Logistic regression: A good example of output transformation

Thanks to Professor Dale Schuurmans
University of Alberta and Google Brain

Instructor: Farzaneh Mirzazadeh
Department of Computer Science, UCSC, Winter 2017

Outline

We will learn that **logistic regression** is

- A useful model for **classification**
- A good example of **non-linear output transformation** method in machine learning

Recall

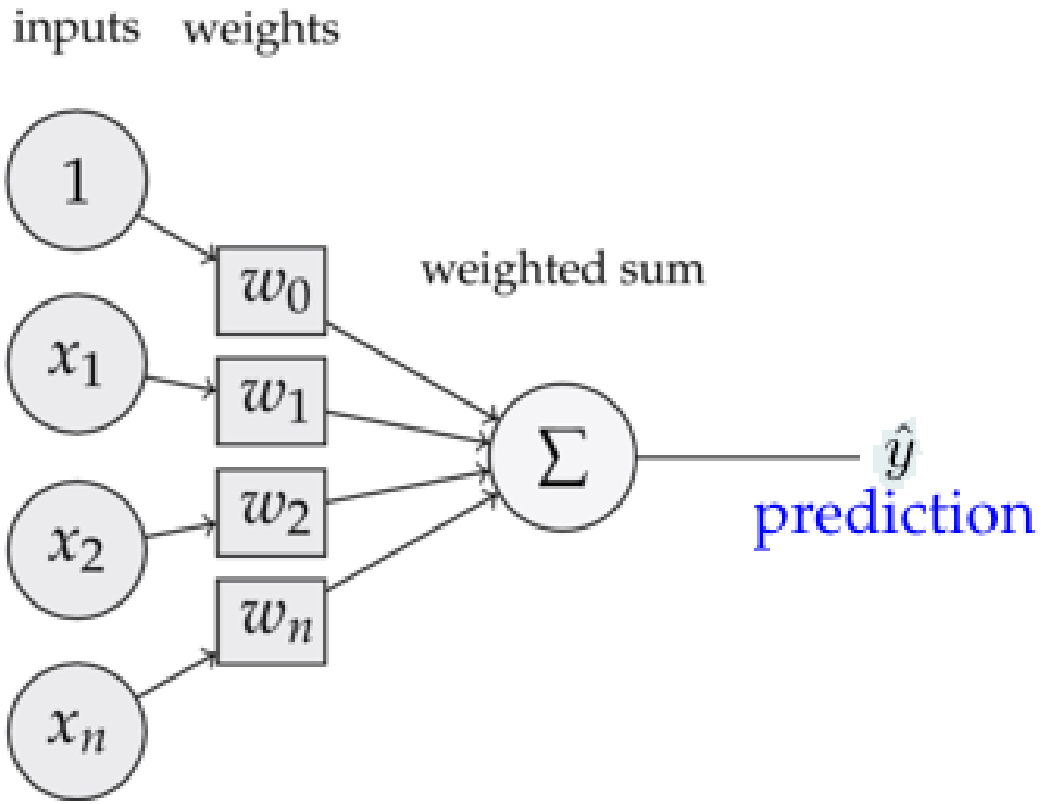
Covered so far

- How to learn a linear function to fit data: **Linear models for regression**
  - Example: predict someone's salary from his/her features.
  - Learns from training data for which both features and the target salary is known and target is a continuous real number
  - Applies the learned model on new data, to predict
- Extension 1: **Basis expansion** on the input side
  - Form new complicated useful features from original features
  - give them as an input to the linear model

  Nonlinearity in the input features is achieved via linear models
- **Regularization**: put a penalty on size of the vector
  - L_1 **regularization**: Gives sparsity (feature selection)
  - L_2 **regularization**: Gives representer theorem, duality, kernels

In this lecture we see another extension, this time on the output side.

Recall: Linear model



- $\hat{y} = \mathbf{x}^\top \mathbf{w}$
- Weight w_i show contribution of feature x_i in the final prediction $\hat{y}$
- Example: predicting salary
  - Features: age, education, laziness
  - Output: salary

Introduction to classification

Targets are categorical variables

Examples:

- cancer/not cancer for patients
- spam/not spam for emails
- invest/not invest on a project
- recruit/not recruit an applicant

Classification

1. Training phase
   Learn a model from training data with known categorical targets. In other words, again fit a function to training data as before.
2. Test phase
   Use the learned model to classify new unseen instances.

The problem is called Binary Classification .
Target represented with a class indicator, $\mathcal{Y} = \{0, 1\}$

Classification problems

All have categorical targets

Types of classification problems

- Binary classification $y \in \{0, 1\}$. $\mathcal{Y} = \{0, 1\}$.
  - Predict one out of two categories.
  - Example: cancer/no cancer diagnosis for a patient
- Multi-class classification
  - Predict one from L categories. **Exclusive classes.**
  - Example: Handwritten digit recognition
    Task: Given image of a single handwritten digit, recognize which of categories of $\{0, 1 \ldots 9\}$ it shows: 10 exclusive categories (classes).
  - Target is a class indicator vector with length L
- Multi-label classification
  - Predict which *ones* out of L categories are relevant. **Non-exclusive labels.**
  - Examples: Forum post or document topic recognition, Image tagging. (Given an image, predict objects inside.)
  - Target is a vector with zero-one components. $\mathbf{y} \in \{0, 1\}^L$

Focus of this lecture:

Binary classification

How to approach a binary classification problem?

Approach of this lecture: Probabilistic approach to classification

Probabilistic approach to classification

- Predict the *probability* of the class being one.
- Training phase
  Fit a function to training data, so that the outputs of the function is close to the value of training targets.
  For training examples, we are given the class probability as zero or one.
- Test phase
  Predict a number in $[0, 1]$ as the probability that the variable is active.
- Example
  Predicting the probability that a patient has cancer from his/her symptoms. Model would be learned from data of past patients with the known outcome.

Linear regression good for predicting probability?

A probability target is in $[0, 1]$.
Would like prediction $\hat{y}$ to respect the same constraints.

Can we use linear regression?

- Good news about probability: Continuous value, as for regression
- Bad news about probability: Not any real number, only real numbers in $[0, 1]$ are meaningful as a probability.

Do we need to change linear model?

Recall

- A linear function has an unconstrained range $\mathcal{Y} = \mathbb{R}$
- So using a linear function for prediction, the predicted probability for a new example can end up invalid, eg -5 or 16.
- But we need a constrained range $\mathcal{Y} = [0, 1]$ to predict a probability.

Any idea how to fix this? Any other type of function to fit to our data?

Linear regression good for predicting probability?

A probability target is in $[0, 1]$.
Would like prediction $\hat{y}$ to respect the same constraints.

Can we use linear regression?

- Good news about probability: Continuous value, as for regression
- Bad news about probability: Not any real number, only real numbers in $[0, 1]$ are meaningful as a probability.

Do we need to change linear model?

Recall

- A linear function has an unconstrained range $\mathcal{Y} = \mathbb{R}$
- So using a linear function for prediction, the predicted probability for a new example can end up invalid, eg -5 or 16.
- But we need a constrained range $\mathcal{Y} = [0, 1]$ to predict a probability.

Any idea how to fix this? Any other type of function to fit to our data?

Linear regression good for predicting probability?

A probability target is in $[0, 1]$.
Would like prediction $\hat{y}$ to respect the same constraints.

Can we use linear regression?

- Good news about probability: Continuous value, as for regression
- Bad news about probability: Not any real number, only real numbers in $[0, 1]$ are meaningful as a probability.

Do we need to change linear model?

Recall

- A linear function has an unconstrained range $\mathcal{Y} = \mathbb{R}$
- So using a linear function for prediction, the predicted probability for a new example can end up invalid, eg -5 or 16.
- But we need a constrained range $\mathcal{Y} = [0, 1]$ to predict a probability.

Any idea how to fix this? Any other type of function to fit to our data?

Linear regression good for predicting probability?

A probability target is in $[0, 1]$.
Would like prediction $\hat{y}$ to respect the same constraints.

Can we use linear regression?

- Good news about probability: Continuous value, as for regression
- Bad news about probability: Not any real number, only real numbers in $[0, 1]$ are meaningful as a probability.

Do we need to change linear model?

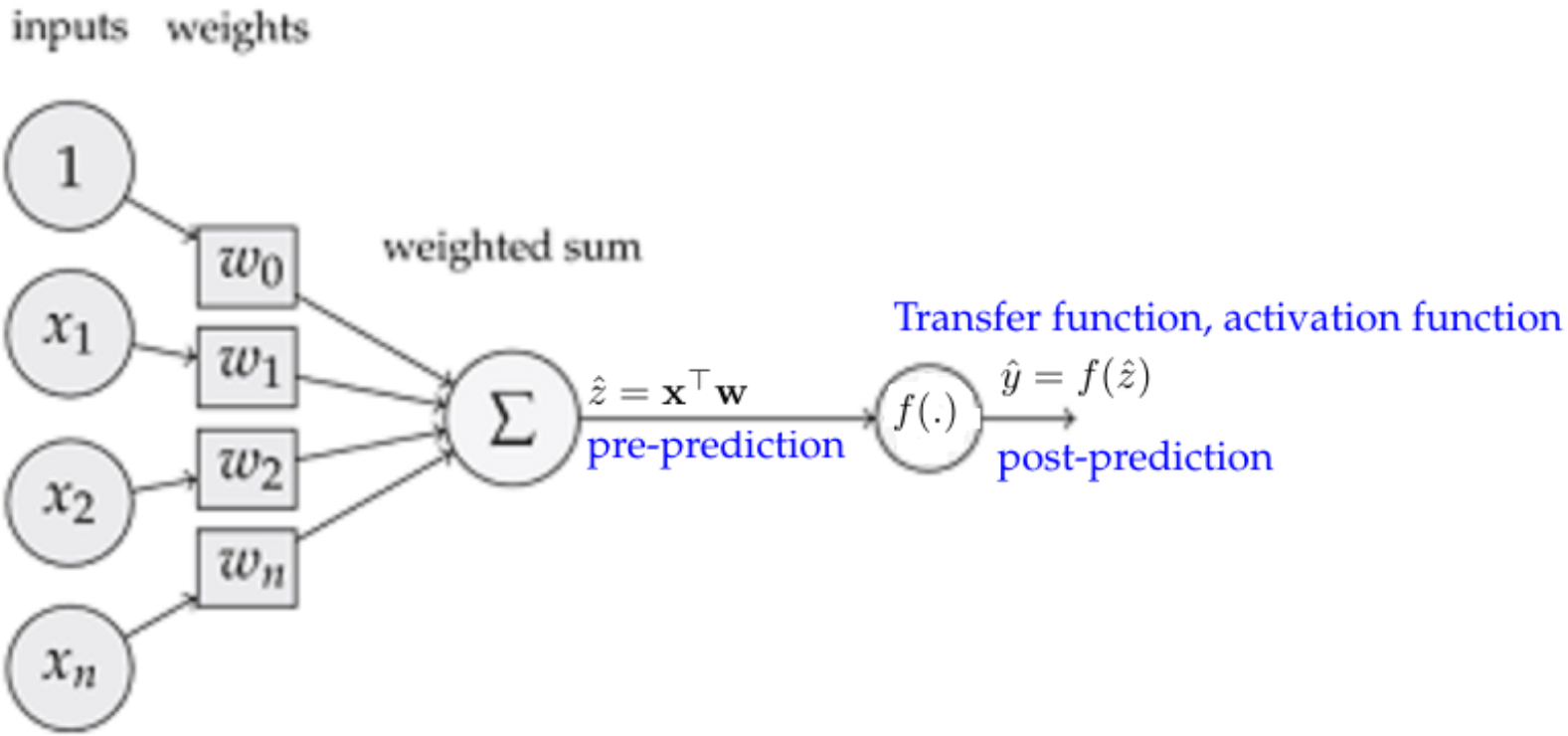
Recall

- A linear function has an unconstrained range $\mathcal{Y} = \mathbb{R}$
- So using a linear function for prediction, the predicted probability for a new example can end up invalid, eg -5 or 16.
- But we need a constrained range $\mathcal{Y} = [0, 1]$ to predict a probability.

Any idea how to fix this? Any other type of function to fit to our data?

Nonlinear output transformation

Idea Use an appropriate non-linear transformation to provide predictions on desired range

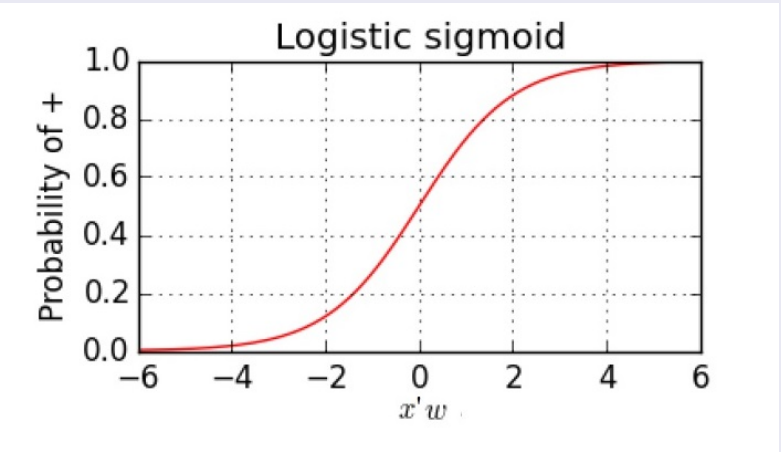


- $\hat{y} = f(\hat{z})$
- $\hat{z} = \mathbf{x}^\top \mathbf{w}$ Pre-prediction
- $\hat{y} = f(\hat{z})$ Post-prediction
- $f(.)$ Transfer function, activation function

I. What transformation is appropriate for a probability outputs?

- Something that changes any real number to a probability.
- If $y \in [0, 1]$ use transfer function $\hat{y} = f(\hat{z}) = \frac{1}{1+\exp(-\hat{z})}$
- Logistic Sigmoid Function
- $f = \sigma(.)$ where

$$\sigma(\hat{z}) = \frac{1}{1 + \exp(-\hat{z})}$$



- Note: Sigmoid is an increasing function
- Properties $\sigma(-\infty) = 0 \quad \sigma(+\infty) = 1 \quad \sigma(0) = 0.5$
- $\sigma'(z) = \sigma(z)(1 - \sigma(z))$

II. What error function to minimize?

Terms *Loss function* and *error function* are used here interchangeably.

What Loss function should we use?

- Note: It is possible to use any loss function with any transfer function But not all such pairs match well, so that training can be done efficiently.
- Risk: Many local minima.

Many local min?

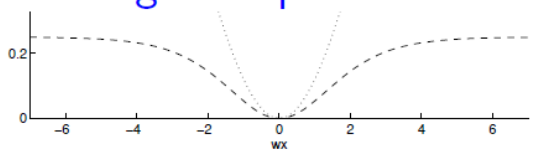
Combining arbitrary f with L can create local minima

E.g.

$$L(\hat{y}; y) = (\hat{y} - y)^2$$
$$f(\hat{z}) = \sigma(\hat{z}) = (1 + \exp(-\hat{z}))^{-1}$$

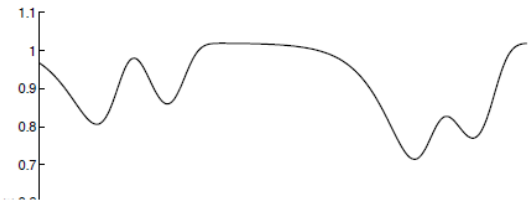
Objective $\sum_i (\sigma(X_{i:}\mathbf{w}) - y_i)^2$ is not convex in $\mathbf{w}$

Consider one training example



(Auer et al. NIPS-95)

Local minima can combine



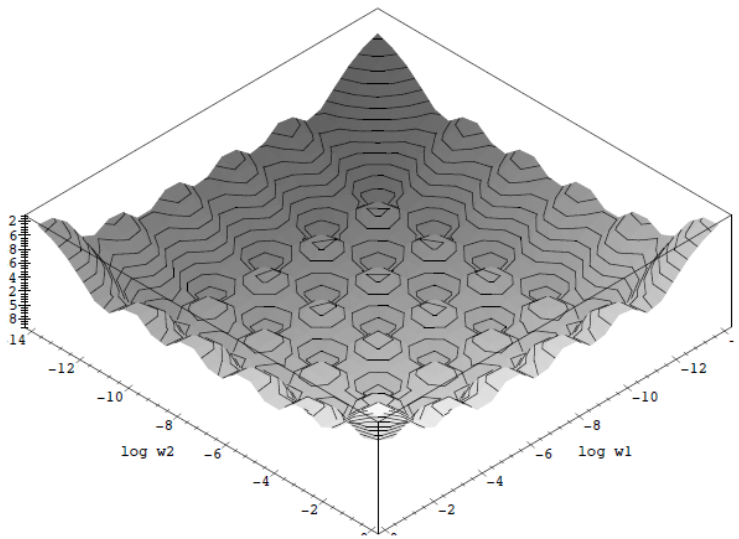
From Dale Schuurmans talk slides

Many local minima?

Risk of creating many local minima

Possible to create exponentially many local minima

t training examples can create $(t/n)^n$ local minima in n dimensions —just local t/n training examples along each dimension



From (Auer et al., NIPS-95)

From Dale Schuurmans talk slides

What loss function to use?

- Good news: Some loss functions and non-linear transfer functions match well. Their composition still remains convex in the parameter
- A matching loss

Matching loss for logistic sigmoid transfer function is:

the cross-entropy loss function

$$\sum_{i=1}^{t} -y_i \ln(\hat{y}_i) - (1 - y_i) \ln(1 - \hat{y}_i)$$

What loss function to use?

- Good news: Some loss functions and non-linear transfer functions match well. Their composition still remains convex in the parameter
- A matching loss

Matching loss for logistic sigmoid transfer function is:

the cross-entropy loss function

$$\sum_{i=1}^{t} -y_i \ln(\hat{y}_i) - (1 - y_i) \ln(1 - \hat{y}_i)$$

What loss function to use?

- Good news: Some loss functions and non-linear transfer functions match well. Their composition still remains convex in the parameter
- A matching loss

Matching loss for logistic sigmoid transfer function is:

the cross-entropy loss function

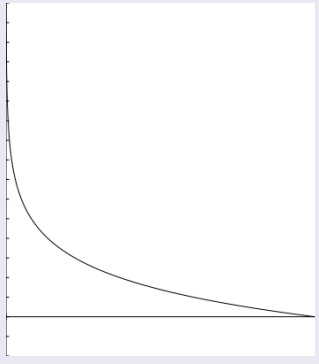
$$\sum_{i=1}^{t} -y_i \ln(\hat{y}_i) - (1 - y_i) \ln(1 - \hat{y}_i)$$

Properties of cross-entropy loss function

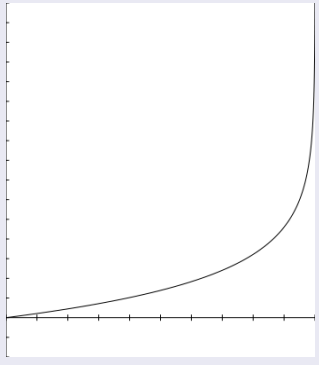
$$l(\hat{y}, y) = -y \ln(\hat{y}) - (1 - y) \ln(1 - \hat{y})$$

Cross-entropy function

- If $y = 1, \hat{y} = 0 \implies loss = \infty$



- If $y = 0, \hat{y} = 1 \implies loss = \infty$



Barrier
$\forall 0 \leq \hat{y} \leq 1, loss(\hat{y}) \geq 0$ (meaningful loss)

Argument 2: Why a cross-entropy loss function is suitable for classification?

- Probabilistic argument: Cross entropy minimizer is a maximum likelihood estimator for data with categorical targets.
- Likelihood of model $\mathbf{w}$ is the probability of getting training labels from this model.

Likelihood

$$
\begin{aligned}
L &= p(\mathbf{y}|X;\mathbf{w}) \\
&= \Pi_{i=1}^{t} p(y_i|\mathbf{x}_i;\mathbf{w}) && \text{(By IID assumption)} \\
&= \Pi_{i=1}^{t} p(y_i=1|\mathbf{x}_i;\mathbf{w})^{y_i}\, p(y_i=0|\mathbf{x}_i;\mathbf{w})^{1-y_i} \\
&= \Pi_{i=1}^{t} (\sigma(\mathbf{x}_i^\top \mathbf{w}))^{y_i} (1-\sigma(\mathbf{x}_i^\top \mathbf{w}))^{(1-y_i)}
\end{aligned}
$$

Log likelihood

$$
\begin{aligned}
&\sum_{i=1}^{t} (\sigma(\mathbf{x}_i^\top \mathbf{w}))^{y_i} + (1-\sigma(\mathbf{x}_i^\top \mathbf{w}))^{1-y_i} \\
&= \sum^{t} y_i \ln(\sigma(\mathbf{x}_i^\top \mathbf{w})) + (1-y_i)\ln(1-\sigma(\mathbf{x}_i^\top \mathbf{w}))
\end{aligned}
$$

III. Complete training formulation

Logistic regression training optimization problem

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{t} -y_i \ln(\sigma(X_{i:}\mathbf{w})) - (1 - y_i) \ln(1 - \sigma(X_{i:}\mathbf{w}))$$

Immediate question

- How to train? How to optimize? In other words, what training algorithm to use?
- Need to look at the type of objective function, constraints, convexity, smoothness to decide.
- Can we find the global min tractably?

IV. How to solve training optimization?

Use a local descent method

- Loss function convex in parameter: Any local min is a global min.
- We have no constraints here

So we can use an *iterative local descent method*.

Examples:

- Gradient descent
- Newton method

Local descent methods

1. Start from any point, set it as the current point
2. While not converged
   1. Find a descent direction at current point
   2. Find a step size (fixed step or a procedure called line search)
   3. Take one step with current step size across the current descent direction, update current point

Simple Optimization Terminology

- A local method finds a local min.
- A descent method finds descent direction until convergence
- Descent direction: in each point is a direction across which the value of the function is guaranteed to decrease taking small enough step size.

- Examples of local descent methods
  - Gradient descent:
    Descent direction$= -$Gradient
  - Newton
    Descent direction$= -(\text{Hessian})^{-1}$ Gradient
- Different in the choice of descent direction
- Trade-off between rate of convergence and low expense of taking each step

Descent directions for logistic regression

- Gradient

$$\begin{aligned}\nabla l(\mathbf{w}) &= X^\top(\mathbf{y} - \hat{\mathbf{y}}) \\ &= X^\top(\mathbf{y} - X\mathbf{w})\end{aligned}$$

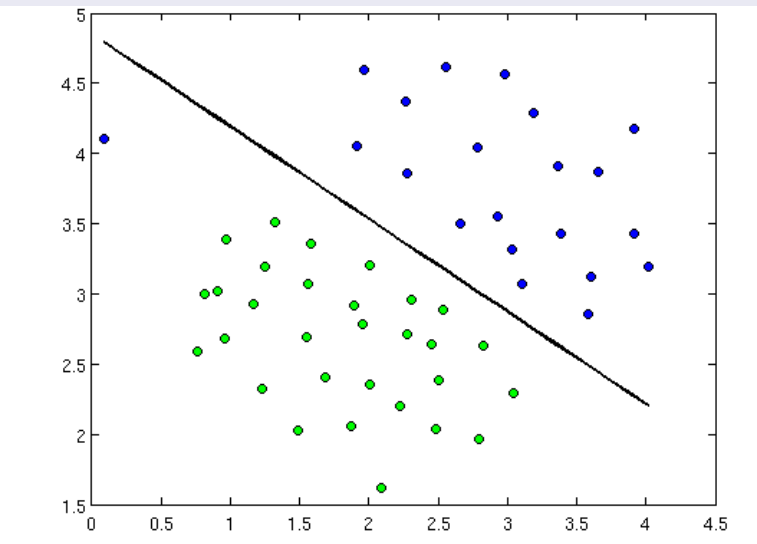
- Hessian

$H = X^\top R X$ where R is a diagonal matrix with $R_{ii} = \hat{y}_i(1 - \hat{y}_i)$

V. Prediction

Prediction formulation

$$\hat{y}(x_\circ) = \begin{cases} 1 & \text{if } \sigma(x_\circ^\top \mathbf{w}) > 0.5 \\ 0 & \text{if } \sigma(x_\circ^\top \mathbf{w}) < 0.5 \end{cases} = \begin{cases} 1 & \text{if } \hat{z} = x_\circ^\top \mathbf{w} > 0 \\ 0 & \text{if } \hat{z} = x_\circ^\top \mathbf{w} < 0 \end{cases} \tag{1}$$



Nonlinear learned function, but linear decision boundary

ML system design note

Combining multiple extensions

It is possible to combine non-linear output transfer (the topic of this lecture) with other extensions from before.

- Non-linear output transform
- Efficient training
- Basis expansion
- Regularization
  - L_2 regularization: Kernels
  - L_1 regularization: Sparsity

Good!

The matching loss paper

- What other output transformation could be useful?
- If the transfer function is increasing there is construction proposed for it to come up with the matching loss.
- What is their matching loss? (First need to check the proposed construction in Auer et al below, NIPS 95)

Exponentially many local minima for single neurons

Peter Auer **Mark Herbster** **Manfred K. Warmuth**

Department of Computer Science
Santa Cruz, California
{pauer,mark,manfred}@cs.ucsc.edu

Question

- Is global optimization still important? Or deep learning has solved every thing?!