Machine Learning (CS 181): 6. Linear Classification and the Perceptron

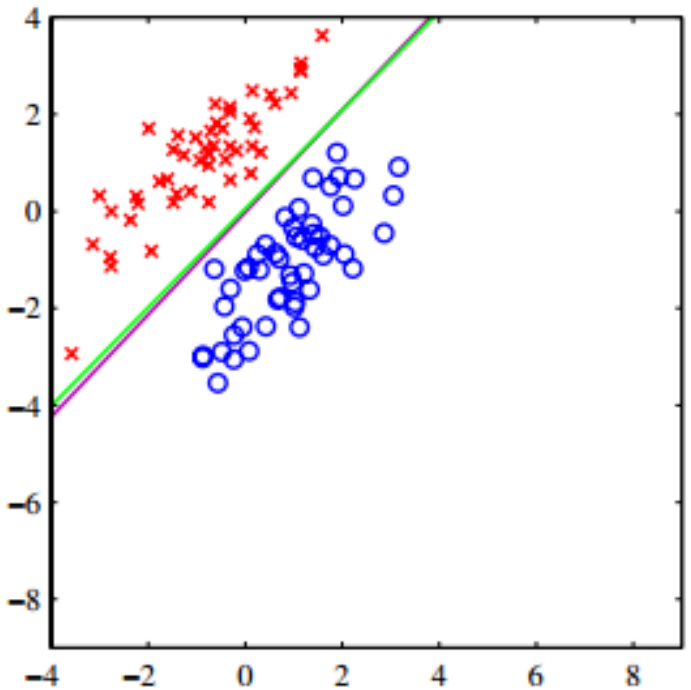
David Parkes and Sasha Rush

Contents

Contents

Binary Classification

- Output space $\mathcal{Y}$ is a fixed set of classes.
- Simplest case $\mathcal{Y} = \{-1, 1\}$ (red/blue)



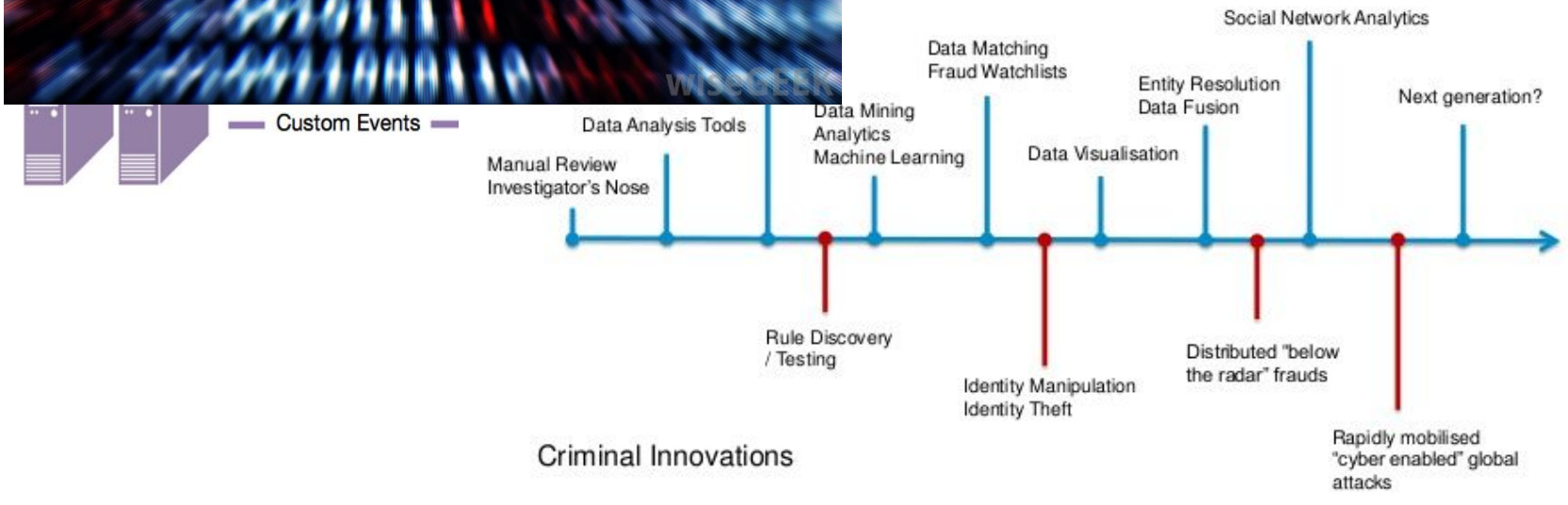
Binary Classification

Linear Classification

Fraud Scores

BAE SYSTEMS

ection Technologies



thenextweb.com

Multiclass Classification

- Output space $\mathcal{Y}$ is a fixed set of classes.
- Made up of c discrete classes $\{C_1, \ldots C_c\}$

e.g.

- Image recognition
- Disease diagnostics
- Product recommendation

Example: Digit Classification

- Data: Handwritten US zip codes

Contents

In Brief: A Non-Parametric Classification Approach

k-Nearest Neighbors

1. Given new input $\mathbf{x}$
2. Find k closest $\mathbf{x}$ training points $(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(k)}, y^{(k)})$
3. Return majority class value

$$\hat{y} = \begin{cases} 1 & \text{if } \sum_{i=1}^{k} y^{(i)} > 0 \\ -1 & o.w \end{cases}$$

Binary Linear Classification

$$h(\mathbf{x}; \mathbf{w}, w_0) = \mathbf{w}^\top \mathbf{x} + w_0$$

$$h : \mathcal{X} \mapsto \mathbb{R} \text{ (slight change)}$$

- w_0; threshold or bias (explicit)
- Predict class $\hat{y} = 1$ if $h(\mathbf{x}; \mathbf{w}, w_0) > 0$
- Else predict class $\hat{y} = -1$

Decision Boundary

All $\mathbf{x}$ such that model score is balanced.

$$h(\mathbf{x}; \mathbf{w}, w_0) = \mathbf{w}^\mathsf{T}\mathbf{x} + w_0 = 0 \tag{1}$$

In the case of linear model, defined by:

$$\mathcal{S} = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{w}^\mathsf{T}\mathbf{x} + w_0 = 0\}$$

Geometrically, the set $\mathcal{S}$ is an *affine hyperplane*,

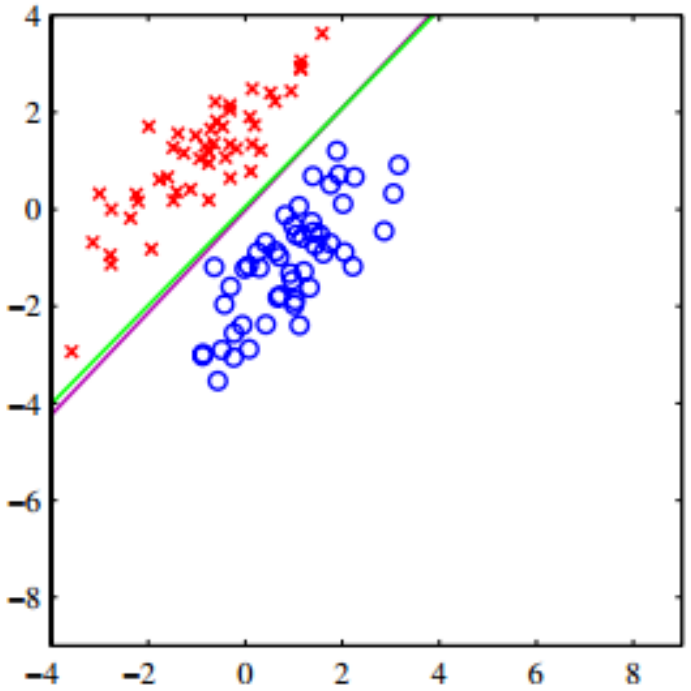
- implicitly partitions/separates the space $\mathcal{X}$ into two parts

Separating Hyperplane / Decision Boundary

$$\mathcal{S} = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{w}^\mathsf{T}\mathbf{x} + w_0 = 0\}$$

In $\mathcal{X} = \mathbb{R}^2$, algebra to solve for x_2,

$$x_2 = \frac{-w_1}{w_2} x_1 - \frac{w_0}{w_2}$$



1) Decision Boundary Orientation

$$\mathcal{S} = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{w}^\mathsf{T}\mathbf{x} + w_0 = 0\}$$

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be in $\mathcal{S}$, and consider vector $\mathbf{s} = \mathbf{x}_1 - \mathbf{x}_2$

$$\begin{aligned}\mathbf{w}^\top\mathbf{s} &= \mathbf{w}^\top(\mathbf{x}_1 - \mathbf{x}_2) \\ &= \mathbf{w}^\top\mathbf{x}_1 - \mathbf{w}^\top\mathbf{x}_2 \\ &= w_0 - w_0 = 0\end{aligned}$$

Therefore:

- orientation is determined by $\mathbf{w}$
- hyperplane orthogonal to $\mathbf{w}$.

1) Decision Boundary Orientation

$$\mathcal{S} = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{w}^\mathsf{T}\mathbf{x} + w_0 = 0\}$$

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be in $\mathcal{S}$, and consider vector $\mathbf{s} = \mathbf{x}_1 - \mathbf{x}_2$

$$\begin{aligned}\mathbf{w}^\top\mathbf{s} &= \mathbf{w}^\top(\mathbf{x}_1 - \mathbf{x}_2) \\ &= \mathbf{w}^\top\mathbf{x}_1 - \mathbf{w}^\top\mathbf{x}_2 \\ &= w_0 - w_0 = 0\end{aligned}$$

Therefore:

- orientation is determined by $\mathbf{w}$
- hyperplane orthogonal to $\mathbf{w}$.

2) Decision Boundary Location 1

Consider point in plane closest to origin (as optimization),

$$\arg\min_{\mathbf{x}\in\mathcal{S}} ||\mathbf{x} - \mathbf{0}||$$

or

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \tfrac{1}{2}\mathbf{x}^\top\mathbf{x} \\ s.t. \quad & \mathbf{w}^\top\mathbf{x} + w_0 = 0 \end{aligned}$$

With Lagrange multiplier

$$L(\mathbf{x}, \lambda) = \frac{1}{2}\mathbf{x}^\top\mathbf{x} + \lambda(\mathbf{w}^\top\mathbf{x} + w_0 - 0)$$

2) Decision Boundary Location 1

Consider point in plane closest to origin (as optimization),

$$\arg\min_{\mathbf{x}\in\mathcal{S}} ||\mathbf{x} - \mathbf{0}||$$

or

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \tfrac{1}{2}\mathbf{x}^\top\mathbf{x} \\ s.t. \quad & \mathbf{w}^\top\mathbf{x} + w_0 = 0 \end{aligned}$$

With Lagrange multiplier

$$L(\mathbf{x}, \lambda) = \frac{1}{2}\mathbf{x}^\top\mathbf{x} + \lambda(\mathbf{w}^\top\mathbf{x} + w_0 - 0)$$

2) Decision Boundary Location 2

$$L(\mathbf{x}, \lambda) = \frac{1}{2}\mathbf{x}^\top \mathbf{x} + \lambda(\mathbf{w}^\top \mathbf{x} + w_0)$$

Partials for both λ and $\mathbf{x}$:

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \mathbf{x}} = \mathbf{x} + \lambda \mathbf{w}$$

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \lambda} = \mathbf{w}^\top \mathbf{x} + w_0$$

Solve for $\mathbf{x}, \lambda$ at min

$$\mathbf{w}^\top (-\lambda \mathbf{w}) + w_0 = 0$$

$$\lambda = \frac{w_0}{\mathbf{w}^\top \mathbf{w}}$$

Solve for $\mathbf{x}$ and compute distance to origin:

$$||\mathbf{x}|| = ||(-w_0/\mathbf{w}^\top \mathbf{w})\mathbf{w}|| = \frac{|w_0|}{||\mathbf{w}||}$$

Linear Separability

- Can $h(\mathbf{x})$ correctly classify all training data?
- Often not. Data must be *linear separable*.
- There exists an affine hyperplane separates data into positive/negative classes.
  - May be noise
  - Underlying linear space may be incorrect.

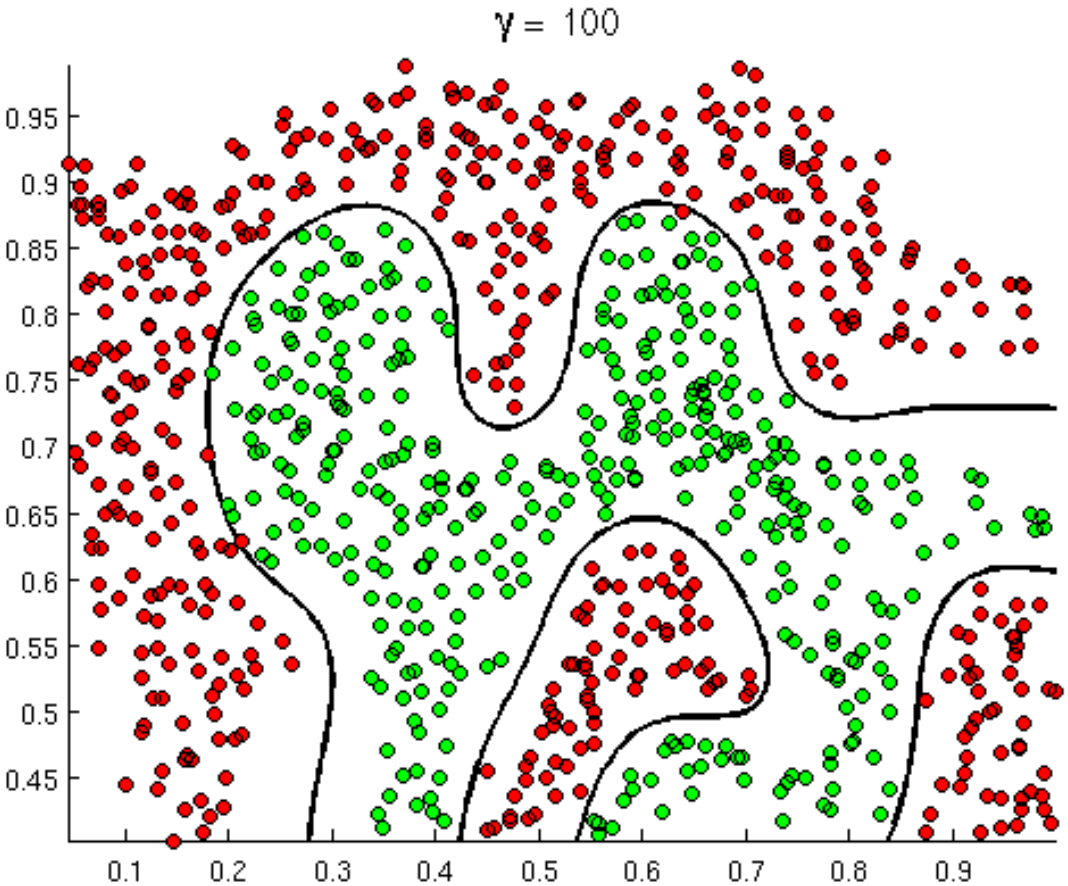
Input Basis

- Same basis technique can be applied in classification.

$$h(\mathbf{x}; \mathbf{w}, w_0) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + w_0.$$

- Data may be not linearly separable in raw form.
- Basis function provides another representation that makes it linear-separable (in $\mathbb{R}^d$)

Separator View with Basis Function



Contents

Least Squares?

Preliminary Idea: treat using least squares loss

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{n} \big(y_i - h(\mathbf{x}_i; \mathbf{w})\big)^2$$

- Target y_i always $\{-1, 1\}$.
- Can use the nice machinery from linear regression.
- What goes wrong?
  - Take loss on correctly classified points.
  - Outlier correct points can change boundary.

Least Squares?

Preliminary Idea: treat using least squares loss

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{n} \big(y_i - h(\mathbf{x}_i; \mathbf{w})\big)^2$$

- Target y_i always $\{-1, 1\}$.
- Can use the nice machinery from linear regression.
- What goes wrong?
  - Take loss on correctly classified points.
  - Outlier correct points can change boundary.

Alternative approach: Perceptron

Loss should target classification accuracy.

High-level algorithm:

1. Check if we classified a point incorrectly.
2. If not, move on.
3. Otherwise, adjust weights.

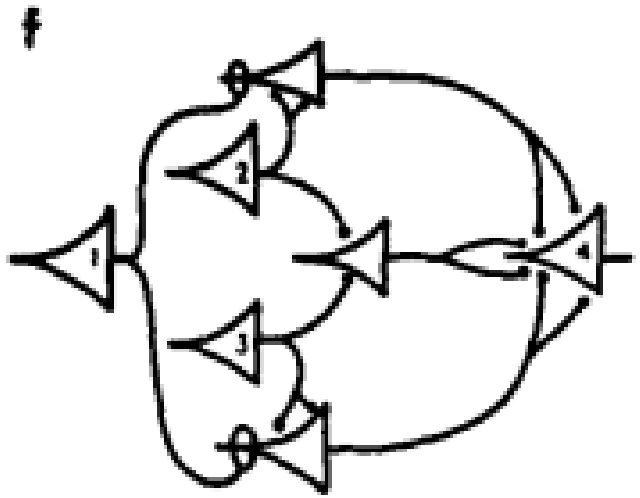
Perceptron History: Origin of Neural Networks

McCullogh-Pitts (1943): explore relationship between logical expression and neurological activations.

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. MCCULLOCH and WALTER H. PITTS

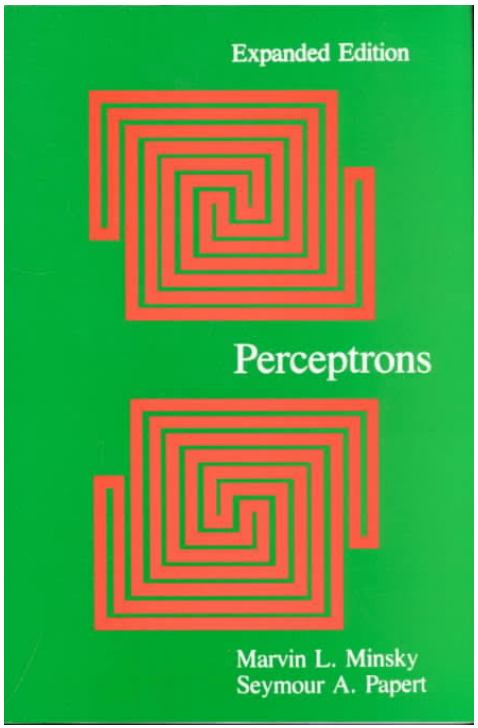
Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.



Rosenblatt (1958): Develops the perceptron learning rule.

Perceptron History: End of 1st wave of Neural Networks

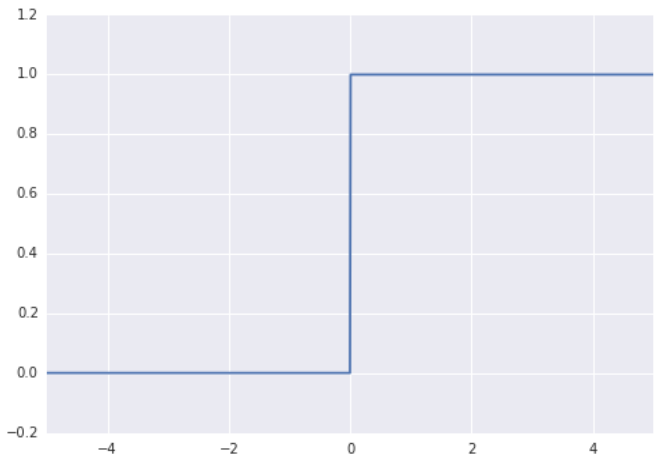
- Minsky and Papert (1969): argue that the linear separability limit single-layer perceptrons hinders their use for AI.



Activation Function

- Over next few class, we will see several functions to transform the reals for classification.
- 0/1 Activation Function:

$$f_{0/1}(z) = \begin{cases} +1 & z > 0 \\ 0 & o.w. \end{cases}$$

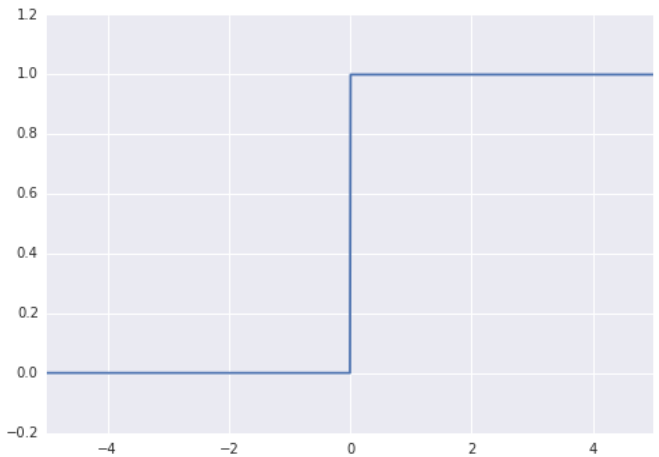


Loss Function

Ideal classification loss function: penalty for misclassification.

- Use multiplication by $-y_i$ as a sign trick.

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \sum_{i=1}^{n} f_{0/1}(-h(\mathbf{x}_i;\mathbf{w})y_i) \\ &= \sum_{i=1:y_i \neq \hat{y}_i}^{n} 1\end{aligned}$$

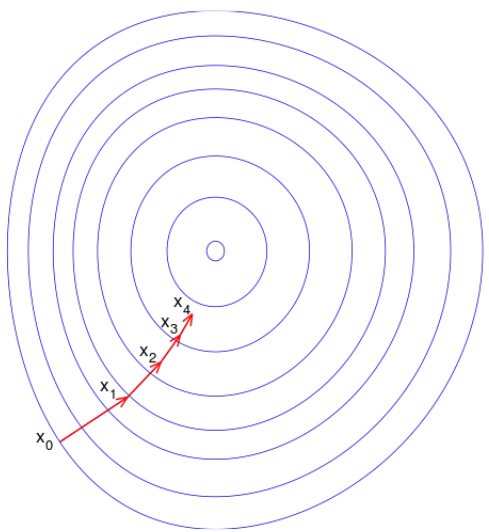


Gradient Descent

Minimize loss by repeated gradient steps (when no closed form):

1. Compute gradient of loss with respect to parameters $\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$
2. Update parameters with rate η

$$\mathbf{w}' \leftarrow \mathbf{w} - \eta \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$$



Gradient steps on a simple $m = 2$ loss function.

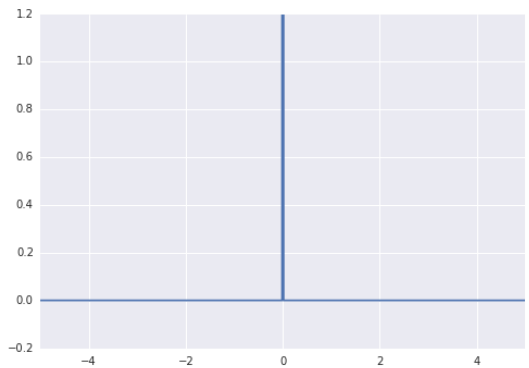
0/1 Loss Function Issues

Want to optimize with gradient descent.

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n} f_{0/1}(-h(\mathbf{x}_i; \mathbf{w})y_i)$$

- Gradient of $f_{0/1}$ is uninformative.

$$f_{0/1}(z) = \begin{cases} +1 & z > 0 \\ 0 & o.w. \end{cases}$$

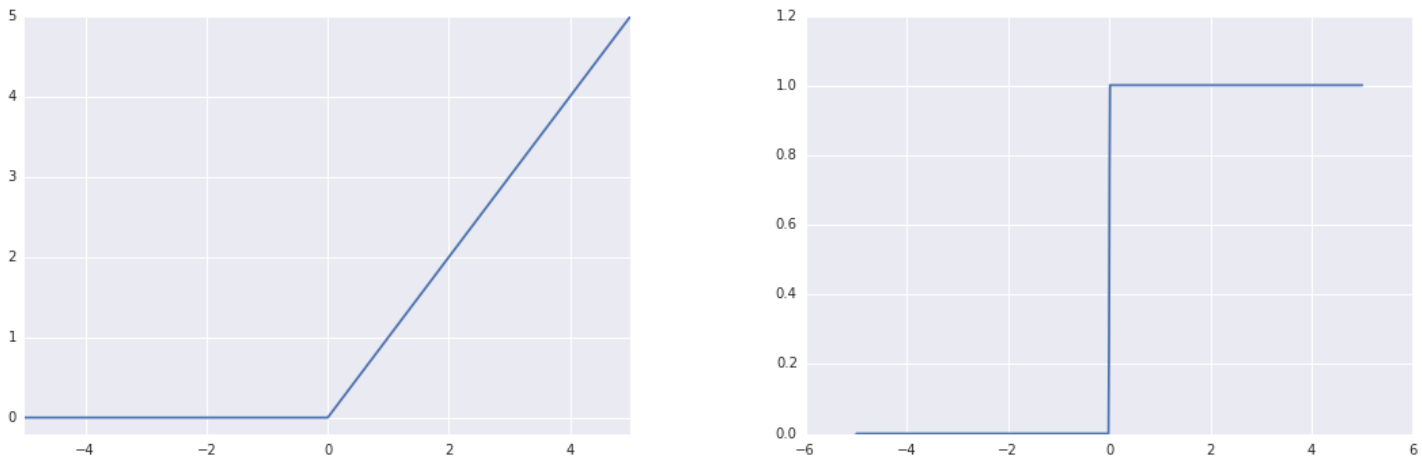


Derivative of 0/1

Solution 1: Hinge / Rectified Linear Activation

- One potential solution to get around this issue.

$$f_{relu}(z) = \begin{cases} z & z > 0 \\ 0 & o.w. \end{cases} = \max\{0, z\}$$



Hinge Function and Derivative

- Value increases as misclassification gets worse.

Perceptron Loss

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \sum_{i=1}^{n} f_{relu}(-h(\mathbf{x}_i)y_i) \\ &= -\sum_{i=1:y_i \neq \hat{y}_i}^{n} (\mathbf{w}^\top \mathbf{x}_i + w_0)y_i\end{aligned}$$

- Penalty scales with number of mistakes and how bad they are.
- Convex, but does not have a closed-form solution.

Stochastic Gradient Descent

Instead of computing gradient of entire loss, compute stochastic gradients.

1. Sample data point i, corresponding to $(\mathbf{x}_i, y_i)$
2. Compute loss and gradient for just that data point $\mathcal{L}^{(i)}(\mathbf{w})$
3. Update $\mathbf{w}$ based on this *stochastic gradient*

Similar to standard gradient descent, often more efficient in practice.

Gradient of Perceptron Loss

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) = \frac{\partial}{\partial \mathbf{w}} \left(- \sum_{i=1:y_i \neq \hat{y}_i}^{n} y_i \mathbf{w}^\top \mathbf{x}_i \right)$$

$$= - \sum_{i=1:y_i \neq \hat{y}_i}^{n} y_i \mathbf{x}_i$$

Stochastic version, just one incorrect datum and learning rate η:

$$\begin{aligned} \mathcal{L}^{(i)}(\mathbf{w}) &= f_{relu}(-h(\mathbf{x}_i) y_i) \\ \frac{\partial}{\partial \mathbf{w}} \mathcal{L}^{(i)}(\mathbf{w}) &= -y_i \mathbf{x}_i \end{aligned}$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta y_i \mathbf{x}_i$$

Because $\mathbf{w}$ doesn't change with scale (see λ), we can take $\eta = 1$ wlog.

Perceptron Algorithm

1. Iterate over the data:
   - If correct ($y_i = \hat{y}_i$), do nothing.
   - If incorrect, add $y_i \mathbf{x}_i$ to weights and y_i to w_0 (exercise)
2. If errors, repeat process.
3. Otherwise separator is found.

Guaranteed to converge if data is linearly separable.

Perceptron Demo

[IPython Demo]

Contents

Generative Classification View

Model the joint probability of call y and data $\mathbf{x}$

$$p(y, \mathbf{x}) = p(y)p(\mathbf{x}|y)$$

Switch to different class representation

$$\mathcal{Y} = \{0, 1\} \text{ or } \{C_1, \ldots C_c\}$$

Process:

- Class is generated with probability $p(y)$
- Input $\mathbf{x}$ is generated conditional on class $p(\mathbf{x}|y)$

Class Probability

Choice of prior can depend on problem format,

- In binary case, we will use Bernoulli distribution (last class),

$$p(y = 1; \theta) = \theta \quad p(y = 0; \theta) = 1 - \theta$$

  or more compactly

$$p(y; \theta) = \theta^y (1 - \theta)^{1-y}$$

- In multiclass (HW), can use categorical distribution,

$$p(y = C_k; \boldsymbol{\pi}) = \pi_k$$

Data Probability

- Choice depends on modeling assumptions
- Select parameteric model for data given class

1. For discrete or indicator data, simple Naive Bayes (next class).

$$p(\mathbf{x}|y) = \prod_{j=1}^{m} p(x_j|y)$$

2. Use continuous data, use multivariate Gaussian (HW)

$$p(\mathbf{x}|y=0) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$$

$$p(\mathbf{x}|y=1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$

Upcoming: Maximum Likelihood

Model is fit using the same approach as for regression.

1. Decide on generating process
2. Fix the parameterization of the model
3. Maximize likelihood of the data.

$$\min_{\pi,\mathbf{w}} \mathcal{L}(\pi, \mathbf{w}) = \min_{\theta,\mathbf{w}} -\sum_{i=1}^{n} \ln p(y_i; \theta) + \ln p(\mathbf{x}_i | y_i; \mathbf{w})$$

- Again, benefits will be having explicit probabilities for classes.
- Can also combine with Bayesian approaches from last class.

Contents

Metrics

Practical concerns when deploying classification models,

- How well does it perform on real data?
- What actions should we take based on predictions?

Error Metrics

Consider the following binary classification statistics:

Type	Symbol	y	$\hat{y}$
True Positive	TP	1	1
False Positive	FP	0	1
False Negative	FN	1	0
True Negative	TN	0	0

These distinctions are particularly important with unbalanced data:

- e.g. banking fraud, anomaly detection, medical diagnostics

Detection Metrics

- Accuracy:

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

- True Positives Rate:

$$TPR = \frac{TP}{TP + FN}$$

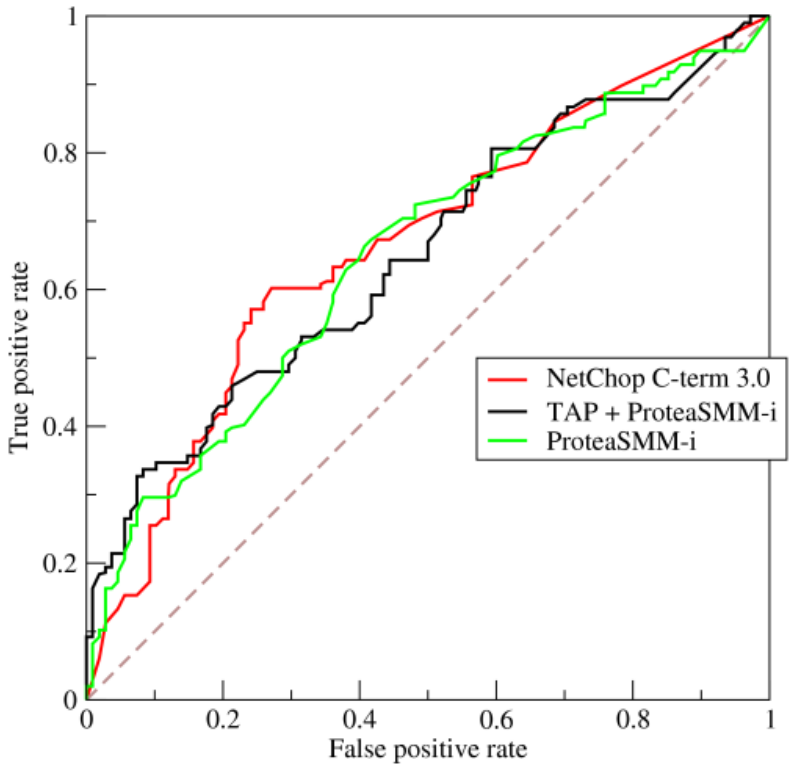
.

- False Positive Rate:

$$FPR = \frac{FP}{FP + TN}$$

ROC Curve

- True Positive Rate by False Positive Rate as sensitivity is adjusted
- Can be produced by varying threshold w_0
- Dotted $y = x$ line is expectation of random classifier.



- Area Under Curve (AUC) used to evaluate a method

F1-Score

Common metric for binary classification and retrieval type problems

- Precision: "Frac of detection are correct"

$$P = \frac{TP}{TP + FP}$$

.

- Recall: "Frac of true cases detected"

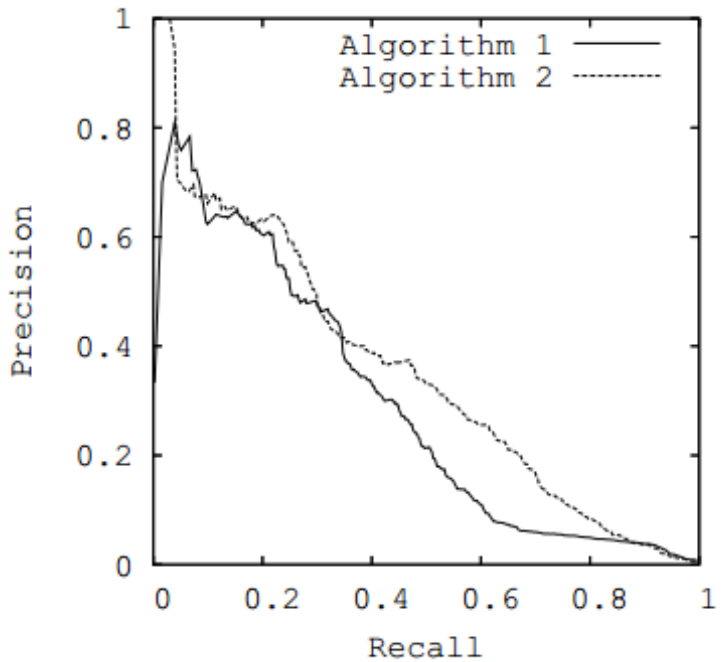
$$R = \frac{TP}{TP + FN}$$

- F1-Score: Harmonic mean of measures

$$2 \times \frac{P * R}{P + R}$$

Precision/Recall curve

Common to show Precision/Recall graph as well.



Fit to the cost function of the application:

- High Recall?
- High Precision?