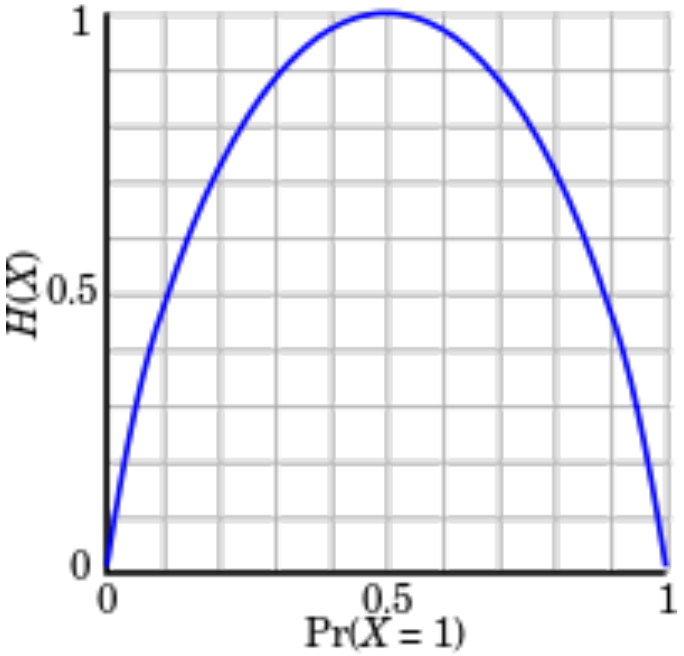
1 bit of information is always needed to represent a binary variable X

Entropy tells us how much of this one bit is uncertain

# Entropy of Binary Random Var.

Goal: Entropy of a distribution captures the amount of uncertainty

$$\text{entropy}(X) = -p(X=1)\log_2 p(X=1) - p(X=0)\log_2 p(X=0)$$




$$H[X] = -\sum_{x \in \{0,1\}} p(X=x)\log_2 p(X=x)$$

$$= -\mathbb{E}_{x \sim p(X)}\left[\log_2 p(X=x)\right]$$

Entropy is the **average** number of bits needed to encode an outcome

Want: low entropy
(low cost storage and transmission!)

# Cross Entropy

Goal: Measure cost of using estimated q to capture true distribution p

$$\text{Entropy}[p(X)] = -\sum_{x \in \{0,1\}} p(X = x) \log_2 p(X = x)$$

$$\text{Cross-Entropy}[p(X), q(X)] = -\sum_{x \in \{0,1\}} p(X = x) \log_2 q(X = x)$$

Info theory interpretation:

Average number of bits needed to encode
samples from a true distribution p(X)
with codes defined by a model q(X)

Goal: Want a model that uses fewer bits!
Lower entropy = more information captured about the outcome labels!

# Log loss is cross entropy!

Let our "true" distribution p(Y) be **empirical** distribution of labels in our observed dataset with N examples

Let our "model" distribution q(Y) be our estimated probabilities

$$\text{Cross-Entropy}[p(Y), q(Y)] = \mathbb{E}_{y \sim p(Y)} \left[ -\log q(Y = y) \right]$$

$$= \frac{1}{N} \sum_{n=1}^{N} -y_n \log \hat{p}_n - (1 - y_n) \log(1 - \hat{p}_n)$$

Same as the average "log loss"!

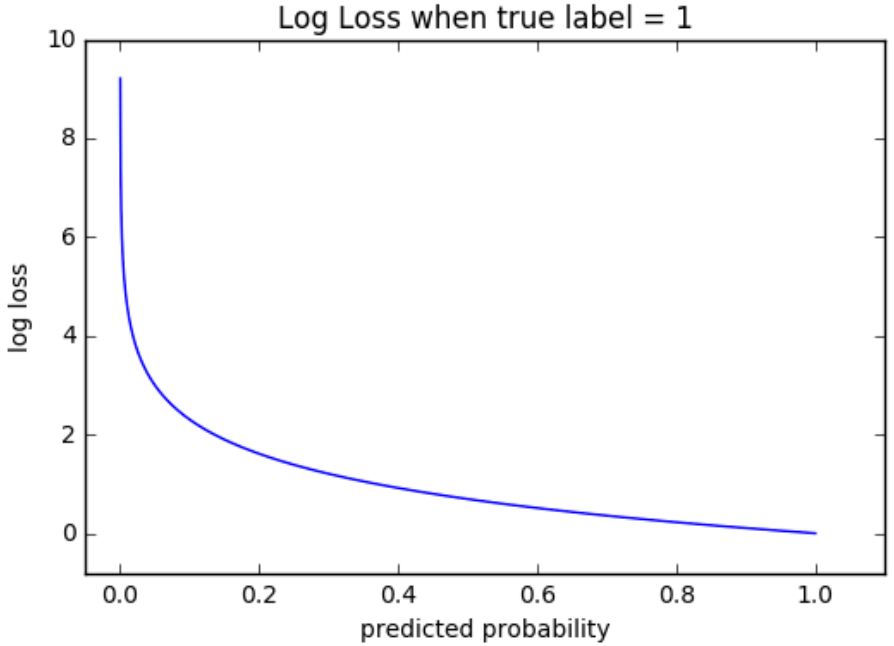
*Info Theory Justification for log loss:*
Want to set model parameters to provide best probabilistic encoding of the training data's label distribution

# The log loss metric

Log loss (aka "binary cross entropy")

**from sklearn.metrics import** log_loss

$$\text{log\_loss}(y, \hat{p}) = -y \log \hat{p} - (1 - y) \log(1 - \hat{p})$$




*Lower is better!*

Advantages:

- smooth and not flat
- easy to take derivatives!
- convex function

# Code for Evaluation Metrics

https://scikit-learn.org/stable/modules/model_evaluation.html

1) To evaluate predicted scores / probabilities

| | |
|---|---|
| `log_loss` (y_true, y_pred[, eps, normalize, …]) | Log loss, aka logistic loss or cross-entropy loss. |
| `hinge_loss` (y_true, pred_decision[, labels, …]) | Average hinge loss (non-regularized) |

2) To evaluate specific binary decisions

| | |
|---|---|
| `confusion_matrix`(y_true, y_pred, *[, ...]) | Compute confusion matrix |

3) To make ROC or PR curves

| | |
|---|---|
| `precision_recall_curve`(y_true, probas_pred, *) | Compute precision-recall pairs for different probability t |
| `roc_curve`(y_true, y_score, *[, pos_label, ...]) | Compute Receiver operating characteristic (ROC) |

# Today's objectives (day 08)
# Evaluating Binary Classifiers

1) Evaluate binary decisions at specific threshold

    accuracy, TPR, TNR, PPV, NPV, …

2) Evaluate across range of thresholds

    ROC curve, Precision-Recall curve

3) Evaluate probabilities / scores directly

    cross entropy loss (aka log loss)